國 立 交 通 大 學

電機與控制工程學系

碩士論文

針對過程控制系統之威脅評估和攻擊偵測模組

Threat Assessment and Model-Based Attack Detection for Precess Control Systems

研 究 生：林宗勳

Student: Zong-Syun Lin

指導教授：黃育綸　博士

Advisor: Dr. Yu-Lun Huang

中華民國九十八年五月

May, 2009

針對過程控制系統之威脅評估和攻擊偵測模組

Threat Assessment and Model-Based Attack Detection for Precess Control Systems

研 究 生：林宗勳 Student: Zong-Syun Lin

指導教授：黃育綸　博士 Advisor: Dr. Yu-Lun Huang

國 立 交 通 大 學

電機與控制工程學系

碩士論文

A Thesis

Submitted to Department of Electrical and Control Engineering

College of Electrical Engineering

National Chiao Tung University

in partial Fulfill of the Requirements

for the Degree of

Master

in

Department of Electrical and Control Engineering

May, 2009

Hsinchu, Taiwan, Republic of China

中華民國九十八年五月

針對過程控制系統之威脅評估和攻擊偵測模組

學生：林宗勳　　指導教授：黃育綸　博士

國 立 交 通 大 學電機與控制工程學系（研究所）碩士班

摘　　要

在這個研究中，我們針對過程控制系統（Process Control System）的安全性進行分析。在假設控制器有足夠自保能力的情況下，我們先試著分析並找出系統中最弱、最關鍵的傳感器節點（sensor nodes）。這些節點在遇到入侵攻擊時，會對系統運轉產生影響，並可能造成系統無法正常運作或引起重大危害。我們在這篇論文中的主要貢獻有：一、提出傳感器攻擊模型，以模擬各種完整性攻擊；二、以化學反應爐作為實驗平台，透過一連串的實驗，探討傳感器攻擊對系統效能產生的影響；三、找出防禦行為之間的重大關聯性，並據此建構出以模型為基底之異常偵測模組（Anomaly Detection Module）。此偵測模組可以透過參數的調整，找出最佳的偵測時間與錯誤警報率，使異常偵測模組能夠精確地應用於不同控制系統中。我們同時透過實驗，證明當傳感器遭受到攻擊時，本論文中所提出的異常偵測模組仍能有效地維持系統運作的穩定度與安全度。此外，考慮到攻擊者可能已經洞察或熟悉偵測機制和系統控制的情況，攻擊者可以發動隱密性攻擊（stealthy attack）以躲避異常偵測模組，因此，本研究亦針對隱密性攻擊可能對系統造成的影響加以實驗、分析與討論。本研究結果證實所提出的異常偵測模組可以有效的防禦針對過程控制系統的完整性攻擊與隱密性攻擊，確保整體系統的正常運作。

Threat Assessment and Model-Based Attack Detection for Precess Control Systems

Student: Zong-Syun Lin Advisor: Dr. Yu-Lun Huang

Department of Electrical and Control Engineering

National Chiao Tung University

Abstract

We present security analysis of process control systems (PCS) when an attacker can compromise sensor measurements that are critical for maintaining the operational goals. We present the general sensor attack model that can represent a wide variety of integrity attacks. By taking example of a well studied process control system, we discuss the consequences of sensor attacks on the performance of the system and important implications for designing defense actions. We develop model-based detection methods that can be tuned to limit the false-alarm rates while detecting a large class of sensor attacks. From the attacker's viewpoint, we show that when the detection mechanisms and control system operations are understood by the attacker, it can carry stealth attacks that maximize the chance of missed detection. From the defender's viewpoint, we show that when an attack is detected, the use of model-based outputs maintains safety under compromised sensor measurements.

誌謝

很高興能夠通過碩士口試這個關卡，這對我的研究成果和學術表現無疑是一大肯定，非常感謝指導老師黃育綸博士的耐心指導，除了研究方法之外，在論文寫作方面也不厭其煩的悉心指導，並給予我機會前往美國加州柏克萊大學深造，跟隨外國頂尖學者Alvaro Cardenas和Saurabh Amin學習研究態度以及做事方法，開闊我的眼界也讓我在研究領域更上一層樓。也感謝老師打造出一個氣氛和樂的實驗室環境，讓我和其他實驗室成員能夠安心和樂的全力衝刺學業和論文，也感謝博班蔡欣宜學姐和黃詠文學長在理論和實作上所給予的建議和幫助。另外也感謝iCAST計畫中所有一起為SWOON爆肝的夥伴、term 3一起出國同甘共苦的戰友，那是碩士生涯最難忘的回憶，沒有你們，我相信碩士生涯定會遜色不少。最後感謝父母從小的栽培以及朋友們在求學路上的陪伴，未來我會更加努力，期望有更好的表現。

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Control systems are widely used by industries and factories to stabilize safety-critical processes. Any disruption of these systems can cause irreparable harm to people who depend on them. In this chapter, we first explain what the control system is and analyze the new vulnerabilities in control systems.

1.1 Control Systems

Control systems are computer-based systems that *monitor* and *control* physical processes. These systems represent a wide variety of networked information technology (IT) systems connected to the physical world. Depending on the application, these control systems are also called Process Control Systems (PCS), Supervisory Control and Data Acquisition (SCADA) systems (in industrial control or in the control of the critical infrastructures), or Cyber-Physical Systems (CPS) (to refer to embedded sensor and actuator networks). Control systems are responsible for vital processes in the generation, transmission and distribution of utility services like electric power, oil and gas, water and waste-water as well as in the operation of industrial manufacturing and production units.

Control systems are usually composed of a set of networked agents, consisting of sensors, actuators, control processing units, and communication devices. For example, the oil and gas industry uses integrated control systems to manage refining operations at plant sites, remotely

monitor the pressure and flow of gas pipelines, and control the flow and pathways of gas transmission. Water utilities can remotely monitor well levels and control the wells pumps; monitor flows, tank levels, or pressure in storage tanks; monitor such as pH, turbidity, and chlorine residual; and control the addition of chemicals to the water.

Modern day industrial control systems have a multi-layer control structure [1], [2]. The overall objectives of such a control structure are: (1) to maintain safe operational goals by limiting the probability of undesirable behavior, (2) to meet the production demands by keeping certain process values within prescribed limits, (3) to maximize production profit.

We distinguish between two control layers that are supposed to handle a set of functionally different control objectives while exchanging necessary information with each other. The *regulatory control layer* has direct access to the sensors that measure the process variables. The process variables are controlled by the actuators that receive input signals from the regulatory layer controllers. These controllers are responsible for nominal safety and operation of the processes in the system and can intervene with high frequency. On the other hand, the controllers in *supervisory control layer* is mainly responsible for controlling certain slowly varying process variables and for providing set-points to the controllers in the regulatory layer.

Since the regulatory layer controllers are required to demonstrate faster response, they are traditionally based on the classic proportional-integral-derivative (PID) algorithms. PID control algorithms still govern much of control system practice because they are easy to tune and provide acceptable performance guarantees in many situations. However, the supervisory layer controllers are generally based on optimization based predictive algorithms [3].

Supervisory layer controllers are often centralized and with strong computing power. We can use complex and robust security solutions developing in IT systems to protect against attacks but sensors in the regulatory layer are spread around the plant and with weak computing power.

Hence, sensors are exposed to the risk and easy to be compromised. According to these reasons, we put our attention on attacks to sensor nodes and assume the computing blocks in supervisory layer (all of controllers) are secured.

1.2 New Vulnerabilities and Threats

Several control systems are safety-critical: any attack may cause a significant impact on public health, the environment, the economy, or even lead to loss of human life. Critical infrastructure control systems are more vulnerable today than in the past due to the increased standardization of technologies, the increased connectivity of control systems to other computer networks and the Internet, insecure connections, etc. In short, control systems are becoming more complex and interdependent---and therefore, more vulnerable.

Not only are control systems more vulnerable now, but there is now more groups interested in attacking them. There are more individuals, organized groups and nation states interested in exploiting the vulnerabilities of control systems.

Because of the increasing risk to computer attacks, there has been a significant effort in recent years to discuss and identify the security issues of control systems [4--15].

1.3 Our Approach

We believe that most of the previous work in the security of control systems has three goals: (1) create awareness of security issues with control systems, (2) help control systems operators and IT security officers design a security policy, and (3) recommend basic security mechanisms for prevention (authentication, access controls, etc), detection, and response to security breaches.

While these recommendations and standards have placed significant importance in the *sur-*

vivability of control systems (their ability to operate while they are under attack); we argue that they have not considered new research problems that arise when control systems are under attack. In particular, we believe that, so far, no one has been able to articulate what is new and fundamentally different in this field from a research point of view compared to traditional IT security.

In particular, research in computer security has focused traditionally on the protection of information. Researchers have not considered how attacks affect the estimation and control algorithms and ultimately, how attacks affect the physical world.

In this work we argue that the major distinction of control systems with respect to other IT systems is the interaction of the control system with the physical world. In this paper we propose to incorporate the physical process dynamics in the security analysis of the control system. In this paper we focus on an attacker that compromises sensor readings. We have two major goals (1) to develop a threat assessment methodology, and (2) to design attack detection and response mechanisms.

The goal of threat assessment is to identify the most valuable targets and to understand the possible attack strategies against the control system. We argue that an appropriate threat assessment of control systems must include an analysis of how computer attacks can affect the physical world in order to (1) understand the consequences of an attack, (2) estimate the possible losses, (3) estimate the response time required by defenders, and (4) identify most cost-effective defenses to our system.

The goal of attack detection and response is to prevent attacks crashing the plant. Unlike the security of IT systems, we don't focus on protecting information in control system. The most important thing for control system is to keep running and stay away from crashes. Crashes in control systems may result in serious damage like explosion or poison gas leak. So we try to

ensure the safety of the plant when the plant is under attack.

Incorporating the physical model of the system can also help to detect and respond to attacks. By modeling the dynamics of the physical process we can detect attacks, and use the model to predict the state of the system in case some sensor nodes are compromised.

1.4 Synopsis

In the next chapter, we present a simplified Tennessee-Eastman process control system (TE-PCS) for which a wide variety of regulatory control algorithms have been developed by control system engineers. The operational goals of the control system is to maintain process safety and ensure economical production. Chapter 3 focusses on threat assessment and provides insights on the strategies that an attacker may follow to maximize damage to operational goals. In Chapter 4, we present the general sensor attack model that can model a number of integrity attacks on sensor measurements that are accessible to the attacker. We present a model-based attack detection method that can detect a wide class of attacks on sensor measurements in Chapter 5. In Chapter 6, the attack detection method is tuned to limit false positive rates and its performance is evaluated for detecting sensor attacks on TE-PCS. Chapter 7 presents the case of stealthy attacks in which the attacker can exploit the knowledge of the attack detection and system operation methods to avoid getting detected. We provide concluding remarks in Chapter 8.

Chapter 2

Process Description

In this paper, we use the Tennessee-Eastman process control system (TE-PCS) model and the associated multi-loop PI control law as proposed by Ricker [16] as our experimental environment. We briefly describe the process architecture and the control loops in Figure 2.1. The original process model is implemented in FORTRAN and the PI control law is implemented in MATLAB. We will use this code for our study.

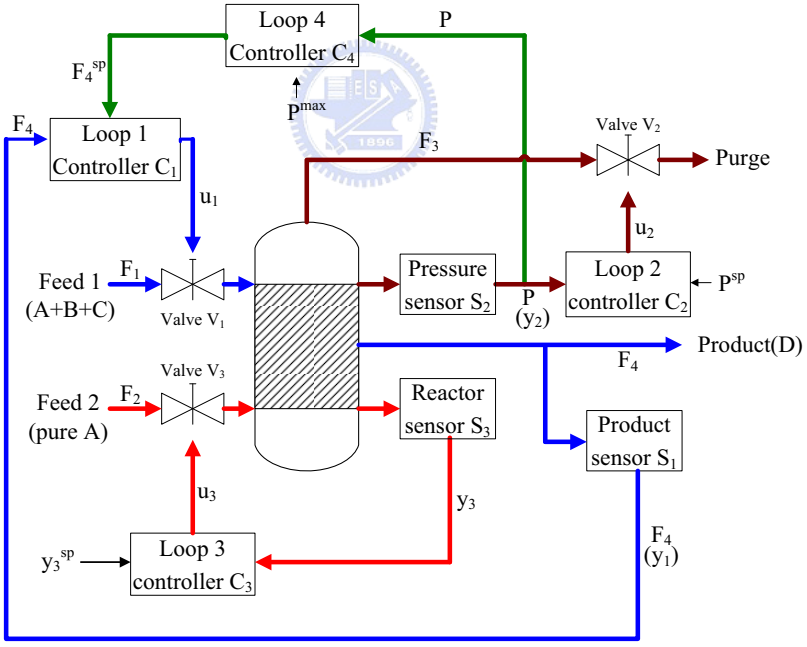


Figure 2.1: Architecture of the Simplified TE Plant

The chemical process consists of an irreversible reaction which occurs in the vapour phase inside a reactor of fixed volume V of 122 (m^3). Two non-condensible reactants A and C react

in the presence of an inert B to form a non-volatile liquid product D:

$$A + C \xrightarrow{B} D.$$

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, C; and stream 4 is the exit for liquid product D. The measured flow rates of stream i is denoted by F_i $(kmol\ h^{-1})$. The *control objectives* are

- to *regulate* F_4, the rate of production of the product D, at a set-point F_4^{sp} $(kmol\ h^{-1})$,
- while maintaining P, the operating pressure of the reactor, below the crash limit of $3000\ kPa$ as dictated *safety* considerations,
- to keep y_3, the ratio of ingredient A in reactor, ensure operation of the chemical reactor.

According to the control objectives mentioned above, Ricker [16] suggests the control loops as follows (also see Figure 2.1):

- Control loop 1:

  The first control loop contains loop 1 controller C_1, feed 1 valve V_1 and product sensor S_1.

  Loop 1 controller monitors production rate $F_4 = y_1$ and computes control signals u_1 to adjust feed 1 valve V_1. By adjusting the value of V_1, loop 1 controller changes the amount of input ingredients and regulates production rate.

- Control loop 2:

  The second control loop contains loop 2 controller C_2, purge valve V_2 and pressure sensor S_2.

Loop 2 controller receives sensor signals $P = y_2$ sended by pressure sensor and computes controls signals u_2. Control signals u_2 open purge valve to let vapor in reactor goes outside if pressure is overhigh, and vice versa.

- Control loop 3:

  The third control loop contains loop 3 controller C_3, feed 2 valve V_3 and reactor sensor S_3.

  Loop 3 controller gets sensor signals y_3 which is the ratio of ingredient A in reactor and computes control signals u_3 to adjust feed 2 valve V_3. Loop 3 controller keeps the ratio of A in reactor by monitoring y_3 and tuning feed 2 valve.

- Control loop 4:

  The fourth control loop contains loop 4 controller C_4, loop 1 controller C_1, feed 1 valve V_1 and pressure sensor S_2.

  Loop 4 controller is needed prevent the loss of pressure control resulting from saturation of the purge rate u_2 in loop-2 controller. When $P = y_2$ is overhigh, the loop 4 controller changes set-point F_4^{sp} and uses u_1 to control pressure $P = y_2$.

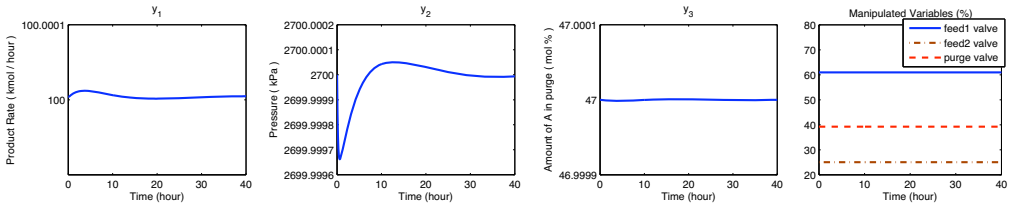


Figure 2.2: Signals without Noise

These four control loops help plant work in steady-state. During the steady-state operation, the production rate $F_4 = y_1$ is 100 $kmol\ h^{-1}$, the pressure $P = y_2$ is 2700 KPa and the fraction of A in reactor y_3 is 47 $mol\%$. We also add a Gaussian disturbance with zero mean and variance

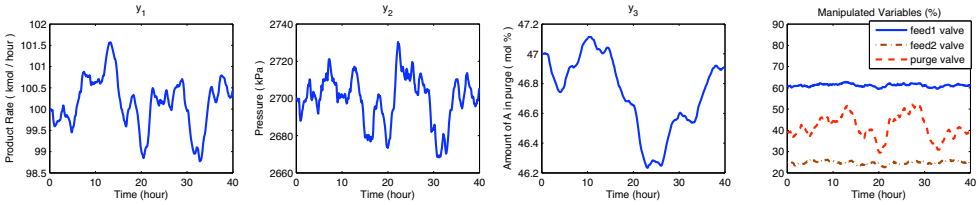


Figure 2.3: Signals with Gaussian noise, mean = 0; var = 0.2

0.2 to the control inputs $u_i(k)$ so that the system is never in a complete steady state. Fig 2.2 and 2.3 show the simulation of the plant signals without noise (at steady state) and with noise respectively.

Chapter 3

Threat Assessment

We study the security issues of control systems by experimenting and simulating cyber attacks on sensor signals in the TE-PCS model. Because operating the chemical reactor with a pressure larger than 3000 kPa is unsafe. It may lead to an explosion or damage of the equipment. Assume that that the goal of the attacker is to raise the pressure level of the tank to a value larger than 3000 kPa, we attack a single sensor at a given time. Let y_i denotes the measurement by sensor i and $\tilde{y}$ denote the received measurements by the controller. The sensor signals y_i equal to $\tilde{y_i}$ if there are no attacks on sensor nodes; y_i differ to $\tilde{y_i}$ when plant is under attack. From the experimental results, we found that the most effective of these attacks were the max/min attacks (make the forged signals the extreme values, i.e, y^{max} or y^{min}); however, not all of them were able to drive the pressure to unsafe levels. We now summarize some of the results.

- By attacking the sensors, a controller is expected to respond with incorrect control signals since it receives wrong information from the compromised sensors. If an attacker does not know the dynamics of the plant, or the control algorithms, it may compromise a given device at random. For example, by forging y_3 as y_3^{max} from $t = 10$ to 30, the controller believes there is a large amount of ingredient A in the tank.

  From the experiments, we found that the plant system can go back to the steady state after the attack finishes, as illustrated in Fig 3.1. Furthermore, the pressure in the main tank never reaches 3000 kPa. In general we found that the plant is very resilient to attacks on y_1 and y_3. Even attacks in the limit of the sensing range (y^{min} and y^max) can't force the

system into an unsafe state.

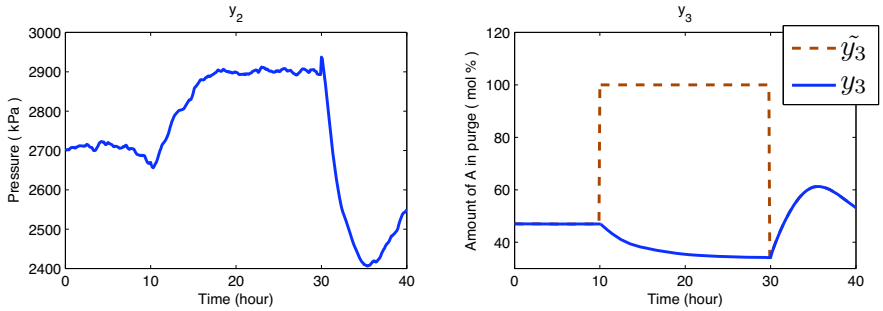
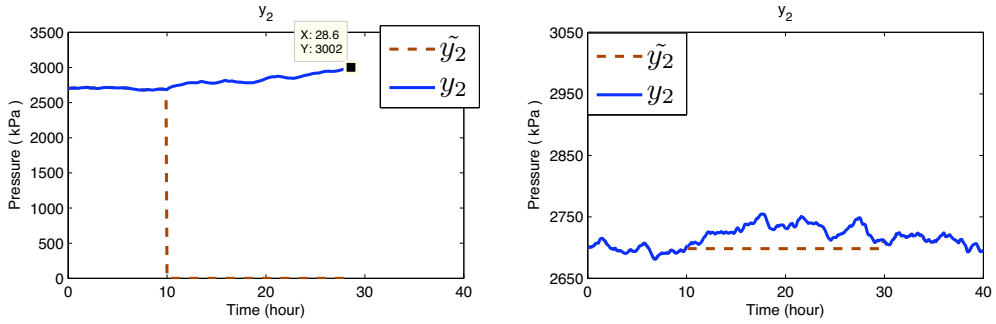


Figure 3.1: Integrity attack y_3^{max} from $t = 10$ to 30. The system remains in a safe state for attacks on y_3.

- By launching attack y_2^{min} the controller turns down the purge valve to increase the pressure and prevent the liquid products from accumulating. We can see that the real pressure of the tank (y_2 in Fig 3.2(a)) keeps increasing past 3000 kPa and the system operates in an unsafe state. In this experiment, it takes about 20 hours ($t = 10$ to $t = 30$) to crash (or cause an explosion to) the plant. It takes long time that the system operators may observe this unusual phenomenon and take proper actions against the attack.



(a) (b)

Figure 3.2: Safety can be breached by compromising sensor y_2 (3.2(a)). DoS attacks, on the other hand, do not cause any damage (and they are easy to detect.) (3.2(b)).

- We found out that in general DoS attacks do not affect the plant. We ran the plant 20 times for 40 hours each and for a DoS attack lasting 20 hours the pressure in the tank never exceeded 2900 kPa.

We conclude that if the plant operator wants to prevent an attack from making the system operate in an unsafe state, it should prioritize the integrity of the sensors rather than their availability. Some of the sensors are more critical to the plant, in this experiment, the sensor y_2 should also be a priority.

Threat assessment helps us find out some easy way to crash TE-PCS. Following we try to model more effective attacks according to the results of threat assessment.

Chapter 4

Attack models

In this paper we focus on attacks on sensor networks and the effects they can have on the process control system. We consider the case when the state of the system is measured by a sensor network of p sensors that observes the measurement vector $y(k) = \{y_1(k), \ldots, y_p(k)\}$, where $y_i(k)$ denotes the measurement by sensor i at time k. All sensors have a dynamic range that defines the domain of y_i for all k. That is, all sensors have defined minimum and maximum values $\forall k, y_i(k) \in [y_i^{min}, y_i^{max}]$. Let $\mathcal{Y}_i = [y_i^{min}, y_i^{max}]$. We assume each sensor has a unique identity protected by a cryptographic key.

Let $\tilde{y}(k) \in \mathbb{R}^p$ denote the *received measurements by the controller* at time k. Based on these measurements the control system defines control actions to maintain certain operational goals. If some of the sensors are under attack, $\tilde{y}(k)$ may be different from the real measurement $y(k)$; however, we assumed that the attacked signals $\tilde{y}_i(k)$ also lie within $\mathcal{Y}_i$ (signals outside this range can be easily detected by fault-tolerant algorithms).

Let $\mathcal{K}_a = \{k_s, \ldots, k_e\}$ represent the attack duration; between the start time k_s and stop time k_e of an attack.

4.1 Additive Attacks

Additive attacks add an arbitrary noise λ_a to sensor signals. λ_a may be either a constant or a random number. In the former case, attackers forge sensor signals by shifting the original signals

a constant offset λ_a. If λ_a is a random number, then attackers alter the original signals by adding some random noises.

To avoid the detection of fault-tolerant detector, the forged signals should lie within the range $[y_i^{min}, y_i^{max}]$. The additive attack can be modeled as follows.

$$\tilde{y}_i^a(t) = \begin{cases} y_i(t) & \text{for } t \notin \mathcal{T}_a \\ y_i(t) + \lambda_a & \text{for } t \in \mathcal{T}_a \ \& \ y_i(t) + \lambda_a \in \mathcal{Y}_i \\ y_i^{min} & \text{for } t \in \mathcal{T}_a \ \& \ y_i(t) + \lambda_a < y_i^{min} \\ y_i^{max} & \text{for } t \in \mathcal{T}_a \ \& \ y_i(t) + \lambda_a > y_i^{max} \end{cases}$$

4.2 Multiplicative Scaling Attacks

Namely, multiplicative scaling attacks attack a system by changing the scale of sensor signals. By carefully choosing an *scaling parameter* $\lambda_m(t)$, the adversary alters the sensor signal y and sends an *scaling value* $\tilde{y}$ to the controller. To avoid the detection of fault-tolerant detector, the forged signals should lie within the range $[y_i^{min}, y_i^{max}]$. Thus, we can model an scaling attack as follows.

$$\tilde{y}_i^m(k) = \begin{cases} y_i(k) & \text{for } k \notin \mathcal{K}_a \\ \lambda_i^m(k) y_i(k) & \text{for } k \in \mathcal{K}_a \\ y_i^{min} & \text{for } k \in \mathcal{K}_a \ \& \ \lambda_i^m(k) y_i(k) < y_i^{min} \\ y_i^{max} & \text{for } k \in \mathcal{K}_a \ \& \ \lambda_i^m(k) y_i(k) > y_i^{max} \end{cases}$$

After launching an scaling attack to the sensor i, the sensor signals y_i are changed to $\tilde{y}_i$. Without a proper defense mechanism, the controllers may mistakenly command the actuators and lead the plant system into an unstable and insecure state.

4.3 Delay Attacks

Real-time requirement is one of the most important issue in industrial control systems. The control systems can not handle disturbances without real-time information. To crash the plant system or make it unstable, an adversary may slow down the reaction of a controller by delaying the sensor signals. The following equations model the delay attack, where λ_d is the delay time of sensor signals.

$$\tilde{y}_i^d(k) = \begin{cases} y_i(k) & \text{for } k \notin \mathcal{K}_a \\ y_i(k - \lambda_i^d(k)) & \text{for } k \in \mathcal{K}_a \text{ \& k} \geq \lambda_i^d(k) \\ y_i(0) & \text{for } k < \lambda_i^d(k) \end{cases}$$

Note that the initial signal $y(0)$ is applied if the current time t is earlier than the specified parameter λ_d.

After modeling variety of attacks, we now study an alternative and more cost-effective approach: automatic detection and response to attacks by using anomaly detection schemes.

Chapter 5

Model-Based Attack Detection

Detecting attacks to control systems can be formulated as anomaly-based intrusion detection systems [17]. One big difference in control systems compared to traditional IT systems, is that instead of creating models of network traffic or software behavior, we can use a representative model of the physical system.

Our argument is that if we know how the output sequence of the physical system, $y(k)$, should react to the control input sequence, $u(k)$, then any attack to the sensor and control data can be potentially detected. Given an input sequence $u(k)$, the attacked sensor measurements will exhibit abnormal deviations from the expected sensor outputs leading to detection of sensor attacks. Similarly, given an output sequence $y(k)$, any attack to the control inputs can also be detected by comparing the deviations of the attacked control inputs with the expected control inputs.

Our proposed attack detection system is presented in Figure 5.1. The control input sequence $u(k)$ is fed to the physical system after being perturbed by an additive Gaussian process noise sequence $w(k)$. The process noise sequence can be thought as unmodeled factors that affect the evolution of system state. The input sequence $u(k)$ is also fed to a system model that is representative of the physical system and is internal to the detection system. (In this case, we use a linear model as described in Section 5.1.) The internal model will produce an output sequence $\hat{y}(k)$. The anomaly detection module (ADM) will compare the two measurement sequences: the sequence $\tilde{y}(k)$ that is received from the sensor measurements and may have been influenced by

the attacker with the sequence $\hat{y}(k)$ that is obtained from internal model. The ADM raises an alert if the deviation between the two sequences is significant.

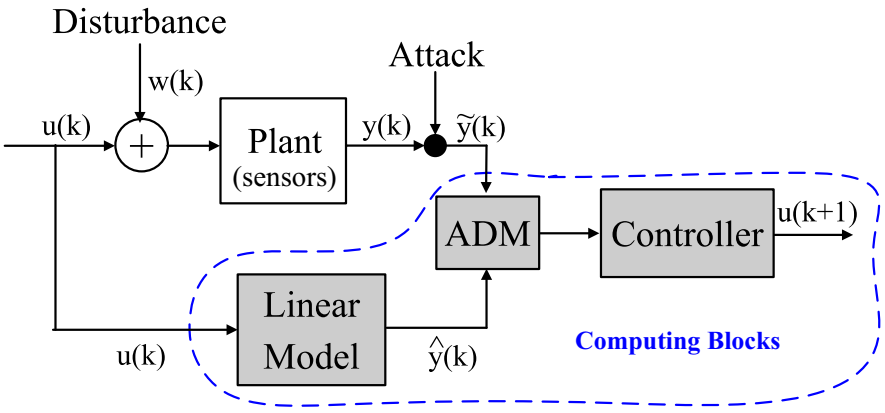


Figure 5.1: The proposed detection module: use Linear Model.

To formalize this problem, we need (1) a simple model that is representative of physical system, and (2) an anomaly detection algorithm. In section 5.1 we discuss our choice of linear model as the representative model. In section 5.2, we describe change detection theory and the detection algorithm we use - a nonparametric cumulative sum (CUSUM) statistic.

5.1 Linear Model

To develop accurate control algorithms, control engineers often construct a representative model that captures the behavior of the physical system in order to predict how the system will react to a given control signal. A representative process model can be derived from first principles --a model based on the fundamental laws of physics-- or from empirical input and output data --a model obtained by simulating the process inputs with a carefully designed test sequence. It is also very common to use a combination of these two models; for example, first-principle models are typically calibrated by using process test data to estimate key parameters. Likewise, empirical models are often adjusted to account for known process physics [2, 3].

For highly safety-critical applications, such as the aerospace industry, it is technically and economically feasible to develop accurate representative models [2]. However, for majority of process control systems, the development of process models from fundamental physics is difficult: in many cases such detailed models are difficult to justify economically, and possibly even impossible to obtain in reasonable time due to the complex nature of many systems and processes. To solve these problems most control vendors provide tools (called identification packages) to develop representative models of physical systems from training data. The most common representative models are *linear* systems.

Linear systems can be used to model dynamics that are linear in state $x(k)$ and control input $u(k)$

$$x(k+1) = Ax(k) + Bu(k) \tag{5.1}$$

where time is represented by $k \in \mathbb{Z}^+$, $x(k) = (x_1(k), \ldots, x_n(k)) \in \mathbb{R}^n$ is the state of the system, and $u(k) = (u_1(k), \ldots, u_m(k)) \in \mathbb{R}^m$ is the control input. The matrix $A = (a_{ij}) \in \mathbb{R}^{n \times n}$ models the physical dependence of state i on state j, and $B = (b_{ij}) \in \mathbb{R}^{n \times m}$ is the input matrix for state i from control input j.

Assume that the system (5.1) is monitored by a *sensor network* with p sensors. We can obtain the representative measurement sequence from the observation equations

$$\hat{y}(k) = Cx(k), \tag{5.2}$$

where $\hat{y}(k) = (\hat{y}_1(k), \ldots, \hat{y}_p(k)) \in \mathbb{R}^p$, and $\hat{y}_l(k) \in \mathbb{R}$ is the estimated measurement collected by sensor l at time k. Matrix $C \in \mathbb{R}^{p \times n}$ is called output matrix.

In this paper we use the linear system, characterized by the matrices A, B, and C, obtained by linearizing the non-linear TE-PCS model about the steady-state operating conditions. (See Ricker [16].) The steady-state conditions are mentioned at the end of Section 2. The linear

model is a good representative of the actual TE-PCS model when the operating conditions are reasonably close to the steady-state.

5.2 Detection Methods

Because we need to detect anomalies in real time, we can use results from sequential detection theory to give a sound foundation to our approach. Sequential detection theory considers the problem where the measurement time is not fixed, but can be chosen online as and when the measurements are obtained. Such problem formulations are called *optimal stopping problems*. Two such problem formulations are: sequential detection (also known as sequential hypothesis testing), and quickest detection (also known as change detection). A good survey of these problems is given by Kailath and Poor [18].

In optimal stopping problems, we are given a time series sequence $z(1), z(2), \ldots, z(N)$, and the goal is to determine the minimum number of samples, N, the anomaly detection scheme should observe before making a decision d_N between two hypotheses: H_0 (normal behavior) and H_1 (attack).

The main difference between sequential detection and change detection is that the former assumes the sequence $z(i)$ is generated either by the normal hypothesis H_0 or by the attack hypothesis H_1. The goal is to decide which hypothesis is true in minimum time. On the other hand, change detection assumes that the observation $z(i)$ starts under H_0 and then, at a given k_s it changes to hypothesis H_1. Here the goal is to detect this change as soon as possible.

Both problem formulations are very popular, but security researchers have used sequential detection more frequently. However, for our attack detection method, the change detection formulation is more intuitive. To facilitate this intuition, we now briefly describe the two for-

mulations.

5.2.1 Sequential Detection

Given a fixed probability of false alarm and a fixed probability of detection, the goal of sequential detection problem is to minimize the number of observations required to make a decision between two hypotheses. The solution is the classic sequential probability ratio test (SPRT) of Wald [19] (also referred as the threshold random walk (TRW) by some security papers). SPRT has been widely used in various problems in information security such as detecting portscans [20], worms [21], proxies used by spammers [22], and botnets [23].

Assuming that the observations $z(k)$ under H_j are generated with a probability distribution p_j, the SPRT can be described by the following equations:

$$S(k+1) = \log \frac{p_1(z(k))}{p_0(z(k))} + S(k)$$

$$N = \inf_n \{n : S(n) \notin [L, U]\}.$$

starting with $S(0) = 0$. The SPRT decision rule d_N is defined as:

$$d_N = \begin{cases} H_1 & \text{if } S(N) \geq U \\ H_0 & \text{if } S(N) \leq L, \end{cases} \tag{5.3}$$

where $L \approx \ln \frac{b}{1-a}$ and $U \approx \ln \frac{1-b}{a}$, and where a is the desired probability of false alarm and b is the desired probability of missed detection (usually chosen as small values).

5.2.2 Change Detection

Goal of the change detection problem is to detect a possible change, at an unknown change point k_s.The cumulative sum (CUSUM) and the Shiryaev-Roberts statistics are the two most

commonly used algorithms for change detection problems, the CUSUM statistic being very similar to the SPRT.

Given a fixed false alarm rate, the CUSUM algorithm attempts to minimize the time N (where $N \geq k_s$) for which the test stops and decides that a change has occurred. Let $S(0) = 0$. The CUSUM statistic is updated according to

$$S(k+1) = \left(\log \frac{p_1(z(k))}{p_0(z(k))} + S(k) \right)^+ \tag{5.4}$$

where $(a)^+ = a$ if $a \geq 0$ and zero otherwise. The stopping time is:

$$N = \inf_n \{ n : S(n) \geq \tau \} \tag{5.5}$$

for a given threshold τ selected based on the false alarm constraint.

We can see that the CUSUM algorithm is an SPRT test with $L = 0$, $U = \tau$, and whenever the statistic reaches the lower threshold L, it re-starts.

One problem that we have in our case is that we do not know the probability distribution for an attack p_1. In general, an adaptive adversary can select any arbitrary (and possibly) non-stationary sequence $z_i(k)$. Assuming a fixed p_1 will thus limit our ability to detect a wide range of attacks.

To avoid making assumptions about the probability distribution of an attacker, we use ideas from nonparametric statistics. We do not assume a parametric distribution for p_1 and p_0; instead, only place mild constraints on the observation sequence. One of the simplest constraints is to assume that the expected value of random process $Z_i(k)$ that generates the sequence $z_i(k)$ under H_0 is less than zero ($\mathbb{E}_0[Z_i] < 0$) and the expected value of $Z_i(k)$ under H_1 is greater than zero ($\mathbb{E}_1[Z_i] > 0$). In the following, we use the subscript i to denote the sequence corresponding to sensor i.

To achieve these conditions let us define

$$z_i(k) := |\tilde{y}_i(k) - \hat{y}_i(k)| - b_i \tag{5.6}$$

where b_i is a small positive constant chosen such that

$$\mathbb{E}_0[|\tilde{y}_i(k) - \hat{y}_i(k)| - b_i] < 0 \tag{5.7}$$

The nonparametric CUSUM statistic for such for sensor i is

$$S_i(k) = (S_i(k-1) + z_i(k))^+, S_i(0) = 0 \tag{5.8}$$

and the corresponding decision rule is

$$d_{N,i} \equiv d_\tau(S_i(k)) = \begin{cases} H_1 & \text{if } S_i(k) > \tau_i \\ H_0 & \text{otherwise.} \end{cases} \tag{5.9}$$

where τ_i is the threshold selected based on the false alarm rate for sensor i.

Following [24], we state the following two important results for CUSUM algorithm (5.8)-(5.9):

- The probability of false alarm decreases exponentially as the threshold τ_i increases,

- The time to detect an attack, $(N_i - k_{s,i})^+$, is inversely proportional to b_i.

5.2.3 Response Using Linear Model

While response is an essential task for information assurance, there are very few intrusion detection papers proposing response strategies.

Our response strategy (shown in Fig 5.1) can be summarized as follows: For sensor i, if $S_i(k) > \tau_i$, the ADM replaces the sensor measurements $\tilde{y}_i(k)$ with measurements generated by the linear model $\hat{y}_i(k)$ (that is the controller will receive as input $\hat{y}_i(k)$ instead of $\tilde{y}_i(k)$). Otherwise, it treats $\tilde{y}_i(k)$ as the correct sensor signal.

We would like to point out that our response strategy is meant as a temporary solution before a human operator responds to the alarm. Based on our results we believe that the time for a human response can be very large (a couple of hours).

Chapter 6

Experiments

In this chapter, we first explain how to determine the two parameters (b and τ) of the non-parametric CUSUM statistic. Then, we describe how our defense system works under different attacks.

6.1 Training Parameters

6.1.1 Training b

In order to detect the anomalies, we need to infer the expected value of the distance $|\hat{y}_i(k) - y_i(k)|$ between the linear model estimate $\hat{y}_i(k)$ and the sensor measurement $y_i(k)$.

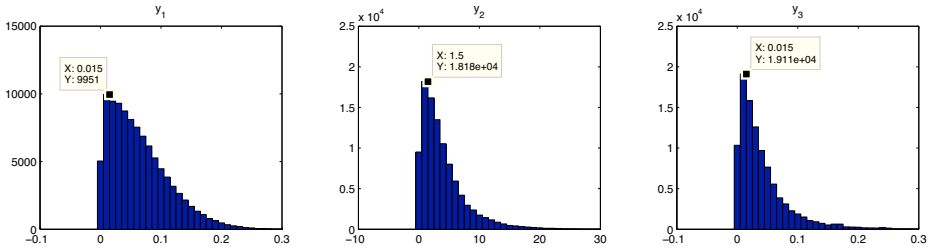


Figure 6.1: The paramenter of ADM: b. For y_1, 9951 bs are 0.015. The mean value of b_{y_1} is 0.0642.

We run experiments for ten thousand times (and for 40 hours each time) without any attacks to gather statistics. Fig 6.1 shows the estimated probability distributions (without normalization).

To obtain b_i, we compute the empirical expected value for each distance and then round up

to the most significant units. We obtain $b_{y_1} = 0.064$, $b_{y_2} = 4.1$, $b_{y_3} = 0.041$.

6.1.2 Training τ

Once we have b_i for each sensor, we need to find a threshold τ_i to balance the tradeoff between false alarms and detection time.

False Alarm Rate

We run simulations for twenty times without attacks and compute the total number of false alarms for different values of τ (and for each sensor). Fig 6.2 shows the results. Taking y_1 as an example, we notice that S_{y_1} alerts frequently if we set $\tau_{y_1} < 6$.

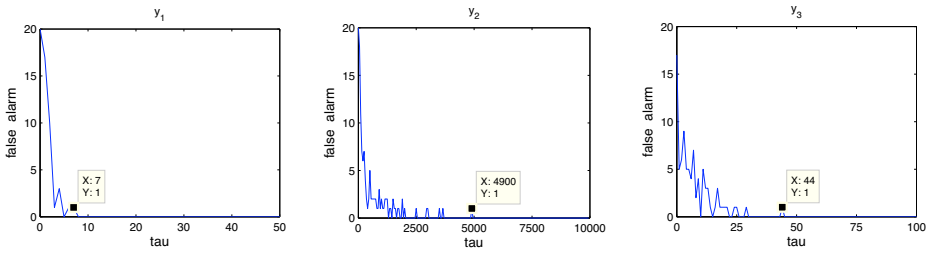


Figure 6.2: The number of false alarms decreases exponentially with increasing τ. This results confirm the theory supporting the nonparametric CUSUM algorithm.

In general, we would like to select τ as high as possible for each sensor to avoid any false alarm; however, increasing τ increases the time to detect attacks.

Detection Time

To measure the time cost of attack detection, we run simulations by launching scaling attacks on sensors y_1, y_2 and y_3. Fig 6.3 shows the experimental results.

- For y_1,

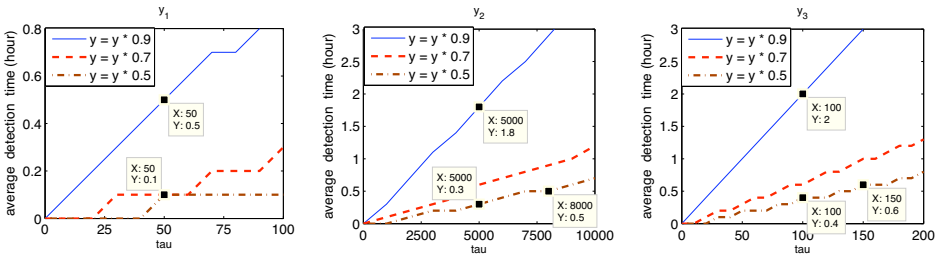


Figure 6.3: The time for detection increases linearly with increasing τ. This results confirm the theory behind the nonparametric CUSUM algorithm.

  - It takes 0.5 hours to detect the scaling attack when λ_m is set to 0.9 and τ_{y_1} is set to 50.
  - It takes 0.1 hours to detect the scaling attack when λ_m is set to 0.5 and τ_{y_1} is set to 50.

- For y_2,
  - It takes 1.8 hours to detect the scaling attack when λ_m is set to 0.9 and τ_{y_2} is set to 5000.
  - It takes 0.3 hours to detect the scaling attack when λ_m is set to 0.5 and τ_{y_2} is set to 5000.
  - It takes 0.5 hours to detect the scaling attack when λ_m is set to 0.5 and τ_{y_2} is set to 8000.

- For y_3,
  - It takes 2 hours to detect the scaling attack when λ_m is set to 0.9 and τ_{y_3} is set to 100.
  - It takes 0.4 hours to detect the scaling attack when λ_m is set to 0.5 and τ_{y_3} is set to 100.
  - It takes 0.5 hours to detect the scaling attack when λ_m is set to 0.5 and τ_{y_3} is set to

100.

The selection of the τ is a trade-off between detection time and the number of false alarms. The appropriate value differs from system to system. Because the large number of false alarms is one of the main problems for anomaly detection systems, we choose the conservative set of parameters $\tau_{y_1} = 50$, $\tau_{y_2} = 10000$, $\tau_{y_3} = 200$.

6.2 Safety of Automatic Response

A comprehensive security posture for any system should include mechanisms for prevention, detection, and response to attacks. Most of the technical efforts for securing systems focus on prevention (authentication, access controls, etc.) and detection (intrusion detection systems, integrity codes, etc.). In contrast, there has been fewer efforts for designing automatic responses to attacks.

The primary problem in designing a computer-automated response is that in several cases, responding to attacks may cause various negative consequences. This is particularly true if a response mechanism is triggered by a false alarm. In our proposed detection and response architecture (Fig. 5.1), we have to make sure that if there is a false alarm, controlling the system by using the estimated values from the linear system will not cause any safety concerns.

To address these concerns we ran the simulation scenario without any attacks 1000 times; each time the experiment ran for 40 hours. As expected, with the parameter set $\tau_{y_1} = 50$, $\tau_{y_2} = 10000$, $\tau_{y_3} = 200$ our system did not detect any false alarm (see Table 6.1); therefore we decided to reduce the detection threshold to $\tau_{y_1} = 5$, $\tau_{y_2} = 1000$, $\tau_{y_3} = 20$ and run the same experiments again. Table 6.2 shows the behavior of the pressure after a response to a false alarm. We can see that while a false response mechanism increases the pressure of the tank, it never

Alarms	Avg y_2	Std Dev	Max y_2
0	2700.4	14.73	2757

Table 6.1: For Thresholds $\tau_{y_1} = 50, \tau_{y_2} = 10000, \tau_{y_3} = 200$ we obtain no false alarm. Therefore we only report the expected pressure, the standard deviation of the pressure, and the maximum pressure reached under no false alarm.

	Alarms	Avg y_2	Std Dev	Max y_2
y_1	61	2710	30.36	2779
y_2	106	2705	18.72	2794
y_3	53	2706	20.89	2776

Table 6.2: Behavior of the plant after response to a false alarm with thresholds $\tau_{y_1} = 5, \tau_{y_2} = 1000, \tau_{y_3} = 20$.

reaches dangerous levels. The maximum pressure obtained while controlling the system based on the linear model was $2779kPa$, which is in the same order of magnitude than the normal variation of the pressure without any false alarm ($2757kPa$).

6.3 Detection and Response

Operating the chemical reactor with a pressure larger than 3000 kPa is unsafe (it can lead to an explosion or damage of the equipment), so one of the main goals of an attacker is to raise the pressure level of the tank to a value larger than 3000 kPa. In this section, we implement the integrity attacks described in previous chapter, including multiplicative scaling attacks, delay attacks, etc.

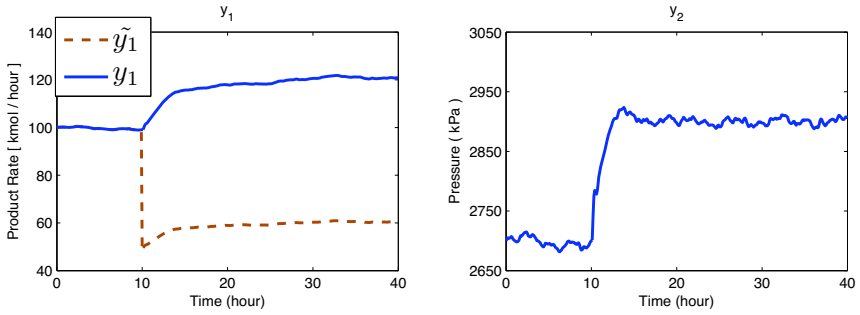
6.3.1 Multiplicative Scaling Attacks

Multiplicative scaling attacks deceive controllers by sending compromised sensor signals and the controller is expected to respond with incorrect control signals. We launch scaling attack on sensor y_1, y_2, y_3 to see how attacks affect plant and the performance of ADM.

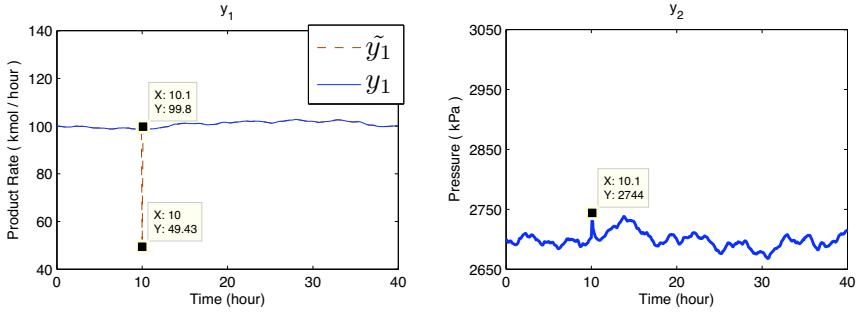
- Attacking y_1

  We first launch scaling attacks to sensor y_1(production rate) at time $T = 10$ (hr). By lowering the production rate, controller opens the input valves to raise production rate and causes pressure raise. However, the plant uses two controllers to protect the pressure. The pressure stops raising when another controller observes pressure is over 2900 kPa and turns down the input valve.

  We enable ADM and repeat the experiment again. ADM detects the attack at time $T = 10.1$ (hr) and the pressure of y_2 remains around 2700 kPa. We conclude that attacking y_1 is not good because it doesn't make plant crash and is detected too fast.



(a) without ADM, pressure raise



(b) With the ADM the attack is detected at time $T = 10.1$ (hr), and the controller uses the signal $\hat{y}_4$ estimated by the Linear Model.

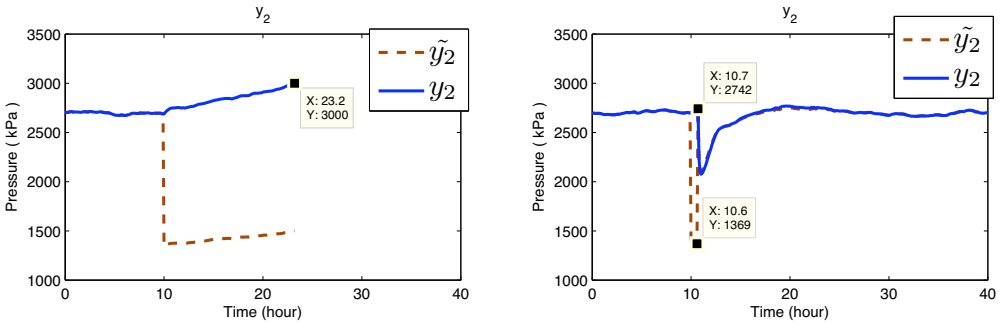
Figure 6.4: $\tilde{y_1} = y_1 * 0.5$

- Attacking y_2

  Sensor y_2 monitors pressure of the reactor. Attacking sensor y_2 by lowering the value makes controller turn down the purge valve to increase pressure. In an unprotected system,

the plant crashes at time $T = 23.5$ (hr) if we set parameter of scaling attack λ_m to 0.5.

With ADM enabled, the attack can be detected at time $T = 10.7$ (hr) and the plant remains stable.



(a) without ADM, plant crashes at $T = 23.5$ (hr) (b) ADM detects and responds to the attack at $T = 10.7$ (hr)

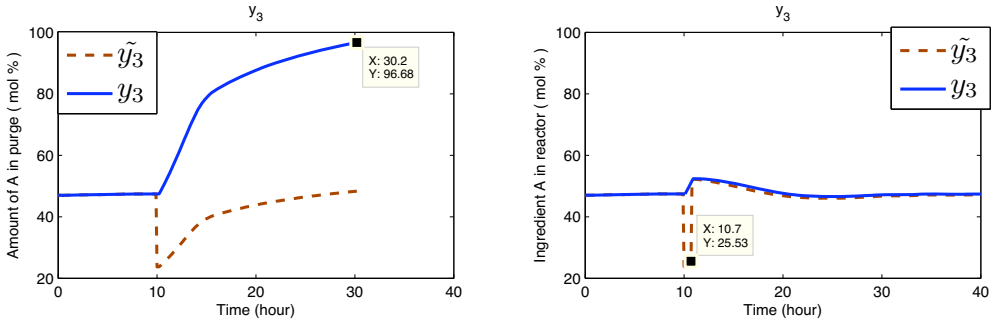
Figure 6.5: $\tilde{y_2} = y_2 * 0.5$

- Attacking y_3

  Sensor y_3 monitors amount of ingredient A in reactor. In an unprotected system, if we set λ_s to 0.5 and launch a scaling attack to the sensor y_3, then the controller detects a low quantity of material A and adds more A into the system. This results in the lack of material B and C. At time $T = 30.2$ (hr), the value of y_3 is 96.68 which means 96.68% of the reactor materials are ingredient A. The plant can't keep running anymore and shuts down.

  With ADM enabled, such an attack can be detected at time $T = 10.8$ (hr) and the plant system continues the reaction without influencing by compromised sensor signals.

Fig 6.7 shows the detection time costs when varying λ_m and launching scaling attacks to the sensors y_1, y_2 and y_3.



(a) without ADM, plant shuts down at $T = 30.2$ (hr) (b) ADM detects and responds to the attack at $T = 10.7$ (hr)

Figure 6.6: $\tilde{y_3} = y_3 * 0.5$

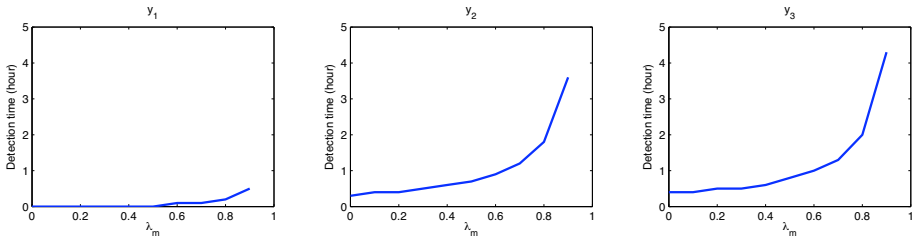


Figure 6.7: detection time v.s. scaling attack. Note that for $\lambda_m = 1$ there is no alarm.

6.3.2 Delay Attacks

Delay attacks delay sensor signals so that the controllers may slow down the reactions and fail to respond in real-time. In this experiment, we launch delay attacks on y_2 monitoring the pressure of reactor.

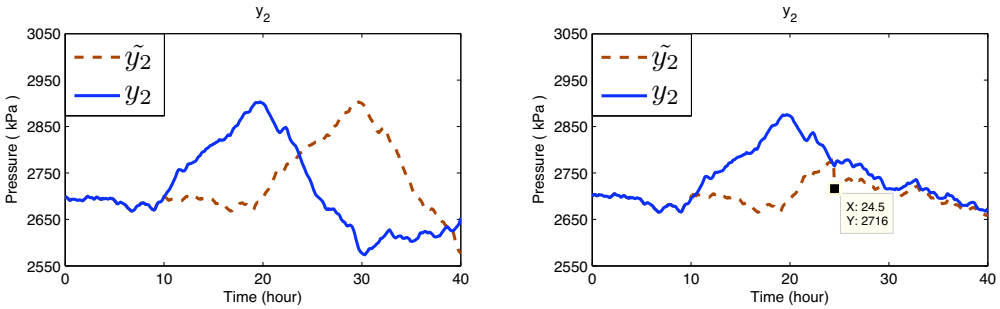
- Delaying y_2

  We delay signals of sensor y_2 so that the controller may ignore the variation of pressure. Fig 6.8 shows the result of the delay attack on y_2. Since the controller of y_2 cannot correctly react in time so the plant enters an unstable state. The experiment result shows that ADM can detect the delay attack at time $T = 24.5$ (hr) even no forged signal is generated in this attack.

  After several experiments with different λ_ds, we conclude that the damages resulted from

delay attacks are related to the delay time λ_d. Larger λ_d leads to higher probability of crashing plant systems, but it may be easier to be detected by ADM. In addition to the delay time, noise is another significant factor that may affect the probability of crashing plant system. Different orders of the same noise may lead to different degree of damage.

In this experiment, we run simulations for 100 times to observe the relationship between delay time, crash time and detection time (see Fig 6.9).



(a) without ADM, system becomes unstable and signals oscillate

(b) ADM detects attack at $T = 24.5$ (hr)

Figure 6.8: $\tilde{y_2}(t) = y_2(t - 10)$

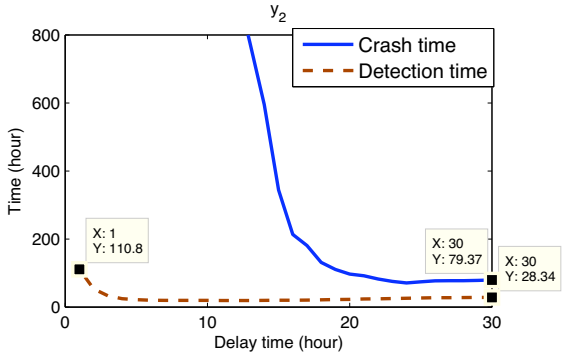


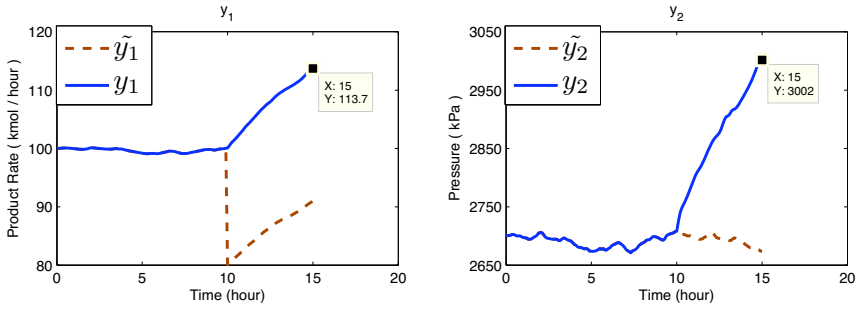
Figure 6.9: delay time v.s. crash time & detection time

From the results, we found that different delay times lead to different crash times and detection times. To eliminate the uncertainty of noise, we combine an scaling and delay attack and study the effect of compound attacks.

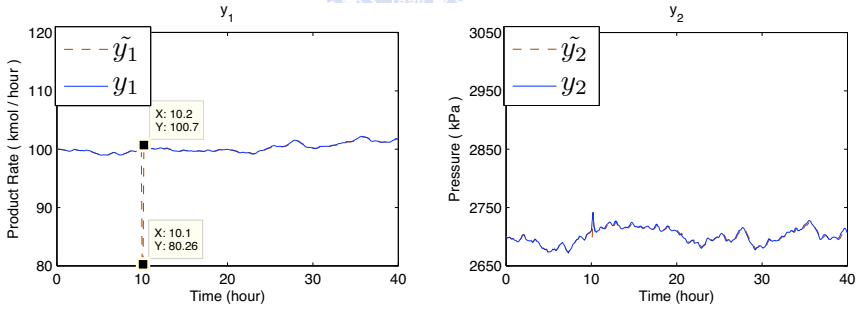
- Delaying y_2 & multiplying y_1

In this example, we launch a delay attack and scaling attack on y_2 and y_1 respectively. Fig 6.10 shows the experiment results, where the scaling attack deceives and delay attack slows down the controllers under attack. Without ADM, the pressure exceeds 3000 kPa and plant crashes at time $T = 15$ (hr). Scaling attacks can be detected at time $T = 10.2$ (hr) when ADM is enabled.

Compared with just launching an single attack, neither scaling attacking on y_1 (see Fig 6.4) nor delay attacking on y_2 can crash the plant system.



(a) without ADM, system crashes at $T = 15$ (hr)



(b) ADM detects attack at $T = 10.2$ (hr)

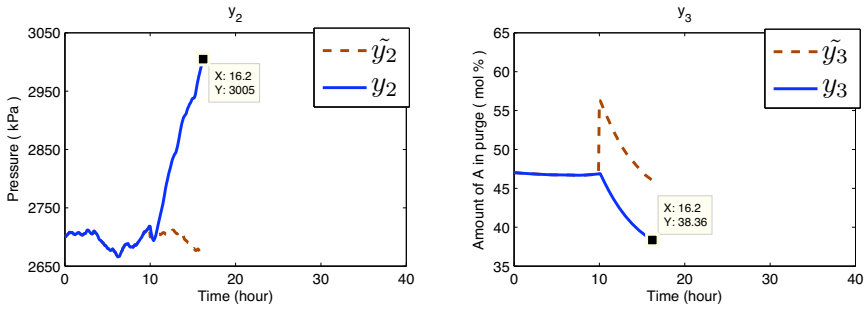
Figure 6.10: $\tilde{y_1}(t) = y_1 * 0.8$ & $\tilde{y_2}(t) = y_2(t - 10)$

- Delaying y_2 & multiplying y_3

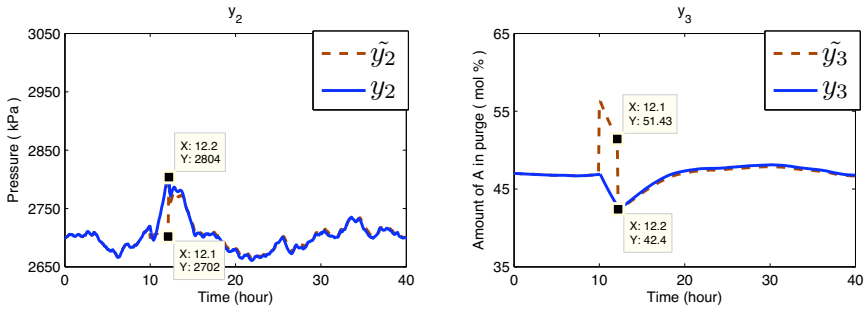
  Next, we attack sensor y_3 instead of y_1 to understand how the compound attack works. In the original plant system (without ADM), Fig 6.11 shows that plant goes to unsafe state at time $T = 16.2$ (hr). Compared with just launching an scaling attack on y_3 (see Fig 6.6), the compound attack takes much less time to crash the plant. The reason is that the delay

attack compromises sensor y_2, so that the controller of y_2 can not obtain real information to fix the warning condition of pressure in time.

Eventually, ADM detects the attack at time $T = 12.2$ (hr) and the plant goes back to control. In short, although the compound attack shortens the crash time, ADM still detects the attack in time.



(a) without ADM the pressure grows past safety levels at $T = 16.2$ (hr).



(b) ADM detects attack at $T = 12.2$ (hr)

Figure 6.11: $\tilde{y_2}(t) = y_2(t-10)$ & $\tilde{y_3} = y_3 * 1.2$

Chapter 7

Stealthy Attacks

We have shown that the ADM can detect a wide range of attacks. We now consider a more powerful adversary that knows about our detection scheme. We take a conservative approach in our models by assuming a very powerful attacker with knowledge of: (1) the exact linear model that we use (i.e., matrices A,B, and C), (2) the parameters of the ADM (τ and b), and (3) the control command signals. Such a powerful attacker may be unrealistic in some scenarios, but we want to test the resiliency of our system to such an attacker to guarantee safety for a wide range of attack scenarios.

The goal of the attacker is to raise the pressure in the tank without being detected (i.e., raise the pressure while keeping the statistic he controls below the corresponding threshold τ).

7.1 Model Stealthy Attacks

Stealthy attackers may attack the plant system stealthily and carefully not to trigger the alarm. To avoid triggering the alarm system, the attacker computes S_ks and makes sure that S_ks don't not reach the detection threshold τ. For this reason, the attacker sets two new thresholds $\lambda_{s_1}\tau$ and $\lambda_{s_2}\tau$ ($\lambda_{s_1} > \lambda_{s_2}$). The value of S_k rises up when the system is under attack, otherwise it falls down. To stealthily attack an system, the attacker pause the attack when S_k exceeds the first threshold ($S_k > \lambda_{s_1}\tau$). The attacker relaunches the attack after S_k falls under the second threshold ($S_k < \lambda_{s_2}\tau$). To pause and restart the attack, the attacker sets a flag F_p for indication

(1 for pause and 0 for restart).

There are a variety of implementations of stealthy attacks. We can realize an stealthy attack by launching additive, scaling, delay attacks or any combination of them. Thus, we model an stealthy attack ($\tilde{y}_i^s(t)$) as follows

$$\tilde{y}_i^s(t) = \begin{cases} \text{No attack:} & \\ y_i(k) & \text{for } k \notin \mathcal{K}_a \\ \text{Under attack:} & \\ y_i^+(k),\ F_p \leftarrow 1 & \text{for } S_k < \lambda_{s_2}\tau \\ y_i^+(k) & \text{for } \lambda_{s_2}\tau \leq S_k < \lambda_{s_1}\tau \text{ and } F_p = 0 \\ \text{Pause attack:} & \\ y_i(k) & \text{for } \lambda_{s_2}\tau \leq S_k < \lambda_{s_1}\tau \text{ and } F_p = 1 \\ y_i(k),\ F_p \leftarrow 0 & \text{for } S_k \geq \lambda_{s_1}\tau \end{cases}$$

In this model, $\tilde{y}_i^+$ is the combination of additive, scaling and delay attacks. It is represented as $\tilde{y}_i^+ \in \{\tilde{y}_i^a, \tilde{y}_i^m, \tilde{y}_i^d, ...\}$.

7.2 Experiments

Theoretically, it is difficult for ADM to detect the stealthy attacks since attackers stop attacking the plant system once S_k exceeds the first threshold $\lambda_{s_1}\tau$ ($\lambda_{s_1} < 1$). In this experiment, we implement stealthy attacks in three ways, by scaling, noises and combining scaling and delay attacks.

In the first example, we stealthily launch a scaling attack on y_2. We specify λ_m as 0.9, $\lambda_{s_1} = 0.9$, and $\lambda_{s_2} = 0.7$. From the experiment results shown in Fig 7.1, we observe that ADM does not detect any attack and the stealthy attack does not cause any injuries in the plant system neither. Before relaunching the next attack, the plant system goes back to an stable state.

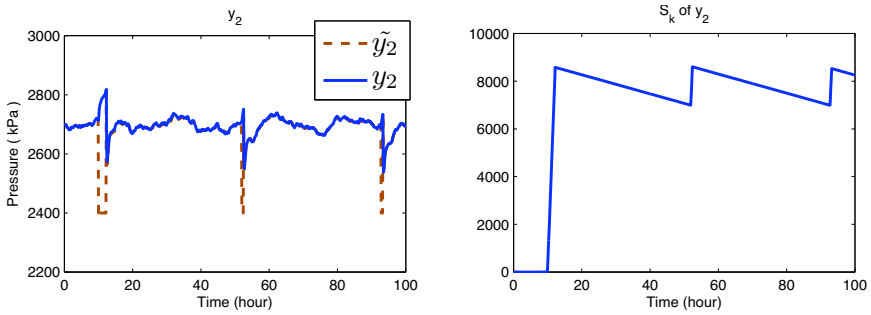


Figure 7.1: stealthy attack($\tilde{y_2} = y_2 * 0.9$)

The second example of a stealthy attack is implemented by an additive attack. Normal distribution noises are inserted into the sensor signals y_2. Fig 7.2 shows that the signals oscillate and the plant becomes unstable during the attack, but the plant system recovers once an attacker pauses the attack for avoiding the detection by ADM.

The last example implements the stealthy attack by launching a scaling attack and a delay attack on y_3 and y_2, respectively. In this example, we specify λ_m as 1.2, λ_d as 10 hours. Fig 7.3 shows the sensor signals of y_2, y_3 and their S_ks. The compound attack is launched at time $T = 10$ (hr) and paused after 2 hours. The results in Fig 6.11 show that the attack duration is too short to crash the plant.

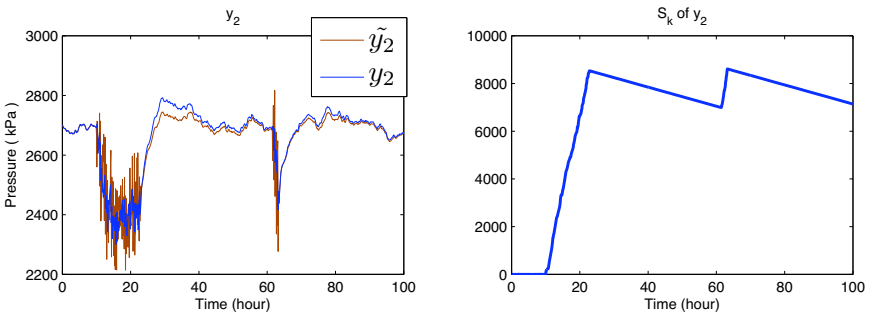
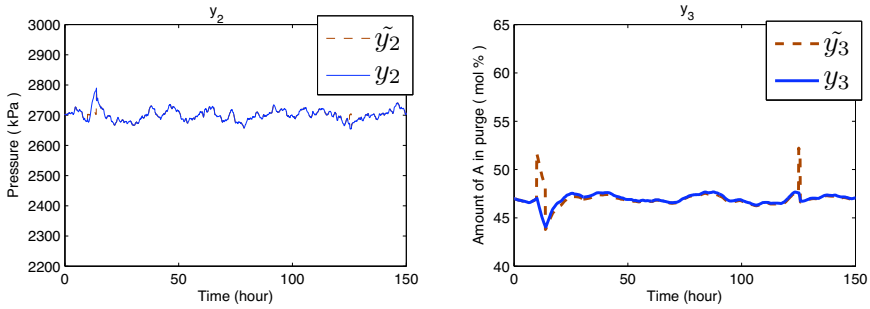
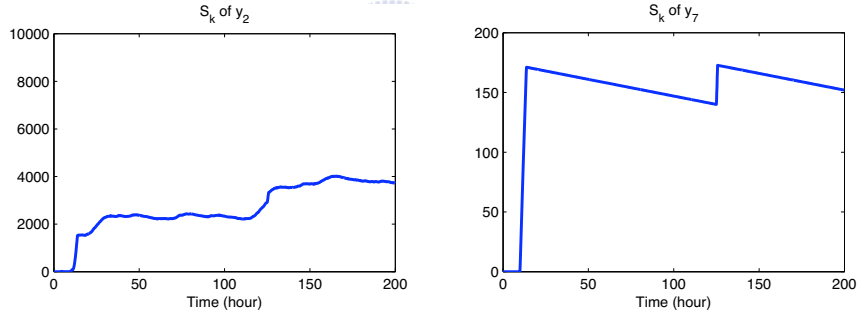


Figure 7.2: stealthy attack($\tilde{y_2} = y_2 + noise$)



(a) $\tilde{y_2}(t) = y_2(t-10)$ & $\tilde{y_3} = y_3 * 1.2$



(b) $S_{k_{y_2}}$ & $S_{k_{y_3}}$

Figure 7.3: stealthy attack

Defending against stealthy attacks is significant since stealthy attacks not only represent insider attacks but also the attacks want to evade ADM. Our approach shows that even though ADM fails to detect stealthy attacks, we can keep the the plant in safe conditions. Conclusively, the attacks want to evade ADM lose their threats to the plant. ADM ensures safety of plant no matter there is a attack or just the disturbance.

Chapter 8

Concluding Remarks and Future Work

In this work we conducted threat assessment of sensors of a well-studied process control system, TE-PCS, and introduced a methodology to design detection and response mechanisms for such control systems.

We showed that an attacker can compromise the safety of the system by attacking different sets of sensors. Since all attacks take several hours to cause a significant damage, we believe that building an anomaly detection module may offer a cost-effective security solution in comparison to investing in more costly tamper resilient devices.

By proposing an anomaly detection scheme we are able to detect and respond to attacks. Our anomaly detection scheme does not have any false alarm under the optimal configuration. Under a suboptimal configuration (designed to raise alarms), we show that a false alarm (which forces us to operate the plant under the representative model instead of the sensor readings) still maintains the safe operation. We also show how our scheme maintains the safe operation of the plant even against powerful stealthy attacker who has knowledge of the parameters of the anomaly detection module and who can compromise all sensors.

Our approach shows the utility of anomaly detection and automatic response by using a representative model of process dynamics of the system. Even though we have focused on the analysis of a chemical reactor system, our principles and techniques can be applied to many other physical processes. An automatic detection and response module will not be a reasonable solution for all control system processes; however, we believe that many processes posses

the characteristics similar to the TE-PCS, and so our approach would potentially offer a good solution for these processes.

Our model-based detection and response method can be extended to other control systems if the attack detection module is equipped with a representative model of the system. Since many control system operators have access to a reasonably large data set of sensor measurements and control inputs under nominal operating mode, a representative model can be constructed using system identification techniques. Moreover, as more data corresponding to operating modes under attacks (often with reduced functionality) become available, representative models for these operating modes can be created. A sophisticated response strategy would then be to *switch* between different representative models corresponding the operating mode identified by the ADM.

Finally, we would like to emphasize that the automatic response mechanism should be considered as a temporary solution before a human investigates the alarm. A full deployment of any automatic response mechanism should take into consideration the amount of time in which it is reasonable for a human operator to respond. In our chemical reactor plant the response time can be very large since the plant dynamics change slowly.

References

[1] J. DOWNS and E. VOGEL, ``A Plant-Wide Industrial Process Control Problem," *Computers & chemical engineering*, vol. 17, no. 3, pp. 245--255, 1993.

[2] S. J. Quin and T. A. Badgwell, ``A survey of industrial model predictive control technology," *Control Engineering Practice*, vol. 11, no. 7, pp. 733--764, July 2003.

[3] J. Rawlings, ``Tutorial overview of model predictive control," *Control Systems Magazine, IEEE*, vol. 20, no. 3, pp. 38--52, Jun 2000.

[4] US-CERT, *Control Systems Security Program*. http://www.us-cert.gov/control_systems/index.html: US Department of Homeland Security, 2008.

[5] GAO, ``Critical infrastructure protection. Multiple efforts to secure control systems are under way, but challenges remain," Report to Congressional Requesters, Tech. Rep. GAO-07-1036, September 2007.

[6] E. Goetz and S. Shenoi, *Critical Infrastructure Protection, Proceedings of the First Annual IFIP Working Group 11.10 International Conference on Critical Infrastructure Protection*. Dartmouth College, Hanover, New Hampshire, USA: Springer, March 2007.

[7] M. Papa and S. Shenoi, *Critical Infrastructure Protection II, Proceedings of the Second Annual IFIP Working Group 11.10 International Conference on Critical Infrastructure Protection*. Springer, March 2008.

[8] T. Kilpatrick, J. Gonzales, R. Chandia, M. Papa, and S. Shenoi, ``Forensic analysis of SCADA systems and networks," *International Journal of Security and Networks*, vol. 2, pp. 95--102, 2008.

[9] V. Igure, S. Laughter, and R. Williams, ``Security issues in SCADA networks," *Computers & Security*, vol. 25, no. 7, pp. 498--506, 2006.

[10] E. Byres and J. Lowe, ``The myths and facts behind cyber security risks for industrial control systems," in *VDE Congress*, 2004.

[11] P. Oman, E. Schweitzer, and J. Roberts, ``Safeguarding IEDs, Substations, and SCADA Systems Against Electronic Intrusions," in *Proceedings of the 2001 Western Power Delivery Automation Conference*, 2001, pp. 9--12.

[12] J. Falco, N. I. of Standards, and T. (US), *IT Security for Industrial Control Systems*. US Dept. of Commerce, Technology Administration, National Institute of Standards and Technology, 2002.

[13] E. Byres, ``Designing Secure Networks for Process Control," *IEEE Industry Applications Magazine*, vol. 6, pp. 33--39, 2000.

[14] A. Wright, J. Kinast, and J. McCarty, ``Low-Latency Cryptographic Protection for SCADA Communications," *LECTURE NOTES IN COMPUTER SCIENCE*, vol. 3089, pp. 263--277, 2004.

[15] P. Tsang and S. Smith, ``YASIR: A Low-Latency, High-Integrity Security Retrofit for Legacy SCADA Systems," *Proceedings of the IFIP TC 11 23rd International Information Security Conference*, 2008.

[16] N. Ricker, ``Model predictive control of a continuous, nonlinear, two-phase reactor," *JOURNAL OF PROCESS CONTROL*, vol. 3, pp. 109--109, 1993.

[17] D. Denning, ``An intrusion-detection model," *Software Engineering, IEEE Transactions on*, vol. SE-13, no. 2, pp. 222--232, Feb. 1987.

[18] T. Kailath and H. V. Poor, ``Detection of stochastic processes," *IEEE Transactions on Information Theory*, vol. 44, no. 6, pp. 2230--2258, October 1998.

[19] A. Wald, *Sequential Analysis*. J. Wiley & Sons, New York, 1947.

[20] J. Jung, V. Paxson, A. Berger, and H. Balakrishan, ``Fast portscan detection using sequential hypothesis testing," in *Proceedings of the 2004 IEEE Symposium on Security and Privacy*, May 2004, pp. 211--225.

[21] S. Schechter and J. J. A. Berger, ``Fast detection of scanning worm infections," in *Proc. of the Seventh International Symposium on Recent Advances in Intrusion Detection (RAID)*, September 2004.

[22] M. Xie, H. Yin, and H. Wang, ``An effective defense against email spam laundering," in *Proceedings of the 13th ACM Conference on Computer and Communications Security*, October 30--November 3 2006, pp. 179--190.

[23] G. Gu, J. Zhang, and W. Lee, ``Botsniffer: Detecting botnet command and control channels in network traffic," in *Proceedings of the 15th Annual Network and Distributed System Security Symposium (NDSS'08)*, San Diego, CA, February 2008.

[24] B. Brodsky and B. Darkhovsky, *Non-Parametric Methods in Change-Point Problems*. Kluwer Academic Publishers, 1993.