Design and Analysis of Intrusion Detection Protocols in Cyber Physical Systems

Robert R. Mitchell III

Dissertation submitted to the Faculty of the Virginia Polytechnic Institute and State University in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Computer Science and Applications

Ing-Ray Chen, Chair
Mohamed Y. Eltoweissy
Wenjing Lou
Chang-Tien Lu
Scott F. Midkiff

April 5, 2013
Falls Church, Virginia

Keywords: Intrusion Detection Systems, Cyber Physical Systems
Copyright 2013, Robert R. Mitchell III
Design and Analysis of Intrusion Detection Protocols in Cyber Physical Systems

Robert R. Mitchell III

ABSTRACT

In this dissertation research we aim to design and validate intrusion detection system (IDS) protocols for a cyber physical system (CPS) comprising sensors, actuators, control units, and physical objects for controlling and protecting physical infrastructures.

The design part includes host IDS, system IDS and IDS response designs. The validation part includes a novel model-based analysis methodology with simulation validation. Our objective is to maximize the CPS reliability or lifetime in the presence of malicious nodes performing attacks which can cause security failures. Our host IDS design results in a lightweight, accurate, autonomous and adaptive protocol that runs on every node in the CPS to detect misbehavior of neighbor nodes based on state-based behavior specifications. Our system IDS design results in a robust and resilient protocol that can cope with malicious, erroneous, partly trusted, uncertain and incomplete information in a CPS. Our IDS response design results in a highly adaptive and dynamic control protocol that can adjust detection strength in response to environment changes in attacker strength and behavior. The end result is an energy-aware and adaptive IDS that can maximize the CPS lifetime in the presence of malicious attacks, as well as malicious, erroneous, partly trusted, uncertain and incomplete information.

We develop a probability model based on stochastic Petri nets to describe the behavior of a CPS incorporating our proposed intrusion detection and response designs, subject to attacks by malicious nodes exhibiting a range of attacker behaviors, including reckless, random, insidious and opportunistic attacker models. We identify optimal intrusion detection settings under which the CPS reliability or lifetime is maximized for each attacker model. Adaptive control for maximizing IDS performance is achieved by dynamically adjusting detection and response strength in response to attacker strength and behavior detected at runtime. We conduct extensive analysis of our designs with four case studies, namely, a mobile group CPS, a medical CPS, a smart grid CPS and an unmanned aircraft CPS. The results show that our adaptive intrusion and response designs operating at optimizing conditions significantly outperform existing anomaly-based IDS techniques for CPSs.
Acknowledgments

I would like to thank Ing-Ray Chen for his tireless dedication, patience and wisdom in transforming me into a scientist. I would like to thank Mohamed Eltoweissy for his many tips on exciting work in related fields. I would like to thank Wenjing Lou for sharing her cyber security expertise. I would like to thank Chang-Tien Lu for contributing his data mining insights. I would like to thank Scott Midkiff for providing his domain expertise in Cyber Physical Systems.

I would like to thank Alexis, Edward and Ellis Mitchell for understanding my passion for this calling and sharing their husband, brother and father with the profession. I would like to thank Bob and Susan Mitchell for raising me to have the hunger to start this journey and the tenacity to stay with it.
## Contents

1 Introduction 1
  1.1 Cyber Physical Systems 1
  1.2 Research Statement 3
  1.3 Research Contributions 3
  1.4 Dissertation Organization 4

2 Literature Search 5
  2.1 Intrusion Detection and Response Functions and Metrics 5
    2.1.1 Core Intrusion Detection Functions 5
    2.1.2 Intrusion Detection Performance Metrics 6
  2.2 Classification 8
    2.2.1 Detection Technique 9
    2.2.2 Collection Process 14
    2.2.3 Trust Model 15
    2.2.4 Response Technique 18
  2.3 Classifying Existing CPS Intrusion Detection Techniques 19
  2.4 Revisiting IDS Techniques and Gaps in CPS IDS Research 19
  2.5 Challenges of CPS Intrusion Detection and Response 21

3 System Model 24
  3.1 CPS Abstraction Model 24
  3.2 System Failure Model 28
3.3 Monitoring, Noise and Mis-monitoring Probability ........................................ 29
3.4 Attack Model ........................................................................................................ 29
3.5 Adversary Behavior Model .................................................................................. 33
  3.5.1 Capture Strength ................................................................................................. 33
  3.5.2 Insider Attack Behavior ..................................................................................... 34
3.6 Stuxnet Attacks ..................................................................................................... 35
3.7 Capture/Insider CPS Attacks in the Literature .................................................... 36

4 Intrusion Detection and Response Design ............................................................. 39
  4.1 Host Intrusion Detection Design ........................................................................... 40
    4.1.1 Compliance Degree .......................................................................................... 40
    4.1.2 Vector Similarity Specification-Based Protocol Designs ................................. 42
    4.1.3 Behavior Rule Based Protocol Design ............................................................. 46
    4.1.4 Testing Methodology for Compliance Degree Distribution ............................ 47
    4.1.5 Threshold-based Host \( p_{fn} \) and \( p_{fp} \) Assessment ................................. 49
  4.2 System Intrusion Detection Design ....................................................................... 49
  4.3 Intrusion Response Design ................................................................................... 52

5 Case Study: Mobile Group CPS ................................................................................ 54
  5.1 MGCPS Reference Model .................................................................................... 54
  5.2 MGCPS Intrusion Detection Design .................................................................... 56
  5.3 Model and Analysis .............................................................................................. 56
  5.4 Parameterization .................................................................................................. 62
    5.4.1 System-Level IDS \( P_{fn} \) and \( P_{fp} \) ................................................................. 62
    5.4.2 Host IDS \( p_{fn} \) and \( p_{fp} \) .............................................................................. 64
    5.4.3 Parameterizing \( C_T \) for Dynamic Intrusion Response ................................. 65
    5.4.4 Energy ........................................................................................................... 67
  5.5 Numerical Data ..................................................................................................... 67
    5.5.1 Effect of Intrusion Detection Strength .......................................................... 68
5.5.2 Effect of Attacker Behavior .......................................... 69
5.5.3 Effect of Capture Strength ............................................. 71
5.5.4 Effect of Intrusion Response .......................................... 72
5.6 Generalization .............................................................. 74
  5.6.1 Stuxnet Attacks ......................................................... 74
  5.6.2 Mixed Attackers ......................................................... 74
5.7 Simulation ................................................................. 75
5.8 Summary ................................................................. 79

6 Case Study: Medical CPS .................................................. 80
  6.1 Background ............................................................ 80
  6.2 MCPS Reference Model .................................................. 83
  6.3 MCPS Intrusion Detection Design ...................................... 84
    6.3.1 Behavior Rules ...................................................... 84
    6.3.2 Transforming Rules to State Machines ............................ 84
    6.3.3 Collect Compliance Degree Data .................................. 91
  6.4 Simulation .......................................................... 94
  6.5 Comparison of ROC under Binary and Distance-Based Grading Policies ... 99
  6.6 Comparative Analysis ............................................... 100
  6.7 Summary .......................................................... 103

7 Case Study: Smart Grid CPS .............................................. 104
  7.1 Background .......................................................... 104
  7.2 SGCPS Reference Model ............................................. 106
  7.3 SGCPS Intrusion Detection Design .................................. 106
    7.3.1 Behavior Rules .................................................. 106
    7.3.2 Transforming Rules to State Machines ......................... 108
    7.3.3 Collect Compliance Degree Data ................................ 117
  7.4 Simulation .......................................................... 117
## List of Figures

1.1 Typical CPS. .......................................................... 1

2.1 Spectrum of IDS Responses. ........................................ 6
2.2 Classification Tree for CPS Intrusion Detection Techniques. .... 8
2.3 Detection Technique Dimension of Intrusion Detection. .......... 9
2.4 Collection Process Dimension of Intrusion Detection. .......... 14
2.5 Trust Model Dimension of Intrusion Detection. ................. 16
2.6 Response Dimension of Intrusion Detection. ..................... 18

3.1 Hierarchically Structured CPS Abstraction Model. ............... 25

4.1 Vector Similarity: \( \langle i, i \rangle \) are the Safe States. .......... 43
4.2 Positional Discontinuity: \( \langle 0, 0 \rangle \) is the Safe State. .......... 44
4.3 Positional Discontinuity Instantaneous Distances. ............... 45
4.4 Positional Discontinuity Time Series Distances. ................. 46
4.5 Actuator Model. ..................................................... 47
4.6 Combined Intrusion Detection Flowchart. ......................... 52

5.1 Reference MGCPS. ................................................... 55
5.2 SPN Model for Intrusion Detection and Response. ............... 57
5.3 The Underlying Semi-Markov Model of the SPN Model. .......... 60
5.4 MTTF vs. \( T_{IDS} \) and \( m \). ...................................... 68
5.5 MTTF vs. \( T_{IDS} \) and \( \lambda_c \). ...................................... 69
5.6 MTTF vs. \( T_{IDS} \) and \( p_{random} \). ................................. 70
5.7 MTTF vs. $T_{IDS}$ and Adversary Type. ........................................... 70
5.8 MTTF vs. $T_{IDS}$ under Reckless Attacker. ........................................... 71
5.9 Effect of Intrusion Response toward Capture Attack Strength under Reckless
Attack Behavior. ........................................................................... 73
5.10 MTTF vs. $m$ and $T_{IDS}$ for Stuxnet Attacker. ............................... 74
5.11 SPN for Modeling Mixed Attackers. ................................................. 75
5.12 Linear Response under Homogeneous Attackers. ............................. 76
5.13 Linear Response under Mixed Attackers. ........................................... 76
5.14 Simulation and Theoretical MTTF vs. $T_{IDS}$ and $m$. ....................... 77
5.15 Simulation and Theoretical MTTF vs. $T_{IDS}$ and $\lambda_c$. .......... 78
5.16 Simulation and Theoretical MTTF vs. $T_{IDS}$ and $p_{\text{random}}$. .... 78

6.1 Medical Physical Components in the Reference MCPS. ..................... 83
6.2 Good VSM Behavior Rule State Machine. ......................................... 90
6.3 Random Attacker VSM Behavior Rule State Machine. ....................... 90
6.4 Sensitivity of Good Node Compliance Degree to $p_{\text{err}}$. ............... 95
6.5 Sensitivity of Bad Node Compliance Degree to $p_{\text{err}}$. ................. 95
6.6 Sensitivity of Bad Node Compliance Degree to $p_a$. ....................... 95
6.7 Probability of False Negative vs. Compliance Threshold for Detecting Random
Attackers. ................................................................................. 97
6.8 Probability of False Negative vs. Compliance Threshold and $p_{\text{err}}$ for Detecting
Reckless Attackers. .......................................................................... 97
6.9 Probability of False Positive vs. Compliance Threshold and $p_{\text{err}}$ for Detecting
Good Nodes. .................................................................................. 98
6.10 Sensitivity of ROC to $p_{\text{err}}$ for Detecting Reckless Attackers. ........ 98
6.11 ROC Graph under Binary Grading for Detecting Random Attackers. ... 99
6.12 ROC Graph under Each Distance-Based Grading Strategy for Detecting Random
Attackers. .................................................................................. 99
6.13 ROC Graph under Distance-Based Grading for Detecting Reckless Attackers. 101
6.14 ROC Graph under Distance-Based Grading for Detecting Random Attackers. 101
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.10</td>
<td>Probability of False Positive Versus Compliance Threshold for a Good Node under Varying $p_{err}$</td>
<td>139</td>
</tr>
<tr>
<td>8.11</td>
<td>UAV Receiver Operating Characteristic Graph</td>
<td>140</td>
</tr>
<tr>
<td>8.12</td>
<td>ACCM Receiver Operating Characteristic Graph</td>
<td>141</td>
</tr>
<tr>
<td>8.13</td>
<td>UIDS and ACCM ROC Graph Comparison</td>
<td>142</td>
</tr>
<tr>
<td>8.14</td>
<td>Detail for AUC Inspection</td>
<td>143</td>
</tr>
</tbody>
</table>
## List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Classification of CPS IDSs</td>
<td>20</td>
</tr>
<tr>
<td>2.2</td>
<td>Commonality and Variability of IDS Techniques for CPSs (white: no research, grey: little research, black: significant research, ×: deserving more research)</td>
<td>20</td>
</tr>
<tr>
<td>3.1</td>
<td>Capture/Insider Attack Models of CPS Attacks in the Literature</td>
<td>36</td>
</tr>
<tr>
<td>5.1</td>
<td>Parameters Used for Analysis of Intrusion Detection and Response Design</td>
<td>57</td>
</tr>
<tr>
<td>5.2</td>
<td>Transition Rates of the SPN Model</td>
<td>58</td>
</tr>
<tr>
<td>5.3</td>
<td>Parameters and their Values for the Reference CPS</td>
<td>62</td>
</tr>
<tr>
<td>5.4</td>
<td>$\beta$ in Beta(1,\beta) and Resulting $p_{fn}$ and $p_{fp}$ Values under Various Attack Models for VSM ($C_T = 0.90, p_{err} = 0.01$)</td>
<td>65</td>
</tr>
<tr>
<td>6.1</td>
<td>MCPS Behavior Rules</td>
<td>87</td>
</tr>
<tr>
<td>6.2</td>
<td>Attack States in Conjunctive Normal Form</td>
<td>87</td>
</tr>
<tr>
<td>6.3</td>
<td>MCPS State Components</td>
<td>88</td>
</tr>
<tr>
<td>6.4</td>
<td>$\beta$ in Beta(1,\beta) and Resulting $p_{fn}$ and $p_{fp}$ Values under Various Attack Models for VSM ($C_T = 0.90, p_{err} = 0.01$)</td>
<td>96</td>
</tr>
<tr>
<td>7.1</td>
<td>HE Behavior Rules</td>
<td>110</td>
</tr>
<tr>
<td>7.2</td>
<td>DAP Behavior Rules</td>
<td>111</td>
</tr>
<tr>
<td>7.3</td>
<td>SM Behavior Rules</td>
<td>111</td>
</tr>
<tr>
<td>7.4</td>
<td>HE Attack States in Conjunctive Normal Form</td>
<td>112</td>
</tr>
<tr>
<td>7.5</td>
<td>DAP Attack States in Conjunctive Normal Form</td>
<td>112</td>
</tr>
<tr>
<td>7.6</td>
<td>SM Attack States in Conjunctive Normal Form</td>
<td>113</td>
</tr>
<tr>
<td>7.7</td>
<td>SGCPS State Components</td>
<td>114</td>
</tr>
</tbody>
</table>
Chapter 7
β in Beta(1,β) and Resulting $p_{fn}$ and $p_{fp}$ Values under Various Attack Models
for HE ($C_T = 0.90, p_{err} = 0.01$).

7.9 β in Beta(1,β) and Resulting $p_{fn}$ and $p_{fp}$ Values under Various Attack Models
for DAP ($C_T = 0.90, p_{err} = 0.015$).

7.10 β in Beta(1,β) and Resulting $p_{fn}$ and $p_{fp}$ Values under Various Attack Models
for SM ($C_T = 0.90, p_{err} = 0.02$).

7.11 Comparison Results for HE.

7.12 Comparison Results for DAP.

7.13 Comparison Results for SM.

Chapter 8
UAV Behavior Rules.

8.2 UAV Attack States in Conjunctive Normal Form.

8.3 UACPS State Components.

8.4 β in Beta(1,β) and Resulting $p_{fn}$ and $p_{fp}$ Values under Various Random Attack
Models for UAV ($C_T = 0.90, p_{err} = 0.01$).

8.5 β in Beta(1,β) and Resulting $p_{fn}$ and $p_{fp}$ Values under Various Opportunistic
Attack Models for UAV ($C_T = 0.90, p_{err} = 0.01, C = 10$).

8.6 $C_T$ to Satisfy $p_{fn}$ (1%) while Maximizing $p_{fp}$ Given $p_{err}$ and Attacker Type as
Input ($p_a = 0.2, \varepsilon = 0.8$).
Chapter 1

Introduction

1.1 Cyber Physical Systems

Figure 1.1: Typical CPS.

Cyber Physical Systems (CPSs) are large scale, geographically dispersed, federated, heterogeneous, life-critical systems that comprise sensors, actuators and control and networking components. Securing CPSs has emerged as a critical interest of all governments. The literature also refers to a CPS as a Distributed Control System (DCS), Networked Control System (NCS), Sensor Actuator Network (SAN), Supervisory Control And Data Acquisition
Chapter 1. Introduction

(SCADA) system or Wireless Industrial Sensor Network (WISN) [127]. Their functions in common are sensing (acquisition) and actuation (control). These systems have wireless segments and are heterogeneous and geographically dispersed. These systems may be federated, mobile, attended or completely inaccessible. **Enclaves** define the edges of the segments of the federated system. Nodes that contain the sensors and actuators are called Remote Terminal Units (RTUs), Intelligent Electronic Devices (IEDs) or Programmable Logic Controllers (PLCs). RTUs may implement some limited tactical control functions. Data Acquisition Systems (DASs) aggregate readings from RTUs and adapt (bridge or tunnel) the local RTU protocol (such as CAN [73], DNP3 [43] or Modbus [1, 2]) with the long-haul protocol shared with the control center (such as TCP). Data processing servers effect the business logic of the CPS; these may be high performance computing clouds that process large datasets produced by economical nodes. Historian servers collect, store and distribute data from sensors [68]. Nodes that contain control logic and provide management services to a Human Machine Interface (HMI) are called Master Terminal Units (MTUs); in contrast with the RTUs, an MTU implements the broad strategic control functions. Figure 1.1 illustrates a typical CPS using these components.

CPSs share several properties: These systems use embedded computers and networks to monitor and control physical processes with feedback to integrate computation with the environment. They consist of a set of networked sensors, actuators, control processing units and communication devices. CPSs are application-specific (purpose-built). Some segments may be resource-constrained: A Wireless Sensor Network (WSN) or Wireless Sensor and Actor Network (WSAN) may form part of a CPS.

Common CPS issues are: availability, reconfigurability, distributed control (distributed management), real-time operation (timeliness), fault-tolerance, scalability, autonomy, reliability, security, heterogeneity, federation and geographic dispersion [53]. Timeliness is critical in CPSs because the environment can change quickly; control loops fail if their period is longer than expected. Automatic control techniques can address CPS reliability [152]. However, security requires distinct measures from reliability. Moreover, compromised nodes may collude to deter or disrupt the CPS functionality. An effective yet energy efficient intrusion detection system (IDS) is of great interest to detect and evict compromised nodes from a CPS whose failure can cause dire consequences. NSF characterizes CPSs as time-critical, positionally precise, energy efficient systems deployed in hostile environments (due to hazardous materials or combatants, for example) that coordinate large scale activities (like warfighting), enhance human capabilities (with sensors or navigation, for example) and improve social welfare (via extended medical care or assisted living, for instance) [53]. This dissertation research concerns design, analysis and validation of intrusion detection systems for CPSs. Our notion of IDS design includes the design of intrusion detection and response.

Federation results in one particular challenge to design of an IDS for CPSs: Multiple control loops may manipulate the same actuators. This is a delicate arrangement that any IDS, the response component in particular, should take care not to upset. Federation also presents a unique class of adversary: Honest but curious users will avoid damaging the CPS but will seek
to gain advantage for their organization (enclave). The general term for this is selfishness; this research addresses selfish as well as malicious adversaries. Geographic dispersal results in another particular challenge: Auditing partial datasets is a requirement because it is difficult to guarantee robust communications across a large area. This implies that decision making in general and intrusion detection and response in particular may have to be made despite malicious, erroneous, partly trusted, uncertain and incomplete information.

1.2 Research Statement

This dissertation research aims to address research challenges of intrusion detection for CPSs, as identified in our literature survey in Chapter 2. Our goal is to design and validate intrusion detection and response protocols for CPSs. The design part includes host IDS, system IDS and IDS response. The validation part includes a novel model-based analysis methodology with simulation validation. Our objective is to maximize the CPS reliability or lifetime in the presence of malicious nodes performing attacks which can cause security failures.

The goal of host IDS design is to result in a lightweight, accurate, autonomous and adaptive protocol that runs on every node in the CPS to detect misbehavior of neighbor nodes based on state-based behavior specifications. The goal of system IDS design is to result in a robust and resilient protocol that can cope with malicious, erroneous, partly trusted, uncertain and incomplete information in a CPS. Finally, the goal of IDS response design is to result in a highly adaptive and dynamic control protocol that can adjust detection strength in response to environment condition changes, especially in changes in attacker strength and behavior. The end result is an energy-aware and adaptive IDS that can maximize the lifetime of the system in the presence of malicious attacks, as well as malicious, erroneous, partly trusted, uncertain and incomplete information.

1.3 Research Contributions

We envision the following contributions from this dissertation research:

1. **Host IDS Protocol Design:** We propose the design concept of compliance degree based on behavior specifications for host IDS design. We analyze two lightweight host IDS designs, namely, vector similarity specification-based design and behavior rule specification-based design. Extending from software testing, we propose a novel testing methodology to characterize the compliance degree of a node based on the history of compliance degree collected during the system’s testing and debugging stage. We also propose a novel model-based analysis methodology by which we determine the false negative probability $p_m$ (misidentifying a bad node as a good node) and false positive
Chapter 1. Introduction

probability $p_{fp}$ (misidentifying a good node as a bad node), as a result of applying a host IDS design for protocol accuracy assessment.

2. **System IDS Protocol Design:** We propose and analyze a system IDS design based on majority voting making use of host $p_{fn}$ and $p_{fp}$ to cope with malicious, erroneous, partly trusted, uncertain and incomplete information in a CPS. The novelty lies in the way to predict the resulting system-level false negative probability $P_{fn}$ and false positive probability $P_{fp}$ depending on the attacker strength and behavior detected dynamically.

3. **Response Protocol Design:** Untreated in the literature, we propose and analyze intrusion response designs that result in a highly adaptive and dynamic control protocol that can adjust detection strength in response to environment changes such as changes in attacker strength and behavior.

4. **Model Based Analysis with Simulation Validation:** We propose a probability model to describe the CPS behavior in response to various models including reckless, random, insidious and opportunistic insider attack behavior models. The novelty lies in the discovery of optimal IDS design settings that will maximize the CPS lifetime. In particular, with the mathematical formulas and SPN models, one could identify optimal IDS settings at static time given an attacker behavior model and dynamically apply optimal IDS settings in response to changes in the attacker strength and behavior at runtime to maximize the system lifetime. We validate the optimal conditions identified for adaptive IDS control with extensive simulation.

1.4 **Dissertation Organization**

The rest of this dissertation research is organized as follows: Chapter 2 summarizes a literature search for CPS intrusion detection systems. Chapter 3 states the CPS model and assumptions. In Chapter 4, we discuss intrusion detection and response designs and rationale. Specifically: the general design principles of host IDS and system IDS, the tradeoff between IDS and energy conservation and the adaptive control of detection intelligence and strength against intrusion intelligence and strength to prolong the system lifetime. In the next four chapters, we apply these design principles to real systems. Preliminary analysis results are reported with physical interpretations given. Chapter 9 presents conclusions and describes future research areas.
Chapter 2

Literature Search

In this chapter, we provide a comprehensive survey of CPS intrusion detection. We also propose a classification tree to organize existing CPS intrusion detection schemes in the hope of identifying gaps in CPS IDS research and shedding light on research directions. We conclude the chapter with CPS IDS research challenges identified and our proposals to answer these challenges. Part of the content of this chapter is from our Elsevier Computer Networks [105] and ACM Computing Surveys [100] submissions.

2.1 Intrusion Detection and Response Functions and Metrics

2.1.1 Core Intrusion Detection Functions

An IDS implements three core functions:

- collecting data regarding suspects;
- analyzing the data;
- responding to the analysis.

Examples of collection are: logging system calls on the local node, recording traffic received on a network interface and saving hearsay reputation scores (multitrust data or recommendations). Examples of analysis are: pattern matching, statistical analysis and data mining. Examples of responses are: spreading an alarm, starting integrity checks, updating routing tables, closing a session, rekeying and launching a counterattack [132]. In our approach, we adjust IDS parameters such as audit interval, intrusion detector quantity and compliance threshold in response to attacker strength detected.
Chapter 2. Literature Search

While the literature is abundant in the first two core functions (collection and analysis), the third function (response) is less treated. Tao and Ruighaver [132] point out it is especially tricky to respond effectively to intrusions. Figure 2.1 illustrates a spectrum of IDS responses. Techniques are organized horizontally based on their impact or disruptive potential: from low to high. Techniques are organized vertically based on their correlation to the adversary’s belligerence: from greedy to malicious. It is tempting to order malicious attacks further, say by indicating that passive attacks such as eavesdropping are less damaging than active attacks such as Denial of Service (DoS). However, it is easy to imagine a scenario where an eavesdropping attack that continues for a long time causes a large amount of private data to leak. While a system is available following a DoS attack, data cannot be unloaded, and privacy cannot be restored once lost. Finally, the system experiences security failure due to functional impairments. Wireless network nodes are much more transient than their wireline counterparts; adversaries come and go quickly. This escalates the importance of the response function (specifically its effectiveness and timeliness) in wireless networks. At the same time an IDS should respond effectively and quickly to an intrusion, it should not disrupt legitimate nodes.

2.1.2 Intrusion Detection Performance Metrics

IDS researchers traditionally use three metrics to measure performance: false positive rate, false negative rate and detection rate [47, 61, 62, 119, 131, 132, 151]. A false positive occurs when an IDS identifies a well-behaved node as an intruder; the literature also refers to this as a false alarm. False positive rate is calculated by false alarms/total normal actions. A
false negative occurs when an IDS identifies a malicious or selfish node as well-behaved; the literature also refers to this as a failure to report \[62, 75, 131\]. On the other hand, a detection (a true positive) occurs when an IDS correctly identifies a malicious or selfish node \[47, 61, 92, 94, 119, 127, 151\]. Detection rate is calculated by intrusions detected/total intrusions. Because there is often a trade-off between detection rate versus false positive probability, a particularly useful technique is to make use of a receiver operating characteristic (ROC) \[59\] curve showing the sensitivity of detection rate versus false positive probability. In our dissertation research we also investigate the best way to trade a high false positive probability off for a low false negative probability when we detect high attacker strength such that bad nodes (including malicious and selfish nodes in general) can be quickly detected (because of a low false negative probability) to avoid impairment-induced security failures.

Some research attempts to establish effective new metrics in order to enrich IDS research. Detection latency is a rarely used but critical means to measure IDS performance \[51, 79\]. Regardless of the attack model (passive or active), an early detection enables an early response. For target systems with resource limitations, power consumption, communications overhead and processor load are important metrics as well. \[92\] measures the time for an arbitrary number of nodes to exhaust their energy when using a given technique. \[94\] measures packet sampling efficiency: the percentage of analyzed packets the IDS identifies as malicious. \[47\] measures time required to train their IDS and time required to analyze test data. In addition to false alarm probability, this dissertation research also considers energy exhaustion failure. In particular, we study the tradeoff between energy consumption versus detection strength. That is, as detection strength increases, the probability of security failure decreases because of a more powerful IDS to detect and evict bad nodes. However, this increases energy consumption and therefore increases the probability of system failure due to energy exhaustion. In our dissertation research one goal is to identify the best intrusion detection settings that best balance energy consumption versus intrusion detection strength so that the system lifetime is maximized.

On the other hand, some studies attempt to establish metrics that are conceptually sound, but have weaknesses in practice. \[152\] advocates using fault-tolerance and adaptability but does not explain how to measure these properties. \[51\] measures survivability as the ability of a system to serve customers and resist security violations; specifically, it is calculated as \(1000 - \Sigma\) failed transactions - \(\Sigma\) security violations. However, the arbitrary nature of this formula is a weakness. \[14\] and \[126\] use application-specific trust or reputation in their analysis. Furthermore, \[126\] supplements this trust metric with quantity of invalid content distributed and fairness of load balance. In general, it is difficult to apply these narrowly focused metrics.

Finally, some studies attempt to establish metrics yet proven useful. \[85\] borrows Equal Error Rate (EER) from the field of biometrics to measure performance; this is the rate at which false negatives (reject error) and false positives (accept error) are equal \[46\]. While this interdisciplinary approach is novel, minimizing false negatives and false negatives is commonly believed to be more important than equalizing their rates. \[60\] tests if a given
IDS technique can detect a number of specific attacks. While this statistic is simple and elegant, it lacks generality and applicability.

### 2.2 Classification

We perform a classification of existing IDS techniques for CPSs to understand the current state of the art in CPS IDS research. Figure 2.2 shows a classification tree for classifying existing IDS techniques in CPSs. We classify the intrusion detection literature based on four criteria (or dimensions):

1. Detection Technique: this criterion distinguishes IDSs based on their basic approach to detection analysis;

2. Collection Process: this criterion contrasts behavior-based IDSs from traffic-based IDSs;

3. Trust Model: this criterion differentiates IDSs that share raw data or analysis results from standalone IDSs;

4. Response Technique: this criterion contrasts active from passive and static from dynamic approaches to repelling an attack.

We use the classification tree to organize existing intrusion detection techniques in the literature in the hope of identifying gaps in IDS research and shedding light on research directions. Below we discuss each classification dimension in detail.
2.2.1 Detection Technique

In this section, we first describe existing IDS detection techniques including signature-based, anomaly-based, specification-based and reputation-based techniques. Then we discuss the effectiveness of IDS detection techniques applying to CPSs. Figure 2.3 shows the Detection Technique dimension.

2.2.1.1 Signature-Based Intrusion Detection

Signature-based intrusion detection approaches look for runtime features that match a specific pattern of misbehavior. Some sources refer to this approach as misuse detection [51, 60, 62, 156], supervised detection [161], pattern-based detection [47] or intruder profiling [150].

One major advantage of this category is a low false positive rate. By definition, these approaches only react to known bad behavior; the theoretical basis is a good node will not exhibit the attack signature. The key disadvantage of this category is that the techniques must look for a specific pattern; a dictionary must specify each attack vector and stay current. An attack signature can be a univariate data sequence: for example, bytes transmitted on a network, a program’s system call history or application-specific information flows (sensor measurements in a CPS). One sophistication is to combine simple data sequences into a multivariate data sequence [25]. The important research problem in this field is creating an effective attack dictionary [62]. Signature length is a coarse indicator of efficiency for
Chapter 2. Literature Search

signature-based approaches; longer signatures suggest a larger memory requirement and higher microprocessor use. Signature-based approaches are more effective against outsider attacks; malicious outsiders presumably will exhibit well known signatures in the course of penetrating the network.

2.2.1.2 Anomaly-Based Intrusion Detection

Anomaly-based intrusion detection approaches look for runtime features that are out of the ordinary. The ordinary can be defined in two ways: Some approaches, known as unsupervised, train with live data (the test signal history) [25]. Other approaches, known as semi-supervised, train with a set of truth data (a collection of training data). Beware some authors [119] refer to training data as a “signature.” Researchers take different approaches for discrete, continuous and multivariate datasets.

The key advantage of anomaly-based approaches is they do not look for something specific. This eliminates the need to fully specify all known attack vectors and keep this attack dictionary current. Specifically, these approaches can detect previously unknown attacks, such as zero day attacks. One major disadvantage of this category is the susceptibility to false positives [61]. Another major disadvantage of this category is the training/profiling phase, during which the system is vulnerable. (This only applies to semi-supervised techniques.) [24] provides a comprehensive survey of anomaly-based intrusion detection that is general to all applications. [150] refers to anomaly-based approaches as user or group profiling; [113] refers to this as a profile-based approach.

Anomaly-based approaches are further classified into conventional statistics-based approaches and data mining (non-parametric) methods. Data mining techniques are based on the concepts of classification or optimization; kernel estimators, data clustering, clustering ellipsoids and support vector machines (SVM) are examples of data mining methods [151]. The size of the feature set is a coarse indicator of efficiency for anomaly-based approaches; larger feature sets suggest a larger memory requirement and higher microprocessor use. [151] points out that feature selection is a key research problem with anomaly-based approaches: More features do not necessarily give better results. Anomaly-based approaches typically use a machine learning module to analyze profile data.

Al-Nashif, Kumar, Hariri, Qu, Luo and Szidarovsky [3] study an anomaly-based IDS that uses traffic-based collection called multi level intrusion detection system (ML-IDS). The multiple levels comprise traffic flow, packet header and payload inspection. The authors make an interesting assertion in stating that signature-based IDSs tend to have high false positive rates. They derive rules for the traffic flow level by using supervised learning. In our specification-based IDS, human experts define behavior rules. Al-Nashif et al. consider scanning (probing), passive scanning (eavesdropping), exploits, R2L attacks, DoS attacks, and worms in their attack model and show against which categories ML-IDS is effective. The authors consider four different time window sizes similar to how we experiment with
Chapter 2. Literature Search

the $T_{IDS}$ parameter. Their results show “more is better” with the longest time window (180 s) performing the best. However, we found that there is an optimal $T_{IDS}$. The difference is that we consider nodes exhausting their energy which is consumed in part by intrusion detection and Byzantine failure caused by good nodes being evicted due to false positives. They report detection rates between 70.5 and 97.0% and “near zero” false positive rates of 0.01%.

Hariri et al. [64] propose a biologically inspired IDS for cloud computing environments called Biologically inspired Resilient Autonomic Cloud (BioRAC). The authors focus on vulnerabilities created by the monoculture environment characteristic of cloud computing. Fundamentally, this is an anomaly-based technique.

Fayssal and Hariri [48] study an anomaly-based IDS for wireless networks called wireless self-protection system (WSPS). The authors’ attack model comprises IEEE 802.11 association request flood attacks.

### 2.2.1.3 Specification-Based Intrusion Detection

The distinction between anomaly and specification-based techniques is subtle. Anomaly-based detection techniques start with a semi-supervised or unsupervised machine learning approach which acts on empirically collected data to mine the features of a normal/good node. Specification-based detection techniques start with a human transforming the characteristics of a healthy system into a formal specification using a grammar or state machine. Specification-based techniques can be viewed as a subset of anomaly detection techniques as the underlying principle is the same.

Specification-based intrusion detection looks for abnormal performance at the system level. Contrast this with anomaly-based intrusion detection that analyzes specific user profiles or data flows and with signature-based intrusion detection that detects “known bad” behavior. Specification-based intrusion detection approaches formally define legitimate behavior and indicate an intrusion when the system departs from this model [19]. One major advantage of specification-based intrusion detection is a low false negative rate. By definition, these approaches only react to known bad behavior; the theoretical basis is a bad node will disrupt the formal specification of the system. Another major advantage of specification-based intrusion detection is the system is immediately effective because there is no training/profiling phase. The key disadvantage of specification-based intrusion detection is the effort required to generate a formal specification. Specification-based intrusion detection approaches are especially effective against insider attacks as they focus on system disruption. Like anomaly-based intrusion detection, these approaches can detect previously unknown attacks, such as zero day attacks.

Uppuluri and Sekar [145] propose a means and a methodology for specifying a system. Their language, Behavioral Monitoring Specification Language (BMSL), specifies both normal and
abnormal behaviors for a specification-based IDS using traffic and behavior collection. BMSL models the event details and event sequencing. The authors’ IDS transforms BMSL programs into detection engines (DEs). The methodology of Uppuluri and Sekar begins with specifying generic system behaviors, then focuses on highly privileged functions, specifies application specific behaviors next, then tailors the specification for each installation and ends by specifying misuse signatures. They combine the BMSL approach to specification-based IDS with a signature-based IDS in order to match the detection rate of a signature-based IDS. Requiring the IDS to store and update a large attack dictionary eliminates one of the benefits of specification over signature-based designs. The authors acknowledge BMSL does not effectively model time. For example, benign user error may account for one failed authentication in one day while ten failed authentications in a minute may indicate an adversary trying to crack the authentication; BMSL cannot distinguish between these situations. BMSL achieved a detection rate of 82% if it was not combined with a signature-based technique; it achieved 100% detection when combined with a signature-based technique.

Sekar et al. [124] use extended finite state automata (EFSAs) to establish a specification for a traffic-based IDS. They combine the EFSA approach to specification-based IDS with an anomaly-based IDS that uses unsupervised machine learning. Requiring the IDS to run a processor and core memory intensive machine learning module eliminates one of the benefits of specification over anomaly-based designs.

Ko et al. [79] propose Distributed Program Execution Monitor (DPEM) which uses the Parallel Environment Grammars (PE-grammars) language for a specification-based IDS using behavior collection.

Tseng et al. [141] use a finite state machine (FSM) to establish a specification for a traffic-based IDS; in particular, they specify the Ad hoc On-demand Distance Vector (AODV) function on their reference system. Distributed network monitors maintain an FSM for each routing transaction (request and reply). Sink states indicate either normal or alarm. Some transient states indicate suspicion; in these states, the network monitor asks its peers for additional insight on the transaction. The authors rely on a modification of AODV to support their design; specifically, this extended AODV has one additional field, previous node, in each message.

Li et al. [86] propose a multitrust IDS that is distributed asymmetrically in a Wireless Mesh Network (WMN). IDS roles are based on the broader network role; gateways, mesh routers and mesh clients perform different IDS functions. Specifically, each role provides different responses. The authors’ attack model only considers selfish adversaries. Specifically, they use a specification-based design that checks for contention window conformance; following a collision, selfish nodes do not stand down for as long as they should.

Zhou et al. [162] use a specification-based detection technique based on traffic data collection in a WMN. Multitrust data plays a key part in Zhou et al.’s design: It compares the communication state a node reports for itself with the state other nodes report for it. Less similarity indicates a higher probability of attack. The communication state consists of base
station (BS) and subscriber station (SS) visibility; the authors extract this state from the mesh network administration messages.

Specification-based approaches typically use a language parser or state machine to analyze profile data.

Another way to distinguish specification-based approaches is whether they are active or passive. Passive specification-based IDSs rely on functional nodes to generate sufficient traffic to drive their parsers or state machines. Active specification-based IDSs prompt RTUs and MTUs to provide traffic to drive their parsers or state machines.

2.2.1.4 Reputation Management

The primary function of a reputation manager is to detect nodes exhibiting selfish behavior rather than violating security. However, in the presence of malfeasance, reputation managers must also guard against colluding nodes intent on enhancing their reputation. [14] identifies the main problem in Reputation Management as distributing reputation scores. Reputation Management approaches are particularly applicable to large networks where establishing a priori trust relationships is not feasible. Examples of metrics reputation managers use are packets forwarded over packets sourced, packets forwarded over non-local packets received and packets sent over packets received. Choosing the metric is a matter of design philosophy. Using packets forwarded over non-local packets received mitigates a bias present in packets forwarded over packets sourced. Weighting data is a key problem for the analysis: Data points have different significance across the time domain (recent data may be more or less valuable than historical) as well as across sources [experienced data is more valuable than observed data which is more valuable than reported (multitrust) data]. Typical reputation manager response measures include distributing the raw data, distributing the reputation score and modifying the routing table. Reputation management is especially relevant to ad hoc applications such as MANETs and VANETs.

2.2.1.5 Effectiveness of Detection Techniques Applying to CPSs

In this section, we reason why certain detection techniques are more effective than others when applying to CPSs.

Anomaly-based designs in general are more effective than the other designs for attended CPSs. These systems have a well defined concept of operations, that is, they are mission-oriented/purpose-built and thus have predictable profiles. These systems also have unique aspects which favor anomaly-based designs. For example, due to federated control and safety criticality, maintenance of attack dictionary updates (for signature-based intrusion detection) is difficult.

Signature-based designs are not a good choice for CPSs because of limited resources (storage
and channel) of devices in a CPS in particular. The channel scarcity does not accommodate dictionary updates, and the limited storage limits the size of the attack dictionary.

Specification-based designs are more effective than the other designs for unattended CPSs because of their predictable profiles. These predictable profiles result in a language parser with fewer rules or an automaton with fewer states, making the IDS lightweight which is a desirable feature for unattended CPSs.

Reputation managers are not a good choice for federated CPSs because reputation managers are multitrust designs. Establishing trust across enclaves is notoriously difficult; for example, an enclave may reward greedy behavior near its edges.

\begin{figure}
\centering
\includegraphics[width=\textwidth]{figure2.4}
\caption{Collection Process Dimension of Intrusion Detection.}
\end{figure}

\section{2.2.2 Collection Process}

Figure 2.4 shows the Data Collection dimension for classifying IDS approaches. There are two ways to collect data before data analysis, namely, behavior-based collection and traffic-based collection. We first describe the essentials of behavior and traffic-based collection. Then we discuss the effectiveness of behavior and traffic-based collection in CPSs.

Here we should note that some papers use the terms host-based intrusion detection system (HIDS) and network-based intrusion detection system (NIDS) when referring to behavior-based collection and traffic-based collection, respectively [67, 109, 156].

\subsection{2.2.2.1 Behavior-Based Collection}

In behavior-based collection, an IDS analyzes logs or other artifacts, such as file system details, to determine if a node is compromised. One major advantage of using behavior-based collection approaches is scalability. This is attractive for large scale applications. Another major advantage of using behavior-based collection approaches is decentralization; this is
attractive for applications with no infrastructure. One major disadvantage of a behavior-based collection is that each node has to perform additional work to collect, if not analyze, its audit data. This is undesirable for resource constrained devices such as RTUs in a CPS. Another major disadvantage of this technique is that a sophisticated attacker can cover their tracks by modifying the audit data on the captured node. A third disadvantage of this technique is that it can be OS or application specific (depending on the particular content of the logs) [62]. Behavior-based collection is not used widely in wireless environment applications [132].

2.2.2.2 Traffic-Based Collection

IDSs using traffic-based collection study network activity to determine if a node is compromised. This audit can be general (traffic/frequency analysis) or protocol-specific (deep packet inspection). The key advantage regarding resource management is that individual nodes are free of the requirement to maintain or analyze their logs. The key disadvantage regarding data collection is that the effectiveness of a traffic-based technique is limited by the visibility of the nodes collecting audit data. Thus, it is challenging to arrange traffic-based collection sensors to get complete intra-cell and inter-cell pictures of network activity [132].

2.2.2.3 Effectiveness of Collection Processes Applying To CPSs

In this section, we reason why certain collection processes are more effective than others when applying to CPSs.

Neither traffic-based nor behavior-based collection is always the best choice for CPS IDS. Either can work well depending on the capability and configuration of the intrusion detector. For example, if code attestation [32] is used for intrusion detection, behavior-based collection works better as the information to be collected is about the code behavior of a host. However, if reputation is used for intrusion detection, traffic-based collection works better as the information to be collected is typically about the network traffic characteristics of a host (for example, selective forwarding). Conceivably it is possible to collect both types of data for intrusion detection. In our dissertation research, we disfavor behavior-based collection because it is difficult if not impossible to examine logs in a federated CPS. Instead we adopt traffic-based collection for a node to monitor its neighbor nodes to measure the compliance degree based on which host $p_{th}$ and $p_{tp}$ are derived.

2.2.3 Trust Model

Figure 2.5 shows the Trust Model dimension for classifying IDS approaches. There are two basic trust models, namely, multitrust and unitrust. We first describe the essentials
Chapter 2. Literature Search

of multitrust and unitrust. Then we discuss the effectiveness of multitrust and unitrust in CPSs.

Figure 2.5: Trust Model Dimension of Intrusion Detection.

Buchegger and Le Boudec [20] propose a distributed reputation management system called CONFIDANT. Buchegger and Le Boudec distinguish three levels of multitrust: *Experienced* data is a firsthand account and CONFIDANT’s reputation system weights it the most strongly. *Observed* data happens in the neighborhood (within radio range) and has a lower weight than experienced data. Finally, *reported* data is an account coming from outside the neighborhood; it receives less weight than experienced or observed data in CONFIDANT’s reputation system. Rather than permanently expelling intruders, the authors afford detected nodes the opportunity to appeal their detection (in the case of false positive) or rehabilitate their reputation (in the case of a reformed greedy node). This is actually a side effect of dealing with the reality of storage limitations and search of complete node histories that grow without bound. Buchegger and Le Boudec’s primary sanction is global stoppage of the intruder’s packet forwarding. Borrowing from the field of ecology, they classify nodes into one of three categories: suckers (who always assist neighbors), cheats (who never assist neighbors) and grudgers (who assist neighbors until they experience non-reciprocation). To measure reputation, CONFIDANT looks for “no forwarding” behavior. The key innovation in Buchegger and Le Boudec’s research is in the intrusion response: Nodes use reputation to guide routing decisions and to inform reputation management itself (a feedback loop/closed loop).

2.2.3.1 Multitrust

Multitrust is the concept of using hearsay/reported information (data from witnesses or third parties). [88] calls this type of information a *recommendation*. Contrast recommendations with what [127] calls *direct monitoring*. This hearsay information can be raw data or an
Chapter 2. Literature Search

analysis result. Using multitrust together with behavior-based collection mitigates a key weakness: the opportunity for capable adversaries to cover their tracks. Multitrust often appears in the context of reputation management which is most applicable to ad hoc applications such as MANET and VANET. However, giving weight to others’ recommendations in a federated environment leads to a dilemma: On one hand, a node places enough trust in neighbors to include their hearsay in reputation calculations. On the other hand, nodes are suspicious enough of their environment to measure and respond to the reputation of their neighbors. Therefore, multitrust is better suited to increasing the security of managed/authenticated environments rather than to establishing a basic level of security in ad hoc environments such as MANETs and VANETs.

Reputation managers require two levels of trust: The “outer circle” of trust regards the system function in general while the “inner circle” of trust regards the multitrust function specifically. The literature uses the term trustworthiness in reference to this “inner circle” credibility [126]. Because the effectiveness of traffic-based approaches are limited by radio range in wireless environments, multitrust offers an advantage for these applications. The literature sometimes calls multitrust approaches cooperative; it further distinguishes them as distributed or hierarchical [127].

2.2.3.2 Unitrust

We classify some IDSs as unitrust, which some research refers to as standalone. In contrast with multitrust designs, a unitrust design does not use reported information; a unitrust design relies on direct monitoring. The advantage of a unitrust design is the data is completely reliable; the IDS does not need to apply safeguards to prevent or tolerate biased reports from adversaries. The disadvantage of a unitrust design is the smaller dataset; the IDS only acts on the data it experiences or observes.

2.2.3.3 Effectiveness of Multitrust versus Unitrust Applying to CPSs

In this section, we discuss the effectiveness of multitrust versus unitrust as applying to CPSs. Here we should note that the discussion is based on the assumption that only multitrust or unitrust is being used. We recognize that many reputation and trust management systems actually consider both multitrust and unitrust in trust composition [10, 26, 39].

Unitrust is more effective than multitrust for CPSs because it may be difficult for them to establish trust. This is due to federated control of CPSs; authentication may not span enclaves of the CPS. In general, the lack of trust and authentication in these systems makes multitrust difficult.

Due to damaged, imprecise and incomplete datasets, node-level unitrust IDS techniques can produce unacceptably low detection rates and unacceptably high false positive rates. To
mitigate this, this dissertation research proposes to combine node-level unitrust IDS with system-level voting multitrust IDS.

2.2.4 Response Technique

Figure 2.6 shows the Response Technique dimension for classifying IDS approaches. There are two basic response techniques, namely, passive and active. We first describe the essentials of passive and active response techniques. Then we discuss the effectiveness of passive and active response techniques in CPSs. Response is the last of the three core IDS functions discussed in Section 2.1.

2.2.4.1 Passive Response Techniques

Logging and administrator notification are examples of passive response techniques; passive responses are more appropriate when an IDS has less confidence in a positive detection result or the cost of an active response is too high. Also, a passive response often allows the IDS to rescind/repeal its response measure; this is desirable to restore nodes sanctioned as a result of a false positive.

2.2.4.2 Active Response Techniques

Active response techniques can be further divided into proactive and reactive response techniques [7]. One example of a proactive response technique is a honeypot: The system authority creates a vulnerable and apparently valuable node with exceptional data collection measures; this bait node attracts attackers, and the system authority gathers useful profile data on the adversary. Performing a vulnerability assessment is another example of a proactive response technique. On the other hand, blocking the data flow for an intruder, collecting more extensive information regarding an intruder or changing IDS parameters
such as audit interval, intrusion detector quantity or compliance threshold are examples of reactive response techniques. Active responses risk negatively impacting the network as a result of a false positive or as a result of an overly heavy handed tactic. Proactive, reactive and passive responses can be distinguished by the time at which responses occur on the time domain as follows [7]: Proactive tactics occur before the time an attack is detected, reactive tactics occur soon after an attack is detected, and passive tactics occur some time after an attack is detected.

### 2.2.4.3 Effectiveness of Response Techniques Applying to CPSs

In this section, we reason why certain response techniques are more effective than others when applying to CPSs.

Because a CPS failure has dire consequences, proactive responses are more effective than reactive or passive responses to prevent attacks from occurring. Our approach for IDS response design is based on detecting attacker strength proactively and adjusting detection strength to prolong the system lifetime. If the attacker strength is low, then the IDS detection strength also can be lowered to reduce energy consumption. Otherwise, the detection strength is increased so as to detect and evict bad nodes to prevent them from causing security failures.

### 2.3 Classifying Existing CPS Intrusion Detection Techniques

This section applies the criteria Section 2.2 established to classify existing IDS techniques for CPSs according to the classification tree in Figure 2.2. The intent is to examine the most and least intensive research in IDS to date and identify research gaps yet to be explored. We summarize our findings of surveying 18 IDS techniques in Table 2.1. We note that none of the 18 IDS techniques surveyed considers intrusion response which is largely an unexplored area.

### 2.4 Revisiting IDS Techniques and Gaps in CPS IDS Research

In this section, we first discuss the most and least studied IDS techniques in the literature based on our survey. Then we identify gaps yet to be explored and revisit IDS techniques that deserve further research for CPSs. Table 2.2 identifies the most and the least researched
### Chapter 2. Literature Search

#### Table 2.1: Classification of CPS IDSs

<table>
<thead>
<tr>
<th>Detection Technique</th>
<th>Collection Approach</th>
<th>Multitrust</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCADA Signature Generator [110]</td>
<td>Signature</td>
<td>Traffic</td>
<td>Test automated XML profile to Snort signature transform in an electricity distribution laboratory</td>
</tr>
<tr>
<td>Model Based EMERALD [87]</td>
<td>Specification</td>
<td>Traffic</td>
<td>Use PVS to transform protocol, communication pattern and service availability specifications for use with EMERALD</td>
</tr>
<tr>
<td>State Based Network IDS [54]</td>
<td>Combined</td>
<td>Traffic</td>
<td>Foundation for [21] that guards against complex attacks with a Modbus/DNP3 state machine</td>
</tr>
<tr>
<td>Retrofit Network Transaction Data Logger [107]</td>
<td>Combined</td>
<td>Traffic</td>
<td>Tamper-proof distributed traffic collection technique to retrofit legacy CPSs reliant on serial links</td>
</tr>
<tr>
<td>ISML [21]</td>
<td>Specification</td>
<td>Traffic</td>
<td>Extension to [54] that distinguishes faults and attacks, describes a language to express a CPS specification and establishes a critical state distance metric</td>
</tr>
<tr>
<td>IDS-NNM [87]</td>
<td>Anomaly</td>
<td>Traffic</td>
<td>Error-back propagation and Levenberg-Marquardt approaches using window-based feature extraction</td>
</tr>
<tr>
<td>INL IDS [147]</td>
<td>Combined</td>
<td>Traffic</td>
<td>Multi-trust hybrid approach using signature-based detection and traffic analysis</td>
</tr>
<tr>
<td>Modbus Based ANN [57]</td>
<td>Anomaly</td>
<td>Traffic</td>
<td>Three stage back propagation ANN based on Modbus features</td>
</tr>
<tr>
<td>T-Rex, T-Prot and T-AxT [163]</td>
<td>Specification</td>
<td>Behavior</td>
<td>Instrument application and use scheduler to confirm timing analysis results</td>
</tr>
<tr>
<td>AAKR/SPRT [153]</td>
<td>Anomaly</td>
<td>Behavior</td>
<td>SNMP drives prediction, residual calculation and detection modules for an experimental testbed</td>
</tr>
<tr>
<td>n-grams and Invariant Induction [17]</td>
<td>Anomaly</td>
<td>Traffic</td>
<td>Demonstrates promising control of detection and false negative rates</td>
</tr>
<tr>
<td>Shin Technique [127]</td>
<td>Combined</td>
<td>Traffic</td>
<td>Analysis results of two tier (one hop, multihop) clustering hierarchical IDS</td>
</tr>
<tr>
<td>HFMI/IDS [95]</td>
<td>Anomaly</td>
<td>Traffic</td>
<td>Vote-based</td>
</tr>
<tr>
<td>EMERALD [113]</td>
<td>Combined</td>
<td>Behavior</td>
<td>Hierarchical and laterally shared detection results</td>
</tr>
<tr>
<td>Xie Technique [152]</td>
<td>Anomaly</td>
<td>Combined</td>
<td>Kernel estimators and voting</td>
</tr>
</tbody>
</table>

Table 2.2: Commonality and Variability of IDS Techniques for CPSs (white: no research, grey: little research, black: significant research, ×: deserving more research)
Chapter 2. Literature Search

IDS techniques, that is, which IDS techniques have been researched the most or the least in CPSs. This table aggregates the survey results from Section 2.3.

In Table 2.2, we identified several gaps in the literature. Many of these gaps do not need investigative attention, but some do.

First, none of these papers has addressed the federated nature of CPSs.

Also, little research has been done on signature-based designs applied to CPSs. This is in line with the reasons we presented in Section 2.2.1.5 that signature-based designs are not suitable for CPSs.

Third, there are only a few existing studies on specification-based intrusion detection applied to CPSs. Researchers should pursue specification-based designs for CPSs. Specification-based IDSs are particularly applicable to CPSs: Signature-based approaches are not workable because of maintenance difficulty and inability to detect unknown attacks. Reputation management-based approaches are not workable because of federation; specifically, enclaves may reward greedy behavior near their boundaries. The false alarm rates for specification-based designs are better than anomaly-based designs, and the well defined use cases for CPSs yield a rich set of phenomena to specify.

Also, none has applied reputation management-based designs to CPSs. For the reasons we presented in Section 2.2.1.5, reputation management-based designs are not suitable for CPSs.

Fifth, none has considered IDS responses, and there is a significant gap for response approaches. Dynamic response approaches should be investigated as applied to CPSs. For the reasons we presented in Section 2.2.4.3, static response approaches are not effective, and the community should regard them as an evolutionary step on the way to more effective dynamic response approaches.

Summarizing above, we mark × in Table 2.2 for IDS techniques which are relatively unexplored in the literature but deserve further research attention, as they have been identified as suitable as well as potentially impactful for CPSs.

2.5 Challenges of CPS Intrusion Detection and Response

In this section, we create a synergy from the challenges identified by the literature survey to our approaches of answering the challenges. We identified five key challenges of CPS intrusion detection and response [105]:

1. pursue specification-based designs for CPS intrusion detection
2. investigate dynamic intrusion response approaches applied to CPSs
Chapter 2. Literature Search

3. develop CPS IDS modeling and analysis tools
4. address the federated nature of CPSs
5. separate IDS and business software

To answer the specification-based CPS IDS design challenge, we propose the design concept of compliance degree based on behavior specifications for host IDS design to result in a lightweight, accurate, autonomous and adaptive protocol that runs on every node in the CPS to detect misbehavior of neighbor nodes. We also propose a system IDS design based on majority voting to cope with malicious, erroneous, partly trusted, uncertain and incomplete information in a CPS, resulting in a robust and resilient protocol for intrusion detection. However, majority voting cannot always guarantee success. Fischer et al. [50] show that the consensus problem is deceptively hard. While the classic Byzantine Generals [81] study tackles a synchronous environment, the authors demonstrate that in an asynchronous environment, a single bad process can deny consensus to a distributed system. The broader literature pursues alternatives to majority voting: Gilbert et al. [58] contrast simple majority systems with quorum systems (also called coteries). Roberto De Prisco [114] deconstructs Lamport et al.’s Paxos algorithm [80] for achieving consensus in a distributed system. Our voting-based IDS design can overcome the limitations of majority voting by transitioning to the quorum system, implementing the Paxos algorithm or applying a novel technique.

To answer the CPS dynamic intrusion response challenge, we propose proactive intrusion response designs to prevent security failures, resulting in a highly adaptive and dynamic control protocol that can adjust detection strength in response to changes in attacker strength and behavior.

To answer the CPS IDS modeling and analysis tool challenge, we propose a model-based analysis methodology with simulation validation to identify optimal IDS design settings that will maximize the CPS reliability and lifetime.

To answer the federation challenge, we aim to investigate an abstraction model for describing a multiple enclave federated CPS and identify as well as apply optimal IDS settings at both the host-level and enclave-level to maximize the multi-enclave federated CPS lifetime.

Finally, to answer the IDS/business software separation challenge, we review the literature: Running IDS at the same location as business software creates problems for both functions: It creates a security weakness and may interfere with legacy operations. Virtual machine monitors (VMMs) and multicore microprocessor designs provide an opportunity to solve this problem effectively. Kirkpatrick et al. [77] apply VMM technology to increase the security of untrusted software modules. The authors assume a powerful attacker model: The adversary has compromised the operating system running on an x86 microprocessor and will cause damage by modifying the memory image of a critical application. They seek to detect and repair these memory modification attacks by applying Luby Transform and Reed-Solomon error-correction codes to main memory. Shih et al. [125] propose a framework
for implementing trusted embedded real-time software for a multicore target called VERTAF Multicore. Shin and Lee [128] solve a similar problem with their Abstract Component Architecture and design Methodology for multicore Embedded Systems (ACaMES) design.
Chapter 3

System Model

CPSs have multiple control loops, strict timing requirements, a wireless network segment, predictable network traffic and legacy components. CPSs fuse cyber (network components and commodity servers) and physical (sensors and actuators) domains. They use federated control due to stakeholders with different interests and concepts of operations. CPSs must self organize due to scale and cannot be readily patched due to certification. They may contain human actors and mobile nodes. The term Mobile CPS indicates a CPS with mobile nodes. Our system model encompasses both stationary and mobile CPSs. Mobile systems are an interesting special case of CPSs, but there are more stationary than mobile systems deployed. CPSs are trending towards heterogeneous, off-the-shelf components and open interfaces. CPSs may operate in locations that are dangerous due to temperature, hazardous materials (HAZMAT) or hostilities. We assume a CPS consisting of heterogeneous nodes dispersed in a geographical area. Some nodes carry sensors and actuators. Some nodes are energy constrained. Some nodes have wireline interfaces while others use wireless terrestrial or satellite interfaces; two wireless nodes are connected when they are within radio range, $R$, of each other. Groups of nodes form enclaves based on national, departmental, service or branch affiliation; this presents a federated environment.

Our work is tailored for CPSs, not cyber systems, which are not constrained by legacy processor and memory limitations and realtime response deadlines which would make anomaly detection feasible. Also, a cyber system can more readily update its attack dictionary which would make signature detection feasible. Therefore specification-based IDS is uniquely suited to CPS because it fills the gap left by these more conventional approaches.

3.1 CPS Abstraction Model

Figure 3.1 illustrates a hierarchical abstraction model for a federated CPS. It represents all of the key CPS artifacts introduced in Chapter 1: enclaves, sensors, actuators, RTUs, DASs,
Chapter 3. System Model

Figure 3.1: Hierarchically Structured CPS Abstraction Model.

MTUs, data processing servers, historian servers, HMIIs, operators and communications links. RTUs comprise sensors and actuators interconnected via local high speed network or bus links. In turn, they are managed by a DAS which bridges the gap between the remote and control segments with a long distance wireless link. They escalate sensor data to the historian server and receive control messages from the MTU. Operators use HMIs to read the sensor data in the collocated historian and exploit it with the assistance of collocated data processors. Multiple enclaves comprise the CPS; highly scrutinized business rules govern the exchange of data between historian servers.

In this dissertation research, we consider four reference models from the abstraction model.

Mobile Group CPS: MGCPs maps to the abstraction model as follows: The MGCPs operates as a single hierarchical, homogeneous enclave. The sensors comprise a GPS receiver, accelerometers, an ISM band (915 MHz) receiver, physiological sensors (e.g., pulse meter, breathing gas meter and internal thermometer) and environmental sensors (e.g., external thermometer and dosimeter). The actuators comprise an ISM band transmitter and a speaker (evacuation tone and aural rescue beacon). The size, weight and power-critical RTUs are human-carried enclosures that contain the sensors and actuators. The DAS/MTU/tactical HMIs are vehicle-mounted components operated by a tactical or on-scene commander (OSC). The data processing servers, historian servers and strategic HMIs are fixed resources located at a remote emergency operations center (EOC). The communication links include proprietary MANET with ISM band TDMA/CDMA waveforms (RTUs), IEEE 802.11/WiFi (MTUs) and IEEE 802.16/WiMAX (EOC). Chapter 5 provides specific details for the MGCPs. We accomplish a case study on a mobile group CPS because of the human impact of these systems. For a company (five firefighters) or battalion (25 firefighters), failure of their MGCPs can be fatal to the group or an individual. One of the primary functions of a first responder MGPCs is to provide situational awareness regarding hazardous
Chapter 3. System Model

materials (HAZMAT). If the MGCPS does not identify a dangerous chemical in the environment and route that information correctly, the entire team is in jeopardy. Another first responder MGCPS primary function is to locate incapacitated workers. If the MGPCS does not function, these workers will succumb to their hazardous environment. The real-world features of an MGCPS that inform our modeling and simulation are: low velocity, two-dimensional mobility and limited energy supply. We assume linear capture strength because it is the most general. On the other hand, logarithmic capture strength assumes the defender becomes more effective over time, and exponential capture strength assumes the attacker becomes more effective over time. Given the limited duration of an emergency, these are not reasonable scenarios. Furthermore, constant capture strength does not consider that existing captured nodes will accelerate the rate of future captures. Given the dynamic, ad hoc topology of an MGCPS, the additional opportunities for propagation offered by already compromised nodes cannot be ignored. We assume reckless insider attack behavior because in order for an attack on an MGCPS to be devastating, timing is critical. If the effect occurs after the emergency is resolved, it will be inconsequential. Therefore, we focus on reckless attackers rather than the others which may use a time-delay. MGCPSs have been the victim of recent attacks [123].

**Medical CPS:** MCPS maps to the abstraction model as follows: The MCPS operates as a single flat, heterogeneous enclave. Three types of nodes comprise the MCPS: vital sign monitors (VSMs), patient controlled analgesia (PCA) units and cardiac devices (CDs). Each of these node types acts as an RTU and carries an array of sensors and actuators. The communications links are IEEE 802.11 (WiFi). Chapter 6 provides specific details for the MCPS. We accomplish a case study on an MCPS because of the human impact of these systems. For a cardiac ward with 156 patients (e.g., Inova Heart and Vascular Institute) or hospital with 833 beds (e.g., Inova Fairfax Hospital), failure of their MCPS can be fatal to an individual. One of the primary functions of an MCPS is to administer analgesics. Overmedicating a patient will cause cardiac arrest. Another MCPS primary function is to provide cardiac support. Doing so when unnecessary or failing to do so when appropriate will kill the patient. The real-world features of an MCPS that inform our modeling and simulation are: heterogeneous architecture, limited energy supply and low communications security (IEEE 802.11 links). We assume constant capture strength because the heterogeneous MCPS composition will limit the accelerating effect of existing compromised nodes. However, the number of these existing compromised nodes is a parameter for logarithmic, linear and exponential capture strength models. Time is not critical so reckless insider attack behavior is not appropriate. Also, the insidious insider attack behavior does not fit because the target for an MCPS attack is a single patient, rather than a ward or hospital. Finally, the random insider attack behavior is arbitrary and does not give enough credit to the adversary. Therefore, we use the opportunistic insider attack behavior which considers a sophisticated attacker who targets a single patient and has a long time to launch the attack. MCPSs have
been the victim of recent attacks. A Department of Veterans Affairs study found 173 incidents of malware infecting medical devices between 2009 and 2011 [138].

**Smart Grid CPS:** SGCPS maps to the abstraction model as follows: The SGCPS operates as a single hierarchical, heterogeneous enclave. Three types of nodes comprise the SGCPS: head-ends (HEs), distribution access points/data aggregation points (DAPs) and smart meters (SMs). The HE encompasses the data processing and historian servers and HMI. The DAPs encompass the DAS and MTU functions. The SMs act as RTUs and carry an array of sensors and actuators. The communications links include IEEE 802.11/WiFi, IEEE 802.16/WiMAX and power line communication (PLC).

Chapter 7 provides specific details for the SGCPS. We accomplish a case study on an SGCPS because of the human impact of these systems. While they are not life-critical, the scope of a smart grid can be enormous. In July 2012, 620 million customers in India lost power for up to two days. The cost of this could be measured in lost productivity: the single day cost for 620 million unproductive citizens given the Indian per capita GDP of $1388 ($3.8 daily) is $2.356 B. The real-world features of an SGCPS that inform our modeling and simulation are: large scale, federated control, unattended operation and high threat of physical capture. We assume exponential capture strength. Linear capture strength assumes neither the defender nor the attacker changes effectiveness over time. Due to the long operational lifetime, this is unlikely. Logarithmic capture strength assumes the protected resource becomes a more effective defender over time. This is too optimistic because of the federation and massive scale of an SGCPS. Furthermore, constant capture strength does not consider that existing captured nodes will accelerate the rate of future captures. Given the mesh topology of the neighborhood area network (NAN) segment of an SGCPS, the additional opportunities for propagation offered by already compromised nodes cannot be ignored. We assume insidious insider attack behavior because the SGCPS adversary can delay attack in order to prepare the cyber battlefield by establishing a preponderance of compromised foothold nodes in the target. SGCPSs have been the victim of recent attacks [134, 135, 136].

**Unmanned Aircraft CPS:** UACPS maps to the abstraction model as follows: The UACPS operates as a single flat, homogeneous enclave. The unmanned air vehicles (UAVs) act as RTUs and carry an array of sensors and actuators. Air operations centers (AOCs) provide the DAS, MTU, data processing, historian and HMI functions. The communication links are line of sight (LOS) VHF, LOS UHF or SATCOM. Chapter 8 provides specific details for the UACPS. We accomplish a case study on a UACPS because of the human impact of these systems. A combat vehicle could use weapons against non-combatants. An MQ-9 Reaper can carry a Hellfire missile or 500 pound bomb [108]. A 500 pound bomb delivering a 200 pound TNT warhead will have a 379 MJ yield; in comparison, an M67 fragmentation grenade has a yield of 0.753 MJ. In addition, a surveillance vehicle could fly into a densely populated area or critical resource (power substation, water treatment plant, center of government). The real-world features of a
UACPS that inform our modeling and simulation are: high velocity, three-dimensional mobility, unattended operation, high communications security (LOS links) and small scale. We assume constant capture strength. Logarithmic capture strength assumes the defender becomes more effective over time, while exponential capture strength assumes the attacker becomes more effective over time. Given the limited duration of a mission, these are not reasonable scenarios. Furthermore, linear capture strength considers that existing captured nodes will accelerate the rate of future captures. Given the strictly hierarchical topology of a UACPS, the additional opportunities for propagation offered by already compromised nodes are limited. We assume reckless insider attack behavior because in order for an attack on a UACPS to be devastating, timing is critical. If the effect occurs after the mission is complete, it will be inconsequential. Therefore, we focus on reckless attackers rather than the others which may use a time-delay. UACPSs have been the victim of recent attacks. While [117] takes the position that the UAV in the incident was lost due to technical error rather than cyber attack, it analyzes the intelligence value of an intact vehicle.

The critical difference between the designs for these systems is the underlying human-created rule set. Behavior rule based intrusion detection is particularly applicable to CPSs because the underlying rule set can express the physical characteristics of the purpose-built system. This results in a novel defense that will present an unfamiliar challenge to attackers. While the system-specific rule set provides an opportunity for strong and unique security, it comes with added responsibility. The human generating the specification must understand the system inside and out or risk creating vulnerabilities due to missing or incorrect rules.

### 3.2 System Failure Model

We consider three system failure conditions for a CPS:

1. **Byzantine Failure** [81]: That is, if one-third or more of the nodes are compromised, then the system fails. The reason is that once the system contains one-third or more compromised nodes, it is impossible to reach a consensus, hence inducing a security failure.

2. **Energy Exhaustion Failure**: That is, for a CPS with limited energy (battery-based), the CPS fails when its energy (sum of device energy) falls below a threshold level that prevents it from functioning.

3. **Functional Impairment Failure**: That is, when the functional impairment caused by attacks exceeds a system tolerance limit, the system is considered as having failed because it cannot execute its intended functions due to significant impairment to its basic functions.
Here we note that a system failure condition can be applied in a federated manner for a CPS consisting of multiple enclaves. That is, each enclave fails if a failure condition is satisfied within the enclave.

### 3.3 Monitoring, Noise and Mis-monitoring Probability

Our behavior-rule based IDS approach relies on the use of monitor nodes. We assume that a monitor node performs intrusion detection on a neighbor trustee node. One possible design is to have a sensor (actuator) monitor another sensor (actuator) within the same RTU. However, this design requires each sensor (actuator) to have multiple sensing functionalities. Another design which we adopt is to have a neighbor RTU or a remote HMI, MTU or DAS monitor an RTU. We model imperfect monitoring by an error parameter, $p_{err}$, representing the probability of a monitor node misidentifying the status of the trustee node due to ambient noise, temporary system faults and/or wireless communication faults in CPS environments. In general, a node may deduce $p_{err}$ at runtime by sensing the amount of ambient noise, system errors and/or wireless communication errors around it.

### 3.4 Attack Model

The first step in investigating network security is to define the attack model. The attack model for a CPS is typically short in duration. The adversary who penetrates a CPS and loiters can gain little intelligence; the critical threat is an adversary who enters the network and disrupts the concerned processes to cause a catastrophe. For this reason, speed of detection is a key challenge in CPS which has not been studied in the literature.

Inside attackers are authenticated while outside attackers are not. Inside attackers have the full capability of a good node; their attack surface is relatively large. Outside attackers are highly restricted in how they interact with the target; their attack surface is relatively small. Some outsider attacks, such as remote to local (R2L) and brute force (dictionary), can be prevented by means such as authentication or confidentiality. Other outsider attacks, such as capture, require physical security measures. R2L and capture attacks create inside attackers. Intrusion detection of and response to insider attacks is the main interest of this dissertation research. We consider several forms of insider attacks:

1. Zero day attacks occur in the wild before they have been published; therefore, any signature-based detection technique will not be effective against them because they do not exist in any dictionary. Zero day attacks are a type of unknown attack; a CPS can counter this attack by using anomaly or specification-based detection approaches. Our host intrusion detection design addresses this attack.
Chapter 3. System Model

2. Advanced Persistent Threats (APTs) are strategically focused, highly sophisticated, resource-rich adversaries such as national governments or large companies that focus on a specific target. Including or excluding APT impacts the scope of a threat model significantly. Contrast APT with the threat imposed by a rogue employee or ephemeral opportunistic criminal or hacker who attacks soft targets of opportunity rather than a single hardened high value target.

3. A shellcode attack places software written by the adversary on the target system. This software could start an interactive shell (hence the namesake), add a user, open a socket to enable R2L attacks [163] or replace a legitimate library. A CPS can counter this attack by using code attestation.

4. A code injection attack sends unexpected input to the target. This input is specifically crafted to bypass normal processing; it can damage the target directly or set up another attack (for example, a shellcode attack). A CPS can counter this attack with rigorous testing/certification.

5. Excuse and newbie-picking attacks are categories of a broader category called policy attacks [63]. An adversary prosecuting an excuse attack claims to be damaged in order to avoid fulfilling its requirements. The premise of a newbie-picking attack is that newcomers to a network are required to temporarily fulfill a disproportionate fraction of system requirements in order to “pay their dues.” An adversary carrying out this attack completely exploits the naivety of newbies seeking to end their probationary period. These attacks primarily concern the ad hoc segments of a CPS. The source of these attacks are the selfish (uncooperative), rather than malicious, adversaries that characterize a federated environment such as a CPS. A CPS can counter this attack with prevention and tolerance: It can prevent these attacks by carefully considering policies in ad hoc segments, and it can tolerate these attacks by using sufficient redundancy in ad hoc segments.

6. Desynchronization (timing) attacks disrupt the clock phase of different nodes. The adversary’s intent is to disrupt the control loops effected by the CPS. Data from a single source will appear earlier or later than its actual time of validity. Data from multiple sources will not be in phase. In either case, the logic unit will be using a damaged dataset for input. Therefore, it will not control the system correctly. This attack seeks to violate the network security concept of availability. A CPS can prevent this attack by strongly authenticating time sources. Unfortunately, it may not be possible to tolerate this attack given the sensitive nature of control loops.

7. Probing attacks survey the subject system without causing damage. The adversary’s intent is to determine vulnerabilities. Arguably, this attack defeats the network security concept of confidentiality. A CPS can counter this attack by disabling ports and services that are not essential to the mission and patching known vulnerabilities.
8. An adversary prosecuting a jellyfish attack takes care to remain protocol compliant. It is more useful to consider jellyfish as a category of attacks or a means of enhancing an attack rather than being an attack vector itself. The best discussion of prevention, detection and tolerance considers specific forms of jellyfish attack. A CPS can counter this attack by using a dynamic response approach; this would minimize the false positives caused by an excessively high (yet ineffective) compliance threshold when the adversary is dormant and the false negatives caused by an excessively low compliance threshold when the adversary is attacking. Because it is based on misbehavior detection, our intrusion response design addresses this attack. While Jellyfish attacks may be 100% protocol compliant, they will not be 100% behavior compliant. The evidence traces as a result of attacks will still lead to deviation from the behavior rules when these rules are specified vigorously. Jellyfish adversaries behave as random attackers when they attack.

9. In a Man-In-The-Middle (MITM) attack, the adversary prepares the battlefield for future attacks. The MITM does not cause damage, rather it establishes an adversary as a middleman between two friendlies to be more favorable for future activities such as blackhole attacks, greyhole attacks and eavesdropping. A CPS can counter MITM attacks by using static routing tables. While this measure penalizes the flexibility of ad hoc segments, it is useful for wireline or infrastructure-based wireless segments.

10. Malicious nodes drop all network traffic in a blackhole attack. This attack damages the routing function in the most direct, unsophisticated way possible. Malicious nodes drop some network traffic in a greyhole attack. This attack seeks to evade simple detection designs while damaging the routing function. A greyhole attack is a very basic jellyfish modification to a blackhole attack. These attacks defeat the network security concept of availability. A CPS can counter this attack with prevention and tolerance. It can prevent black and greyhole attacks based on the form that they take: Static routing tables prevent an MITM scenario, and robust physical security prevents a node capture attack. Also, a CPS can tolerate this attack by using secure routing protocols.

11. “Honest but curious” actors may launch the following two attacks: Nodes prosecuting eavesdropping attacks record data for which they are not the destination. These attacks defeat the network security concept of confidentiality. A CPS can prevent this attack by encrypting payload. Nodes prosecuting traffic analysis collect metadata for traffic they are not intended to receive; then, these nodes calculate statistics over the metadata. Examples of metadata include: source, sink, search terms or publish-subscribe tags. A CPS can prevent this attack by encrypting metadata and rigorously specifying the policy governing shared data. Because these attacks are passive, there is no question of tolerance.

12. An exfiltration attack makes sensitive data available outside of the target system. This attack defeats the network security concept of privacy. Attacks such as probing,
eavesdropping or traffic analysis naturally combine with an exfiltration attack when launched by an insider.

13. Spoofing attacks misrepresent the identity of a node. Sybil attacks are a special case of spoofing attacks where one node misrepresents itself with many identities, rather than just one. Sybil attacks can increase the effectiveness of denial of service (DoS) or other attacks. The strategy behind this is to decrease the density of misuses or anomalies. That is, the adversary aims to keep the attack from surpassing some detection threshold by distributing malicious actions over many identities. These attacks defeat the network security concept of authenticity. A CPS can counter spoofing by strongly authenticating nodes; cryptography can provide a strong authentication solution with brute force (long key lengths) or elegance (game changing techniques). Also, the concept of operations can provide a solution using multi-factor authentication. Possible authentication factors are: ownership (a badge for example), knowledge (like a password), biometric (for example, a fingerprint) and relationship (a third-party introduction).

14. A DoS attack intends to overwhelm the capacity of the subject system. For example, if a web server can handle 1000 simultaneous sessions, a DoS establishes this many sessions to prevent a legitimate user from being served. This attack defeats the network security concept of availability. A CPS can counter DoS using robust designs whose performances fade gradually under high demand: For example, sensors should push data into the network via multicast or publish/subscribe. CPS segments can tolerate a DoS attack by remaining loosely coupled: For example, RTUs should be able to “run free” without input from a DAS or MTU.

15. A user to root (U2R) attack presupposes the adversary can authenticate as a normal user; having authenticated as a normal user, this attack proceeds to escalate their privileges. This attack defeats the network security concept of authenticity. A CPS can counter a U2R attack by following information and communications technology (ICT) best practices for privilege separation and principle of least authority.

16. Capture attacks create an insider threat which enables insider attacks. The geographic dispersal that characterizes CPSs make node capture a key part of the attack model, because it is difficult to guarantee physical security across a large area. A CPS can counter this attack with prevention and tolerance: It can prevent some captured node attacks with robust physical security; however, complete physical security is not practical for the broad geographic dispersion that characterizes CPSs. Also, a CPS can apply a voting-based IDS to both detect and tolerate captured node attacks. Our system intrusion detection design addresses this attack.

17. Data modification attacks on CPSs are further classified as command or response injections, command or response modifications or command or response omissions. A CPS can counter this attack with prevention and tolerance: It can prevent data
modification attacks by using message integrity codes (MICs) or authentication. Also, a CPS can tolerate data modification attacks by not single-sourcing data; specifically, critical processes should evaluate input that should be equal from multiple producers.

18. With slander or ballot stuffing attacks the adversary falsely reports unfavorable evaluations of good nodes or favorable evaluations of bad nodes. The intent is to damage the reputation of friendly nodes or inflate the reputation of colluding adversaries. A CPS can apply a voting-based IDS to both detect and tolerate slander attacks. Our system intrusion detection design addresses this attack.

All the above attacks can cause impairment security failures when the basic system functionality is severely damaged. When a bad node is actively performing attacks, however, it leaves more traces to detection and consequently eviction from the system. In Chapter 4, we discuss how our IDS techniques can detect and respond to some of these attacks.

3.5 Adversary Behavior Model

There are two aspects to adversary behavior: capture strength (to capture/compromise nodes) and insider attack behavior (to damage the system after compromise).

3.5.1 Capture Strength

We consider four capture strength models:

**Constant Capture Strength Model:** The attacker compromises each node in the system randomly with rate $\lambda_c$. This models the baseline scenario where the effectiveness of neither the defender nor the attacker changes over time. That is, the capture strength is modeled by:

$$\Lambda_c = \lambda_c \times N_g$$  \hspace{1cm} (3.1)

**Logarithmic Capture Strength Model:** The attacker increasingly takes longer time to compromise nodes in the system, following a logarithmic function curve. This models the scenario where the system has detected attackers (i.e., compromised nodes) and enhanced the defenses of the remaining nodes, making it increasingly harder for the attacker to compromise more nodes. Under this model, the compromising rate increases in logarithm form with the number of compromised nodes. That is, the capture strength is modeled by:

$$\Lambda_c = \lambda_c \times N_g \times \log_P(N_a^a + 2)$$ \hspace{1cm} (3.2)

where $P$ is the logarithm base which is a design parameter to be set based on the application scenario, $N_g$ is the number of good nodes and $N_a^a$ is the number of active attackers.
**Linear Capture Strength Model:** The attacker compromises nodes one after the other with a rate linearly related to the number of compromised nodes in the system. This applies to the case in which compromised nodes do not collude and just perform constant time attacks. Under this model, the compromising rate increases linearly with the number of compromised nodes. That is, the capture strength is modeled by:

\[ \Lambda_c = \lambda_c \times N_g \times (N_b^a + 1) \] (3.3)

**Exponential Capture Strength Model:** The attacker increasingly takes shorter time to compromise nodes in the system, following an exponential function curve. This models the scenario where the attacker learns secret information from compromised nodes in the system and exploits it to more easily compromise other nodes within a shorter time. Under this model, the compromising rate increases exponentially with the number of compromised nodes. That is, the capture strength is modeled by:

\[ \Lambda_c = \lambda_c \times N_g \times (N_b^a + 1)^P \] (3.4)

where \( P \) is the exponential attacker strength exponent which is a design parameter.

### 3.5.2 Insider Attack Behavior

Our insider attack behavior model considers four types of adversarial behavior:

- **Reckless:** A reckless attacker performs attacks with probability one (that is, whenever it has a chance); the primary objective is to cause impairment failure.

- **Random:** A random attacker performs attacks randomly with probability \( p_{\text{random}} \); hence, its attack probability \( p_a = p_{\text{random}} \). The primary objective is to evade detection. It may take a longer time for a random attacker to cause impairment failure since the attack is random. However, random attackers are hidden so it may increase the probability of Byzantine security failure once the number of bad nodes equals or exceeds one third of the node population.

- **Insidious:** An insidious attacker hides to evade detection until a critical mass of compromised nodes is reached, then they emerge from hiding to perform “all in” attacks. The primary objective is to maximize the failure probability caused by either impairment or Byzantine security failure.

- **Opportunistic:** An opportunistic attacker exploits ambient noise modeled by \( p_{\text{err}} \) (probability of mis-monitoring) to perform attacks. While a random attacker’s \( p_a \) is fixed, an opportunistic attacker decides its attack probability \( p_a \) based on \( p_{\text{err}} \) sensed. When \( p_{\text{err}} \) is higher, the system is more vulnerable, so its \( p_a \) is higher. An opportunistic attacker can be conservative or aggressive. We borrow from the demand-pricing relationship in
the field of Economics [4, 29, 33, 36, 155] to model the opportunistic attacker’s attack probability $p_a$ as a function of $p_{err}$. Specifically,

$$p_a = C \times p_{err}^\varepsilon.$$  

(3.5)

With $C > 0$, this formula covers both conservative and aggressive attack behaviors:

1. $\varepsilon = 1$: $p_a$ increases linearly with $p_{err}$; this models a conservative opportunistic attacker.

2. $\varepsilon < 1$: $p_a$ increases exponentially with $p_{err}$; this models an aggressive opportunistic attacker, the extent of which is modeled by $\varepsilon$.

### 3.6 Stuxnet Attacks

Stuxnet is an APT that security consultant VirusBlokAda discovered in June 2010 [76]. Highly sophisticated, resource-rich adversaries such as national governments or large companies that focus on a specific target distinguish APTs from other threats. This designation impacts the scope of the threat model significantly. Contrast APT with the threat imposed by a rogue employee or individual criminal with limited resources or hacker seeking only soft targets of opportunity. Security consultants Kaspersky Lab and F-Secure opine that Stuxnet is state sponsored due to its level of sophistication. An infected USB drive first introduces Stuxnet to a network. Stuxnet has two functions: propagate and damage. While the propagate (worm) function appears to be indiscriminate, 58.5% of infected computers were in Iran; Stuxnet indiscriminately copies itself to three other hosts via RPC and deletes itself on June 24, 2012. The damage function targets two types of nodes: First, it targets computers running Siemens Step-7 which comprises WinCC and Process Control System (PCS) 7 MTU software. Using four zero day attacks, including a hardcoded database password, it launches a shellcode attack on these nodes to redirect the communication library (s7otbxdx.dll) interface from the programmable logic controller to a rogue MITM library (s7otbxxs.dll). The rogue Step-7 MITM code has two functions: leapfrog the attack to programmable logic controllers and manipulate responses from the programmable logic controller to make it appear normal. After establishing a foothold on a Step-7 node, Stuxnet launches shellcode attacks on attached S7-300 programmable logic controllers that host Vacan or Fararo Paya variable frequency drives, specifically those that spin motors between 807 and 1210 Hz. The programmable logic controller shellcode first launches a Profibus-based MITM attack. Finally, the programmable logic controller shellcode launches data modification attacks: It injects motor control commands for a sequence of specific speeds (1410, 2, 1064 Hz) to disrupt their intended function. It modifies sensor data to prevent the Step-7 (MTU) from detecting the drive misbehavior; this takes away the Step-7 (MTU)’s opportunity to shut down the drives. The earliest evidence of the first Stuxnet variant appears in June 2009, the second variant in March 2010, the third variant in April 2010 and the fourth variant (Duqu) in September...
2011. While the first three variants are similar, Duqu logs keystrokes, records configuration and exfiltrates this data [130].

We use the exponential capture strength and insidious attack behavior for Stuxnet because it aggressively compromises nodes but waits until a critical mass is gathered to perform “all in” attacks to break down the system completely. The literature indicates that Stuxnet spread aggressively and attacked when it encountered a specific type of RTU.

### 3.7 Capture/Insider CPS Attacks in the Literature

In this section, we perform a literature search of existing CPS attacks and classify them in Table 3.1 based on the capture model and insider attack model used in these CPS attacks to support our capture/insider attack behavior model assumptions discussed earlier in Section 3.5.

Amin et al. [5] study a water distribution CPS. They conservatively assume that the attacker is an insider with expert knowledge of the system design. Their system model distinguishes between a regulatory control segment and a supervisory control segment. The authors classify attacks based on the seven layers of their system model: regulatory segment sensors and actuators, the uplinks joining sensors and actuators with PLCs in the regulatory segment, the regulatory segment lateral links meshing PLCs, the uplinks joining PLCs with the control segment, the control segment MTU, the control segment HMI and the corporate network. For detailed analysis, they focus on one highly specific attack model; it involves the attacker modifying the readings from a water level meter in order to steal water. Their insider attack

<table>
<thead>
<tr>
<th>Attack Case</th>
<th>Capture Strength</th>
<th>Insider Attack Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stuxnet [76, 130]</td>
<td>Exponential</td>
<td>Insidious</td>
</tr>
<tr>
<td>Water Distribution CPS [5]</td>
<td>Constant</td>
<td>Reckless</td>
</tr>
<tr>
<td>Exfiltration Attack [71]</td>
<td>Constant</td>
<td>Reckless</td>
</tr>
<tr>
<td>Petri Net Model of Coordinated Smart Grid Attackers [34]</td>
<td>Exponential</td>
<td>Insidious</td>
</tr>
<tr>
<td>Multi-Step Attack [89]</td>
<td>Linear</td>
<td>Insidious</td>
</tr>
<tr>
<td>Smart Grid HAG [65]</td>
<td>Constant</td>
<td>Reckless</td>
</tr>
<tr>
<td>MCPS Attack [13, 70, 122]</td>
<td>Constant</td>
<td>Reckless</td>
</tr>
<tr>
<td>Fifty Hz Attack [49]</td>
<td>Linear</td>
<td>Reckless</td>
</tr>
<tr>
<td>Telvent Attack [134, 135, 136]</td>
<td>Exponential</td>
<td>Insidious</td>
</tr>
<tr>
<td>UACPS Attack [143, 144]</td>
<td>Linear</td>
<td>Reckless</td>
</tr>
</tbody>
</table>
behavior is reckless because it runs for a substantial length of time: 420 min. The authors only consider lone-wolf type attacks, so we use the constant capture strength model.

Hu et al. [71] focus on data exfiltration attacks. The authors model an adversary who downloads a lot of files in a short amount of time. This is not random insider attack behavior which would spread the downloads over time. This is also not opportunistic insider attack behavior which would accomplish downloads based on periods of high noise [e.g., mass downloads may occur periodically (weekly) or based on some event (release of a new version)]. They do not mention a distributed attack, which we could model with insidious behavior. Therefore, reckless insider attack behavior is the best model for their adversary. The authors only consider lone-wolf type attacks, so we use the constant capture strength model.

Chen et al. [34] propose using Petri nets to model smart grid attackers; we were eager to read this study as it is contemporaneous with our work [102] that uses the stochastic variation of Petri nets. The authors focus on coordinated attacks so we brand their model as insidious insider attack behavior. They single out smart meters as the most vulnerable segment of the smart grid. Chen et al. believe that coordinated attacks can weaken the defense of a CPS, so we brand their model as exponential capture strength.

Liu et al. [89] model attacks with discrete stages for: probing, scanning, intrusion and the goal. The probing or reconnaissance stage amounts to creating a network picture using tools such as ping or traceroute. During the scanning stage, the attacker scans TCP and UDP ports to identify potential entry points. Next, the attacker applies exploits based on the scanning results. We claim that unless the attacker assumes a monocultural victim, another step is required to identify the application, operating system and hardware of each node identified in the probing step. This profiling should be contemporaneous with the scanning step. Finally, the adversary realizes the attack goal (e.g., copying or deleting a valuable file). The authors consider a coordinated, multiple-stage attack plan which suggests insidious insider attack behavior. Because it considers a coordinated attack, we use one of the variable capture strength models; in the absence of information regarding the attacker or defender’s change in efficacy during the attack, we use the “middle of the road” linear capture strength.

Hybrid attack graphs (HAGs) extend traditional attack graphs to allow continuous components; Louthan et al. [90] show that this is an important aspect for modeling CPSs. Hawrylak et al. [65] apply HAGs to the smart grid. The authors’ specific goal is to overheat and destroy a transformer. The transformer attack involves a single actor who does not consider ambient conditions or try to match a normal profile, so we model this attack using constant capture strength and reckless insider attack behaviors.

[13, 70, 122] consider a single actor who does not consider ambient conditions or try to match a normal profile, so we model this adversary using constant capture strength and reckless insider attack behaviors.

We model the attack in [49] using reckless insider attack behavior because its effect was
complete and apparent immediately. Because it used a botnet, we use one of the variable
capture strength models; in the absence of information regarding the attacker or defender’s
change in effectiveness during the attack, we use the “middle of the road” linear capture
strength.

[134, 135, 136] report on an attack on Telvent which planted malware and exfiltrated data.
Although the apparent goal of the Telvent attack is exfiltration, these media sources compare
its sophistication with Stuxnet. Therefore, we model this attack using exponential capture
and insidious insider attack behaviors.

The 2011 attack on the USAF’s UAV fleet [143, 144] involved malware that logged keystrokes.
The spread of the malware was limited to the control segment (i.e., HMIs and MTU) and
did not reach the remote segment (UAVs). The US Defense Department’s Host Based
Security System (HBSS) was the security appliance that detected the attack. Because it is
a coordinated attack, we use one of the variable capture strength models; in the absence of
information regarding the attacker or defender’s change in efficacy during the attack, we use
the “middle of the road” linear capture strength. Nothing indicates the malware is especially
deceptive so we model it using reckless insider attack behavior.

We sketch a classification algorithm based on statistical analysis for the system to do run-
time detection of the capture and insider attack behavior type. The classifier runs at either
the voting-based system IDS level or specification-based host IDS level. The basic idea is to
classify the attacker in two dimensions: capture and insider attack behavior. To determine
the capture strength type, the classifier analyzes detections for unique nodes over time. No
unique nodes generating detections over time suggests the attacker uses constant capture
strength. A burst followed by progressively fewer unique nodes generating detections over
time suggests the attacker uses logarithmic capture strength. If about the same number of
unique nodes generate detections over time, this suggests the attacker uses linear capture
strength. Progressively more unique nodes generating detections over time suggests the at-
tacker uses exponential capture strength. To determine the insider attack behavior type, the
classifier analyzes the detection history for given nodes over time. A node that consistently
generates detections uses reckless insider attack behavior. A node that generates detections
that are distributed uniformly over time uses random insider attack behavior. A node that
generates detections only when others do so uses insidious or opportunistic insider attack
behavior. We distinguish opportunistic from insidious insider attack behavior by correlat-
ing detections with some environmental feature (e.g., time of day, movement or channel
utilization).
Chapter 4

Intrusion Detection and Response Design

In this chapter, we discuss intrusion detection and response designs for CPSs. Specifically: the general design principles of host IDS and system IDS, the tradeoff between IDS and energy conservation and the adaptive control of detection intelligence and strength against intrusion intelligence and strength to prolong the system lifetime. Section 4.1 discusses host IDS protocol designs based on compliance degree to result in a lightweight, accurate, autonomous and adaptive protocol that runs on every node in the CPS to detect misbehavior of neighbor nodes using state-based behavior specifications. Section 4.2 discusses system IDS design to result in a robust and resilient protocol that can cope with malicious, erroneous, partly trusted, uncertain and incomplete information. Section 4.3 discusses IDS response designs to result in a highly adaptive and dynamic control protocol that can adjust detection strength in response to environment changes in attacker strength and behavior. Part of the content in this chapter is based on our publications [95, 96, 104, 106].

While our design principles work primarily at the application layer, information may be supplied from lower layers to the application layer to make intrusion detection and response decisions. We have experimented with auditing physical layer information but have focused at the application layer in the mobile group CPS, medical CPS, smart grid and unmanned aircraft system (UAS) domains. Application layer data includes attributes of the physical environment (such as position or temperature); this should not be confused with properties of the physical layer of the communications stack (such as strength or phase).

Scalability is a key concern in CPSs. Broadly, our design principle is scalable by two means: At the host IDS level, each node monitors its neighbor nodes based on simple behavior rules mapping to one or more small state machines; specifications must produce state machines that grow linearly or better with respect to network size. At the system IDS level, only a number of neighbor nodes around a suspect rather than all nodes are used to make intrusion detection decisions or supply information to an upper-level control node to make intrusion
response detections.

4.1 Host Intrusion Detection Design

All host IDS techniques for CPSs in this section are derived from one central design principle: that is, they are specification-based and state-based. Contrast this with anomaly, signature and reputation-based designs and with language-based approaches to specification-based designs. Specification-based intrusion detection is different from signature-based intrusion detection because it looks for the effect of an attack rather than the attack itself. Specification-based intrusion detection is different from anomaly-based intrusion detection because a human explicitly specifies attributes of a correctly operating system rather than a machine learning module inferring these same attributes from historical data. While a human will create sound theory-based specifications, a machine learning module may derive invalid data-driven specifications. Our hypothesis is that while the predeployment cost is higher for specification-based IDSs, it will outperform anomaly-based IDSs (it requires less resources and generates fewer false positives) and signature-based IDSs (it will have a higher detection rate for unknown attacks). This hypothesis is to be verified in this dissertation research.

Key activities in specification-based intrusion detection are: identifying critical system functions, formally describing the correct behavior of these functions and comparing collected behavior or traffic data with the specification. Specification-based designs seek to achieve low false positive rates like signature-based designs and the ability to detect novel attacks like anomaly-based designs. Our specification-based IDS design is state-based because of its small resource requirement to run on individual nodes, especially for sensors and actuators.

4.1.1 Compliance Degree

The output from the specification-based IDS design is a specification compliance degree in the range of [0, 1], based on the proportion of time a node conforms to the behavior specification. We propose to investigate two compliance degree grading strategies:

Sojourn Time Based Grading: Our IDS design for CPSs distinguishes early warnings from attacks in progress by means of a compliance degree. Contrast this with the contemporary tendency to classify behavior or traffic in a binary manner. The important problem here is to generate a grade, rather than a binary result, from a specification. The way we distinguish grades is to use a state machine approach and consider sojourn time. A CPS has many secure states; we measure the percentage of time a node is in a secure state. This maps naturally to the compliance degree which is in [0, 1]. A longer sojourn time indicates a larger compliance degree while a shorter sojourn time
indicates a smaller compliance degree. A larger compliance degree indicates an attack is less likely while a smaller compliance degree indicates an attack is more likely.

**Distance Based Grading:** In this strategy, a node’s grade is based on proximity to a known good state. This distance can be expressed discretely as the minimum number of state components that are different from a secure state; this is the Hamming distance. Also, this distance can be expressed quantitatively as the minimum total difference between each state component and the corresponding component in a secure state; this is the Euclidean or Manhattan distance. For example, consider a system with a state space of \(\langle a, b, c \rangle\) and secure states \(\langle 0, 1, 2 \rangle\) and \(\langle 4, 4, 4 \rangle\). Runtime state \(\langle 1, 4, 1 \rangle\) has a Hamming distance of 2 (two components differ between it and the second secure state), a Manhattan distance of 5 (based on its proximity to the first secure state: \(\sum \Delta x\)) and a Euclidean distance of \(\sqrt{\sum (\Delta x)^2}\). Given a pair of time-series (one trustee, one monitor), Longest Common Subsequence (LCS), Levenshtein and Damerau-Levenshtein measurements can show the distance between the two datasets. LCS distance should not be confused with longest common substring distance; a common subsequence does not need to be contiguous so extra values can appear within a common subsequence.

\[
c = \frac{\text{LCS}(\text{monitor}, \text{trustee})}{\text{time series length}}
\]

Levenshtein distance is the minimum number of edits required to transform one sequence into another; Levenshtein edits comprise insertion, deletion and substitution.

\[
c = 1 - \frac{\text{Levenshtein}(\text{monitor}, \text{trustee})}{\text{time series length}}
\]

Damerau-Levenshtein is based on Levenshtein but adds transposition of two contiguous values to the set of allowed edits.

We propose to investigate these two grading policies or combine them into one in calculating the compliance degree of a node. For sojourn time-based grading, essentially we assign a value of 1 to state \(i\) if state \(i\) is a secure state and a value of 0 if state \(i\) is an insecure state. Let \(c_{d,i}\) be compliance degree assigned to state \(i\). Then,

\[
c_{d,i} = \begin{cases} 
1 & \text{if state } i \text{ is secure} \\
0 & \text{otherwise}
\end{cases}
\]

For distance-based grading, we still assign a value of 1 to state \(i\) if state \(i\) is a secure state. However, if state \(i\) is an insecure state, we assign state \(i\) a value in \([0, 1]\) representing the distance of state \(i\) from a secure state. Therefore, \(c_{d,i}\) is assigned as follows:

\[
c_{d,i} = 1 - \frac{\text{distance}_i}{\text{maximum distance}}
\]

where distance\(_i\) is the distance between state \(i\) and the nearest secure state, and maximum distance is the longest distance between any insecure state and the nearest secure state in
the state machine. By this assignment, if state $i$ is a secure state, distance$ _i$ is zero, hence $c_{d,i} = 1$. If state $i$ is an insecure state, $c_{d,i}$ is still close to 1 if state $i$ is close to a secure state, but is close to 0 if state $i$ is close to the worst insecure state. With $c_{d,i}$ assigned, we can then calculate the compliance degree, $c_d$, of a node in state machine $s$, as follows:

$$c_d = \sum_i c_{d,i} \times (\text{sojourn time}_i / T)$$ (4.1)

where $T$ is the periodic observation interval over which $c_d$ is calculated, and sojourn time$ _i / T$ gives the proportion of time a node stays in state $i$ over the observation period $T$.

In case there are multiple state machines per suspect per corresponding node, we combine the compliance degree, $c_d$, for each state machine using the following formula:

$$\bar{c}_d = \frac{\sum_s w_s \cdot c_{d,s}}{\sum_s w_s}$$ (4.2)

where $\bar{c}_d$ is the average compliance degree measured at a particular time point, $w_s$ is the weighting factor associated with state machine $s$ which varies depending on whether the state machine input is experienced (higher weight) or observed (lower weight) and $c_{d,s}$ is the compliance degree associated with state machine $s$.

We develop two host intrusion detection techniques based on compliance degree: behavior rule specification and vector similarity specification, discussed in detail below in Sections 4.1.2 and 4.1.3, respectively. The basic idea of vector similarity specification is to compare similarity of a sequence of sensor readings, commands or votes among entities performing the same set of functions. A state machine is automatically derived from which a similarity test is performed to detect outliers. The basic idea of behavior rule specification is to specify the behavior of an entity (a sensor or an actuator) by a set of rules from which a state machine is automatically derived. Then, node misbehavior can be assessed by observing the behaviors of the node against the state machine (or behavior rules). Essentially, the states derived in a state machine would be labeled as secure versus insecure. A monitoring node then applies snooping and overhearing techniques observing the percentage of time a neighbor node is in secure states over a detection interval. A longer sojourn time in secure states indicates greater specification compliance while a shorter sojourn time indicates less specification compliance.

### 4.1.2 Vector Similarity Specification-Based Protocol Designs

Our first host IDS technique is called Vector Similarity and can be applied to a wide variety of data attacks including slandering attacks (a bad node providing bad recommendations toward a good node) and modification attacks (a bad sensor providing incorrect sensor readings).
Chapter 4. Intrusion Detection and Response Design

Under vector similarity, each node would collect a vector of values measuring the same physical entity or event. One example is that a vehicle may be equipped with up to four sensors reporting its motion. The readings from these four sensors should be more or less the same. Periodically, each sensor through overhearing would collect a vector of four values (including its own) reporting the vehicle’s motion. Another example is for a reputation system, a node may periodically receive recommendations from its neighbors regarding the reputation of a target node. In this case a node can construct a vector of recommendations (including its own) periodically and detect similarity of the recommendations in the vector to detect potential outliers.

Figure 4.1: Vector Similarity: \langle i, i \rangle are the Safe States.

Figure 4.1 illustrates potential sensor readings in integer format for convenience. States \( S \in \forall i, j | x_i = x_j \) are the safest. These drawings show a two sensor space for clarity but extend to four sensor space easily. Manhattan or Euclidean distance are appropriate for grading with this approach. Because of the low number of sensors (four, nominally) and the continuous domain of readings, using Hamming distance would result in an unacceptably high false positive rate.

4.1.2.1 Applying Vector Similarity: Positional Discontinuity

One instance of vector similarity is a positional discontinuity design. The basic idea is to track the movements of a node and assess if the node only visits secure states over time. It can detect spoofing, including Sybil, attacks. This design addresses the heterogeneity that characterizes CPSs by using a different specification for nodes based on motion profile: stationary, terrestrial movers and three dimensional movers. We implement positional discontinuity specifications with state machines.

Figure 4.2 illustrates potential spatial coordinates of a suspect. \( \langle 0, 0 \rangle \) is the safe state; it represents a position update that matches the last known location of the suspect. Figure 4.3 illustrates how to grade nodes using Positional Discontinuity Hamming, Manhattan and
Euclidean approaches, respectively. These drawings show a two dimensional space for clarity but extend to three dimensions easily to accommodate airborne, underwater or subterranean applications. Manhattan distance is more appropriate for urban and subterranean environments while Euclidean distance is more appropriate for environments with more freedom of movement.

Figure 4.4 illustrates how to grade nodes using Positional Discontinuity LCS, Levenshtein and Damerau-Levenshtein approaches, respectively. For the sake of simplicity, the values in the time series represent the index of some area in physical space in this example. More generally, this technique could be used with any coordinate system, such as: Latitude-Longitude-Altitude (LLA), Earth-Centered-Earth-Fixed (ECEF) or East-North-Up (ENU).

As an LCS example, consider a system using a three point time series with three possible values. Monitor and trustee time series have 27 possible values: 000, 001, 002, ..., 222. If we consider a monitor time series of 000 and a trustee time series of 000, $c_{LCS} = 3/3 = 1$. If we consider a monitor time series of 001 and a trustee time series of 010, $c_{LCS} = 2/3 = 0.667$. If we consider a monitor time series of 000 and a trustee time series of 111, $c_{LCS} = 0/3 = 0$.

As a Levenshtein example, consider the same system. If we consider a monitor time series of 000 and a trustee time series of 000, $c_L = 1 - 0/3 = 1$. If we consider a monitor time series of 012 and a trustee time series of 120, $c_L = 1 - 2/3 = 0.333$. If we consider a monitor time series of 000 and a trustee time series of 111, $c_L = 1 - 3/3 = 0$. Reusing the Levenshtein example to illustrate Damerau-Levenshtein, if we consider a monitor time series of 000 and a trustee time series of 000, $c_{DL} = 1 - 0/3 = 1$. If we consider a monitor time series of 012 and a trustee time series of 021, $c_{DL} = 1 - 1/3 = 0.667$. If we consider a monitor time series of 000 and a trustee time series of 111, $c_{DL} = 1 - 3/3 = 0$.

Because stationary nodes are not equipped with localization modules, we apply an analogy: we audit the layer three (IP), layer two (MAC) and layer one (port) addresses for stationary nodes. We specify that there is a fixed one to one mapping for each address tuple $\langle \text{IP}, \text{MAC}, \text{port} \rangle$. We use the Hamming distance for the stationary suspect variant. This is
because the concepts of ordinal, interval or ratio measurement do not apply. 192.168.1.100 is no “further” away from 192.168.1.101 than it is from 192.168.1.102; CA:FE:CA:FE:CA:FE is no “further” away from FE:ED:FE:ED:FE:ED than it is from BE:EF:BE:EF:BE:EF.

Each intrusion detector will grade a neighbor node within its radio range based on its position updates. During the evaluation period, the intrusion detector will record position data for the suspect. At the end of the evaluation period, the intrusion detector will calculate $c_d$ using Equation 4.1, exchange $c_d$ measurements with other intrusion detectors and calculate $\bar{c}_d$ using Equation 4.2.

**4.1.2.2 Applying Vector Similarity: Recommendation Similarity**

Another instance of vector similarity is a recommendation similarity design. The basic idea is to track the results of an intrusion detector and assess if the node shows a bias compared with other intrusion detectors. This design can detect data modification and slander attacks. Recommendation similarity addresses the federation that characterizes CPSs by critically studying intrusion detection results from sources which may not share a common charter.
and addresses the heterogeneity that characterizes CPSs by critically studying intrusion detection results from sources with dissimilar capabilities. For sensors that are responsible for providing sensor reports periodically, sensor reports can be input to this technique.

4.1.3 Behavior Rule Based Protocol Design

Our second host IDS technique is called Behavior Rule. The basic idea is to specify the behavior of an entity (a sensor or an actuator) by a set of rules from which a state machine is automatically derived. Then, $c_d$ of a node can be obtained by observing the behaviors of the node against the state machine (or behavior rules). This host IDS technique is effective toward nodes whose behaviors can be specified by a set of rules and consequently by a state machine. Here we note that Xie et al. point out that while environmental measurements from sensors is a well studied source of audit data, the control input to the actuators is less studied [152]. This design is generally applicable to both sensors and actuators.

As an example, a CPS entity “elevator” may be specified by the following rules:

- do not close door if it is blocked (light barrier is broken)
- do not move elevator if door is open
• do not move elevator if any movement would exceed cable strength due to payload
• do not open door while elevator is moving

Figure 4.5: Actuator Model.

Figure 4.5 exhaustively illustrates safe and unsafe lift states. We use Hamming distance because the concepts of ordinal, interval or ratio measurement do not apply. However, it makes sense to count the number of deviant states. A state one component away from a safe state is more healthy than a state two components away from a safe state.

4.1.4 Testing Methodology for Compliance Degree Distribution

We propose a novel testing methodology to characterize the compliance degree of a node based on the history of compliance degree collected during the system’s testing and debugging stage. During the system testing and debugging phase, the system would be tested with its anticipated operating profile from which failure data are collected. In particular, behaving and misbehaving nodes (with various attacker behaviors) would be injected into the system as input to test the IDS design effectiveness. An an example, one can apply positional discontinuity and recommendation similarity designs to the reference mobile group CPS as
follows: (a) a monitoring node periodically determines a sequence of locations of a sensor-carried mobile node within radio range through ranging and detects if the location sequence (corresponding to the state sequence) deviates from the expected location sequence; (b) a monitoring node periodically collects votes from neighbor nodes who have participated in system intrusion detection (described below) and detects dissimilarity of vote sequences among these neighbors for outlier detection.

The measurement of compliance degree of a node frequently is not perfect and can be affected by noise and unreliable wireless communication in the CPS. We model the compliance degree by a random variable $X$ with $G(\cdot) = \text{Beta}(\alpha, \beta)$ distribution [116], with the value 0 indicating that the output is totally unacceptable (zero compliance) and 1 indicating the output is totally acceptable (perfect compliance), such that the cumulative density function, $G(a)$, $0 \leq a \leq 1$, is given by

$$G(a) = \int_0^a \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha) \Gamma(\beta)} x^{\alpha-1}(1-x)^{\beta-1} \, dx$$

and the expected value of $X$ is given by

$$E_B[X] = \int_0^1 x \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha) \Gamma(\beta)} x^{\alpha-1}(1-x)^{\beta-1} \, dx = \frac{\alpha}{\alpha + \beta}$$

The $\alpha$ and $\beta$ parameters are to be estimated based on the method of maximum likelihood by using the compliance degree history collected during the system’s testing phase in which the system is tested with its anticipated attacker event profile and where the compliance degree is assessed using the specification-based host IDS technique described earlier. The compliance degree history collected in this way is the realization of a sequence of random variables $(c_1, c_2, \ldots, c_n)$ where $c_i$ is the $i^{th}$ compliance degree output observed during the testing phase, and $n$ is the total number of compliance degree outputs observed. The maximum likelihood estimates of $\alpha$ and $\beta$ are obtained by numerically solving the following equations:

$$\frac{n \partial \Gamma(\hat{\alpha}+\hat{\beta})}{\partial \hat{\alpha}} - \frac{n \partial \Gamma(\hat{\alpha})}{\partial \hat{\alpha}} + \sum_{i=1}^n \log c_i = 0$$

$$\frac{n \partial \Gamma(\hat{\alpha}+\hat{\beta})}{\partial \hat{\beta}} - \frac{n \partial \Gamma(\hat{\beta})}{\partial \hat{\beta}} + \sum_{i=1}^n \log(1 - c_i) = 0$$

where

$$\frac{\partial \Gamma(\hat{\alpha}+\hat{\beta})}{\partial \hat{\alpha}} = \int_0^\infty (\log x)x^{\hat{\alpha}+\hat{\beta}-1}e^{-x} \, dx.$$
The maximum likelihood estimate of $\beta$ is

$$\hat{\beta} = \frac{n}{\sum_{i=1}^{n} \log \left( \frac{1}{1 - c_i} \right)}$$ (4.6)

The reason we choose the Beta distribution as described above is that the domain of the Beta distribution can be viewed as a probability, so it can be used to describe the prior distribution over the probability (of a distribution) which models the node compliance degree. By applying Bayesian inference, the Beta distribution then can be used as the posterior distribution of the probability after observing sufficient instances.

### 4.1.5 Threshold-based Host $p_{fn}$ and $p_{fp}$ Assessment

Host intrusion detection is characterized by per-node false negative and false positive probabilities, denoted by $p_{fn}$ and $p_{fp}$, respectively. While many detection criteria [12, 27, 28] are possible, we consider a threshold criterion as follows: if a bad node’s compliance degree denoted by $X_b$ is higher than a system minimum compliance threshold $C_T$, then there is a false negative. Note that during the testing and debugging phase, we would plant behaving and misbehaving nodes in the system to test the IDS effectiveness, so we know exactly if a node is a bad node. Suppose that the compliance degree $X_b$ of a bad node is modeled by a $G(\cdot) = Beta(\alpha, \beta)$ distribution as described above. Then, the host IDS false negative probability $p_{fn}$ is given by:

$$p_{fn} = Pr\{X_b > C_T\} = 1 - G(C_T).$$ (4.7)

On the other hand, if a good node’s compliance degree denoted by $X_g$ is less than $C_T$, then there is a false positive. Again suppose that the compliance degree $X_g$ of a good node is modeled by a $G(\cdot) = Beta(\alpha, \beta)$ distribution. Then the host false positive probability $p_{fp}$ is given by:

$$p_{fp} = Pr\{X_g \leq C_T\} = G(C_T).$$ (4.8)

Here we observe that these two probabilities are largely affected by the setting of the minimum compliance threshold $C_T$. A large $C_T$ induces a small false negative probability at the expense of a large false positive probability. Conversely, a small $C_T$ induces a small false positive probability at the expense of a large false negative probability. A proper setting of $C_T$ in response to attacker strength detected at runtime helps maximize the system lifetime.

### 4.2 System Intrusion Detection Design

Our system IDS technique is based on majority voting of host IDS results to cope with incomplete and uncertain information available to nodes in the CPS. By limiting voters to
enclave members, this design addresses the federation characteristic of CPSs. Our system-level IDS technique involves the selection of $m$ detectors as well as the invocation interval $T_{IDS}$ to best balance energy conservation versus intrusion tolerance for achieving high reliability.

Each node periodically exchanges its routing information, location and identifier with its neighbor nodes. A coordinator is selected randomly among neighbors so that the adversaries can not target a single point of failure. We add randomness to the coordinator selection process by introducing a hashing function that takes in the identifier of a node concatenated with the current location of the node as the hash key. The node with the smallest returned hash value would then become the coordinator. Because candidate nodes know each other’s identifier and location, they can independently execute the hash function to determine which node would be the coordinator. The coordinator then selects $m$ detectors randomly (including itself), and lets all detectors know each others’ identities so that each voter can send its yes/no vote to other detectors. Vote authenticity is achieved via preloaded public keys. At the end of the voting process, all detectors will know the same result, that is, the node is diagnosed as good, or as bad based on the majority vote.

The system IDS is characterized by system false negative and false positive probabilities, denoted by $P_{fn}$ and $P_{fp}$, respectively. These two false alarm probabilities are not constant but vary dynamically, depending on the percentage of bad nodes in the system when majority voting is performed.

\[
P_{fn} = \sum_{i=0}^{m-m_a} \left[ \binom{N_b}{m_a + i} \binom{N_g}{m - (m_a + i)} \right] + \sum_{j=0}^{m-m_a} \frac{m-j}{m} \sum_{k=m_a-j}^{m} \left[ \binom{N_g}{k} (p_{fn})^k \binom{N_g - k}{m - j - k} (1 - p_{fn})^{m - j - k} \right] \frac{\binom{N_g + N_b}{m}}{m}
\]

For notational convenience, let the number of bad nodes be $N_b$, the number of good nodes be $N_g$, the number of detectors be $m$ and the majority of $m$ be $m_a$.

We explain Equation 4.9 for obtaining $P_{fn}$ in detail below. The explanation of Equation 4.10 for obtaining $P_{fp}$ follows the same logic. In Equation 4.9, $m$ is the number of voters and $m_a$ is the majority of $m$. The first summation is the special case; it aggregates the probability of a false negative stemming from selecting a majority of bad nodes. That is, it is equal to the number of ways to choose a majority of bad nodes from the set of all bad nodes times the number of ways to choose a minority of good nodes from the set of all good nodes divided by
Chapter 4. Intrusion Detection and Response Design

\[ P_{fp} = \sum_{i=0}^{m-m_a} \left[ \binom{N_b}{m_a+i} \binom{N_g}{m-(m_a+i)} \right] \left( \frac{N_g + N_b}{m} \right) + \sum_{j=0}^{m-m_a} \left[ \binom{N_b}{j} \sum_{k=m_a-j}^{m-j} \left[ \binom{N_g}{k} \left( p_{fp} \right)^k \binom{N_g-k}{m-j-k} (1-p_{fp})^{(m-j-k)} \right] \right] \left( \frac{N_g + N_b}{m} \right) \]

Equation 4.10

the number of ways to choose \( m \) nodes from the set of all good and bad nodes. The second summation is the general case; it aggregates the probability of a false negative stemming from selecting a majority of good nodes, some of which cast incorrect votes, coupled with selecting some number of bad nodes. That is, it is equal to the number of ways to choose a minority of bad nodes from the set of all bad nodes times the aggregate probability of a sufficient number of good nodes casting incorrect votes also divided by the number of ways to choose \( m \) nodes from the set of all good and bad nodes. The aggregate probability is a nested summation of the number of ways to choose a sufficient number of good nodes which cast incorrect votes and the remaining good nodes which cast correct votes. Algorithm 1 below illustrates Equation 4.9 programmatically. The algorithm for \( P_{fp} \) substitutes \( p_{fp} \) for \( p_{fn} \).

**Algorithm 1** \( P_{fn} \) Calculation Step by Step.

\[
\begin{align*}
\Sigma_0 &= 0 \\
\Sigma_1 &= 0 \\
\{ &\text{Case 1: a majority of voters are intruders} \} \\
\text{for all } i \text{ such that } 0 \leq i \leq m - N_m &\text{ do} \\
\Sigma_0 &= \Sigma_0 + C(N_b, N_m + i) \times C(N_g, m - (N_m + i)) / C(N_g + N_b, m) \\
\text{end for} \\
\{ &\text{Case 2: a minority of voters are intruders} \} \\
\text{for all } j \text{ such that } 0 \leq j \leq m - N_m &\text{ do} \\
\sigma &= 0 \\
\text{for all } k \text{ such that } N_m - j \leq k, k \leq m - j &\text{ and } k \leq N_g \text{ do} \\
\sigma &= \sigma + C(N_g, k) \times p_{fn}^k \times C(N_g - k, m - j - k) \times (1 - p_{fn})^{(m-j-k)} \\
\text{end for} \\
\Sigma_1 &= \Sigma_1 + C(N_b, j) \times \sigma / C(N_g + N_b, m) \\
\text{end for} \\
\text{return } \Sigma_0 + \Sigma_1
\end{align*}
\]
Figure 4.6 illustrates how our dynamic voting-based intrusion detection technique and our per-host intrusion detection technique interoperate.

### 4.3 Intrusion Response Design

In this section, we discuss our response protocol design. Our intrusion response protocol is particularly effective against jellyfish attacks which can be modeled with a random attacker who attacks with probability $p_a$. The basic idea is to react to malicious events detected at runtime by adjusting the minimum compliance threshold $C_T$. The effectiveness of our response to jellyfish attackers hinges on detection which relies on a rigorous and complete set of behavior rules for the subject.

When the IDS senses an increasing attack rate (e.g., if a dormant random attacker becomes active), it can increase the compliance threshold $C_T$ with the objective to prevent impairment security failure. This results in a smaller false negative probability, which has a positive effect of reducing the number of bad nodes in the system and decreasing the probability of impairment security failure. However, it also results in a larger false positive probability, which has a negative effect of reducing the number of good nodes in the system and consequently increasing the probability of Byzantine security failure. To compensate for the negative effect, the IDS increases the audit rate (decreases the intrusion detection interval) or increases the number of detectors to reduce the false positive probability at the expense of more energy consumption.

When the IDS senses a lower attack rate (e.g., if an active random attacker becomes dormant), it can decrease the compliance threshold $C_T$ with the objective to prevent Byzantine security failure. This results in a smaller false positive probability, which has a positive effect of increasing the number of good nodes in the system and decreasing the probability of Byzantine security failure.

This dissertation research aims to investigate the best way to dynamically control $C_T$. We consider three designs to counter the three attacker strength models discussed in Section 3.5.1:
Linear Dynamic $C_T$ Control: This is a simple yet efficient linear one-to-one mapping function as follows:

$$C_T(t) = C_T^p + \delta_{C_T} \times (N_a^b - 1) \quad (4.11)$$

Logarithmic Dynamic $C_T$ Control: This is a logarithmic mapping function as follows:

$$C_T(t) = C_T^p + \delta_{C_T} \times \log_P(N_a^b + 1) \quad (4.12)$$

The logarithmic mapping function adjusts $C_T$ more conservatively than the linear mapping function. This avoids false positives caused by increasing $C_T$ too quickly.

Exponential Dynamic $C_T$ Control: This is an exponential mapping function as follows:

$$C_T(t) = C_T^p + \delta_{C_T} \times (N_a^b + 1)^P \quad (4.13)$$

The exponential mapping function adjusts $C_T$ more aggressively than the linear mapping function. This avoids false negatives caused by increasing $C_T$ too slowly.

In the above formulation $C_T(t)$ refers to the $C_T$ value set at time $t$ as a response to the attacker strength measured by $N_a^b(t)$ detected at time $t$; $C_T^p$ is the minimum threshold set by the system after the testing and debugging phase for the reckless attack case; and $\delta_{C_T}$ is the increment to $C_T$ per active bad node detected. $N_a^b$ and $N_i^b$ refer to the number of “active” and “inactive” bad nodes, respectively, with $N_a^b + N_i^b = N_b$. The difference between an active bad node and an inactive bad node is that an inactive bad node behaves as if it were a good node in order to evade detection, including casting votes the same way as a good node would, when it participates in the system-level IDS voting process. In practice, we do not know $N_b$, $N_a^b$ or $N_i^b$. However, we could predict them probabilistically based on a probability model predicting the dynamics of a CPS which we will exemplify with the reference MGCP$S$ in Chapter 5.

Here we note that when $C_T$ is closer to 1, a node (good or bad) will more likely be considered as compromised even if it wanders only for a small amount of time in insecure states. A large $C_T$ induces a small host false negative probability $p_{fn}$ at the expense of a large host false positive probability $p_{fp}$.

Also we note that $C_T$ affects $p_{fn}$ and $p_{fp}$, and the relationship between the minimum compliance threshold $C_T$ set versus $p_{fn}$ and $p_{fp}$ may be determined at design time by means of model-based analysis to be introduced in Chapters 5, 6, 7 and 8. The ultimate goal is to use and apply optimal IDS settings identified, i.e., adjusting $C_T$ in this case, in response to attacker strength detected at runtime to maximize the CPS lifetime.
Chapter 5

Case Study: Mobile Group CPS

This chapter demonstrates the working of a system IDS design for the mobile group CPS. We develop a probability model based on stochastic Petri nets [118, 31, 30] to describe the behavior of the CPS in the presence of both malicious nodes exhibiting a range of attacker behaviors and an IDS for detecting and responding to malicious events at runtime. Our results indicate that adjusting detection and response strength in response to attacker strength and behavior detected can significantly improve the reliability of the CPS. We report numerical data for a CPS subject to various capture strength and insider attack behaviors with physical interpretations given. The content of this chapter derives from work we published in [103].

5.1 MGCPS Reference Model

Our reference MGCPS model is based on the infrastructure described in [142] comprising at the sensor layer 128 sensor-carried mobile nodes. Each node ranges its neighbors periodically. Each node uses its sensor to measure any detectable phenomena nearby. Each node transmits a CDMA waveform. Neighbors receiving that waveform transform the timing of the PN code (1023 symbols) and RF carrier (915 MHz) into distance. Each node performs sensing and reporting functions to provide information to upper layer control devices to control and protect the infrastructure and in addition uses its ranging function for node localization and intrusion detection. Specifically, the ranging algorithm used by our reference MGCPS for localization has four key functions:

1. process data from inertial and barometric sensors for navigation and multipath mitigation;

2. calculate phase shift between code and carrier of a CDMA waveform to range;
3. distinguish multipath reflections from line of sight in RF input to improve ranging;

4. blend inertial, barometric and range inputs using a Kalman filter.

The reference model is a special case of a single-enclave system with homogeneous nodes. The
IDS functionality is distributed to all nodes in the system for intrusion/fault tolerance. On
top of the sensor-carried mobile nodes sits an enclave control node responsible for setting sys-
tem parameters in response to dynamically changing conditions such as changes of attacker
strength. The control module is assumed fault/intrusion free through security/hardware
protection mechanisms against capture attacks/hardware failure.

Figure 5.1 contextualizes our reference MGCPS which comprises 128 sensor-carried mobile
nodes, a control unit and physical objects for controlling and protecting a physical infra-
structure. The mobile nodes are capable of sensing physical environments as well as actuating
and controlling the underlying physical objects in the MGCPS. They function as sensors and
actuators, each carrying sensors for sensing physical phenomena, as well as actuating devices
for controlling physical objects. The CPS literature identifies these mobile nodes as RTUs.
Sitting on top of these mobile nodes is a control unit which receives sensing data from the
mobile nodes and determines actions to be performed by individual nodes or a group of mo-
bile nodes. This triggers their actuating devices to control and protect the physical objects
in the CPS. We exemplify a number of applications to which our reference CPS can apply,
including:

1. disaster recovery (say after an earthquake) in which a group of mobile nodes with
motion and video sensing and actuating capability cooperate under the control of a
disaster corrective control unit to protect and recover physical objects (e.g., people or
a physical infrastructure);

2. emergency rescue (say a burning building) in which a group of mobile fighters equipped
with motion and video sensing and fighting capability cooperate under the control of
a control unit to rescue physical objects (e.g., people trapped or seized);
3. military patrol (combat or reconnaissance) [38] in which a group of mobile patrol nodes equipped with motion sensing and fighting capability cooperate under the control of a control unit to protect and control physical objects (e.g., geographic areas or critical resources);

4. pervasive healthcare [97] in which a group of mobile medical personnel equipped with motion and video sensing and actuating capability cooperate under the control of a control unit to protect and provide healthcare to physical objects (e.g., patients or medical devices);

5. unmanned aircraft systems [99] in which a group of unmanned aerial vehicles equipped with sensing and aircraft fighting capability cooperate under the control of a remote control unit to control and protect physical objects (e.g., geographic areas).

The control unit contains control logic and provides management services. The CPS literature identifies this control unit as an MTU. In contrast with the RTUs, an MTU implements the broad strategic control functions. Our reference CPS is distinct from Wireless Sensor Networks (WSNs); WSNs are resource constrained, mostly stationary and have a specific traffic profile. On the other hand, our reference CPS is safety-critical, mobile and uses ad hoc networking with bidirectional flows. We do not make any assumptions regarding the network structure used to connect nodes in a CPS. In our reference CPS, nodes are mobile, and they are connected through wireless link to the control node.

5.2 MGCPS Intrusion Detection Design

Our host intrusion detection protocol design is based on two core techniques, behavior rule specification and vector similarity specification, as described in Section 4.1. The end product is $p_{fn}$ and $p_{fp}$ as computed from Equations 4.7 and 4.8. Our system IDS technique is based on majority voting of host IDS results to cope with incomplete and uncertain information available to nodes in the CPS. The end product is $P_{fn}$ and $P_{fp}$ as computed from Equations 4.9 and 4.10. Our dynamic IDS response design is to adjust the compliance degree threshold, $C_T$, in response to the attacker strength detected with the goal to maximize the system lifetime, as described in Section 4.3.

5.3 Model and Analysis

Table 5.1 lists the set of parameters used in our model-based analysis of intrusion detection and response designs. The parameter $N$ defines the starting network size (i.e., the number of nodes). The hostility of the network is characterized by $\lambda_c$; $m$ is the number of detectors used in the system IDS.
Table 5.1: Parameters Used for Analysis of Intrusion Detection and Response Design

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Meaning</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N$</td>
<td>Number of nodes in a CPS</td>
<td>Input</td>
</tr>
<tr>
<td>$\lambda_c$</td>
<td>Per node compromise rate (Hz)</td>
<td>Input</td>
</tr>
<tr>
<td>$p_{FP}$</td>
<td>Probability of per-host IDS false positive</td>
<td>Input</td>
</tr>
<tr>
<td>$p_{FN}$</td>
<td>Probability of per-host IDS false negative</td>
<td>Input</td>
</tr>
<tr>
<td>$T_{IDS}$</td>
<td>Intrusion detection interval (s)</td>
<td>Input</td>
</tr>
<tr>
<td>$m$</td>
<td>Number of detectors in the system IDS</td>
<td>Input</td>
</tr>
<tr>
<td>$P_{FP}$</td>
<td>Probability of system IDS false positive</td>
<td>Derived</td>
</tr>
<tr>
<td>$P_{FN}$</td>
<td>Probability of system IDS false negative</td>
<td>Derived</td>
</tr>
<tr>
<td>$p_{random}$</td>
<td>Random attack probability by a random attacker</td>
<td>Input</td>
</tr>
<tr>
<td>$p_a$</td>
<td>Attack probability by an insidious attacker</td>
<td>Derived</td>
</tr>
<tr>
<td>$N_{IDS}$</td>
<td>Maximum IDS cycles before energy exhaustion</td>
<td>Derived</td>
</tr>
<tr>
<td>MTTF</td>
<td>System lifetime</td>
<td>Output</td>
</tr>
</tbody>
</table>

Our theoretical model uses Stochastic Petri Net (SPN) techniques [40]. Dalton et al. [42] previously studied using SPNs to model cyber attacks. Figure 5.2 shows the SPN model describing the ecosystem of a CPS with intrusion detection and response under capture, impairment and Byzantine security attacks. The underlying model of the SPN model is a continuous-time semi-Markov process with a state representation ($N_g$, $N_b$, $N_e$, impaired, energy) where $N_g$ is the number of good nodes, $N_b$ is the number of bad nodes, $N_e$ is the number of nodes evicted (as they are considered as bad nodes by intrusion detection), impaired is a binary variable with 1 indicating impairment security failure, and energy is a binary variable with 1 indicating energy availability and 0 indicating energy exhaustion.

![Figure 5.2: SPN Model for Intrusion Detection and Response.](image-url)
Table 5.2: Transition Rates of the SPN Model.

<table>
<thead>
<tr>
<th>Transition Name</th>
<th>Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>TENERGY</td>
<td>$1/(N_{IDS} \times T_{IDS})$</td>
</tr>
<tr>
<td>TCP</td>
<td>$N_g \times \lambda_c$</td>
</tr>
<tr>
<td>TFP</td>
<td>$N_g \times P_{fp}/T_{IDS}$</td>
</tr>
<tr>
<td>TIDS</td>
<td>$N_b \times (1 - P_{fn})/T_{IDS}$</td>
</tr>
<tr>
<td>TIF</td>
<td>$p_a \times N_b \times \lambda_i$</td>
</tr>
</tbody>
</table>

- We use places to hold tokens each representing a node. Initially, all $N$ nodes are good nodes (e.g., 128 in our reference CPS) and put in place $N_g$ as tokens.

- We use transitions to model events. Specifically, TCP models good nodes being compromised; TFP models a good node being falsely identified as compromised; TIDS models a bad node being detected correctly.

- Good nodes may become compromised because of capture attacks with rate $\lambda_c$. This is modeled by associating transition TCP with an aggregate rate $\lambda_c \times N_g$. Firing TCP will move tokens one at a time (if it exists) from place $N_g$ to place $N_b$. Tokens in place $N_b$ represent bad nodes performing impairment attacks with probability $p_a$.

- When a bad node is detected by the system IDS as compromised, the number of compromised nodes evicted will be incremented by 1, so place $N_e$ will hold one more token. On the other hand, the number of undetected compromised nodes will be decremented by 1, i.e., place $N_b$ will hold one less token. These detection events are modeled by associating transition TIDS with a rate of $N_b \times (1 - P_{fn})/T_{IDS}$ with $1 - P_{fn}$ accounting for the system IDS true positive probability.

- The system-level IDS can incorrectly identify a good node as compromised. This is modeled by moving a good node in place $N_g$ to place $N_e$ from firing transition TFP with a rate of $N_g \times P_{fp}/T_{IDS}$ with $P_{fp}$ accounting for the system IDS false positive probability.

- The system energy is exhausted after time $N_{IDS} \times T_{IDS}$ where $N_{IDS}$ is the maximum number of intrusion detection intervals the CPS can possibly perform before it exhausts its energy due to performing ranging, sensing and intrusion detection functions. It can be estimated by considering the amount of energy consumed in each $T_{IDS}$ interval. This energy exhaustion event is modeled by placing a token in place $energy$ initially and firing transition TENERGY with rate $1/(N_{IDS} \times T_{IDS})$. When the energy exhaustion event occurs, the token in place $energy$ will be vanished and the system enters an absorbing state meaning the lifetime is over. This is modeled by disabling all transitions in the SPN model.
• When the number of bad nodes (i.e., tokens in place $N_b$) is at least 1/3 of the total number of nodes (tokens in place $N_g$ and $N_b$), the system fails because of a Byzantine failure. The system lifetime is over and is modeled again by disabling all transitions in the SPN model.

• Bad nodes in place $N_b$ perform attacks with probability $p_a$ and cause impairment to the system. After an impairment-failure time period is elapsed, heavy impairment due to attacks will result in a security failure. We model this by firing transition TIF with a rate of $p_a \times N_b \times \lambda_{if}$ indicating the amount of time needed by $p_a \times N_b$ bad nodes to reach this level of impairment beyond which the system cannot sustain the damage. The value of $\lambda_{if}$ is system specific and is determined by domain experts. A token is flown into place impaired when such a security failure occurs. Once a token is in place impaired, the system enters an absorbing state meaning the lifetime is over. Again it is modeled by disabling all transitions in the SPN model.

Here we note that the last two bullet points cover the two conditions that would cause a security failure.

We use the SPN model to analyze two design tradeoffs:

• Detection strength versus energy consumption: As we increase the detection frequency (a smaller $T_{IDS}$) or the number of detectors (a larger $m$), the detection strength increases, thus preventing the system from running into a security failure. However, this increases the rate at which energy is consumed, thus resulting in a shorter lifetime. Consequently, there is an optimal setting of $T_{IDS}$ and $m$ under which the system MTTF is maximized, given the node capture strength and attack model.

• Detection response versus attacker strength: As the random attack probability $p_a$ decreases, the attacker strength decreases, thus lowering the probability of security failure due to impairment attacks. However, compromised nodes become more hidden and difficult to detect because they leave less evidence traceable, resulting in a higher per-host false negative probability $p_{fn}$ and consequently a higher system-level false negative probability $P_{fn}$. This increases the probability of security failure due to Byzantine attacks. The system can respond to instantaneous attacker strength detected and adjust $C_T$ to trade a high per-host false positive probability $p_{fp}$ off for a low per-host false negative probability $p_{fn}$, or vice versa, so as to minimize the probability of security failure. Hence, there exists an optimal setting of $C_T$ as a function of attacker strength detected at time $t$ under which the system security failure probability is minimized.

Let $L$ be a binary random variable denoting the life of the system such that it takes on the value of 1 if the system is alive at time $t$ and 0 otherwise. Then, the expected value of $L$ is the reliability of the system $R(t)$ at time $t$. Consequently, the integration of $R(t)$ from $t = 0$ to $\infty$ gives the mean time to failure (MTTF) or the average lifetime of the system we aim
to maximize. The binary value assignment to \( L \) can be done by means of a reward function assigning a reward \( r_i \) of 0 or 1 to state \( i \) at time \( t \) as follows:

\[
r_i = \begin{cases} 
1 & \text{if system is alive in state } i \\
0 & \text{if system fails due to security or energy failure}
\end{cases}
\]

A state is represented by the distribution of tokens to places in the SPN model. For example, with the SPN model defined in Figure 5.2, the underlying state is represented by \((N_g, N_b, N_e, impaired, energy)\). When place \( energy \) contains zero tokens, it indicates energy exhaustion. When \( N_g \) is less than or equal to twice of \( N_b \), it indicates a Byzantine failure. When place \( impaired \) contains a token, it indicates a security failure due to significant functional impairment. Once the binary value of 0 or 1 is assigned to all states of the system as described above, the reliability of the system \( R(t) \) is the expected value of \( L \) weighted on the probability of the system stays at a particular state at time \( t \), which we can obtain easily from solving the SPN model using SPNP [40]. The MTTF of the system is equal to the cumulative reward to absorption, i.e.,

\[
MTTF = \int_0^\infty R(t)dt 
\]  

(5.1)

which we can again compute easily using SPNP.

Figure 5.3: The Underlying Semi-Markov Model of the SPN Model.

Figure 5.3 shows the corresponding semi-Markov model of the SPN model for the case in which the number of nodes is 128 in our reference MGCPS. Note that only a part of the underlying semi-Markov model is shown in Figure 5.3 to illustrate the concept of states and state transitions. The SPN model is constructed as follows:

- We use places to hold tokens each representing a node. Initially, all \( n \) nodes are good nodes (e.g., 128 in our reference MGCPS) and put in place \( N_g \) as tokens. The initial
state thus is \((128, 0, 0, 1, 0)\) in the underlying semi-Markov model shown in Figure 5.3.

- We use transitions to model events. Specifically, TCP models good nodes being compromised; TFP models a good node being falsely identified as compromised; TIDS models a bad node being detected correctly; TIF models a bad node impairing the MGCPS.

- Good nodes may become compromised because of insider attacks with per-node compromising rate \(\lambda_c\). This is modeled by associating transition TCP with an aggregate rate \(\lambda_c \times N_g\). Firing TCP will move tokens one at a time (if it exists) from place \(N_g\) to place \(N_b\). Tokens in place \(N_b\) represent compromised but undetected nodes. For example, if in state \((128, 0, 0, 1, 0)\) a good node is compromised, a token will flow from \(N_g\) to \(N_b\) and the resulting state is \((127, 1, 0, 1, 0)\).

- When a bad node is detected as compromised, the number of evicted nodes will be incremented by 1, so place \(N_e\) will hold one more token. On the other hand, the number of undetected compromised nodes will be decremented by 1, i.e., place \(N_b\) will hold one less token. This event is modeled by associating transition TIDS with a rate of \(N_b \times (1 - P_{fn})/T_{IDS}\) accounting for the false negative probability of voting-based intrusion detection. For example, if in state \((127, 1, 0, 1, 0)\) a bad node is detected and evicted, a token will flow from \(N_b\) to \(N_e\) and the resulting state is \((127, 0, 1, 1, 0)\).

- Voting-based intrusion detection can also incorrectly identify a good node as compromised. This is modeled by moving a good node in place \(N_g\) to place \(N_e\) from firing transition TFP with a rate of \(N_g \times P_{fp}/T_{IDS}\) accounting for the false positive probability of voting-based intrusion detection. For example, if in state \((127, 1, 0, 1, 0)\) a good node is misdiagnosed as a bad node and evicted, a token will flow from \(N_g\) to \(N_e\) and the resulting state is \((126, 1, 1, 1, 0)\).

- The system energy is exhausted after \(N_{IDS} \times T_{IDS}\) intervals where \(N_{IDS}\) is the maximum number of intrusion detection intervals the MGCPS can possibly perform before it exhausts its energy due to performing ranging, sensing and intrusion detection functions. It can be estimated by considering the amount of energy consumed in each \(T_{IDS}\) interval. This energy exhaustion event is modeled by placing a token in place energy initially and firing transition TENERGY with rate \(1/(N_{IDS} \times T_{IDS})\). When the energy exhaustion event occurs, the token in place energy will be vanished. For example, if in state \((128, 0, 0, 1, 0)\) the energy exhaustion event occurs, the token in place energy will be vanished and the resulting state is \((128, 0, 0, 0, 0)\). We use a dashed-line arrow in Figure 5.3 to indicate the energy exhaustion transition. Note that the energy exhaustion event can possibly occur in any state in which the penultimate state component’s value is 1, that is, when energy is still available. To avoid clutter, we pick only 4 states in Figure 5.3 to illustrate the energy exhaustion transition.
• The MGCPS receives critical damage from bad nodes with per-node impairment rate $\lambda_{if}$. This is modeled by associating transition TIF with an aggregate rate $N_b \times p_a \times \lambda_{if}$. Firing TIF will create a token in place \textit{impaired}. For example, if in state (127, 1, 0, 1, 0) the bad node critically damages the MGCPS, a token will appear in \textit{impaired} and the resulting state is (127, 1, 0, 1, 1).

## 5.4 Parameterization

Table 5.3: Parameters and their Values for the Reference CPS.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Meaning</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N$</td>
<td>Number of nodes or network size</td>
<td>128</td>
</tr>
<tr>
<td>$\bar{n}$</td>
<td>Number of neighbors within radio range</td>
<td>32</td>
</tr>
<tr>
<td>$p_{fn}$</td>
<td>Per-host false negative probability</td>
<td>$1 - 20%$</td>
</tr>
<tr>
<td>$p_{fp}$</td>
<td>Per-host false positive probability</td>
<td>$1 - 20%$</td>
</tr>
<tr>
<td>$\lambda_c$</td>
<td>Per-node capture strength</td>
<td>$1/(1 \text{ hr}) - 1/(24 \text{ hr})$</td>
</tr>
<tr>
<td>$T_{IDS}$</td>
<td>Intrusion detection interval</td>
<td>$1 - 60 \text{ min}$</td>
</tr>
<tr>
<td>$m$</td>
<td>Number of intrusion detectors per node</td>
<td>[3, 11]</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>Number of ranging operations</td>
<td>5</td>
</tr>
<tr>
<td>$E_t$</td>
<td>Energy for transmission per node</td>
<td>125 $\mu$J</td>
</tr>
<tr>
<td>$E_r$</td>
<td>Energy for reception per node</td>
<td>50 $\mu$J</td>
</tr>
<tr>
<td>$E_a$</td>
<td>Energy for analyzing data per node</td>
<td>1740 $\mu$J</td>
</tr>
<tr>
<td>$E_s$</td>
<td>Energy for sensing per node</td>
<td>500 $\mu$J</td>
</tr>
<tr>
<td>$E_o$</td>
<td>Initial system energy</td>
<td>16.128 MJ</td>
</tr>
</tbody>
</table>

We consider the reference CPS model introduced in Section 5.1 operating in a $2 \times 2$ area with a network size ($N$) of 128 nodes. Hence, the number of neighbors within radio range, denoted by $\bar{n}$, initially is about $128/4 = 32$ nodes. Our IDS design is based on local monitoring, so it can be generically applied to any network structure. A node in our reference CPS uses a 35 Wh battery, so its energy is 126 kJ. The system energy initially, denoted by $E_o$, is therefore $126 \text{ kJ} \times 128 = 16.128 \text{ MJ}$. Table 5.3 lists the set of parameters and their values for the reference CPS.

### 5.4.1 System-Level IDS $P_{fn}$ and $P_{fp}$

We first parameterize the system IDS $P_{fn}$ and $P_{fp}$ given per-host IDS false negative probability $p_{fn}$ and per-host IDS false positive probability $p_{fp}$ as input. We first note that $P_{fn}$ and $P_{fp}$ highly depend on the attacker behavior. A reckless attacker constantly performs slandering
attacks such that it will vote a bad node as a good node, and conversely a good node as a bad node, to eventually cause a security failure. However, a random or an insidious attacker will only perform slandering attacks randomly with probability \( p_a \) to avoid detection.

We first differentiate the number of active bad nodes, \( N_b^a \), from the number of inactive bad nodes, \( N_b^i \), with \( N_b^a + N_b^i = N_b \), such that at any time:

\[
N_b^a = p_a \times N_b \tag{5.2}
\]

\[
N_b^i = (1 - p_a) \times N_b \tag{5.3}
\]

The difference between an active bad node and an inactive bad node is that an inactive bad node behaves as if it were a good node in order to evade detection, including casting votes the same way as a good node would, when it participates in the system-level IDS voting process.

\[
P_{fn} = \sum_{i=0}^{m-m_a} \left[ \binom{N_b^a}{m_a + i} \binom{N_b^i}{m - (m_a + i)} \right] + \sum_{j=0}^{m-m_a} \left[ \binom{N_b^a}{j} \sum_{k=m_a-j}^{m-j} \binom{N_b^a + N_b^i}{k} (p_{fn})^k \left( \binom{N_g + N_b^i - k}{m - j - k} (1 - p_{fn})^{m-j-k} \right) \right] \quad (5.4)
\]

For a reckless attacker, \( p_a = 1 \). For a random attacker, \( p_a = p_{random} \). For an insidious attacker, to maximize the benefit of colluding attacks, a compromised node stays dormant until a critical mass of compromised nodes is gathered so that \( p_a = 1 \) when \( N_b \geq N_b^T \) and \( p_a = 0 \) otherwise, where \( N_b^T \) is a parameter reflecting the insidiousness degree. In other words, all bad nodes engage in active attacks when there is a critical mass of compromised nodes in the system.

We calculate \( P_{fn} \) by Equation 5.4. This derives from Equation 4.9 to account for the presence of active and inactive attackers due to attacker behavior type. The equation for \( P_{fp} \) is the same except replacing \( p_{fn} \) by \( p_{fp} \) in the right hand side expression.

We explain Equation 5.4 for obtaining \( P_{fn} \) in detail below. The explanation for \( P_{fp} \) follows the same logic. In Equation 5.4, \( m \) this is the number of detectors and \( m_a \) is the majority of \( m \). The first summation aggregates the probability of a false negative stemming from selecting a majority of active bad nodes. That is, it is equal to the number of ways to choose a majority of \( m \) nodes from the set of active bad nodes times the number of ways to choose
Chapter 5. Case Study: Mobile Group CPS

a minority of $m$ nodes from the set of good nodes and inactive bad nodes divided by the number of ways to choose $m$ nodes from the set of all good and bad nodes. The second summation aggregates the probability of a false negative stemming from selecting a minority of $m$ nodes from the set of active bad nodes which always cast incorrect votes, coupled with selecting a sufficient number of nodes from the set of good nodes and inactive bad nodes which make incorrect votes with probability $p_{fn}$, resulting in a majority of incorrect votes being cast.

5.4.2 Host IDS $p_{fn}$ and $p_{fp}$

Next, we parameterize the host IDS false negative probability $p_{fn}$ and false positive probability $p_{fp}$ for reckless, random, insidious and opportunistic attacks. The system, after a thorough testing and debugging phase, determines a minimum threshold $C_T$ such that $p_{fn}$ and $p_{fp}$ measured based on Equations 4.7 and 4.8 are acceptable to system design. Let $p_{fn}^k$ and $p_{fp}^k$ be the false negative probability and the false positive probability of the host IDS when $p_a = 1$ (e.g., under reckless attackers). Let the minimum threshold $C_T$ value set for the reckless attack case be denoted by $C_T^p$.

Let $p_{fn}^a$ and $p_{fp}^a$ be the false negative and positive probabilities of the host IDS when $p_a < 1$ (e.g., under random attackers). For the case of random attackers with probability $p_a < 1$, conceivably the amount of evidence observable from a bad node would be diminished proportional to $p_a$. Consequently, with the same minimum threshold $C_T^p$ being used, the host false negative probability would increase. We again use Equations 4.7 and 4.8 to obtain $p_{fn}^a$ and $p_{fp}^a$ for each given $p_a$ value during the testing and debugging phase. Here we note that the host false positive probability would remain the same, i.e., $p_{fp}^a = p_{fp}^k$ because the attacker behavior does not affect false positives, given the same minimum threshold $C_T^p$ being used.

Next, let $p_{fn}^i$ and $p_{fp}^i$ be the false negative and positive probabilities of the host IDS under insidious attackers. Obviously, the false positive probability is not affected, so $p_{fp}^i = p_{fp}^k$. Since insidious nodes stay dormant until a critical mass is achieved to perform “all in” attacks, the false negative probability is one during the dormant period and is equal to that under reckless attacks during the “all in” attack period. Specifically,

$$p_{fn}^i = \begin{cases} p_{fn}^k & \text{if } N_b \geq N_b^T \\ 1 & \text{otherwise} \end{cases} \quad (5.5)$$

Lastly, let $p_{fn}^o$ and $p_{fp}^o$ be the false negative and positive probabilities of the host IDS under opportunistic attackers. As with the other attacker types, the false positive probability is not affected, so $p_{fp}^o = p_{fp}^k$. The probability that an opportunistic node is active is a function of $p_{err}$ described by Equation 3.5. We choose $\varepsilon = 0.9$ in order to cause $p_a$ to increase exponentially with $p_{err}$ because this models an aggressive opportunistic attacker. We again use Equations 4.7 and 4.8 to obtain $p_{fn}^o$ and $p_{fp}^o$ for each given $p_a$ value during the testing and debugging phase.
Table 5.4: $\beta$ in Beta(1,$\beta$) and Resulting $p_{fn}$ and $p_{fp}$ Values under Various Attack Models.

<table>
<thead>
<tr>
<th>Attack Type</th>
<th>$\beta$</th>
<th>$p_{fn}$</th>
<th>$p_{fp}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random with $p_a = 1.000$ (reckless)</td>
<td>1.20</td>
<td>6.3%</td>
<td>7.3%</td>
</tr>
<tr>
<td>Random with $p_a = 0.800$</td>
<td>1.00</td>
<td>10.0%</td>
<td>7.3%</td>
</tr>
<tr>
<td>Random with $p_a = 0.400$</td>
<td>0.75</td>
<td>17.8%</td>
<td>7.3%</td>
</tr>
<tr>
<td>Random with $p_a = 0.200$</td>
<td>0.50</td>
<td>31.6%</td>
<td>7.3%</td>
</tr>
<tr>
<td>Random with $p_a = 0.100$</td>
<td>0.20</td>
<td>63.1%</td>
<td>7.3%</td>
</tr>
<tr>
<td>Random with $p_a = 0.050$</td>
<td>0.13</td>
<td>74.1%</td>
<td>7.3%</td>
</tr>
<tr>
<td>Random with $p_a = 0.025$</td>
<td>0.09</td>
<td>81.3%</td>
<td>7.3%</td>
</tr>
<tr>
<td>Insidious</td>
<td>0; 1.20</td>
<td>100%; 6.3%</td>
<td>7.3%</td>
</tr>
<tr>
<td>Opportunistic</td>
<td>0.35</td>
<td>44.7%</td>
<td>7.3%</td>
</tr>
</tbody>
</table>

Here we note that $p_{fn}$ and $p_{fp}$ obtained above for reckless, random, insidious or opportunistic attackers would be a function of time as input to Equation 5.4 for calculating system-level IDS $P_{fn}$ and $P_{fp}$ dynamically.

We apply the statistical analysis described by Equations 4.3 through 4.6 to get the maximum likelihood estimates of $\beta$ (with $\alpha$ set as 1) under each attacker behavior model and then use Equations 4.7 and 4.8 to yield $p_{fn}$ and $p_{fp}$. The system minimum threshold $C_T$ is set to $C_T^p = 0.9$ to yield $p_{fn}^k = 6.3\%$ and $p_{fp}^k = 7.3\%$. Table 5.4 summarizes $\beta$ values and the resulting $p_{fn}$ and $p_{fp}$ values under various attacker behavior models. The reckless attack model is a special case in which $p_a = 1$. The insidious attack model is another special case in which $p_a = 1$ during the “all in” attack period and $p_a = 0$ during the dormant period.

### 5.4.3 Parameterizing $C_T$ for Dynamic Intrusion Response

The parameterization of $p_{fn}$ and $p_{fp}$ above is based on a constant $C_T$ being used (i.e., $C_T^p = 0.9$). A dynamic IDS response design is to adjust $C_T$ in response to the attacker strength detected with the goal to maximize the system lifetime. The attacker strength of a node, say node $i$, may be estimated periodically by node $i$’s intrusion detectors. That is, the compliance degree value of node $i$, $X_i(t)$, as collected by $m$ intrusion detectors based on observations collected during $[t - T_{IDS}, t]$, is compared against the minimum threshold $C_T^p$ set for reckless attackers. If $X_i(t) < C_T^p$, then node $i$ is considered a bad node performing active attacks at time $t$; otherwise it is a good node. This information is passed to the control module who subsequently estimates $N_a^p(t)$ representing the attacker strength at time $t$.

In this dissertation research, we aim to identify the best IDS response design in response to attacker strength detected at runtime. The basic idea is to decrease the per-host false negative probability $p_{fn}$ when the attacker strength is high, so we may quickly remove active
attacker from the system to prevent impairment failure. This is achieved by increasing the $C_T$ value. Conversely, when there is little attacker evidence detected, we lower $C_T$ so we may quickly decrease the probability of a good node being misidentified as a bad node, i.e., lowering the per-host false positive probability, to prevent Byzantine failure.

While there are many possible ways to dynamically control $C_T$, we consider the three dynamic $C_T$ control functions described in Equations 4.11, 4.12 and 4.13 where $C_T(t)$ refers to the $C_T$ value set at time $t$ as a response to the attacker strength measured by $N^a_b(t)$ detected at time $t$; $C_T^p$ is the minimum threshold set by the system for the reckless attack case; and $\delta C_T$ is the increment to $C_T$ per active bad node detected. For example, when $C_T^p = 0.9$, $\delta C_T = 0.05$, $P = 10$ for logarithmic dynamic $C_T$ control and $P = 0.1$ for exponential dynamic $C_T$ control, we have:

**Linear Dynamic $C_T$ Control:**

$$C_T(t) = \begin{cases} 0.85000 & \text{if } N^a_b = 0 \\ 0.90000 & \text{if } N^a_b = 1 \\ 0.95000 & \text{if } N^a_b = 2 \\ 0.99000 & \text{if } N^a_b \geq 3 \end{cases} \quad (5.6)$$

**Logarithmic Dynamic $C_T$ Control:**

$$C_T(t) = \begin{cases} 0.90000 & \text{if } N^a_b = 0 \\ 0.91505 & \text{if } N^a_b = 1 \\ 0.92386 & \text{if } N^a_b = 2 \\ 0.93010 & \text{if } N^a_b = 3 \end{cases} \quad (5.7)$$

**Exponential Dynamic $C_T$ Control:**

$$C_T(t) = \begin{cases} 0.95000 & \text{if } N^a_b = 0 \\ 0.95359 & \text{if } N^a_b = 1 \\ 0.95581 & \text{if } N^a_b = 2 \\ 0.95744 & \text{if } N^a_b = 3 \end{cases} \quad (5.8)$$

Note that when $C_T$ is closer to 1, a node will more likely be considered as compromised even if it wanders only for a small amount of time in insecure states. A large $C_T$ induces a small per-host false negative probability $p_{fn}$ at the expense of a large per-host false positive probability $p_{fp}$. 
5.4.4 Energy

Lastly, we parameterize \( N_{\text{IDS}} \), the maximum number of intrusion detection cycles the system can possibly perform before energy exhaustion, as follows:

\[
N_{\text{IDS}} = \frac{E_o}{E_{T_{\text{IDS}}}}
\]  

(5.9)

where \( E_o \) is the initial energy of the reference CPS and \( E_{T_{\text{IDS}}} \) is the energy consumed per \( T_{\text{IDS}} \) interval due to ranging, sensing and intrusion detection functions, calculated as:

\[
E_{T_{\text{IDS}}} = n \times (E_{\text{ranging}} + E_{\text{sensing}} + E_{\text{detection}})
\]  

(5.10)

where \( E_{\text{ranging}}, E_{\text{sensing}} \) and \( E_{\text{detection}} \) stand for energy spent for ranging, sensing and intrusion detection in a \( T_{\text{IDS}} \) interval, respectively. Here the energy spent per node is multiplied with the node population in the CPS to get the total energy spent by all nodes per cycle.

In Equation 5.10, \( E_{\text{ranging}} \) stands for the energy spent for periodic ranging. It is calculated as:

\[
E_{\text{ranging}} = \alpha \times [E_t + \bar{n} \times (E_r + E_a)]
\]  

(5.11)

Here a node spends \( E_t \) energy to transmit a CDMA waveform. Its \( \bar{n} \) neighbors each spends \( E_r \) energy to receive the waveform, and each spends \( E_a \) energy to transform it into distance. This operation is repeated for \( \alpha \) times for determining a sequence of locations. In Equation 5.10, \( E_{\text{sensing}} \) stands for the amount of energy consumed due to periodic sensing. It is computed as:

\[
E_{\text{sensing}} = \bar{n} \times (E_s + E_a).
\]  

(5.12)

Here a node spends \( E_s \) energy for sensing navigation and multipath mitigation data and \( E_a \) energy for analyzing sensed data for each of its \( \bar{n} \) neighbors. Finally, \( E_{\text{detection}} \) stands for the energy used for performing intrusion detection on a target node. It can be calculated by:

\[
E_{\text{detection}} = m \times (E_t + \bar{n} \cdot E_r) + m \times (E_t + (m - 1) \cdot (E_r + E_a)).
\]  

(5.13)

Here we consider the energy required to choose \( m \) intrusion detectors to evaluate a target node (the first term) and the energy required for \( m \) intrusion detectors to vote (the second term). Specifically, the first term is the number of intrusion detectors times the cost of transmitting plus the number of nodes in radio range times the cost of receiving. The second term is the number of intrusion detectors times the cost of transmitting plus the number of peer intrusion detectors times the cost of receiving plus the cost of analyzing the vote.

5.5 Numerical Data

In this section, we present numerical data for reliability assessment as a result of executing intrusion detection and response in a CPS. Our objective is to identify optimal design
settings in terms of the optimal values of $T_{IDS}$, $m$ and $C_T$ under which we can best trade off energy consumption versus intrusion detection, as well as response effectiveness versus impairment security failure to maximize the system MTTF, when given a set of parameter values characterizing the operational and networking conditions.

### 5.5.1 Effect of Intrusion Detection Strength

We first examine the effect of intrusion detection strength measured by the intrusion interval, $T_{IDS}$, and the number of intrusion detectors, $m$. We only present results for the reference CPS under reckless attackers, as the results for other types of attackers show similar trends.

Figure 5.4 shows MTTF versus $T_{IDS}$ as the number of detectors ($m$) in the system-level IDS varies over the range of $[3, 11]$ in increments of 2. We see that there exists an optimal $T_{IDS}$ value at which the system lifetime is maximized to best tradeoff energy consumption versus intrusion tolerance. Initially when $T_{IDS}$ is too small, the system performs ranging, sensing and intrusion detection too frequently and quickly exhausts its energy, resulting in a small lifetime. As $T_{IDS}$ increases, the system saves more energy and its lifetime increases. Finally, when $T_{IDS}$ is too large, although the system can save even more energy, it fails to catch bad nodes often enough, resulting in the system having many bad nodes. Bad nodes through active attacks can cause impairment security failure. Furthermore, when the system has 1/3 or more bad nodes out of the total population, a Byzantine failure ensues. We observe that the optimal $T_{IDS}$ value at which the system MTTF is maximized is sensitive to the $m$ value. The general trend is that as $m$ increases, the optimal $T_{IDS}$ value decreases. Here we observe that $m = 7$ is optimal to yield the maximum MTTF because too many intrusion detectors would induce energy exhaustion failure, while too few intrusion detectors would induce security failure. Using $m = 7$ can best balance energy exhaustion failure versus security failure for high reliability.

![Figure 5.4: MTTF vs. $T_{IDS}$ and $m$.](image)
Figure 5.5: MTTF vs. $T_{IDS}$ and $\lambda_c$.

Figure 5.5 shows MTTF versus $T_{IDS}$ as the compromising rate $\lambda_c$ varies over the range of once per 4 hours to once per 24 hours to test the sensitivity of MTTF with respect to $\lambda_c$ (with $m$ fixed at five to isolate its effect). We first observe that as $\lambda_c$ increases, MTTF decreases because a higher $\lambda_c$ will cause more compromised nodes to be present in the system. We also observe that the optimal $T_{IDS}$ decreases as $\lambda_c$ increases. This is because when more compromised nodes exist, the system needs to execute intrusion detection more frequently to maximize MTTF. Figure 5.5 identifies the best $T_{IDS}$ to be used to maximize the lifetime of the reference CPS to balance energy exhaustion versus security failure, when given $C_T$ and $\lambda_c$ characterizing the operational and networking conditions of the system.

5.5.2 Effect of Attacker Behavior

In this section, we analyze the effect of various attacker behavior models, including reckless (with $p_a = 1$, $p_{in}^k$ and $p_{fp}^k$ given as input), random (with $p_a = p_{\text{random}}$, $p_{in}^a$ and $p_{fp}^a$ given as input), insidious (with $p_a = 1$ when $N_b \geq N_b^T = 10$ and $p_a = 0$ otherwise, $p_{in}^i$ and $p_{fp}^i$ defined by Equation 5.5 given as input) and opportunistic attackers (with $\varepsilon = 0.9$). The analysis conducted here is based on static $C_T$. In the next section, we will analyze the effect of dynamic $C_T$ as a response to attacker strength detected at runtime.

Figure 5.6 shows MTTF versus $T_{IDS}$ with varying $p_{\text{random}}$ values. We first observe that the system MTTF is low when $p_{\text{random}}$ is small (e.g., $p_{\text{random}} = 0.025$). This is because when $p_{\text{random}}$ is small, most bad nodes are dormant and remain in the system without being detected. Thus, the system suffers from Byzantine failure quickly, leading to a low MTTF. As $p_{\text{random}}$ increases from 0.025 to 0.2, the system MTTF increases because of a higher chance of bad nodes being detected and removed from the system, thus reducing the probability of Byzantine security failure. As $p_{\text{random}}$ increases further, however, the system MTTF decreases again because of a higher probability of impairment security failure as there will be more
bad nodes actively performing impairment attacks. In the extreme case of $p_{random} = 1$, all bad nodes perform attacks and the system failure is mainly caused by impairment. The maximum MTTF occurs when $p_{random} = 0.2$ at which point the probability of security failure due to either type of security attacks is minimized. Here we should note that the result of $p_{random} = 0.2$ yielding the highest MTTF is a balance of impairment security failure rate versus Byzantine failure rate dictated by the parameter settings of the reference CPS as given in Tables 5.3 and 5.4.

Figure 5.7 compares MTTF versus $T_{IDS}$ of the reference CPS under the four attacker types head-to-head. It shows that the MTTF is the highest for the reference CPS under random attackers. The MTTF of the CPS under opportunistic and reckless attackers are second and third highest, respectively. As expected, the reference CPS under insidious attackers has the lowest MTTF. We attribute this to the fact that unlike reckless attackers which aim to cause
impairment failure, insidious attackers while dormant can cause Byzantine failure and while “all in” can also cause impairment failure. The extent to which the system MTTF differs depends on the relative rate at which impairment failure versus Byzantine failure occurs. The former is dictated by $\lambda_{if}$ and the latter is dictated by how fast the Byzantine failure condition is satisfied. The result that the MTTF difference between reckless attackers (the third curve) and insidious attackers (the last curve) is relatively significant is due to a large Byzantine failure rate compared with the impairment failure rate. On the other hand, the reference CPS under random attackers can more effectively prevent either Byzantine failure or impairment failure from occurring by removing bad nodes as soon as they perform attacks. The system MTTF difference between random versus reckless attackers again depends on the relative rate at which impairment failure versus Byzantine failure occurs. Both the endurance of the CPS and the optimal $T_{IDS}$ under opportunistic attackers are bracketed by the results for reckless and random attackers. This is because, in our configuration, opportunistic attackers are less aggressive than reckless, but more aggressive than random attackers. (Note this is not necessarily true in general, as it depends on the $\varepsilon$ value used and the resulting $p_a$ value compared with $p_{random}$.) Opportunistic attackers are also more cunning than reckless attackers which demonstrates that additional complexity does not guarantee more effectiveness.

5.5.3 Effect of Capture Strength

We can generalize the analysis from the constant capture strength model by defining a capture strength function as a state dependent function in the SPN model. The transition rate of TCP would be calculated by Equations 3.2, 3.3 or 3.4 for the logarithmic, linear or exponential capture strength models respectively.

![Figure 5.8: MTTF vs. $T_{IDS}$ under Reckless Attacker.](image)

Figure 5.8 compares the performance of the MGCPS under reckless attackers for various
capture strengths. The MTTF and the optimal $T_{IDS}$ are the highest for logarithmic capture strength, followed by linear and exponential strengths. After capture, a bad node contributes to CPS failure in one of three ways: In the first case, it is detected and evicted which leads to attrition failure. In the second case, it prosecutes an insider attack which leads to impairment failure. In the third case, it lies in wait which leads to Byzantine failure.

5.5.4 Effect of Intrusion Response

In this section, we analyze the effect of intrusion response, i.e., dynamic $C_T$ control as a response to attacker strength detected at runtime, on the system MTTF. We organize this section by the capture attacker strength model described in Section 3.5.1, so we can see the effectiveness of our dynamic $C_T$ countermeasure design described in Section 5.4.3.

We consider the attacker strength being characterized by the attacker behavior type (reckless, random, insidious or opportunistic) and/or the capture attack strength type (constant, logarithmic, linear or exponential). For ease of disposition, we only present the analysis results for the reckless attacker behavior type, as other attacker behavior types produce results with a similar trend. We show four graphs visualizing the effect of intrusion response. The four figures are for the four capture strengths. In each figure we will see which $C_T$ control method (among static, logarithmic, linear and exponential) is the best countermeasure.

Figures 5.9a through 5.9d show MTTF versus $T_{IDS}$ under the static, logarithmic, linear and exponential $C_T$ designs for constant, logarithmic, linear and exponential capture strengths, respectively, under the reckless attack behavior. Specifically, Figure 5.9a (upper left) indicates that static to linear dynamic $C_T$ control is most effective against constant capture strength. Figure 5.9b (upper right) indicates that logarithmic to linear dynamic $C_T$ control is most effective against logarithmic capture strength. Figure 5.9c (bottom left) indicates that linear dynamic $C_T$ control is most effective against linear capture strength. Figure 5.9d (bottom right) indicates that exponential dynamic $C_T$ control is most effective against exponential capture strength.

In general, we see a one-to-one match of the dynamic $C_T$ control function against the capture attack strength type for MTTF maximization. A more striking match is that linear $C_T$ dynamic control matches well against the linear capture attack strength as exhibited in Figure 5.9c. We also observe from Figure 5.9d that exponential dynamic $C_T$ control matches well against exponential capture attack strength, although the improvement in MTTF is less significant since the system is under heavy capture attacks.

The analysis results imply that one should be able to apply control theory to design a general $C_T$ control function with several control parameters (such as $C_T^p$, $\delta_{CT}$ and $P$ in Equations 4.11-4.13) such that the system can adjust the values of these parameters in response to dynamically changing attacker strength detected at runtime. This will be explored further as a future research direction.
Figure 5.9: Effect of Intrusion Response toward Capture Attack Strength under Reckless Attack Behavior.
5.6 Generalization

5.6.1 Stuxnet Attacks

We model Stuxnet attacks with exponential capture strength and insidious attacker behaviors. Figure 5.10 shows the MTTF of the MGCPS under Stuxnet attacks. Figure 5.10 visualizes the $m$ and $T_{IDS}$ settings representing the best intrusion detection strength against Stuxnet attacks modeled by exponential capture strength and insidious attack behaviors. When $P$ increases, one has to increase the detection strength by lowering $T_{IDS}$ and raising $m$ to maximize MTTF. For $P = 0.0$, the optimal $(m, T_{IDS})$ is $(11, 1600 \text{ s})$. For $P = 0.5$, the optimal $(m, T_{IDS})$ is $(11, 480 \text{ s})$. For $P = 1.0$, the optimal $(m, T_{IDS})$ is $(11, 20 \text{ s})$. For $P = 2.0$, the optimal $(m, T_{IDS})$ is $(11, 20 \text{ s})$. We obtained these results under dynamic $C_T$ control. The optimal $m$ is 11 within $[3, 11]$.

5.6.2 Mixed Attackers

To model mixed attackers, we adjust the composition of the SPN model as shown in Figure 5.11. This SPN is an extension of the one described in Section 5.3. We transform the $N_b$ place into a subsystem of places where nodes flow immediately through captured and probabilistically (TMIXX) mix into places for specific attacker behaviors: reckless, random, insidious and opportunistic. Each attacker behavior will impair the system at a unique rate (TIFX) and will be detected at a unique rate (TIDSX).

Figure 5.12 compares the performance of the MGCPS in the presence of four homogeneous attacker scenarios. The random adversary causes the least disruption in the MGCPS; the MTTF curve is the highest. The opportunistic adversary is close behind the random adver-
Chapter 5. Case Study: Mobile Group CPS

Figure 5.11: SPN for Modeling Mixed Attackers.

sary, but there is more area between their curves where the random adversary causes less disruption than the converse. The reckless adversary clearly disrupts the MGCPS more than random or opportunistic adversaries while the aptly-named insidious adversary disrupts the MGCPS to the largest degree. In general, the optimal $T_{IDS}$ moves to the left as the total area under the curve decreases, however while the areas under the curve for random and opportunistic adversaries are similar, the optimal $T_{IDS}$ for opportunistic attackers is about half that for random attackers.

Figure 5.13 compares the performance of the MGCPS in the presence of four mixed attacker scenarios. The 80/20 mix dominated by random and augmented by insidious attackers is the least disruptive. The 80/20 mix dominated by opportunistic and augmented by reckless attackers is more disruptive. The 80/20 mix dominated by reckless and augmented by random attackers is still more disruptive. The 80/20 mix dominated by insidious and augmented by opportunistic attackers is the most disruptive. This matches the results from the homogeneous comparison in Figure 5.12. For the more disruptive attacker types (e.g., insidious), augmentation with a less disruptive attacker type attenuates their effect (increases MTTF). On the other hand, augmentation with a more disruptive attacker type amplifies the effect (decreases MTTF) of less disruptive attacker types (e.g., random).

5.7 Simulation

First, we describe the simulation tool used, the simulation environment setup and the parameter values used. We instrumented a simulation using SMPL [93]. The simulation tracks
Figure 5.12: Linear Response under Homogeneous Attackers.

Figure 5.13: Linear Response under Mixed Attackers.
node state with components for membership ( eviction) and goodness (capture). The simulation schedules events for node capture, intrusion detection audits, impairment failure and energy exhaustion. A simulation run ends for one of four reasons: there is a Byzantine security failure, the system exhausts its energy, all of the nodes have been evicted or there is an impairment failure. We configure the simulation with the values as described in Table 5.3 given in Section 5.4.

Next, we discuss how we obtained the results (i.e., number of runs) and if the results are of statistical significance. We collect an MTTF observation by recording the starting simulator time, running a simulation, recording the ending simulator time and calculating the difference between simulator times. The average MTTF value reported represents a grand mean out of a large number of MTTF observations with statistical significance. Specifically, we apply batch means analysis [93] to satisfy 95% confidence level and 10% accuracy requirements. We use a batch size of 100 MTTF observations to compute a batch mean out of 100 MTTF observations. The general idea is that for each given configuration, we run a number of batches to get a number of batch means. Then we calculate the grand mean out of the batch means and determine if the grand mean falls within 10% of the true mean with 95% confidence. If it does not, we increase our sample data by one more batch to collect another batch mean and then compute the grand mean again. We repeat this until the grand mean satisfies the 95% confidence level and 10% accuracy requirements.

![Figure 5.14: Simulation and Theoretical MTTF vs. $T_{IDS}$ and $m$.](image)

Below we report simulation results for simulation validation of analytical results shown in Figures 5.4, 5.5 and 5.6.

Figure 5.14 shows MTTF versus $T_{IDS}$ simulation results with respect to analytical results obtained earlier in Figure 5.4. The shapes of both plots are remarkably similar: unimodal with similar kurtosis, a left/positive skew and a pronounced right tail. In both plots, MTTF peaks near $T_{IDS} = 160$ s near 150 minutes, and $m = 5$ is the optimal value. The mean percentage error (MPE) separating the analysis and simulation results are between 13 and 22% curve by curve, as shown in Figure 5.14. We conclude that while there is some bias, simula-
tion results match up with analytical results very well, thereby validating our survivability analysis methodology.

Figure 5.15: Simulation and Theoretical MTTF vs. $T_{\text{IDS}}$ and $\lambda_c$.

Figure 5.15 shows MTTF versus $T_{\text{IDS}}$ simulation results with respect to analytical results obtained earlier in Figure 5.5. The shapes of both plots are similar in the same way as Figure 5.14. In Figure 5.15, MTTF peaks near $T_{\text{IDS}} = 300$ s between 600 and 3200 min, and lower $\lambda_c$ corresponds with higher MTTF. The MPE separating the analysis and simulation results are between 36 and 65% curve by curve, as shown in Figure 5.15.

Figure 5.16: Simulation and Theoretical MTTF vs. $T_{\text{IDS}}$ and $p_{\text{random}}$.

Figure 5.16 shows MTTF versus $T_{\text{IDS}}$ simulation results with respect to analytical results obtained earlier in Figure 5.6. The shapes of both plots are similar in the same way as Figures 5.14 and 5.15. In Figure 5.16, MTTF peaks near $T_{\text{IDS}} = 300$ s between 500 and 2200 min, and lower $p_{\text{random}}$ corresponds with higher MTTF and higher optimal $T_{\text{IDS}}$. The MPE separating the analysis and simulation results are between -28 and 72% curve by curve, as shown in Figure 5.16. We conclude that the analytical results and especially the trends exhibited in Figure 5.6 are valid.
5.8 Summary

In this chapter, we applied our IDS design to a mobile group CPS. In particular, we developed a probability model with simulation validation to analyze the resulting reliability of the mobile group CPS subject to various capture and inside attacker behaviors. For each attacker behavior, we identified the best detection strength (in terms of the detection interval and the number of detectors) and the best response strength (in terms of the per-host minimum compliance threshold for setting the false negative and positive probabilities) under which the reliability of the system may be maximized.
Chapter 6

Case Study: Medical CPS

In this chapter, we apply our IDS design to a medical CPS (MCPS). We propose and analyze a behavior-rule specification-based technique for intrusion detection of medical devices embedded in an MCPS in which the patient’s safety is of the utmost importance. We propose a methodology to transform behavior rules to a state machine, so that a device that is being monitored for its behavior can easily be checked against the transformed state machine for deviation from its behavior specification. Using vital sign monitor medical devices as an example, we demonstrate that our intrusion detection technique can effectively trade false positives off for a high detection probability to cope with more sophisticated and hidden attackers to support ultra safe and secure MCPS applications. Moreover, through a comparative analysis, we demonstrate that our behavior-rule specification-based IDS technique outperforms an existing anomaly-based technique for detecting abnormal patient behaviors in pervasive healthcare applications. The content of this chapter derives from our publications [97, 98].

6.1 Background

The most prominent characteristic of an MCPS is its feedback loop that acts on the physical environment. In other words, the physical environment provides data to the MCPS sensors whose data feed the MCPS control algorithms that drive the actuators which change the physical environment. MCPSs are often characterized by sophisticated patient treatment algorithms interacting with the physical environment including the patient. In this chapter, we are concerned with intrusion detection mechanisms for detecting compromised sensors or actuators embedded in an MCPS for supporting safe and secure MCPS applications upon which patients and healthcare personnel can depend with high confidence.

IDS design for CPSs has attracted considerable attention [6, 22] because of the dire consequence of CPS failure. However, IDS techniques for MCPSs are still in their infancy with
very little work reported. Intrusion detection techniques in general can be classified into four types: signature, anomaly, trust and specification-based techniques. In this chapter, we consider specification rather than signature-based detection to deal with unknown attacker patterns. We consider specification rather than anomaly-based techniques to avoid using resource-constrained sensors or actuators in an MCPS for profiling anomaly patterns (e.g., through learning) and to avoid high false positives. We consider specification rather than trust-based techniques [9, 55] to avoid delay due to trust aggregation and propagation to promptly react to malicious behaviors in safety critical MCPSs.

To accommodate resource-constrained sensors and actuators in an MCPS, we propose the design notion of behavior rules for specifying acceptable behaviors of medical devices in an MCPS. Rule-based intrusion detection thus far has been applied only in the context of communication networks which have no concern of physical environments and the closed-loop control structure as in an MCPS. For example, Da Silva et al. [41] propose an IDS that applies seven types of traffic-based rules to detect intruders: interval, retransmission, integrity, delay, repetition, radio transmission range and jamming. Ioannis et al. [72] propose a multitrust IDS with traffic-based collection that audits the forwarding behavior of suspects to detect blackhole and greyhole attacks launched by captured devices based on the rate of specification violations.

Our contribution relative to prior work cited above is that we specifically consider behavior rules for MCPS actuators controlling patient treatment algorithms as well as for physiological sensors providing information concerning the physical environment. Further, we propose a methodology to transform behavior rules to a state machine, so that a device that is being monitored for its behavior can easily be checked against the transformed state machine for deviation from its behavior specification. Existing work [37, 45] only considered specification-based state machines for intrusion detection of communication protocol misbehaving patterns.

Untreated in the literature, in this chapter we also investigate the impact of attacker behaviors on the effectiveness of MCPS intrusion detection. We demonstrate that our specification-based IDS technique can effectively trade higher false positives off for lower false negatives to cope with more sophisticated and hidden attackers. We show results for a range of configurations to illustrate this trade. Because the key motivation in MCPS is safety, our solution is deployed in a configuration yielding a high detection rate without compromising the false positive probability. Our approach is monitoring-based relying on the use of peer devices to monitor and measure the compliance degree of a trustee device connected to the monitoring node by the CPS network. The rules comparing monitor and trustee physiology (blood pressure, oxygen saturation, pulse, respiration and temperature) exceeds protection possible by considering devices in isolation.

The fundamental difference in designing IDSs for safety critical MCPSs versus for communication systems is that the intrusion detection is closely tied with the physical components of the CPS, so the detection is less about communication protocol compliance but more about
behavior compliance specific to the physical components to be controlled in the CPS. The behavior rules proposed in our work specifically address the expected behavior of individual physical components in the MCPS. The compliance threshold proposed in this chapter specifically measures the goodness of a physical component. A challenge is to provide a high detection rate without introducing high false positives. We demonstrate that our IDS design based on the compliance threshold can effectively distinguish benign abnormalities from malicious attacks. To the best of our knowledge, there is no prior work discussing the difference between CPS intrusion detection and communication systems intrusion detection.

It is necessary to build an IDS per CPS domain or application since the behavior rules for specifying the behaviors of physical components/devices in a CPS are inherently domain or application specific.

In the literature, ISML [21] and T-Rex [163] are also specification-based approaches for intrusion detection in CPSs. However, none of them considered MCPSs. In the field of intrusion detection for MCPSs or healthcare systems, Asfaw et al. [8] studied an anomaly-based IDS for MCPSs. The authors focus on attacks that violate privacy of an MCPS; in contrast, our investigation focuses on attacks that violate the integrity of an MCPS. They use an anomaly-based approach while we use a specification-based approach. Asfaw et al. do not provide numerical results in the form of false negatives or false positives which are the critical metrics for this research area; our investigation does provide these results.

Venkatasubramanian and Gupta [146] survey security solutions for pervasive healthcare applications. Like [8], the authors focus on attacks on a passive pervasive healthcare system that violate patient privacy while our investigation considers integrity attacks on an MCPS that harm a patient. Their countermeasures focus on encryption and authentication/access control.

Yang and Hwang [154] investigated an approach to fraud and abuse detection in healthcare applications. In contrast, our investigation focuses on the treatment, rather than the administrative, domain of healthcare. The authors use an anomaly-based approach while we use a specification-based approach. They provide numerical results that measure internal validity (the effectiveness of the data mining implementation) but do not provide externally valid metrics like ROC which can reveal the tradeoff between the detection rate versus the false positive probability.

Porras and Neumann [113] study a hierarchical multitrust behavior-based IDS called Event Monitoring Enabling Responses to Anomalous Live Disturbances (EMERALD) [37] using complementary signature-based and anomaly-based analysis. The authors identify a signature-based analysis trade between the state space created/runtime burden imposed by rich rule sets and the increased false negatives that stem from a less expressive rule set. Porras and Neumann highlight two specific anomaly-based techniques using statistical analysis: one studies user sessions (to detect live intruders), and the other studies the runtime behavior of programs (to detect malicious code). EMERALD provides a generic analysis framework that is flexible enough to allow anomaly detectors to run with different scopes.
of multitrust data (service, domain or enterprise). However, Porras and Neumann did not report false negative or false positive probability data. While EMERALD pursues a domain-independent CPS security solution combining anomaly and signature-based analysis, our investigation focuses on one that is relevant for MCPSs using specification-based analysis.

Tsang and Kwong [140] propose a multitrust IDS called Multi-agent System (MAS) that includes an analysis function called Ant Colony Clustering Model (ACCM). The authors intend for ACCM to reduce the characteristically high false positive rate of anomaly-based approaches while minimizing the training period by using an unsupervised approach to machine learning. MAS is hierarchical and contains a large number of roles: monitor agents collect audit data, decision agents perform analysis, action agents effect responses, coordination agents manage multitrust communication, user interface agents interact with human operators, and registration agents manage agent appearance and disappearance. Their results indicate ACCM slightly outperforms the detection rates and significantly outperforms the false positive rates of k-means and expectation-maximization approaches. Like [113], MAS pursues a domain-independent CPS security solution using anomaly-based analysis; our investigation focuses on MCPS-specific IDS using specification-based analysis.

6.2 MCPS Reference Model

![Figure 6.1: Medical Physical Components in the Reference MCPS.](image)

We consider a pervasive health monitoring system embedding medical devices as our MCPS reference model as illustrated in Figure 6.1. For ease of disposition, we are particularly concerned with three types of sensor/actuator devices embedded in this MCPS: vital sign
monitor (VSM), patient controlled analgesia (PCA) and cardiac device (CD). Many healthcare examples exist with these three devices. An example is an automated anesthesiologist where vital sign monitors (VSMs) sense patient physiology and control intravenous delivery of sedative. Specifically, VSMs sense blood pressure (mmHg\(^2\)), oxygen saturation (SpO\(_2\)), heart rate (Hz), respiration (Hz) and temperature (C). Another example is PCA where patient analgesic requests and physiological sensor readings from VSMs drive infusion pumps [84]. A third example is an intensive care situation where the CD frequency and pulse readings from VSMs drive biomedical devices such as a ventilator or automatic external defibrillator. The Welch Allyn Connex 6000 is the real system we base this investigation on. It has a 11.1 V 3.80 Ah (42 Wh) or 10.8 V 6.75 Ah (73 Wh) Li-ion battery, blood pressure sensor, thermometer, oxygen saturation sensor and two pulse rate sensors (one each integrated with blood pressure and oxygen saturation sensors). It uses USB for internal communication and IEEE 802.11 for external communication.

### 6.3 MCPS Intrusion Detection Design

#### 6.3.1 Behavior Rules

Our IDS design for the reference MCPS model relies on the use of lightweight specification-based behavior rules for each sensor or actuator medical device. They are oriented toward detecting an inside attacker attached to a specific physical component, provide a continuous (versus a binary) output between 0 and 1 (to account for transient faults and human errors) and allow a monitor device to perform intrusion detection on a neighboring trustee through monitoring. Here a monitor device is itself a sensor or monitor capable of doing intrusion detection on many trustees of different types. For example, a sensor might need to audit dissimilar sensors or even actuators for a small system. Therefore, a monitor device might have several sets of behavior rules (and thus several state machines), one for each trustee. Table 6.1 lists the MCPS behavior rules for PCA, CD and VSM. This table specifies the trustee and monitor devices for applying our IDS technique.

The behavior rule set specifies expected normal behaviors for each device and can detect deviation of normal behaviors regardless of the attacker’s patterns. It does not rely on knowledge of known attacker patterns as in signature-based intrusion detection.

#### 6.3.2 Transforming Rules to State Machines

The following procedure transforms a behavior specification into a state machine: First, we identify the “attack state” as a result of a behavior rule being violated. Then, we transform this attack state into a conjunctive normal form predicate and identify the involved state components in the underlying state machine. Next, for each device, we combine the attack
states into a Boolean expression in disjunctive normal form. Then we transform the union of all predicate variables into the state components of a state machine and establish their corresponding ranges. Finally, we manage the number of states by state collapsing and identifying combinations of values that are not legitimate. Below we exemplify how a state machine is derived from the behavior specification in terms of behavior rules for the reference MCPS model.

Unsafe states in our state machine are not those “hazardous” states generated due to design faults (e.g., software bugs). Such “hazardous” states, once identified, indeed would be removed as a result of design faults being identified and removed during the testing and debugging phase. The unsafe states (and safe states) in our approach are device-specific and are not removable because they are not caused by design faults. A CPS device will enter an unsafe state only when it is seen to deviate from the normal behavior specified by the behavior rule. This is the idea of our specification-based behavior rule intrusion detection.

6.3.2.1 Identify Attack States

Attacks performed by a compromised sensor/actuator will drive the MCPS into certain attack states identifiable through analyzing the specification-based behavior rules.

For the PCA device, there are 4 attack states as a result of violating the 4 PCA behavior rules listed in Table 6.1. The first PCA attack state is that a patient requesting analgesic has a pulse below some threshold. One way an attacker could exploit this is to cause an overdose of analgesic delivered by a PCA system. A patient will lose consciousness after receiving a sufficient amount of analgesic; if the PCA receives additional requests for analgesic, then an intruder is involved. The IDS can infer consciousness from pulse data. For this attack state, the PCA module is the trustee, and the VSM is the monitor.

The second PCA attack state is that a patient requesting analgesic has a respiration rate below some threshold. A compromised PCA device performing this attack will drive the MCPS into this state. One way an attacker could exploit this is to cause an overdose of analgesic delivered by a PCA system. A patient will lose consciousness after receiving a sufficient amount of analgesic; if the PCA receives additional requests for analgesic, then an intruder is involved. The IDS can infer consciousness from respiration data. For this attack state, the PCA module is the trustee, and the VSM is the monitor.

The third PCA attack state is that an analgesic request rate exceeds some threshold. One way an attacker could exploit this is to cause an overdose of analgesic delivered by a PCA system. It is important to distinguish physical button presses from requests actually generated. While a patient in pain may press the button more frequently than is safe due to pain, the PCA module should only fulfill requests within the safe threshold. If the PCA module fulfills requests too frequently, then an intruder is involved. For this attack state, the PCA module is the trustee, and the VSM is the monitor.
Chapter 6. Case Study: Medical CPS

The fourth PCA attack state is that the PCA infusion rate, $x$, is in $(0, 100\%]$ and the cardiac device mode, $y$, is defibrillation, yielding a state with two components. As the device being evaluated transitions from one state $(x_0, y_0)$ to another $(x_1, y_1)$, the monitor can check if $(x_0, y_0)$ and $(x_1, y_1)$ are both good states. For this attack state, the PCA module is the trustee, and the VSM is the monitor.

For the CD device, there are two attack states. The first CD attack state is that pulse average is not equal to CD frequency when acting as a pacemaker. One way an attacker could exploit this is to change the pacemaker frequency. If the CD frequency when acting as a pacemaker is substantially different from the patient’s heart rate, then an intruder is involved. For this attack state, the CD is the trustee, and the VSM is the monitor.

The second CD attack state is that pulse average is within a normal range when the CD enters defibrillator mode. One way an attacker could exploit this is to defibrillate a stable patient. If the CD enters defibrillator mode unnecessarily, then an intruder is involved. For this attack state, the CD is the trustee, and the VSM is the monitor.

For the VSM device, there are 5 attack states in which a trustee sensor reading (blood pressure, oxygen saturation, pulse, respiration or temperature) is beyond 100% of the corresponding monitor sensor reading. A peer VSM in the neighborhood of the trustee sensor serves as the monitor, measuring the same physical phenomenon. In this rule there is a variable “sensor reading % deviation” which can go from 0 to 100% in 10% increments, yielding 11 possible values. The monitor observing a trustee sensor will check the status of this variable. As the trustee sensor goes from one state to another, say, from 10 to 20%, the monitor will assess the deviation of good behaviors of the trustee by means of host IDS techniques.

### 6.3.2.2 Express Attack States in Conjunctive Normal Form

Table 6.2 lists the attack states in conjunctive normal form.

### 6.3.2.3 Consolidate Predicates in Disjunctive Normal Form

**PCA**

$((\text{Analgesic Request} = \text{TRUE}) \land (\text{Pulse} < T)) \lor ((\text{Analgesic Request} = \text{TRUE}) \land (\text{Respiration} < T)) \lor (\text{Analgesic Request Rate} > T) \lor ((\text{Analgesic Infusion Rate} > 0) \land (\text{Mode} = \text{DEFIBRILLATOR}))$

**CD**

$((\text{Analgesic Infusion Rate} > 0) \land (\text{Mode} = \text{DEFIBRILLATOR})) \lor ((\text{Mode} = \text{PACEMAKER}) \land (|\text{Pulse} - \text{Pacemaker Frequency}| > \delta)) \lor ((\text{Mode} = \text{DEFIBRILLATOR}) \land (L < \text{Pulse} < H))$
Table 6.1: MCPS Behavior Rules

<table>
<thead>
<tr>
<th>Description</th>
<th>Trustee</th>
<th>Monitor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pulse above threshold during analgesic request</td>
<td>PCA</td>
<td>VSM</td>
</tr>
<tr>
<td>Respiration above threshold during analgesic request</td>
<td>PCA</td>
<td>VSM</td>
</tr>
<tr>
<td>Analgesic request rate below safe threshold</td>
<td>PCA</td>
<td>VSM</td>
</tr>
<tr>
<td>No analgesic infusion during defibrillation</td>
<td>PCA</td>
<td>VSM</td>
</tr>
<tr>
<td>Pulse matches pacemaker frequency</td>
<td>CD</td>
<td>VSM</td>
</tr>
<tr>
<td>Patient is unstable before defibrillation</td>
<td>CD</td>
<td>VSM</td>
</tr>
<tr>
<td>Trustee blood pressure matches monitor</td>
<td>VSM</td>
<td>Peer VSM</td>
</tr>
<tr>
<td>Trustee oxygen saturation matches monitor</td>
<td>VSM</td>
<td>Peer VSM</td>
</tr>
<tr>
<td>Trustee pulse matches monitor</td>
<td>VSM</td>
<td>Peer VSM</td>
</tr>
<tr>
<td>Trustee respiration matches monitor</td>
<td>VSM</td>
<td>Peer VSM</td>
</tr>
<tr>
<td>Trustee temperature matches monitor</td>
<td>VSM</td>
<td>Peer VSM</td>
</tr>
</tbody>
</table>

Table 6.2: Attack States in Conjunctive Normal Form

\[
\begin{align*}
(\text{Analgesic Request} = \text{TRUE}) \land (\text{Pulse} < T) \\
(\text{Analgesic Request} = \text{TRUE}) \land (\text{Respiration} < T) \\
(\text{Analgesic Request Rate} > T) \\
(\text{Analgesic Infusion Rate} > 0) \land (\text{Mode} = \text{DEFIBRILLATOR}) \\
(\text{Mode} = \text{PACEMAKER}) \land (|\text{Pulse} - \text{Pacemaker Frequency}| > \delta) \\
(\text{Mode} = \text{DEFIBRILLATOR}) \land (L < \text{Pulse} < H) \\
|\text{Monitor Blood Pressure} - \text{Trustee Blood Pressure}| > \delta \\
|\text{Monitor Oxygen Saturation} - \text{Trustee Oxygen Saturation}| > \delta \\
|\text{Monitor Pulse} - \text{Trustee Pulse}| > \delta \\
|\text{Monitor Respiration} - \text{Trustee Respiration}| > \delta \\
|\text{Monitor Temperature} - \text{Trustee Temperature}| > \delta
\end{align*}
\]
### Table 6.3: MCPS State Components

<table>
<thead>
<tr>
<th>Name</th>
<th>Control or Reading</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analgesic request</td>
<td>Reading</td>
<td>{false, true}</td>
</tr>
<tr>
<td>Pulse</td>
<td>Reading</td>
<td>[0, 240 bpm]</td>
</tr>
<tr>
<td>Respiration</td>
<td>Reading</td>
<td>[0, 60 bpm]</td>
</tr>
<tr>
<td>Analgesic request rate</td>
<td>Reading</td>
<td>[0, 4/hour]</td>
</tr>
<tr>
<td>Blood pressure</td>
<td>Reading</td>
<td>[0, 240 mmHg] × [0, 160 mmHg]</td>
</tr>
<tr>
<td>Oxygen saturation</td>
<td>Reading</td>
<td>[0, 100%]</td>
</tr>
<tr>
<td>Temperature</td>
<td>Reading</td>
<td>[32, 42 C]</td>
</tr>
<tr>
<td>Analgesic infusion rate</td>
<td>Control</td>
<td>[0, 100%]</td>
</tr>
<tr>
<td>Mode</td>
<td>Control</td>
<td>{passive, pacemaker, defibrillator}</td>
</tr>
<tr>
<td>Pacemaker frequency</td>
<td>Control</td>
<td>[0, 240 bpm]</td>
</tr>
</tbody>
</table>

**VSM**  
\( (|\text{Monitor Blood Pressure - Trustee Blood Pressure}| > \delta) \lor (|\text{Monitor Oxygen Saturation - Trustee Oxygen Saturation}| > \delta) \lor (|\text{Monitor Pulse - Trustee Pulse}| > \delta) \lor (|\text{Monitor Respiration - Trustee Respiration}| > \delta) \lor (|\text{Monitor Temperature - Trustee Temperature}| > \delta) \)

#### 6.3.2.4 Identify State Components and Component Ranges

We quantize continuous components at integer scale in permissible ranges. For example, pulse is in the range of [0, 240 bpm] and respiration is in the range of [0, 60 bpm]. Table 6.3 shows a complete list of the permissible ranges of MCPS state components. The resulting PCA automaton has \( 2 \times 241 \times 61 \times 5 \times 101 \times 3 = 4.45 \times 10^7 \) states. The resulting CD automaton has \( 241 \times 101 \times 3 \times 241 = 1.760 \times 10^7 \) states. The resulting VSM automaton has \( 241 \times 161 \times 241 \times 161 \times 101 \times 101 \times 241 \times 241 \times 61 \times 61 \times 11 \times 11 = 4.016 \times 10^{23} \) states. All of these automata are too large; we deal with this state explosion in the next step.

#### 6.3.2.5 Manage State Space

To manage the number of states, we reduce the size of the state machine by abbreviating the values for some components. For the PCA device, only three values are relevant for pulse, respiration and analgesic request rate: normal, beyond warning threshold and beyond unsafe threshold. Therefore, we collapse the domain for each of these components to three values. Likewise only two values are relevant for analgesic infusion rate: zero or nonzero. Therefore, we collapse the domain for this component to two values. This treatment yields a modest PCA state machine with \( 2 \times 3 \times 3 \times 3 \times 2 \times 3 = 324 \) states. 50 of these states
Chapter 6. Case Study: Medical CPS

are safe because they fully comply with all of the behavior rules from Table 6.1. 80 are warning states because they exceed the warning threshold for at least one behavior rule. 194 of these states are unsafe because they violate or exceed the unsafe threshold for at least one of the behavior rules. Rather than their values, the VSM behavior rules only need to know whether each vital sign trustee reading matches, is farther than the warning threshold or is farther than the unsafe threshold from the corresponding monitor reading. Therefore, we collapse the domain for each of these components to three values. This treatment yields a modest VSM state machine with \(3 \times 3 \times 3 \times 3 \times 3 = 243\) states. One of these states is safe because the monitor and trustee readings match for all five components as described in Table 6.1. 31 are warning states because the monitor and trustee readings differ by more than the warning margin for at least one component but not more than the unsafe threshold for any component. 211 of these states are unsafe because at least one component differs by more than the unsafe threshold.

### 6.3.2.6 Behavior Rule State Machines

Here we describe how to generate the behavior rule state machine of a medical device. We use the VSM device as an example. The VSM state machine consisting of one safe, 31 warning and 211 unsafe states based on the behavior rules is generated as follows. First, we label these states as 1, 2, \ldots, n = 243. Next, we assign \(p_{ij}\), the probability that state \(i\) goes to state \(j\), for each \((i, j)\) pair in the state machine to reflect a good or bad VSM’s behavior. A good VSM should stay in safe states 100% of the time. This will give the compliance degree of a good VSM close to one. However, occasionally it may be detected by the monitor node as staying in a warning or unsafe state due to ambient noise resulting from unexpected environment or system condition changes, as well as wireless communication faults. Let \(p_{err}\) be the error probability of a monitor node misidentifying the status of a trustee node due to ambient noise and wireless communication faults. During the testing phase, we seed a good VSM in the system and assign a monitor node to observe and measure \(p_{ij}\) of the good VSM in the presence of the error probability \(p_{err}\): \(p_{ij}\) is \(31 / (31 + 211) \times p_{err}\) when \(j\) is one of the 31 warning states, \(p_{ij}\) is \(211 / (31 + 211) \times p_{err}\) when \(j\) is one of the 211 unsafe states, and \(p_{ij}\) is \(1 - p_{err}\) when \(j\) is the one good state. Figure 6.2 illustrates the behavior rule state machine for a good VSM in the MCPS. One dotted slash and crossed dotted slashes over a state indicate a warning state and an unsafe state, respectively. Each state component represents how one of the trustee node attributes matches its counterpart from the monitor. For the VSM device, blood pressure, oxygen saturation, pulse, respiration and temperature are the device attributes of interest. Note that each device has its own state machine with device-specific attributes being the state components of the state machine.

For a compromised VSM, \(p_{ij}\) depends on its attacker type: A reckless attacker presumably will stay in unsafe or warning states 100% of the time; however, occasionally it may be detected by the monitor node as staying in a safe state due to ambient noise and wireless
Chapter 6. Case Study: Medical CPS

Figure 6.2: Good VSM Behavior Rule State Machine.

Figure 6.3: Random Attacker VSM Behavior Rule State Machine.
communication faults. During the testing phase, we seed a reckless attacker in the system
following its attacker profile and assign a monitor node to observe and measure $p_{ij}$: $p_{ij}$ is
\[ \frac{211}{31 + 211} \times (1 - p_{err}) \] when $j$ is one of the 211 unsafe states, \[ \frac{31}{31 + 211} \times (1 - p_{err}) \] when $j$ is one of the 31 warning states and $p_{err}$ when $j$ is the one good state where $p_{err}$ is
the error probability of misidentifying the status of a reckless attacker due to ambient noise
and wireless communication faults. For a random attacker with attack probability $p_a$, $p_{ij}$
is \[ \frac{211}{31 + 211} \times (p_a \times (1 - p_{err}) + (1 - p_a) \times p_{err}) \] when $j$ is one of the 211 bad states, \[ \frac{31}{31 + 211} \times (p_a \times (1 - p_{err}) + (1 - p_a) \times p_{err}) \] when $j$ is one of the 31 warning states and $p_a \times p_{err} + (1 - p_a) \times (1 - p_{err})$ when $j$ is the one good state. We note that a random attacker
with attack probability $p_a$ will stop attacking with probability $1 - p_a$, which will be detected
by the monitor node with probability $1 - p_{err}$. Figure 6.3 illustrates the behavior rule state
machine for a random attacker VSM in the MCPS.

### 6.3.3 Collect Compliance Degree Data

We use the state machines to collect compliance degree data of a good and a bad medical
device during the system testing and debugging phase before deployment. The behaviors of
normal devices and devices controlled by a random attacker are simulated and compliance
degree data are collected to allow us to predict the false negative and false positive probabil-
ities. While we experimented with a range of configurations, our solution is deployed with
settings yielding a high detection rate because the key motivation in MCPS is safety.

Specifically, we profile the analgesic request, pulse, respiration, blood pressure, oxygen satu-
reration, temperature, analgesic infusion rate, cardiac device mode and pacemaker frequency,
given that they are being controlled by a good or a bad medical device.

We model the behavior of a medical device by a stochastic process such that it may be in
state $0, 1, 2, \ldots, n$ in a state machine for intrusion detection of this medical device, with
$p_{ij}$ parameterized as discussed earlier in Section 6.3.2.6. Then, the probability that the
stochastic process is in state $j$ is given by:

\[ \pi_j = \sum_{i=0}^{n} \pi_i \cdot p_{ij} \] (6.1)

Since we have $n + 1$ states, we have $n + 1$ equations above, one for each state. This will yield
infinite solutions, so we replace one equation with:

\[ \sum_{i=0}^{n} \pi_i = 1 \] (6.2)

The physical meaning of $\pi_j$ is the probability that a device is in state $j$ at any time.

Let $c$ be the compliance degree of a node. With the above formulation, it is calculated as
the sum of the products of each state’s grade and probability:

\[ c = \sum_j c^j \times \pi_j \]  

(6.3)

where \( c^j \) is the “grade” assignment to state \( j \), measuring the closeness between the observed behavior (in state \( j \)) and the specified “good” behavior. We consider two grading strategies: binary and distance-based. For binary grading, we assign a value of 1 to state \( j \) if it is secure and a value of 0 otherwise:

\[ c^j = \begin{cases} 
1 & \text{if state } j \text{ is a secure state} \\
0 & \text{otherwise} 
\end{cases} \]

With binary grading, the compliance degree \( c \) of a device essentially is equal to the proportion of the time the device is in secure states.

For distance-based grading, we still assign a value of 1 to state \( j \) if it is secure. However, if state \( j \) is insecure, we assign it a value in \([0, 1]\) representing the distance of state \( j \) from a secure state. Therefore, \( c^j \) is assigned as follows:

\[ c^j = 1 - \text{distance}_j / \text{maximum distance} \]

where distance\(_j\) is the distance between state \( j \) and the nearest secure state, and maximum distance is the longest distance between any insecure state and the nearest secure state in the state machine. By this assignment, if state \( j \) is secure, distance\(_j\) is zero, hence \( c^j = 1 \). If state \( j \) is insecure, \( c^j \) is still close to 1 if \( j \) is close to a secure state but is close to 0 if \( j \) is far from a secure state. With \( c^j \) assigned, we can then calculate the compliance degree, \( c \), of a node in state machine \( s \) using Equation 6.3 where \( \pi_j \) gives the proportion of time a node stays in \( j \) over the observation period. We study six distance-based grading strategies for measuring \( c^j \) and for computing the compliance degree based on Equation 6.3.

- Hamming distance, also called signal distance, applies to a pair of multidimensional data points. This is the number of state components that differ between two sequences: in our application, state \( j \) and the closest secure state.

\[ c^j = 1 - \text{Hamming}(j) / \text{max(Hamming(\cdot))} \]

For example, consider a system with one state component with two possible values. This system has two states: 0, 1. Consider state 0 safe and state 1 unsafe. Therefore, \( c^0 = 1 - 0/1 = 1 \) and \( c^1 = 1 - 1/1 = 0 \). Consider a node with \( \pi_0 = 0.9 \) and \( \pi_1 = 0.1 \). Therefore, for this node, \( c = 1 \times 0.9 + 0 \times 0.1 = 0.9 \).

- Manhattan distance, also called rectilinear distance, applies to a pair of multidimensional data points. This is the sum of the differences between state components of two sequences: in our application, state \( j \) and the closest secure state.

\[ c^j = 1 - \text{Manhattan}(j) / \text{max(Manhattan(\cdot))} \]
For example, consider a system with two state components with two possible values. This system has four states: 00, 01, 10, 11. Consider state 00 safe and the rest unsafe. Therefore, $c^{00} = 1 - 0/2 = 1$, $c^{01} = 1 - 1/2 = 0.5$, $c^{10} = 1 - 1/2 = 0.5$ and $c^{11} = 1 - 2/2 = 0$. Consider a node with $\pi_{00} = 0.9$, $\pi_{01} = 0.045$, $\pi_{10} = 0.045$ and $\pi_{11} = 0.01$. Therefore, for this node, $c = 1 \times 0.9 + 0.5 \times 0.045 + 0.5 \times 0.045 + 0 \times 0.01 = 0.945$.

- Euclidean distance applies to a pair of multidimensional data points. This is the square root of the sum of the squares of the state component differences between two sequences: in our application, state $j$ and the closest secure state.

$$c^j = 1 - \text{Euclidean}(j)/\text{max(Euclidean(·))}$$

For example, consider a system with two state components with two possible values. This system has four states: 00, 01, 10, 11. Consider state 00 safe and the rest unsafe. Therefore, $c^{00} = 1 - \sqrt{0^2 + 0^2}/\sqrt{2} = 1$, $c^{01} = 1 - \sqrt{0^2 + 1^2}/\sqrt{2} = 0.707$, $c^{10} = 1 - \sqrt{1^2 + 0^2}/\sqrt{2} = 0.707$ and $c^{11} = 1 - \sqrt{1^2 + 1^2}/\sqrt{2} = 0$. Consider a node with $\pi_{00} = 0.9$, $\pi_{01} = 0.045$, $\pi_{10} = 0.045$ and $\pi_{11} = 0.01$. Therefore, for this node, $c = 1 \times 0.9 + 0.707 \times 0.045 + 0.707 \times 0.045 + 0 \times 0.01 = 0.963$.

- Longest common subsequence (LCS) distance, not to be confused with longest common substring distance, applies to a pair of time series. Longest common subsequence differs from longest common substring because a common subsequence does not need to be contiguous; extra values can appear within a common subsequence.

$$c = \text{LCS(monitor, trustee)}/\text{time series length}$$

For example, consider a system using a two point time series with two possible values. Monitor and trustee time series have four possible values: 00, 01, 10, 11. If we consider a monitor time series of 00 and a trustee time series of 00, $c = 2/2 = 1$. If we consider a monitor time series of 00 and a trustee time series of 10, $c = 1/2 = 0.5$. If we consider a monitor time series of 00 and a trustee time series of 11, $c = 0/2 = 0$.

- Levenshtein distance, also called edit distance, applies to a pair of time series. This is the minimum number of edits required to transform one sequence into another; Levenshtein edits comprise insertion, deletion and substitution.

$$c = 1 - \text{Levenshtein(monitor, trustee)}/\text{time series length}$$

For example, consider a system using a three point time series with three possible values. Monitor and trustee time series have 27 possible values: 000, 001, 002, ..., 222. If we consider a monitor time series of 000 and a trustee time series of 000, $c = 1 - 0/3 = 1$. If we consider a monitor time series of 012 and a trustee time series of 120, $c = 1 - 2/3 = 0.333$. If we consider a monitor time series of 000 and a trustee time series of 111, $c = 1 - 3/3 = 0$. 
**Chapter 6. Case Study: Medical CPS**

- Damerau-Levenshtein is based on Levenshtein but adds transposition of two contiguous values to the set of allowed edits. For example, consider a system using a three point time series with three possible values. Monitor and trustee time series have 27 possible values: 000, 001, 002, . . . , 222. If we consider a monitor time series of 000 and a trustee time series of 000, \( c = 1 - 0/3 = 1 \). If we consider a monitor time series of 012 and a trustee time series of 021, \( c = 1 - 1/3 = 0.667 \). If we consider a monitor time series of 000 and a trustee time series of 111, \( c = 1 - 3/3 = 0 \).

Trust is a belief about if a node is trustworthy or not based on many trust metrics including cooperativeness, honesty, connectivity, capability (such as energy), competence, etc. [39]. Cooperativeness and honesty may be measured by protocol compliance in communication networks [39], so node compliance may also be used to measure cooperativeness and honesty in a CPS. Thus, node compliance is considered as part of trust since it can be used to measure cooperativeness and honesty, but it is not the whole of trust which may comprise a number of trust metrics.

## 6.4 Simulation

We collect compliance degree history \( c_1, c_2, \ldots, c_n \) of a device by means of Monte Carlo simulation. We use the VSM device in the reference MCPS defined in Section 6.2 to exemplify the utility of our IDS technique for secure MCPS applications. The Welch Allyn Connex 6000 is an example of a VSM that fits into our model.

Specifically, we simulate the procedure described in Section 6.3.2.6 to construct the state machines of a good VSM device and a bad VSM device. For a good VSM device, we simulate \( p_{ij} \) (see 6.3.2.6 for its definition) as \( 1 - p_{err} \) when \( j \) is the single good state, and as \( p_{err}/242 \) when \( j \) is one of the 242 bad states (treating both 31 warning and 211 unsafe states as bad). For a bad VSM device with random attack probability \( p_a \), we simulate \( p_{ij} \) as \( ((1 - p_a) \times (1 - p_{err}) + p_a \times p_{err}) \) when \( j \) is the single good state, and as \( (p_a \times (1 - p_{err}) + (1 - p_a) \times p_{err})/242 \) when \( j \) is one of the 242 bad states.

Given the state machine of a VSM device generated above, we collect a sequence of compliance degree values \( (c_1, c_2, \ldots, c_n) \) with \( n = 1000 \) Monte Carlo simulation test runs. In each simulation test run, we start from state 0 and then follow the stochastic process of this device as it goes from one state to another. We continue doing this until at least one state is traversed sufficiently (say 100 times). Then we calculate the limiting probability that the device is in state \( j \), \( \pi_j \), using the ratio of the number of transitions leading to state \( j \) to the total number of state transitions. Then we collect one instance of \( c \) using Equation 4.1. We repeat a sufficiently large \( n = 1000 \) test runs to collect \( c_1, c_2, \ldots, c_n \) needed for computing the distribution of the compliance degree of a good or a bad medical device performing reckless or random attacks.
Figure 6.4: Sensitivity of Good Node Compliance Degree to $p_{err}$.

Figure 6.5: Sensitivity of Bad Node Compliance Degree to $p_{err}$.

Figure 6.6: Sensitivity of Bad Node Compliance Degree to $p_a$. 

Chapter 6. Case Study: Medical CPS
Table 6.4: $\beta$ in Beta(1, $\beta$) and Resulting $p_{fn}$ and $p_{fp}$ Values under Various Attack Models for VSM ($C_T = 0.90, p_{err} = 0.01$).

<table>
<thead>
<tr>
<th>Attack Type</th>
<th>$\beta$</th>
<th>$p_{fn}$</th>
<th>$p_{fp}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reckless attacker ($p_a = 1.00$)</td>
<td>98.5</td>
<td>0.001%</td>
<td>14.8%</td>
</tr>
<tr>
<td>Random attacker ($p_a = 0.80$)</td>
<td>4.29</td>
<td>0.005%</td>
<td>14.8%</td>
</tr>
<tr>
<td>Random attacker ($p_a = 0.40$)</td>
<td>1.08</td>
<td>8.33%</td>
<td>14.8%</td>
</tr>
<tr>
<td>Random attacker ($p_a = 0.20$)</td>
<td>0.621</td>
<td>23.9%</td>
<td>14.8%</td>
</tr>
<tr>
<td>Random attacker ($p_a = 0.10$)</td>
<td>0.441</td>
<td>36.3%</td>
<td>14.8%</td>
</tr>
</tbody>
</table>

Figure 6.4 shows compliance degree raw data with $X = 1, 2, \ldots, n$ and $Y = c_1, c_2, \ldots, c_n$, for $n = 1000$ points, for a good VSM node with several $p_{err}$ values. There are five clouds of compliance degree data, one corresponding with each $p_{err}$ setting. We see that as $p_{err}$ (representing ambient noise) increases, the cloud of compliance degree data moves down, i.e., the compliance degree of the good node decreases. This is because as the noise increases, there is a higher probability of the monitoring node misidentifying the good state status of the good VSM node.

Figure 6.5 shows the sensitivity of $c_1, c_2, \ldots, c_n$ to $p_{err}$ for a bad VSM node. Like Figure 6.4, there are five clouds of compliance degree data, one corresponding with each $p_{err}$ setting. However, in this case as $p_{err}$ increases, the cloud of compliance degree data moves up, i.e., the compliance degree of the bad VSM node increases. This is because as the noise increases, there is a higher probability of the monitoring node misidentifying the bad state status of the bad VSM node.

Figure 6.6 shows the sensitivity of $c_1, c_2, \ldots, c_n$ to $p_a$, the random attack probability by a bad node. There are six clouds of compliance degree data, one corresponding with each $p_a$ setting. As $p_a$ increases, the cloud of compliance degree data moves down, i.e., the bad node’s compliance degree decreases. This is because as the bad VSM node performs more frequent attacks, it is more easily detected, so its measured compliance degree decreases.

With $c_1, c_2, \ldots, c_n$ of a good or bad VSM device in hand, we apply Equation 4.6 to compute the $\beta$ parameter value of $G(\cdot) = Beta(\alpha, \beta)$ for the probability distribution of the compliance degree for a good or a bad VSM device. We then calculate $p_{fn}$ and $p_{fp}$ by Equations 4.7 and 4.8, respectively, given the minimum compliance degree $C_T$ as input reflecting the consequence of false negatives over false positives for the VSM device. For an MCPS we prioritize achieving a low false negative probability because the key motivation is safety.

Table 6.4 shows the $\beta$ values and the resulting $p_{fn}$ and $p_{fp}$ values when $C_T = 0.9, p_{err} = 0.01$, and the binary grading strategy is being used to assign $c_j$ to state $j$. $C_T$ is a design parameter to be fine-tuned to trade high false positives for low false negatives due to safety criticality as described below. We observe that when the random attack probability $p_a$ is high, the
attacker can be easily detected, as evidenced by a low false negative probability. Especially when \( p_a = 1 \), a reckless attacker can hardly be missed. On the other hand, as \( p_a \) decreases, the attacker becomes more hidden and insidious, and the false negative probability increases. The false positive probability remains the same regardless of the random attack probability because it is a metric measuring the detection error against a good node only.

![Figure 6.7: Probability of False Negative vs. Compliance Threshold for Detecting Random Attacker](image1)

![Figure 6.8: Probability of False Negative vs. Compliance Threshold and \( p_{err} \) for Detecting Reckless Attackers.](image2)

Our behavior-rule based IDS allows one to adjust the minimum compliance degree threshold \( C_T \) to obtain an acceptable \( p_{fn} \) while keeping \( p_{fp} \) as low as possible.

Figure 6.7 shows the relationship between \( p_{fn} \) and \( C_T \) for detecting a random attacker with varying \( p_a \) values. Our intent is to analyze the effect of \( p_a \) on \( p_{fn} \). For each curve, \( p_{fn} = 1 \) at \( C_T = 0 \), and \( p_{fn} = 0 \) at \( C_T = 1 \). We see \( p_{fn} \) decreases as \( p_a \) increases because bad nodes are more likely to behave in a way that reveals their malintent.

Figure 6.8 shows the relationship between \( p_{fn} \) and \( C_T \) for detecting a reckless attacker (\( p_a = 1 \))
with varying $p_{\text{err}}$ values. Our intent is to analyze the effect of $p_{\text{err}}$ on $p_{\text{fn}}$. Like Figure 6.7, $p_{\text{fn}} = 1$ at $C_T = 0$, and $p_{\text{fn}} = 0$ at $C_T = 1$ for each curve. We see $p_{\text{fn}}$ decreases as $p_{\text{err}}$ decreases because noise is less likely to mask the malicious behavior of bad nodes.

Correspondingly, Figure 6.9 shows the relationship between $p_{\text{fp}}$ and $C_T$ for detecting a good node with varying $p_{\text{err}}$ values. Our intent is to analyze the effect of $p_{\text{err}}$ on $p_{\text{fp}}$. For each curve, $p_{\text{fp}} = 0$ at $C_T = 0$. $p_{\text{fp}}$ decreases as $p_{\text{err}}$ decreases because noise is less likely to distort the behavior of good nodes to appear malicious.

Combining the results in Figures 6.8 and 6.9, Figure 6.10 plots the ROC curves under different $p_{\text{err}}$ values for detecting a reckless attacker to illustrate the effect of $p_{\text{err}}$ on ROC. For a ROC curve (detection rate $1 - p_{\text{fn}}$ vs. false positive probability $p_{\text{fp}}$), the objective is to maximize the Area Under the Curve (AUC) such that the same detection rate can be achieved with a smaller $p_{\text{fp}}$ value. This figure illustrates AUC decreasing with $p_{\text{err}}$. This is because for a given data point, a larger $p_{\text{err}}$ increases the apparent compliance of bad
nodes which lowers the detection rate (moves the point down) and decreases the apparent compliance of good nodes which raises the false positive rate (moves the point to the right).

Figure 6.11: ROC Graph under Binary Grading for Detecting Random Attackers.

Figure 6.12: ROC Graph under Each Distance-Based Grading Strategy for Detecting Random Attackers.

6.5 Comparison of ROC under Binary and Distance-Based Grading Policies

By adjusting $C_T$, our specification-based IDS technique can effectively trade higher false positives off for lower false negatives to cope with more sophisticated and hidden random attackers. That is, by increasing $C_T$, one can effectively reduce $p_{fn}$ at the expense of $p_{fp}$. This
is especially desirable for ultra safe and secure MCPS applications for which a false negative may have a dire consequence. Figure 6.11 shows a ROC graph of intrusion detection rate \((1 - p_{fn})\) versus false positive probability \((p_{fp})\) under the binary grading policy, obtained as a result of adjusting \(C_T\). In Figure 6.11, there are several curves, one for each random attacker case with a different attack probability \(p_a\). We fix \(p_{err}\) to 0.01 to isolate out its effect. As we increase \(C_T\), the detection rate increases (vertically up on a ROC graph) while the false probability increases (toward the right of a ROC graph). We see that in this environment setting with our specification-based IDS technique, the detection rate of the VSM medical device can approach 100% for detecting attackers, that is, an attacker is always detected with probability 1 without false negatives, while bounding the false positive probability to below 5% (for reckless attackers) and 20% (for random attackers).

Figure 6.12 compares the performance of distance-based grading strategies over a range of attacker types. AUC is a common criterion for relating ROC graphs. Figure 6.12 shows that AUC increases as \(p_a\) increases; each distance-based grading strategy performs better for more aggressive attackers.

Figures 6.13 and 6.14 compare the performances of distance-based grading strategies for reckless and random attackers, respectively. For reckless attackers, Hamming grading performs the best followed by Euclidean, Manhattan, Levenshtein, Damerau-Levenshtein and LCS. However for random attackers, Levenshtein grading performs the best followed by Damerau-Levenshtein, Hamming, LCS, Euclidean and Manhattan. One aspect common to Hamming and Levenshtein measurements which may cause their high performance is that the magnitudes of the component differences do not inform their distance. Based on this, we expect Euclidean and Manhattan measurements to perform the worst, and Figure 6.14 supports this. One aspect that distinguishes Hamming from Levenshtein measurements which may account for each performing better for a different attacker is that Hamming grading operates instantaneously on multivariate data, and Levenshtein grading operates on a time series of univariate data. Since reckless attackers are always noncompliant, an audit at any moment should be able to reveal an attacker. Based on this, we expect LCS, Levenshtein and Damerau-Levenshtein measurements to perform worse than Hamming, Manhattan and Euclidean measurements for reckless attackers, and Figure 6.13 supports this. Since random attackers are compliant some fraction of the time, auditing a node over an interval of time may work better for these types of attackers. Based on this, we expect Hamming, Manhattan and Euclidean measurements to perform worse than LCS, Levenshtein and Damerau-Levenshtein for random attackers, and Figure 6.14 supports this.

### 6.6 Comparative Analysis

We perform a comparative study using the IDS design by Park et al. [112] as a baseline scheme. Their IDS scheme is designed for detecting abnormal patient behaviors in a pervasive healthcare system. We justify our comparison between the abnormal behavior of Park’s
Figure 6.13: ROC Graph under Distance-Based Grading for Detecting Reckless Attackers.

Figure 6.14: ROC Graph under Distance-Based Grading for Detecting Random Attackers.
patients and our reckless adversary because of the way they synthesized their abnormal
patients; time shifting data for normal patients is similar to a replay attack.

First, their IDS applies a similarity function, to grade four aspects of sensor data: LCS of
events ($s_1$), number of common events that are not part of the LCS ($s_2$), event start time
similarities ($s_3$) and event duration similarities ($s_4$). They experiment with two variants
each of $s_3$ and $s_4$: one considers events in the LCS ($s_D$ trials) and the other does not
($s_I$ trials). Their intent is to control the effects of interdependence between the similarity
measures. Second, their IDS calculates a threshold for classifying good and bad behavior
using a training data set. Third, the authors’ IDS determines the weight for each of the
four sensor data aspects. Park et al. measure patient activity using 3-tuple events, $e_i$, which
comprise <sensor ID, time, duration>. They form episodes, $E_i$, from sequences of events.
They use 70% of the dataset in [133] as normal training data and synthesize abnormal
training data by random generation and time shifting normal training data by four, eight
and 12 hours. They use the remaining 30% of the dataset as test data. Finally, they optimize
the performance by weighting the LCS ($s_1$) and duration ($s_4$) aspects of sensor data more
heavily than non-LCS common events ($s_2$) and start times ($s_3$) aspects. Figure 6.15 shows
the resulting ROC curves for several distinct configurations out of which we use the best
ROC curve for performance comparison.

![Figure 6.15: ROC Graph for Park’s IDS.](image)

Figure 6.15 compares the performance of Park’s design with our behavior-rule specification-
based intrusion detection (BSID) design using the LCS grading strategy and a reckless
attacker. BSID using the LCS grading strategy outperforms Park’s IDS. The AUC of BSID
using LCS dominantly covers that of Park’s IDS. This could be due to our longer history;
we use 100 events compared to 20. In addition to LCS ($s_1$), Park et al. consider three
other measurements of audit data: number of common events that are not part of the
LCS ($s_2$), event start time similarities ($s_3$) and event duration similarities ($s_4$). They use
a weighting scheme that optimizes the performance, but our preliminary results indicate
weights of 0 for $s_2$, $s_3$ and $s_4$ are best. This means one good measurement with a long
history outperforms multiple optimally weighted measurements. In Figure 6.16, we also
clearly see that BSID using the Hamming grading policy performs the best among all in detecting a reckless attacker.

6.7 Summary

For safety-critical MCPSs, being able to detect attackers while limiting the false alarm probability to protect the welfare of patients is of the utmost importance. In this chapter, we proposed a behavior-rule specification-based IDS technique for intrusion detection of medical devices embedded in an MCPS. We exemplified the utility with VSMs and demonstrated that the detection probability of the medical device approaches one (that is, we can always catch the attacker without false negatives) while bounding the false alarm probability to below 5% for reckless attackers and below 20% for random attackers. Through a comparative analysis, we demonstrated our behavior-rule specification-based IDS technique outperforms an existing technique based on anomaly intrusion detection.
Chapter 7

Case Study: Smart Grid CPS

In this chapter, we apply our IDS design to a smart grid CPS (SGCPS). Specifically, we propose a behavior-rule based smart grid intrusion detection system (SGIDS) for securing infrastructure (sensors or actuators) embedded in an SGCPS in which continuity of operation is of the utmost importance. We investigate the impact of attacker behaviors on the effectiveness of our behavior-rule intrusion detection design. Using head-ends (HEs), distribution access points/data aggregation points (DAPs) and smart meters (SMs) as examples, we demonstrate that our behavior-rule based intrusion detection technique can effectively trade false positives for a high detection probability to cope with sophisticated and hidden attackers to support ultra safe and secure SGCPS applications. We show that our SGIDS outperforms contemporary anomaly-based IDSs via comparative analysis. The content of this chapter derives from our publication [102].

7.1 Background

The most prominent characteristic of an SGCPS is its feedback loop that acts on the physical environment. In other words, the physical environment provides data to the SGCPS sensors whose data feed the SGCPS control algorithms that drive the actuators which change the physical environment. SGCPSs are often characterized by sophisticated reliability, efficiency, sustainability and utility algorithms interacting with the physical environment including subscriber appliances. In this chapter, we are concerned with intrusion detection mechanisms for detecting compromised sensors or actuators embedded in an SGCPS for supporting safe and secure SGCPS applications that subscribers can depend on with confidence.

IDS techniques for SGCPSs are still in their infancy with very little work reported in the literature. Only [16, 23, 35, 66, 74, 78, 91, 137, 148, 149, 159, 160] have reported on SGCPS intrusion detection. However, nine of these had no numerical data regarding the false negative probability \( p_{fn} \) (i.e., missing a bad node) and the false positive probability \( p_{fp} \) (i.e.,
misidentifying a good node as a bad node). The other three had minimal numerical data: one or two data points characterizing $p_{fa}/p_{fp}$ instead of a dataset that could be transformed into a ROC plot, i.e., a $p_{fa}$ versus $p_{fp}$ curve that describes the relationship between $p_{fa}$ and $p_{fp}$ obtained as a result of applying IDS techniques.

Specifically, Zhang et al. [159, 160] studied two detection algorithms called CLONALG and AIRS2Parallel. CLONALG is unsupervised. AIRS2Parallel is semi-supervised. They reported that CLONALG had a detection accuracy between 80.1% and 99.7% and AIRS2Parallel had an accuracy between 82.1% and 98.7%, where the detection accuracy is the likelihood that a node is classified correctly, calculated by $1 - p_{fp} - p_{fa}$. He and Blum [66] investigated a series of anomaly-based IDSs including Locally Optimum Unknown Direction (LOUD), Locally Optimum Estimated Direction (LOED), LOUD-Generalized Likelihood Ratio (LOUD-GLR) and LOED-Generalized Likelihood Ratio (LOED-GLR). He and Blum’s LOUD-GLR approach performed the best: The maximum detection rate (i.e., $1 - p_{fa}$) is reportedly 95%. However, no ROC data were given in [66, 159, 160].

Intrusion detection techniques in general can be classified into three types: signature-based, anomaly-based and specification-based techniques. In this chapter, we consider specification-based detection rather than signature-based detection to deal with unknown attacker patterns. We consider specification-based rather than anomaly-based techniques such as those by Zhang et al. [159, 160] and He and Blum [66] to avoid using resource constrained sensors or actuators in an SGCPS for profiling anomaly patterns (for example, through learning) and to avoid high false positives (treating good nodes as bad nodes).

To accommodate resource constrained sensors and actuators in an SG, we propose the design notion of behavior rules for specifying acceptable behaviors of nodes (sensors and actuators) in an SG. To the best of our knowledge, rule-based intrusion detection thus far has been applied only in the context of communication networks which have no concern of physical environments and the closed-loop control structure as in an SG. For example, Da Silva et al. [41] propose an IDS that applies seven types of traffic-based rules to detect intruders: interval, retransmission, integrity, delay, repetition, radio transmission range and jamming. Ioannis et al. [72] propose a multitrust IDS with traffic-based collection that audits the forwarding behavior of suspects to detect blackhole and greyhole attacks launched by captured devices based on the rate (versus the count) of specification violations.

Our contribution relative to prior work cited is that we specifically consider behavior rules for SG actuators controlling reliability, efficiency, sustainability and utility algorithms as well as for electrical, mechanical and environmental sensors providing information concerning the physical environment. Further, we propose a method to transform behavior rules to a state machine, so that a device that is being monitored for its behavior can be checked against the transformed state machine for deviation from its behavior specification. Existing work [37, 45] only considered specification-based state machines for intrusion detection of communication protocol misbehaving patterns. Untreated in the literature, in this chapter we also investigate the impact of attacker behaviors on the effectiveness of SGCPS intrusion.
detection. Using HEs, DAPs and SMs as examples, we demonstrate that our intrusion detection technique can effectively trade false positives for a high detection probability to cope with more sophisticated and hidden attackers to support ultra safe and secure SGCPS applications. Moreover, we show that our SGIDS design outperforms contemporary anomaly-based IDSs [66, 159, 160] via comparative analysis.

### 7.2 SGCPS Reference Model

We consider a smart electrical grid CPS embedding physical components as our SGCPS reference model as illustrated in Figure 7.1. For ease of disposition, we are particularly concerned with three types of sensor/actuator devices embedded in this SG: an HE, DAPs and SMs. Many SGCPS examples exist with these three devices. One example is a reliability function where phasor measurement units (PMUs) drive the opening and closing of switches to avoid interruptions. Another example is an efficiency function where usage data from SMs drive the billing rates and load control policies the HE dictates. A third example is a sustainability function where usage data from SMs drive the pitch setting for a wind distributed energy resource (DER) generator.

### 7.3 SGCPS Intrusion Detection Design

#### 7.3.1 Behavior Rules

Our IDS design for the reference SGCPS model relies on the use of lightweight specification-based behavior rules for each sensor/actuator component. They are oriented toward detecting an inside attacker attached to a specific physical component, provide a continuous output between 0 and 1 (to account for transient faults and human errors) and allow a monitor to perform intrusion detection on a neighbor trustee through monitoring. Here a monitor is itself a sensor/actuator device with capability to do intrusion detection on trustee nodes assigned to it. For example, an SM may monitor another SM within radio range. An HE may monitor other HE or DAP trustee devices within radio range. Therefore, an HE might have several sets of behavior rules (and thus several state machines), one for each trustee. Tables 7.1, 7.2 and 7.3 list the SGCPS behavior rules for the HE, DAPs and SMs. These tables specify the trustee and monitor devices for applying our IDS technique.

The behavior rule set specifies expected normal behaviors for each device and can detect deviation of normal behaviors regardless of the attacker’s patterns. It does not rely on knowledge of known attacker patterns as in signature-based intrusion detection.
Figure 7.1: Smart Grid CPS.
7.3.2 Transforming Rules to State Machines

The following procedure transforms a behavior specification into a state machine: First, we identify the “attack state” as a result of a behavior rule being violated. Then, we transform this attack state into a conjunctive normal form predicate and identify the involved state components in the underlying state machine. Next, for each device (that is, an HE, DAP or SM), we combine the attack states into a Boolean expression in disjunctive normal form. Then we transform the union of all predicate variables into the state components of a state machine and establish their corresponding ranges. Finally, we manage the number of states by state collapsing and identifying combinations of values that are not legitimate. Below we exemplify how a state machine is derived from the behavior specification in terms of behavior rules for the reference SGCPS model.

7.3.2.1 Identify Attack States

Attacks performed by a compromised sensor/actuator will drive the SGCPS into certain attack states identifiable through analyzing the specification-based behavior rules.

For the HE device, there are nine attack states as a result of violating the nine behavior rules for HEs listed in Table 7.1. The first HE attack state is that the HE activates a block of appliances, but the system demand is above some threshold. The second HE attack state is that the HE increases the duty cycle for a block of appliances, but the system demand is above some threshold. The third HE attack state is that the HE deactivates a block of appliances, but the system demand is below some threshold. The fourth HE attack state is that the HE decreases the duty cycle for a block of appliances, but the system demand is below some threshold. The fifth HE attack state is that the HE decreases the billing rate, but the system demand is above some threshold. The sixth HE attack state is that the HE increases the billing rate, but the system demand is below some threshold. The seventh HE attack state is that the HE opens the switch for a microgrid, but there is no associated fault or maintenance. The eighth HE attack state is that DERs are disconnected, but the system demand is above some threshold. The ninth HE attack state is that an interruption is present, but the HE has not generated an alert. For all of these HE attack states, the HE is the trustee and all DAPs are monitors.

For the DAP device, there are eight attack states as a result of violating the nine behavior rules for DAPs listed in Table 7.2. The first DAP attack state is that the HE requests a load increase, but microgrid demand is above some threshold. The second DAP attack state is that the HE requests a load decrease, but microgrid demand is below some threshold. For these first two DAP attack states, the HE is the trustee, and a DAP is the monitor. The third DAP attack state is that the microgrid is islanded, but there is no interruption. For the third DAP attack state, a DAP is the trustee, and the HE is the monitor. The fourth DAP attack state is that the number of packets forwarded by the DAP does not equal the
number of packets received by the DAP. This state corresponds with the two behavior rules concerning packet handling. For the fourth DAP attack state, a DAP is the trustee, and the HE and SMs are monitors. The fifth DAP attack state is that the community DER is not connected, but it is available. The sixth DAP attack state is that an interruption is present, but the DAP has not generated an alert. The seventh DAP attack state is that the DAP decreases the pitch of wind DER generator blades, but the microgrid demand is above some threshold. The eighth DAP attack state is that the DAP increases the pitch of wind DER generator blades, but the microgrid demand is below some threshold. For the fifth through eighth DAP attack states, a DAP is the trustee, and the HE is the monitor.

For the SM device, there are nine attack states as a result of violating the nine behavior rules for SMs listed in Table 7.3. The first SM attack state is that the SM is not generating usage data. The second SM attack state is that time-independent smart appliances are active, but the billing rate is above some threshold. The third SM attack state is that the subscriber is not banking electricity, but the billing rate is below some threshold. The fourth SM attack state is that time-independent smart appliances are active, but the demand is above some threshold. The fifth SM attack state is that the subscriber is not banking electricity, but the demand rate is below some threshold. The sixth SM attack state is that the subscriber DER is not connected, but it is available. The seventh SM attack state is that an interruption is present, but the SM has not generated an alert. The eighth SM attack state is that the SM decreases the pitch of wind DER generator blades, but the subscriber demand is above some threshold. The ninth SM attack state is that the SM increases the pitch of wind DER generator blades, but the subscriber demand is below some threshold. For all of these SM attack states, an SM is the trustee, and the DAP is the monitor.

7.3.2.2 Express Attack States in Conjunctive Normal Form

Tables 7.4, 7.5 and 7.6 list the attack states in conjunctive normal form for HE, DAP and SM nodes, respectively.

7.3.2.3 Consolidate Predicates in Disjunctive Normal Form

HE \(((\text{Appliance Block} = \text{ACTIVE}) \land (\text{System Demand} > \mu_d + \epsilon_d)) \lor ((\text{New Appliance Duty Cycle} > \text{Old Appliance Duty Cycle}) \land (\text{System Demand} > \mu_d + \epsilon_d)) \lor ((\text{Appliance Block} = \text{INACTIVE}) \land (\text{System Demand} < \mu_d - \epsilon_d)) \lor ((\text{New Appliance Duty Cycle} < \text{Old Appliance Duty Cycle}) \land (\text{System Demand} < \mu_d - \epsilon_d)) \lor ((\text{New Billing Rate} < \text{Old Billing Rate}) \land (\text{System Demand} > \mu_d + \epsilon_d)) \lor ((\text{New Billing Rate} > \text{Old Billing Rate}) \land (\text{System Demand} < \mu_d - \epsilon_d)) \lor ((\text{Switch Position} = \text{OPEN}) \land (\text{Fault} = \text{FALSE}) \land (\text{Maintenance} = \text{FALSE})) \lor ((\text{DER} = \text{DISCONNECTED}) \land (\text{System Demand} > \mu_d + \epsilon_d)) \lor ((\text{Interruption} = \text{TRUE}) \land (\text{Alert} = \text{NULL}))\)
Table 7.1: HE Behavior Rules

<table>
<thead>
<tr>
<th>Description</th>
<th>Trustee</th>
<th>Monitor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Turn off appliance block (for example, all water heaters in a microgrid) if system demand is above threshold</td>
<td>HE</td>
<td>HE</td>
</tr>
<tr>
<td>Decrease duty cycle ($t/T$, where $t =$ pulse width and $T =$ period) for appliance block if system demand is above threshold</td>
<td>HE</td>
<td>HE</td>
</tr>
<tr>
<td>Turn on appliance block if system demand is below threshold</td>
<td>HE</td>
<td>HE</td>
</tr>
<tr>
<td>Increase duty cycle ($t/T$, where $t =$ pulse width and $T =$ period) for appliance block if system demand is below threshold</td>
<td>HE</td>
<td>HE</td>
</tr>
<tr>
<td>Increase rate if system demand is above threshold</td>
<td>HE</td>
<td>HE</td>
</tr>
<tr>
<td>Decrease rate if system demand is below $\mu_d - \epsilon_d$</td>
<td>HE</td>
<td>HE</td>
</tr>
<tr>
<td>Close switch if no fault or maintenance</td>
<td>HE</td>
<td>HE</td>
</tr>
<tr>
<td>Connect DER to SGCPS if system demand is above $\mu_d + \epsilon_d$</td>
<td>HE</td>
<td>HE</td>
</tr>
<tr>
<td>If sensors indicate an interruption, notify affected nodes</td>
<td>HE</td>
<td>HE</td>
</tr>
</tbody>
</table>
### Table 7.2: DAP Behavior Rules

<table>
<thead>
<tr>
<th>Description</th>
<th>Trustee</th>
<th>Monitor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request subscriber load decrease if subscriber demand is above $\mu_d + \epsilon_d$</td>
<td>DAP</td>
<td>DAP, HE</td>
</tr>
<tr>
<td>Request subscriber load increase if subscriber demand is below $\mu_d - \epsilon_d$</td>
<td>DAP</td>
<td>DAP, HE</td>
</tr>
<tr>
<td>Island if bulk generation is interrupted</td>
<td>DAP</td>
<td>DAP, HE</td>
</tr>
<tr>
<td>Relay packets</td>
<td>DAP</td>
<td>DAP, HE</td>
</tr>
<tr>
<td>Don’t source (replay or inject) packets</td>
<td>DAP</td>
<td>DAP, HE</td>
</tr>
<tr>
<td>Use community DER generators if available</td>
<td>DAP</td>
<td>DAP, HE</td>
</tr>
<tr>
<td>If sensors indicate an interruption, notify affected nodes</td>
<td>DAP</td>
<td>DAP, HE</td>
</tr>
<tr>
<td>If demand above threshold, increase pitch of (unfurl) wind DER generator to maximize power</td>
<td>DAP</td>
<td>DAP, HE</td>
</tr>
<tr>
<td>If demand below threshold, decrease pitch of (furl) wind DER generator to maximize lifetime</td>
<td>DAP</td>
<td>DAP, HE</td>
</tr>
</tbody>
</table>

### Table 7.3: SM Behavior Rules

<table>
<thead>
<tr>
<th>Description</th>
<th>Trustee</th>
<th>Monitor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generate usage data periodically</td>
<td>SM</td>
<td>SM, DAP</td>
</tr>
<tr>
<td>Deactivate time independent appliances if rate is above $\mu_r + \epsilon_r$</td>
<td>SM</td>
<td>SM, DAP</td>
</tr>
<tr>
<td>Activate time independent appliances or store energy if rate is below $\mu_r - \epsilon_r$</td>
<td>SM</td>
<td>SM, DAP</td>
</tr>
<tr>
<td>Deactivate time independent appliances if microgrid demand is above $\mu_d + \epsilon_d$</td>
<td>SM</td>
<td>SM, DAP</td>
</tr>
<tr>
<td>Activate time independent appliances or store energy if microgrid demand is below $\mu_d - \epsilon_d$</td>
<td>SM</td>
<td>SM, DAP</td>
</tr>
<tr>
<td>Use subscriber DERs if available</td>
<td>SM</td>
<td>SM, DAP</td>
</tr>
<tr>
<td>If sensors indicate an interruption, notify affected nodes</td>
<td>SM</td>
<td>SM, DAP</td>
</tr>
<tr>
<td>If subscriber demand above threshold, increase pitch of (unfurl) wind DER generator to maximize power</td>
<td>SM</td>
<td>SM, DAP</td>
</tr>
<tr>
<td>If subscriber demand below threshold, decrease pitch of (furl) wind DER generator to maximize lifetime</td>
<td>SM</td>
<td>SM, DAP</td>
</tr>
</tbody>
</table>
Table 7.4: HE Attack States in Conjunctive Normal Form

| (Appliance Block = ACTIVE) \(\land\) (System Demand > \(\mu_d + \epsilon_d\)) |
| (New Appliance Duty Cycle > Old Appliance Duty Cycle) \(\land\) (System Demand > \(\mu_d + \epsilon_d\)) |
| (Appliance Block = INACTIVE) \(\land\) (System Demand < \(\mu_d - \epsilon_d\)) |
| (New Appliance Duty Cycle < Old Appliance Duty Cycle) \(\land\) (System Demand < \(\mu_d - \epsilon_d\)) |
| (New Billing Rate < Old Billing Rate) \(\land\) (System Demand > \(\mu_d + \epsilon_d\)) |
| (New Billing Rate > Old Billing Rate) \(\land\) (System Demand < \(\mu_d - \epsilon_d\)) |
| (Switch Position = OPEN) \(\land\) (Fault = FALSE) \(\land\) (Maintenance = FALSE) |
| (DER = DISCONNECTED) \(\land\) (System Demand > \(\mu_d + \epsilon_d\)) |
| (Interruption = TRUE) \(\land\) (Alert = NULL) |

Table 7.5: DAP Attack States in Conjunctive Normal Form

| (New Load Request > Old Load Request) \(\land\) (Microgrid Demand > \(\mu_d + \epsilon_d\)) |
| (New Load Request < Old Load Request) \(\land\) (Microgrid Demand < \(\mu_d - \epsilon_d\)) |
| (Island = FALSE) \(\land\) (Interruption = TRUE) |
| \(|\text{Forwarded Packets - Received Packets}\| > \epsilon_f\) |
| (DER Connection = FALSE) \(\land\) (DER Availability = TRUE) |
| (Interruption = TRUE) \(\land\) (Alert = NULL) |
| (New Pitch < Old Pitch) \(\land\) (Microgrid Demand > \(\mu_d + \epsilon_d\)) |
| (New Pitch > Old Pitch) \(\land\) (Microgrid Demand < \(\mu_d - \epsilon_d\)) |
Table 7.6: SM Attack States in Conjunctive Normal Form

<table>
<thead>
<tr>
<th>Time &gt; Usage.Timestamp + ( \epsilon )</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Appliance = ACTIVE) &amp; (Billing Rate &gt; ( \mu_r + \epsilon_r ))</td>
</tr>
<tr>
<td>(Energy Storage = FALSE) &amp; (Billing Rate &lt; ( \mu_r - \epsilon_r ))</td>
</tr>
<tr>
<td>(Appliance = ACTIVE) &amp; (Microgrid Demand &gt; ( \mu_d + \epsilon_d ))</td>
</tr>
<tr>
<td>(Energy Storage = FALSE) &amp; (Microgrid Demand &lt; ( \mu_d - \epsilon_d ))</td>
</tr>
<tr>
<td>(DER Connection = FALSE) &amp; (DER Availability = TRUE)</td>
</tr>
<tr>
<td>(Interruption = TRUE) &amp; (Alert = NULL)</td>
</tr>
<tr>
<td>(New Pitch &lt; Old Pitch) &amp; (Subscriber Demand &gt; ( \mu_d + \epsilon_d ))</td>
</tr>
<tr>
<td>(New Pitch &gt; Old Pitch) &amp; (Subscriber Demand &lt; ( \mu_d - \epsilon_d ))</td>
</tr>
</tbody>
</table>

**DAP** \((\text{New Load Request} > \text{Old Load Request}) \& (\text{Microgrid Demand} > \( \mu_d + \epsilon_d \)) \lor ((\text{New Load Request} < \text{Old Load Request}) \& (\text{Microgrid Demand} < \( \mu_d - \epsilon_d \)) \lor ((\text{Island} = \text{FALSE}) \& (\text{Interruption} = \text{TRUE})) \lor \left( |\text{Forwarded Packets} - \text{Received Packets}| > \epsilon_f \right) \lor ((\text{DER Connection} = \text{FALSE}) \& (\text{DER Availability} = \text{TRUE})) \lor ((\text{Interruption} = \text{TRUE}) \& (\text{Alert} = \text{NULL})) \lor ((\text{New Pitch} < \text{Old Pitch}) \& (\text{Microgrid Demand} > \( \mu_d + \epsilon_d \)) \lor ((\text{New Pitch} > \text{Old Pitch}) \& (\text{Microgrid Demand} < \( \mu_d - \epsilon_d \))))

**SM** \( (\text{Time} > \text{Usage.Timestamp} + \epsilon) \lor ((\text{Appliance} = \text{ACTIVE}) \& (\text{Billing Rate} > \( \mu_r + \epsilon_r \))) \lor ((\text{Energy Storage} = \text{FALSE}) \& (\text{Billing Rate} < \( \mu_r - \epsilon_r \))) \lor ((\text{Appliance} = \text{ACTIVE}) \& (\text{Microgrid Demand} > \( \mu_d + \epsilon_d \))) \lor ((\text{Energy Storage} = \text{FALSE}) \& (\text{Microgrid Demand} < \( \mu_d - \epsilon_d \))) \lor ((\text{DER Connection} = \text{FALSE}) \& (\text{DER Availability} = \text{TRUE})) \lor ((\text{Interruption} = \text{TRUE}) \& (\text{Alert} = \text{NULL})) \lor ((\text{New Pitch} < \text{Old Pitch}) \& (\text{Subscriber Demand} > \( \mu_d + \epsilon_d \))) \lor ((\text{New Pitch} > \text{Old Pitch}) \& (\text{Subscriber Demand} < \( \mu_d - \epsilon_d \))))

### 7.3.2.4 Identify State Components and Component Ranges

We quantize continuous components at integer scale in permissible ranges. For example, system demand is in the range of \([0, 1000 \text{ GW}]\) and duty cycle is in the range of \([0, 100\%]\). Table 7.7 shows a complete list of the permissible ranges of SGCPS state components. The resulting HE automaton has \(2 \times 1001 \times 101 \times 100 \times 2 \times 2 \times 2 \times 2 \times 2 = 1.294 \times 10^9\) states. The resulting DAP automaton has \(1001 \times 1001 \times 2 \times 2 \times 11 \times 11 \times 2 \times 2 \times 2 = 7.759 \times 10^9\) states. The resulting SM automaton has \(2^{2^2} \times 2 \times 100 \times 2 \times 1001 \times 2 \times 2 \times 2 \times 2 \times 91 \times 1001 = 2.506 \times 10^{21}\) states. All of these automata are too large; we deal with this state explosion in the next step.
Table 7.7: SGCPS State Components

<table>
<thead>
<tr>
<th>Name</th>
<th>Control or Reading</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Appliance block activation</td>
<td>Control</td>
<td>{false, true}</td>
</tr>
<tr>
<td>System demand</td>
<td>Reading</td>
<td>[0, 1000 GW]</td>
</tr>
<tr>
<td>Appliance duty cycle</td>
<td>Control</td>
<td>[0, 100%]</td>
</tr>
<tr>
<td>Billing rate</td>
<td>Control</td>
<td>(0, 1 USD/kWh]</td>
</tr>
<tr>
<td>Switch position</td>
<td>Control</td>
<td>{open, closed}</td>
</tr>
<tr>
<td>Fault</td>
<td>Reading</td>
<td>{false, true}</td>
</tr>
<tr>
<td>Maintenance</td>
<td>Control</td>
<td>{false, true}</td>
</tr>
<tr>
<td>DER connection</td>
<td>Control</td>
<td>{false, true}</td>
</tr>
<tr>
<td>Interruption</td>
<td>Reading</td>
<td>{false, true}</td>
</tr>
<tr>
<td>Alert</td>
<td>Control</td>
<td>{false, true}</td>
</tr>
<tr>
<td>Load request</td>
<td>Control</td>
<td>[0, 1000 kW]</td>
</tr>
<tr>
<td>Microgrid demand</td>
<td>Reading</td>
<td>[0, 1000 MW]</td>
</tr>
<tr>
<td>Island</td>
<td>Control</td>
<td>{false, true}</td>
</tr>
<tr>
<td>Forwarded packets</td>
<td>Reading</td>
<td>[0, 10/s]</td>
</tr>
<tr>
<td>Received packets</td>
<td>Reading</td>
<td>[0, 10/s]</td>
</tr>
<tr>
<td>DER availability</td>
<td>Reading</td>
<td>{false, true}</td>
</tr>
<tr>
<td>Wind DER generator pitch</td>
<td>Control</td>
<td>[0, 90°]</td>
</tr>
<tr>
<td>Usage age</td>
<td>Reading</td>
<td>[0, 2^{32}]</td>
</tr>
<tr>
<td>Appliance activation</td>
<td>Control</td>
<td>{false, true}</td>
</tr>
<tr>
<td>Energy storage</td>
<td>Control</td>
<td>{false, true}</td>
</tr>
<tr>
<td>Subscriber demand</td>
<td>Reading</td>
<td>[0, 1000 kW]</td>
</tr>
</tbody>
</table>
7.3.2.5 Manage State Space

To manage the number of states, we reduce the size of the state machine by abbreviating the values for some components. Only three values are relevant for system, microgrid and subscriber demand: below threshold, normal and above threshold. Therefore, we collapse the domain for these components to three values for the HE, DAP and SM, respectively. This treatment yields a modest HE state machine with $2 \times 3 \times 3 \times 2 \times 2 \times 2 \times 2 = 3456$ states, out of which 1008 are identified as safe states and 2448 are unsafe states. Only two values are relevant for networking: whether or not packets forwarded and packets received differ by more than some threshold. Therefore, we collapse the domain for this component to two values. This treatment yields a modest DAP state machine with $3 \times 3 \times 2 \times 2 \times 2 \times 2 \times 3 = 1728$ states, out of which 255 are identified as safe states and 1473 are unsafe states. Only three values are relevant for rate: below threshold, normal and above threshold. Also, only two values are relevant for usage reporting: current or missing. Therefore, we collapse the domains for these components to three and two values, respectively. This treatment yields a modest SM state machine with $2 \times 2 \times 3 \times 2 \times 3 \times 2 \times 2 \times 2 \times 3 = 3456$ states, out of which 396 are identified as safe states and 3060 are unsafe states.

7.3.2.6 Behavior Rule State Machines

Here we describe how to generate the behavior rule state machine of an SGCPS device. We use the HE device as an example. The HE state machine consisting of 1008 safe and 2448 unsafe states based on the behavior rules is generated as follows. First, we label these states as $1, 2, \ldots, n = 3456$. Next, we assign $p_{ij}$, the probability that state $i$ goes to state $j$, for each $(i, j)$ pair in the state machine to reflect a good or bad HE’s behavior.

A good HE should stay in safe states 100% of the time. This will give the compliance degree of a good HE close to one. However, occasionally it may be detected by the monitor node as staying in a warning or unsafe state due to ambient noise resulting from unexpected environment or system condition changes, as well as wireless communication faults. Let $p_{\text{err}}$ be the error probability of a monitor node misidentifying the status of a trustee node due to ambient noise and wireless communication faults. During the testing phase, we seed a good HE in the system and assign a monitor node to observe and measure $p_{ij}$ of the good HE in the presence of the error probability $p_{\text{err}}$: $p_{ij}$ is $2448/(2448+1008) \times p_{\text{err}}$ when $j$ is one of the 2448 unsafe states, and $p_{ij}$ is $1008/(2448+1008) \times (1 - p_{\text{err}})$ when $j$ is one of the 1008 good states. Figure 7.2 illustrates the behavior rule state machine for a good HE in the SGCPS. One dotted slash and crossed dotted slashes over a state indicate a warning state and an unsafe state, respectively. Each state component represents how one of the trustee node attributes matches its counterpart from the monitor. For the HE device, appliance block enable, system demand, duty cycle trend, rate adjustment, switch position, fault condition, maintenance condition, DER connection, interruption condition and alert condition are the device attributes of interest. Note that each device has its own state machine with device-
specific attributes being the state components of the state machine.

![State Machine Diagram](image)

**Figure 7.2:** Good HE Behavior Rule State Machine.

![State Machine Diagram](image)

**Figure 7.3:** Random Attacker HE Behavior Rule State Machine.

For a compromised HE, $p_{ij}$ depends on its attacker type: A reckless attacker presumably will stay in unsafe or warning states 100% of the time; however, occasionally it may be detected by the monitor node as staying in a safe state due to ambient noise and wireless communication faults. During the testing phase, we seed a reckless attacker in the system following its attacker profile and assign a monitor node to observe and measure $p_{ij}$: aggregate $p_{ij}$ is $1 - p_{err}$ when $j$ is one of the 2448 unsafe states. $p_{err}$ is the error probability of misidentifying the status of a reckless attacker due to ambient noise and wireless communication faults. For a random attacker with attack probability $p_a$, aggregate $p_{ij}$ is $p_a \times (1 - p_{err}) + (1 - p_a) \times p_{err}$ when $j$ is one of the 2448 bad states and $p_a \times p_{err} + (1 - p_a) \times (1 - p_{err})$ when $j$ is one of the 1008 good states. We note that a random attacker with attack probability $p_a$ will stop attacking with probability $1 - p_a$, which will be detected by the monitor node with probability $1 - p_{err}$. Figure 7.3 illustrates the behavior rule state machine for a random attacker HE in the SGCPS.
7.3.3 Collect Compliance Degree Data

Our SGIDS relies on the use of monitor nodes, e.g., an SM or a DAP is a monitor node of another SM. The monitor node knows the state machine of the trustee node assigned to it. The monitor node periodically measures the amount of time the trustee node stays in safe and unsafe states as the trustee node migrates from one state to another triggered by events causing state transitions. We consider a binary grading policy, i.e., assigning a compliance degree of 1 to a safe state and 0 to an unsafe state. Let $c$ be the compliance degree of a device. The compliance degree $c$ of a device essentially is equal to the proportion of the time the device is in safe states. Let $S$ be the set of safe states the trustee node traverses over a period of time $T$. Let $t_i$ be the sojourn time that the trustee node stays in a safe state $i$, as measured by the monitor node. Then the monitor node collects an instance of $c$ by:

$$c = \frac{\sum_{i \in S} t_i}{T}$$  \hspace{1cm} (7.1)

If a node stays only in safe states during $T$, then by Equation 7.1, its compliance degree $c$ is one. On the other hand, if a node stays only in unsafe states only during $T$, then its compliance degree $c$ is zero. The monitor node monitors and collects the trustee node’s compliance degree history $c_1, c_2, \ldots, c_n$ for $n$ monitoring periods, where $n$ is sufficiently large, based on which it concludes whether or not the trustee node is compromised.

We leverage the state machines generated to collect compliance degree data of a good and a bad node. With Equation 7.1, the compliance degree $c$ is essentially equal to the sum of the probabilities of safe states i.e., $c = \sum_{j \in S} \pi_j$, where $\pi_j$ is the limiting probability that the node is in state $j$ of the state machine and $S$ is the set of safe states in the state machine. We then collect compliance degree history $c_1, c_2, \ldots, c_n$ of a node by means of Monte Carlo simulation. That is, given a good (or a bad) node’s state machine we start from state 0 and then follow the stochastic process of this node as it goes from one state to another. We continue doing this until at least one state is reentered sufficiently often (say 100 times). Then we calculate $\pi_j$ using the ratio of the number of transitions leading to state $j$ to the total number of state transitions. Then we collect one instance of compliance degree. We repeat a sufficiently large $n$ test runs to collect $c_1, c_2, \ldots, c_n$ needed for computing the distribution of the compliance degree of a normal node or a node controlled by a reckless or random attacker.

7.4 Simulation

We collect compliance degree history $c_1, c_2, \ldots, c_n$ of a device by means of Monte Carlo simulation. We use the HE device in the reference SGCPS defined in Section 7.2 to exemplify the utility of our IDS technique for secure SGCPS applications.
Specifically, we simulate the procedure described in Section 7.3.2.6 to construct the state machines of a good HE device and a bad HE device. For a good HE device, we simulate $p_{ij}$ (see 7.3.2.6 for its definition) as $(1 - p_{err})/1008$ when $j$ is one of the 1008 good states, and as $p_{err}/2448$ when $j$ is one of the 2448 bad states. For a bad HE device with random attack probability $p_{a}$, we simulate $p_{ij}$ as $((1 - p_{a}) \times (1 - p_{err}) + p_{a} \times p_{err})/1008$ when $j$ is one of the 1008 good states, and as $(p_{a} \times (1 - p_{err}) + (1 - p_{a}) \times p_{err})/2448$ when $j$ is one of the 2448 bad states.

Given the state machine of an HE device generated above, we collect a sequence of compliance degree values $(c_{1}, c_{2}, \ldots, c_{n})$ with $n = 1000$ Monte Carlo simulation test runs. In each simulation test run, we start from state 0 and then follow the stochastic process of this device as it goes from one state to another. We continue doing this until at least one state is traversed sufficiently (say 100 times). Then we calculate the limiting probability that the device is in state $j$, $\pi_{j}$, using the ratio of the number of transitions leading to state $j$ to the total number of state transitions. Then we collect one instance of $c$ using Equation 7.1. We repeat a sufficiently large $n = 1000$ test runs to collect $c_{1}, c_{2}, \ldots, c_{n}$ needed for computing the distribution of the compliance degree of a good or a bad HE performing reckless or random attacks.

Figure 7.4: Sensitivity of Good Node Compliance Degree to $p_{err}$.

Figure 7.4 shows compliance degree raw data with $X = 1, 2, \ldots, n$ and $Y = c_{1}, c_{2}, \ldots, c_{n}$, for $n = 1000$ points, for a good HE node with several $p_{err}$ values. There are five clouds of compliance degree data, one corresponding with each $p_{err}$ setting. We see that as $p_{err}$ (representing ambient noise) increases, the cloud of compliance degree data moves down, i.e., the compliance degree of the good node decreases. This is because as the noise increases, there is a higher probability of the monitoring node misidentifying the good state status of the good HE node.

Figure 7.5 shows the sensitivity of $c_{1}, c_{2}, \ldots, c_{n}$ to $p_{err}$ for a bad HE node. Like Figure 7.4, there are five clouds of compliance degree data, one corresponding with each $p_{err}$ setting. However, in this case as $p_{err}$ increases, the cloud of compliance degree data moves up, i.e.,
Figure 7.5: Sensitivity of Bad Node Compliance Degree to $p_{err}$.

Figure 7.6: Sensitivity of Bad Node Compliance Degree to $p_a$. 
Chapter 7. Case Study: Smart Grid CPS

Table 7.8: $\beta$ in Beta(1,$\beta$) and Resulting $p_{fn}$ and $p_{fp}$ Values under Various Attack Models for HE ($C_T = 0.90, p_{err} = 0.01$).

<table>
<thead>
<tr>
<th>Attack Type</th>
<th>$\beta$</th>
<th>$p_{fn}$</th>
<th>$p_{fp}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reckless ($p_a = 1.00$)</td>
<td>99.5</td>
<td>&lt;0.001%</td>
<td>2.30%</td>
</tr>
<tr>
<td>Random ($p_a = 0.80$)</td>
<td>4.33</td>
<td>0.0047%</td>
<td>2.30%</td>
</tr>
<tr>
<td>Random ($p_a = 0.40$)</td>
<td>1.10</td>
<td>7.99%</td>
<td>2.30%</td>
</tr>
<tr>
<td>Random ($p_a = 0.20$)</td>
<td>0.633</td>
<td>23.3%</td>
<td>2.30%</td>
</tr>
<tr>
<td>Random ($p_a = 0.10$)</td>
<td>0.449</td>
<td>35.5%</td>
<td>2.30%</td>
</tr>
<tr>
<td>Random ($p_a = 0.05$)</td>
<td>0.353</td>
<td>44.3%</td>
<td>2.30%</td>
</tr>
</tbody>
</table>

Table 7.9: $\beta$ in Beta(1,$\beta$) and Resulting $p_{fn}$ and $p_{fp}$ Values under Various Attack Models for DAP ($C_T = 0.90, p_{err} = 0.015$).

<table>
<thead>
<tr>
<th>Attack Type</th>
<th>$\beta$</th>
<th>$p_{fn}$</th>
<th>$p_{fp}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reckless ($p_a = 1.00$)</td>
<td>49.6</td>
<td>&lt;0.001%</td>
<td>4.59%</td>
</tr>
<tr>
<td>Random ($p_a = 0.80$)</td>
<td>4.19</td>
<td>0.0064%</td>
<td>4.59%</td>
</tr>
<tr>
<td>Random ($p_a = 0.40$)</td>
<td>1.10</td>
<td>7.89%</td>
<td>4.59%</td>
</tr>
<tr>
<td>Random ($p_a = 0.20$)</td>
<td>0.644</td>
<td>22.7%</td>
<td>4.59%</td>
</tr>
<tr>
<td>Random ($p_a = 0.10$)</td>
<td>0.464</td>
<td>34.3%</td>
<td>4.59%</td>
</tr>
<tr>
<td>Random ($p_a = 0.05$)</td>
<td>0.372</td>
<td>42.5%</td>
<td>4.59%</td>
</tr>
</tbody>
</table>

the compliance degree of the bad HE node increases. This is because as the noise increases, there is a higher probability of the monitoring node misidentifying the bad state status of the bad HE node.

Figure 7.6 shows the sensitivity of $c_1, c_2, \ldots, c_n$ to $p_a$, the random attack probability by a bad node. There are six clouds of compliance degree data, one corresponding with each $p_a$ setting. As $p_a$ increases, the cloud of compliance degree data moves down, i.e., the bad node’s compliance degree decreases. This is because as the bad HE node performs more frequent attacks, it is more easily detected, so its measured compliance degree decreases.

With $c_1, c_2, \ldots, c_n$ of a good or bad HE device in hand, we apply Equation 4.6 to compute the $\beta$ parameter value of $G(\cdot) = Beta(\alpha, \beta)$ for the probability distribution of the compliance degree for a good or a bad HE device. We then calculate $p_{fn}$ and $p_{fp}$ by Equations 4.7 and 4.8, respectively, given the minimum compliance degree $C_T$ as input reflecting the consequence of false negatives over false positives for the HE device. For an SGCPS we prioritize achieving a low false negative probability because the key motivation is safety.

Tables 7.8, 7.9 and 7.10 show the $\beta$ values and the resulting $p_{fn}$ and $p_{fp}$ values when $C_T$
Table 7.10: $\beta$ in Beta($1,\beta$) and Resulting $p_{fn}$ and $p_{fp}$ Values under Various Attack Models for SM ($C_T = 0.90, p_{err} = 0.02$).

<table>
<thead>
<tr>
<th>Attack Type</th>
<th>$\beta$</th>
<th>$p_{fn}$</th>
<th>$p_{fp}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reckless ($p_a = 1.00$)</td>
<td>32.8</td>
<td>$&lt;0.001%$</td>
<td>6.87%</td>
</tr>
<tr>
<td>Random ($p_a = 0.80$)</td>
<td>4.06</td>
<td>0.0086%</td>
<td>6.87%</td>
</tr>
<tr>
<td>Random ($p_a = 0.40$)</td>
<td>1.11</td>
<td>7.78%</td>
<td>6.87%</td>
</tr>
<tr>
<td>Random ($p_a = 0.20$)</td>
<td>0.656</td>
<td>22.1%</td>
<td>6.87%</td>
</tr>
<tr>
<td>Random ($p_a = 0.10$)</td>
<td>0.479</td>
<td>33.2%</td>
<td>6.87%</td>
</tr>
<tr>
<td>Random ($p_a = 0.05$)</td>
<td>0.390</td>
<td>40.7%</td>
<td>6.87%</td>
</tr>
</tbody>
</table>

is 0.9 ($C_T$ is a design parameter to be fine-tuned to trade high false positives for low false negatives). Because the expected compliance degree following a Beta($\alpha, \beta$) distribution is $\alpha/(\alpha + \beta)$ as given by Equation 4.4, we see that $\beta$ is close to 0 for a good node or a hidden bad node with a low attack probability (e.g., $p_a = 0.05$) since such a node will have the average compliance degree close to 1. On the other hand, $\beta$ is much larger than 0 for a bad node with a high attack probability (e.g., $p_a = 1$) since such a node will have the average compliance degree much lower than 1.

Figure 7.7: Probability of False Negative vs. Compliance Threshold and $p_a$ for Detecting Random Attackers for HE.

Figures 7.7, 7.8 and 7.9. show the relationship between $p_{fn}$ and $C_T$ for detecting a random attacker with varying $p_a$ values for the three different node types. Our intent is to analyze the effect of $p_a$ on $p_{fn}$. For each curve, $p_{fn} = 1$ at $C_T = 0$ and $p_{fn} = 0$ at $C_T = 1$. We see $p_{fn}$ decreases as $p_a$ increases because bad nodes are more likely to behave in a way that reveals their malintent.

Correspondingly, Figure 7.10 shows the relationship between $p_{fp}$ and $C_T$ for detecting a good node for the three different node types. Our intent is to analyze the effect of node type on
Figure 7.8: Probability of False Negative vs. Compliance Threshold and $p_a$ for Detecting Random Attackers for DAP.

Figure 7.9: Probability of False Negative vs. Compliance Threshold and $p_a$ for Detecting Random Attackers for SM.

Figure 7.10: Probability of False Positive vs. Compliance Threshold for Detecting Good Nodes.
For each curve, \( p_{fp} = 0 \) at \( C_T = 0 \). \( p_{fp} \) is lowest for HE and highest for SM due to their corresponding \( p_{err} \) (0.01, 0.015 and 0.02 for HEs, DAPs and SMs, respectively). Lower \( p_{err} \) equates to lower \( p_{fp} \) because noise is less likely to distort the behavior of good nodes to appear malicious.

Figure 7.11: HE Receiver Operating Characteristic Graph.

By adjusting \( C_T \), our specification-based IDS technique can effectively trade higher false positives for lower false negatives to cope with more sophisticated and hidden random attackers. This is especially desirable for ultra safe and secure SGCPS applications for which a false negative may have a dire consequence. Figure 7.11 shows a ROC graph of intrusion detection rate (i.e., \( 1 - p_{fn} \)) versus false positive probability (\( p_{fp} \)) obtained as a result of adjusting \( C_T \). In Figure 7.11, there are several curves for each node type, one for each random attacker case with a different attack probability \( p_a \). As we increase \( C_T \), the detection rate increases (vertically up on a ROC graph) while the false probability increases (toward the right of a ROC graph). We see that with our specification-based IDS technique, the detection rate of the SGCPS node can approach 100% for detecting attackers, that is, an attacker is always detected with probability 1 without false negatives, while bounding the false positive probability to below 0.2% for reckless attackers and below 6% for random attackers.

7.5 Comparative Analysis

We compare the performance of our SGIDS with contemporary anomaly-based IDSs for SGCPSs, including CLONALG and AIRS2Parallel [159, 160], Locally Optimum Unknown Direction (LOUD), Locally Optimum Estimated Direction (LOED), LOUD-Generalized Likelihood Ratio (LOUD-GLR) and LOED-Generalized Likelihood Ratio (LOED-GLR) [66].

Zhang et al. [160] reported that CLONALG had a false positive rate of 0.7% and a false negative rate of 21.02% and AIRS2Parallel had a false positive rate of 1.3% and a false negative rate of 26.32%. Zhang et al. [159] further compared the effectiveness of audit
Table 7.11: Comparison Results for HE.

<table>
<thead>
<tr>
<th>Approach</th>
<th>Detection Accuracy</th>
<th>Device</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLONALG SG [160]</td>
<td>$100 - 0.7 - 21.02 = 78.28%$</td>
<td>HE</td>
</tr>
<tr>
<td>AIRS2Parallel SG [160]</td>
<td>$100 - 1.3 - 26.32 = 78.38%$</td>
<td>HE</td>
</tr>
<tr>
<td>LOUD [66]</td>
<td>$100 - 0.1 - 9 = 90.90%$</td>
<td>HE</td>
</tr>
<tr>
<td>LOED [66]</td>
<td>$100 - 0.1 - 16 = 83.90%$</td>
<td>HE</td>
</tr>
<tr>
<td>LOUD-GLR [66]</td>
<td>$100 - 0.1 - 5 = 94.90%$</td>
<td>HE</td>
</tr>
<tr>
<td>LOED-GLR [66]</td>
<td>$100 - 0.1 - 9 = 90.90%$</td>
<td>HE</td>
</tr>
<tr>
<td>CLONALG WIDS [159]</td>
<td>$99.70%$</td>
<td>HE</td>
</tr>
<tr>
<td>AIRS2Parallel WIDS [159]</td>
<td>$91.50%$</td>
<td>HE</td>
</tr>
<tr>
<td>SGIDS ($C_T = 0.50$)</td>
<td>$100 - 0.70 - 0.00 = 99.30%$</td>
<td>HE</td>
</tr>
<tr>
<td>SGIDS ($C_T = 0.73$)</td>
<td>$100 - 1.30 - 0.00 = 98.70%$</td>
<td>HE</td>
</tr>
<tr>
<td>SGIDS ($C_T = 0.09$)</td>
<td>$100 - 0.10 - 0.01 = 99.89%$</td>
<td>HE</td>
</tr>
</tbody>
</table>

Table 7.12: Comparison Results for DAP.

<table>
<thead>
<tr>
<th>Approach</th>
<th>Detection Accuracy</th>
<th>Device</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLONALG NIDS [159]</td>
<td>[80.10, 97.00%]</td>
<td>DAP</td>
</tr>
<tr>
<td>AIRS2Parallel NIDS [159]</td>
<td>[82.10, 96.10%]</td>
<td>DAP</td>
</tr>
<tr>
<td>SGIDS ($C_T = 0.37$)</td>
<td>$100 - 0.70 - 0.00 = 99.30%$</td>
<td>DAP</td>
</tr>
<tr>
<td>SGIDS ($C_T = 0.58$)</td>
<td>$100 - 1.30 - 0.00 = 98.70%$</td>
<td>DAP</td>
</tr>
<tr>
<td>SGIDS ($C_T = 0.06$)</td>
<td>$100 - 0.10 - 1.68 = 98.22%$</td>
<td>DAP</td>
</tr>
</tbody>
</table>

The authors provided no $p_n$ and $p_f$ information, but presumably the worst detection accuracy is obtained when $p_f$ is very low. He and Blum [66] investigated LOUD, LOED, LOUD-GLR and LOED-GLR approaches to anomaly-based IDS. They fixed the false positive probability (i.e., $p_f$) at 0.1% and showed that the detection rate (i.e., $1 - p_n$) for each approach varies over a wide range based on the parameterization. The LOUD-GLR approach reportedly performs the best with the detection accuracy of $100 - 0.1 - 5 = 94.9\%$.

Tables 7.11, 7.12 and 7.13 summarize the comparative performances among contemporary SGCPS IDSs for HE, DAP and SM devices, respectively. The performance metric is detection accuracy defined as $1 - p_f - p_n$. For cases where $p_n$ and $p_f$ are reported [66, 160], we show the
Table 7.13: Comparison Results for SM.

<table>
<thead>
<tr>
<th>Approach</th>
<th>Detection Accuracy</th>
<th>Device</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLONALG HIDS [159]</td>
<td>[93.90, 99.30%]</td>
<td>SM</td>
</tr>
<tr>
<td>AIRS2Parallel HIDS [159]</td>
<td>[95.10, 98.70%]</td>
<td>SM</td>
</tr>
<tr>
<td>SGIDS ($C_T = 0.29$)</td>
<td>$100 - 0.70 - 0.00 = 99.30%$</td>
<td>SM</td>
</tr>
<tr>
<td>SGIDS ($C_T = 0.47$)</td>
<td>$100 - 1.30 - 0.00 = 98.70%$</td>
<td>SM</td>
</tr>
<tr>
<td>SGIDS ($C_T = 0.05$)</td>
<td>$100 - 0.10 - 7.82 = 92.08%$</td>
<td>SM</td>
</tr>
</tbody>
</table>

detection accuracy value following the $1 - p_{fp} - p_{fn}$ format. For cases where $p_{fn}$ and $p_{fp}$ are not reported [159], we only show the detection accuracy value or a range of detection accuracy values. For comparison, we configure our adversary with $p_a = 1$ (reckless attacker). We show SGIDS performance for $C_T = 0.50$ for HE, $C_T = 0.37$ for DAP and $C_T = 0.29$ for SM to approximate the CLONALG $p_{fp}$ of 0.7% [160]. We show SGIDS performance for $C_T = 0.73$ for HE, $C_T = 0.58$ for DAP and $C_T = 0.47$ for SM to approximate the AIRS2Parallel $p_{fp}$ of 1.3% [160]. We show SGIDS performance for $C_T = 0.09$ for HE, $C_T = 0.06$ for DAP and $C_T = 0.05$ for SM to approximate the LOUD, LOED, LOUD-GLR and LOED-GLR $p_{fp}$ of 0.1% [66].

Tables 7.11, 7.12 and 7.13 support the claim that by effectively adjusting $C_T$ to trade false positives for low false negatives, our behavior-rule specification-based SGIDS outperforms existing anomaly-based IDS approaches, especially for HE and DAP devices.

### 7.6 Summary

For smart grids, being able to detect attackers while limiting the false positive probability to protect the continuity of operation is of the utmost importance. In this chapter, we proposed a behavior-rule specification-based IDS technique for intrusion detection of nodes (sensors or actuators) embedded in a smart grid system. We exemplified the utility by head-ends, distribution access points/data aggregation points and smart meters and demonstrated that the detection probability approaches one (that is, we can always catch the attacker without false negatives) while bounding the false positive probability to below 0.2% for reckless attackers and below 6% for random attackers (that is, we can bound the probability of misidentifying a good node as a bad node to a very low level). Through a comparative analysis, we demonstrated our behavior-rule specification-based IDS technique outperforms existing anomaly-based IDS approaches for detecting intruders in smart grid systems.
Chapter 8

Case Study: Unmanned Aircraft CPS

In this chapter, we apply our IDS design to an unmanned aircraft CPS (UACPS). Specifically, we propose an adaptive specification-based intrusion detection system for securing infrastructure (sensors or actuators) embedded in a UACPS in which continuity of operation is of the utmost importance. We investigate the impact of reckless, random and opportunistic attacker behaviors on the effectiveness of our intrusion detection technique. Using unmanned air vehicles (UAVs) as examples, we demonstrate that our adaptive intrusion detection technique can effectively trade false positives for a high detection probability to cope with more sophisticated random and opportunistic attackers to support ultra safe and secure UACPS applications. Through a comparative analysis, we demonstrate a high detection accuracy of our UAV intrusion detection technique. The content of this chapter derives from our publications [99, 101].

8.1 Background

UACPSs comprise a large part of the warfighting capability of modern militaries. Also, they are emerging in civilian applications such as surveillance for law enforcement, situational awareness for emergency services, content for news outlets and data collection for researchers. While they pose the same risk as piloted aircraft, the operator is removed from the vehicle in time and space; this calls for enhanced automated security systems to guarantee the safe operation of UACPSs.

Intrusion detection techniques in general can be classified into three types: signature-based, anomaly-based and specification-based techniques. This work proposed and measured a specification-based IDS for UACPSs. It requires minimal run time resources, detects unknown attacks and effectively trades between false positive and detection rates. We consider specification-based detection rather than signature-based detection to deal with unknown attacks. We consider specification-based rather than anomaly-based techniques to avoid high
false positives in mission-critical UACPSs.

The state of the art in intrusion detection for UACPSs is limited. Only [82], [83] and [139] have studied this topic. Trafton and Pizzi [139] described the role of intrusion detection in UACPS applications. The authors proposed Joint Airborne Network Services Suite (JANSS) which provides a framework to integrate an airborne military network. Among other services, JANSS covers intrusion detection. However, neither concrete solutions nor detection rate data were reported. Lauf and Robinson [82, 83] investigated HybrIDS, an anomaly-based approach, for UACPSs. HybrIDS comprises two intrusion detection methods: Maxima Detection System (MDS) and Cross-Correlative Detection System (CCDS). MDS detects single intruders using audit data after a short training phase and accomplishes a more in-depth training phase for CCDS. CCDS can detect cooperating intruders after the longer training phase provided by MDS. HybrIDS was evaluated using a metric called “pervasion” defined as the percentage of bad nodes in the system. It was reported that intruders can be detected even with a 22% pervasion. Existing work [82, 83, 139] had no numerical data regarding the false negative probability $p_{fn}$ (i.e., missing a bad node) and the false positive probability $p_{fp}$ (i.e., misidentifying a good node as a bad node). Our work is the first to generate a ($p_{fn}$, $p_{fp}$) UACPS intrusion detection system (IDS) dataset that could be transformed into a ROC plot ($1 - p_{fn}$ versus $p_{fp}$) that describes the tradeoff between $p_{fn}$ and $p_{fp}$ obtained as a result of applying IDS techniques.

The UAV Intrusion Detection System (UIDS) we propose is adaptive, i.e., it can adapt to the attacker type and environmental changes by adjusting its intrusion detection strength dynamically, so as to satisfy the maximum $p_{fn}$ requirement while minimizing $p_{fp}$. We demonstrate that our adaptive intrusion detection technique can effectively trade false positives for a high detection probability to cope with sophisticated random and opportunistic attackers to support ultra safe and secure UACPS applications. Through a comparative analysis, we demonstrate a high detection accuracy of our UAV intrusion detection technique.

8.2 UACPS Reference Model

![Figure 8.1: Reference UACPS Key Components.](image-url)
We consider a UACPS comprising tens or hundreds of UAVs each embedding CPS physical components (sensors and actuators) as a UACPS reference model. For example, the United States Air Force 42d Attack Squadron has 18 assigned [52]. Figure 8.1 illustrates the reference UACPS. One cyber physical loop in this model is the flight control system in each UAV. Inertial sensors drive realtime adjustment of control surfaces and thrust. In addition, locative sensors (navigational components), such as Global Positioning System (GPS), Global Navigation Satellite System (GLONASS), Compass, Galileo or inertial sensors drive nonrealtime adjustment of control surfaces and thrust. Another cyber physical loop in this model is the threat countermeasures system. Radar components detect physical presence of threats, and specifically tuned radios detect RF signatures of threats. These sensors drive the realtime deployment of countermeasures like flare, chaff and electronic countermeasures (ECM). On top of UAVs sit the historian and human machine interface (HMI) devices which may be replicated to provide UACPS control functions over long distance; the literature also refers to this segment as a ground control station (GCS). UACPSs have been the victim of actual cyber physical attacks [117, 143, 144].

8.3 UACPS Intrusion Detection Design

8.3.1 Behavior Rules

Our IDS design for the reference UACPS model relies on the use of lightweight specification-based behavior rules for each sensor or actuator component embedded in a UAV. They are oriented toward detecting an inside attacker attached to a specific physical component, provide a continuous output between 0 and 1 and allow a monitor device to perform intrusion detection on a neighboring trustee through simple monitoring. Table 8.1 lists the UACPS behavior rules for the UAVs. This table specifies the trustee and monitor devices for applying our IDS technique.

8.3.2 Transforming Rules to State Machines

The following procedure transforms a behavior specification into a state machine: First, we identify the “attack state” as a result of a behavior rule being violated. Then we transform this attack state into a conjunctive normal form predicate and identify the involved state components in the underlying state machine. Next we combine the attack states into a Boolean expression in disjunctive normal form. Then we transform the union of all predicate variables into the state components of a state machine and establish their corresponding ranges. Finally, we manage the number of states by state collapsing and identifying combinations of values that are not legitimate. Below we exemplify how a state machine is derived from the behavior specification in terms of behavior rules for the reference UACPS model.
Table 8.1: UAV Behavior Rules

<table>
<thead>
<tr>
<th>Description</th>
<th>Trustee</th>
<th>Monitor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Safe weapons if outside target area</td>
<td>UAV</td>
<td>UAV or HMI</td>
</tr>
<tr>
<td>Use minimum thrust if loitering</td>
<td>UAV</td>
<td>UAV or HMI</td>
</tr>
<tr>
<td>Turn off countermeasures if no threat</td>
<td>UAV</td>
<td>UAV or HMI</td>
</tr>
<tr>
<td>Stow landing gear if outside air base</td>
<td>UAV</td>
<td>UAV or HMI</td>
</tr>
<tr>
<td>Do not send to non-whitelisted</td>
<td>UAV</td>
<td>UAV or HMI</td>
</tr>
<tr>
<td>destinations</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Produce accurate data</td>
<td>UAV</td>
<td>UAV or HMI</td>
</tr>
<tr>
<td>Provide true recommendations</td>
<td>UAV</td>
<td>UAV or HMI</td>
</tr>
</tbody>
</table>

8.3.2.1 Identify Attack States

Attacks performed by a compromised sensor (actuator) embedded in a UAV will drive the UAV into certain attack states identifiable through analyzing the specification-based behavior rules. There are seven attack states as a result of violating the seven behavior rules for a UAV listed in Table 8.1.

The first UAV attack state is that a UAV readies its weapon when outside its target area (not within its air base, ingress corridor or egress corridor). This state catches attackers that intend to direct a UAV’s weapon against a friendly resource; these attackers attack to the UAV weapon module. The second UAV attack state is that a loitering UAV uses more than the minimum thrust required to maintain altitude. This state catches attackers that intend to decrease a UAV’s endurance by wasting its energy; these attackers attack to the UAV thrust module. The third UAV attack state is that a UAV uses countermeasures without identifying a threat. This state catches attackers that intend to increase the vulnerability of a UAV; these attackers attach to the UAV countermeasures module. A fourth UAV attack state is that a UAV deploys landing gear when outside its air base. This state catches attackers that intend to capture the UAV; these attackers attach to the UAV landing gear module. One way an attacker could pursue this goal is to launch a shellcode attack that diverts the UAV to an area they control. A fifth UAV attack state is that a node sends bytes to unauthorized parties. This state catches attackers that intend to exfiltrate mission data; these attackers attach to the database. A sixth UAV attack state is that a trustee’s embedded sensor reading differs from the monitor’s embedded sensor reading. The monitor is in the neighborhood of the trustee, measuring the same physical phenomenon. A seventh UAV attack state is that a monitor provides bad-mouthing attacks, i.e., providing bad recommendations regarding a well-behaving trustee node, or good-mouthing attacks, i.e., providing good recommendations regarding a misbehaving trustee node. This is detected by comparing recommendations
Table 8.2: UAV Attack States in Conjunctive Normal Form

| (Weapons = READY) ∧ (Location ≠ TARGET) |
| (Thrust > T) ∧ (Status = LOITER) |
| (Countermeasures = ACTIVE) ∧ (Threat = FALSE) |
| (Gear = DEPLOYED) ∧ (Location ≠ AIRBASE) |
| Destination ≠ WHITELISTED |
| | (|Trustee Data - Monitor Data| > δ) |
| | Trustee Audit ≠ Monitor Audit |

provided by multiple monitor nodes and detecting discrepancies.

8.3.2.2 Express Attack States in Conjunctive Normal Form

Table 8.2 lists the UAV attack states in conjunctive normal form.

8.3.2.3 Consolidate Predicates in Disjunctive Normal Form

\[( (\text{Weapons} = \text{READY}) \land (\text{Location} \neq \text{TARGET})) \lor ( (\text{Thrust} > T) \land (\text{Status} = \text{LOITER})) \lor ( (\text{Countermeasures} = \text{ACTIVE}) \land (\text{Threat} = \text{FALSE})) \lor ( (\text{Gear} = \text{DEPLOYED}) \land (\text{Location} \neq \text{AIRBASE})) \lor (\text{Destination} \neq \text{WHITELISTED}) \lor (|\text{Trustee Data} - \text{Monitor Data}| > \delta) \lor (\text{Trustee Audit} \neq \text{Monitor Audit})\]

8.3.2.4 Identify State Components and Component Ranges

We limit the range of the altitude parameter to the lowest point on earth and maximum altitude of the MQ-9.

We quantize continuous components at integer scale in permissible ranges. For example, altitude is in the range of \([-423, 15000 \text{ m}]\) and bank is in the range of \([-180, 180^\circ]\). Table 8.3 shows a complete list of the permissible ranges of UACPS state components. The resulting HMI and UAV automatons have \(181 \times 361 \times 15424 \times 361 \times 361 \times 2 \times 2 \times 2 \times 2 \times 101 \times 201 \times 201 \times 2 \times 2 \times 2 \times 2 = 9.955 \times 10^{27}\) states. Both of these automata are too large; we deal with this state explosion in the next step.
Table 8.3: UACPS State Components

<table>
<thead>
<tr>
<th>Name</th>
<th>Control or Reading</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Latitude</td>
<td>Reading</td>
<td>$[-90, 90^\circ]$</td>
</tr>
<tr>
<td>Longitude</td>
<td>Reading</td>
<td>$[-180, 180^\circ]$</td>
</tr>
<tr>
<td>Altitude</td>
<td>Reading</td>
<td>$[-423, 15000 \text{ m}]$</td>
</tr>
<tr>
<td>Bank</td>
<td>Reading</td>
<td>$[-180, 180^\circ]$</td>
</tr>
<tr>
<td>Pitch</td>
<td>Reading</td>
<td>$[-180, 180^\circ]$</td>
</tr>
<tr>
<td>Yaw</td>
<td>Reading</td>
<td>$[-180, 180^\circ]$</td>
</tr>
<tr>
<td>Threat</td>
<td>Reading</td>
<td>{false, true}</td>
</tr>
<tr>
<td>Stall</td>
<td>Reading</td>
<td>{false, true}</td>
</tr>
<tr>
<td>Data</td>
<td>Reading</td>
<td>{match, mismatch}</td>
</tr>
<tr>
<td>Audit</td>
<td>Reading</td>
<td>{match, mismatch}</td>
</tr>
<tr>
<td>Thrust</td>
<td>Control</td>
<td>[0, 100%]</td>
</tr>
<tr>
<td>Aileron</td>
<td>Control</td>
<td>[100% left, 100% right]</td>
</tr>
<tr>
<td>Elevator</td>
<td>Control</td>
<td>[100% down, 100% up]</td>
</tr>
<tr>
<td>Rudder</td>
<td>Control</td>
<td>[100% left, 100% right]</td>
</tr>
<tr>
<td>Gear</td>
<td>Control</td>
<td>{deployed, stowed}</td>
</tr>
<tr>
<td>Countermeasure</td>
<td>Control</td>
<td>{active, inactive}</td>
</tr>
<tr>
<td>Weapon</td>
<td>Control</td>
<td>{safe, ready}</td>
</tr>
<tr>
<td>Destination</td>
<td>Control</td>
<td>{whitelisted, unauthorized}</td>
</tr>
</tbody>
</table>
8.3.2.5 Manage State Space

Reducing the size of the state machine is an important subproblem for this study. Rather than approaching state machine reduction in an hoc fashion, we applied this broad strategy:

1. identify maximum acceptable state machine size
2. while the state machine is too large
   (a) choose the state component with the largest domain
   (b) compress the domain of this state component

This strategy hinges on our observation that the distribution of the domain size for state components is not uniform. Specifically, the wide domain for relatively few components dictates the size of the resulting state machine. We considered the following options for the compression technique: A linear encoding is intuitive, but resolution may be limited due to non uniform distribution of values. A logarithmic encoding will solve this problem for datasets clustered near the low end of the domain. An exponential encoding will solve this problem for datasets clustered near the high end of the domain. An adaptive encoding increases resolution in dense areas of the domain and reduces resolution in sparse areas of the domain. To manage the number of states, we reduce the size of the state machine by abbreviating the values for and consolidating some components. Our rules only consider three values for position (air base, corridor or target), so we consolidate latitude, longitude and altitude into a single position component and restrict this component to three values. Our rules only consider four values for flight status (takeoff, travel, loiter or land), so we consolidate bank, pitch, yaw, stall, aileron, elevator and rudder into a single status component and restrict this component to four values. Our rules only consider three values for thrust (sub-stall, minimal or super-minimal) so we restrict this component to three values. This treatment yields a modest state machine with \(3 \times 4 \times 2 \times 2 \times 2 \times 3 \times 2 \times 2 \times 2 \times 2 = 4608\) states, out of which 165 are identified as safe states and 4443 are unsafe states.

8.3.2.6 Behavior Rule State Machines

For the UAV state machine, there are 165 safe states and 4443 unsafe states. First, we label these states as states \(1, 2, \ldots, n = 4608\). Next, we decide \(p_{ij}\), the probability that state \(i\) goes to state \(j\), for each \((i,j)\) pair in the state machine to reflect a good (or bad) UAV’s behavior.

For a compromised UAV, \(p_{ij}\) depends on its attacker type: A reckless attacker will not go from an unsafe state to a safe state because it continuously attacks. So \(p_{ij} = 0\) if \(i\) is a bad state and \(j\) is a good state. However, the monitoring process is imperfect with error probability \(p_{err}\), so the monitor node may not observe exactly the same state transitions
performed by the reckless attacker. As a result, aggregate $p_{ij} = p_{err}$ instead of $p_{ij} = 0$ when $i$ is a bad state and $j$ is a good state.

For a random attacker with attack probability $p_a$, $p_{ij}$ values sum to $p_a$ for a given $i$ for all bad states $j$ and $p_{ij}$ values sum to $1 - p_a$ for a given $i$ for all good states $j$ because it will stop attacking with probability $1 - p_a$. With imperfect monitoring, a monitor node sees: $p_{ij}$ values sum to $p_a \times (1 - p_{err}) + (1 - p_a) \times p_{err}$ for a given $i$ for all bad states $j$ and $p_{ij}$ values sum to $(1 - p_a) \times (1 - p_{err}) + p_a \times p_{err}$ for a given $i$ for all good states $j$.

In practice, during the testing phase one will seed an attacker and assign a monitor node to observe the states this attacker enters to assign individual $p_{ij}$ values. For the special case in which every bad state among all is entered with equal probability and every good state among all is also entered with equal probability, a compromised UAV with random attack probability $p_a$ will be observed as having $p_{ij}$ as $((1 - p_a) \times (1 - p_{err}) + p_a \times p_{err})/165$ when $j$ is one of the 165 good states, and as $(p_a \times (1 - p_{err}) + (1 - p_a) \times p_{err})/4443$ when $j$ is one of the 4443 bad states. Figure 8.2 illustrates the behavior rule state machine for a compromised UAV. Crossed dotted slashes over a state indicate an unsafe state.

Here we note that the random attacker behavior-rule state machine derived above covers all three attacker models: reckless for which $p_a = 1$, random for which $p_a < 1$ and opportunistic for which $p_a$ is related to $p_{err}$ by Equation 3.5.

For a good UAV, it should stay in safe states 100% of the time; however, occasionally it may be misidentified by the monitor node as staying in an unsafe state due to ambient noise, temporary system faults and wireless communication faults with error probability $p_{err}$. For the special case in which every bad state among all is entered with equal probability and every good state among all is also entered with equal probability, a good UAV node will be observed as having $p_{ij}$ as $(1 - p_{err})/165$ when $j$ is one of the 165 good states, and as $p_{err}/4443$ when $j$ is one of the 4443 bad states. Figure 8.3 illustrates the behavior rule state machine for a good UAV. Again, crossed dotted slashes over a state indicate an unsafe state.
8.3.3 Collect Compliance Degree Data

Our UIDS relies on the use of monitor nodes, e.g., a UAV or HMI is a monitor node of another UAV. The monitor node knows the state machine of the trustee node assigned to it. The monitor node periodically measures the amount of time the trustee node stays in safe and unsafe states as the trustee node migrates from one state to another triggered by events causing state transitions. We consider a binary grading policy, i.e., assigning a compliance degree of 1 to a safe state and 0 to an unsafe state.

Let $c$ be the compliance degree of a device. The compliance degree $c$ of a device essentially is equal to the proportion of the time the device is in safe states. Let $S$ be the set of safe states the trustee node traverses over a period of time $T$. Let $t_i$ be the sojourn time that the trustee node stays in a safe state $i$, as measured by the monitor node. Then the monitor node collects an instance of $c$ by:

$$c = \frac{\sum_{i \in S} t_i}{T}$$  \hspace{1cm} (8.1)

If a node stays only in safe states during $T$, then by Equation 8.1, its compliance degree $c$ is one. On the other hand, if a node stays only in unsafe states only during $T$, then its compliance degree $c$ is zero. The monitor node monitors and collects the trustee node’s compliance degree history $c_1, c_2, \ldots, c_n$ for $n$ monitoring periods, where $n$ is sufficiently large, based on which it concludes whether or not the trustee node is compromised.

We leverage the state machines generated to collect compliance degree data of a good and a bad UAV during the testing phase. With Equation 8.1, the compliance degree $c$ is essentially equal to the sum of the probabilities of safe states i.e., $c = \sum_{j \in S} \pi_j$, where $\pi_j$ is the limiting probability that the node is in state $j$ of the state machine and $S$ is the set of safe states in the state machine. We then collect compliance degree history $c_1, c_2, \ldots, c_n$ of a UAV by means of Monte Carlo simulation. That is, given a good (or a bad) UAV’s state machine we
start from state 0 and then follow the stochastic process of this node as it goes from one state to another. We continue doing this until at least one state is reentered sufficiently often (say 100 times). Then we calculate $\pi_j$ using the ratio of the number of transitions leading to state $j$ to the total number of state transitions. Then we collect one instance of compliance degree. We repeat a sufficiently large $n$ test runs to collect $c_1, c_2, \ldots, c_n$ needed for computing the distribution of the compliance degree of a normal UAV or a UAV controlled by a reckless or random attacker.

### 8.4 Simulation

We collect compliance degree history $c_1, c_2, \ldots, c_n$ of a device by means of Monte Carlo simulation. We use the UAV in the reference UACPS defined in Section 8.2 to exemplify the utility of our IDS technique for secure UACPS applications.

Specifically, we simulate the procedure described in Section 8.3.2.6 to construct the state machines of a good UAV and a bad UAV. For a good UAV, we simulate $p_{ij}$ (see 8.3.2.6 for its definition) as $(1 - p_{err})/165$ when $j$ is one of the 165 good states, and as $p_{err}/4443$ when $j$ is one of the 4443 bad states. For a bad UAV with random attack probability $p_a$, we simulate $p_{ij}$ as $((1 - p_a) \times (1 - p_{err}) + p_a \times p_{err})/165$ when $j$ is one of the 165 good states, and as $(p_a \times (1 - p_{err}) + (1 - p_a) \times p_{err})/4443$ when $j$ is one of the 4443 bad states.

Given the state machine of a UAV generated above, we collect a sequence of compliance degree values $(c_1, c_2, \ldots, c_n)$ with $n = 1000$ Monte Carlo simulation test runs. In each simulation test run, we start from state 0 and then follow the stochastic process of this device as it goes from one state to another. We continue doing this until at least one state is traversed sufficiently (say 100 times). Then we calculate the limiting probability that the device is in state $j$, $\pi_j$, using the ratio of the number of transitions leading to state $j$ to the total number of state transitions. Then we collect one instance of $c$ using Equation 8.1. We repeat a sufficiently large $n = 1000$ test runs to collect $c_1, c_2, \ldots, c_n$ needed for computing the distribution of the compliance degree of a good or a bad UAV performing reckless or random attacks.

Figure 8.4 shows compliance degree raw data with $X = 1, 2, \ldots, n$ and $Y = c_1, c_2, \ldots, c_n$, for $n = 1000$ points, for a good UAV with several $p_{err}$ values. There are five clouds of compliance degree data, one corresponding with each $p_{err}$ setting. We see that as $p_{err}$ (representing ambient noise) increases, the cloud of compliance degree data moves down, i.e., the compliance degree of the good node decreases. This is because as the noise increases, there is a higher probability of the monitoring node misidentifying the good state status of the good UAV.

Figure 8.5 shows the sensitivity of $c_1, c_2, \ldots, c_n$ to $p_{err}$ for a bad UAV. Like Figure 8.4, there are five clouds of compliance degree data, one corresponding with each $p_{err}$ setting. However, in this case as $p_{err}$ increases, the cloud of compliance degree data moves up, i.e., the compliance degree of the bad UAV increases. This is because as the noise increases,
Figure 8.4: Sensitivity of Good Node Compliance Degree to $p_{err}$.

Figure 8.5: Sensitivity of Bad Node Compliance Degree to $p_{err}$.

Figure 8.6: Sensitivity of Bad Node Compliance Degree to $p_a$. 
Table 8.4: $\beta$ in Beta(1,$\beta$) and Resulting $p_{fn}$ and $p_{fp}$ Values under Various Random Attack Models for UAV ($C_T = 0.90$, $p_{err} = 0.01$).

<table>
<thead>
<tr>
<th>Model</th>
<th>$\beta$</th>
<th>$p_{fn}$</th>
<th>$p_{fp}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reckless ($p_a = 1.00$)</td>
<td>99.3</td>
<td>$&lt; 0.001%$</td>
<td>2.30%</td>
</tr>
<tr>
<td>Random ($p_a = 0.80$)</td>
<td>4.33</td>
<td>0.005%</td>
<td>2.30%</td>
</tr>
<tr>
<td>Random ($p_a = 0.40$)</td>
<td>1.10</td>
<td>8.02%</td>
<td>2.30%</td>
</tr>
<tr>
<td>Random ($p_a = 0.20$)</td>
<td>0.632</td>
<td>23.3%</td>
<td>2.30%</td>
</tr>
<tr>
<td>Random ($p_a = 0.10$)</td>
<td>0.449</td>
<td>35.5%</td>
<td>2.30%</td>
</tr>
</tbody>
</table>

Table 8.5: $\beta$ in Beta(1,$\beta$) and Resulting $p_{fn}$ and $p_{fp}$ Values under Various Opportunistic Attack Models for UAV ($C_T = 0.90$, $p_{err} = 0.01$, $C = 10$).

<table>
<thead>
<tr>
<th>Model</th>
<th>$\beta$</th>
<th>$p_{fn}$</th>
<th>$p_{fp}$</th>
<th>$\varepsilon$</th>
<th>$p_a$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aggressive</td>
<td>0.734</td>
<td>18.5%</td>
<td>2.30%</td>
<td>0.8</td>
<td>0.251</td>
</tr>
<tr>
<td>Aggressive</td>
<td>0.555</td>
<td>27.9%</td>
<td>2.30%</td>
<td>0.9</td>
<td>0.158</td>
</tr>
<tr>
<td>Conservative</td>
<td>0.449</td>
<td>35.5%</td>
<td>2.30%</td>
<td>1.0</td>
<td>0.1</td>
</tr>
</tbody>
</table>

there is a higher probability of the monitoring node misidentifying the bad state status of the bad UAV.

Figure 8.6 shows the sensitivity of $c_1, c_2, \ldots, c_n$ to $p_a$, the random attack probability by a bad node. There are six clouds of compliance degree data, one corresponding with each $p_a$ setting. As $p_a$ increases, the cloud of compliance degree data moves down, i.e., the bad node’s compliance degree decreases. This is because as the bad UAV performs more frequent attacks, it is more easily detected, so its measured compliance degree decreases.

With $c_1, c_2, \ldots, c_n$ of a good or bad UAV in hand, we apply Equation 4.6 to compute the $\beta$ parameter value of $G(\cdot) = Beta(\alpha, \beta)$ for the probability distribution of the compliance degree for a good or a bad UAV. We then calculate $p_{fn}$ and $p_{fp}$ by Equations 4.7 and 4.8, respectively, given the minimum compliance degree $C_T$ as input reflecting the consequence of false negatives over false positives for the UAV. For a UACPS we prioritize achieving a low false negative probability because the key motivation is safety.

Tables 8.4 and 8.5 exemplify the $\beta$ values and the resulting $p_{fn}$ and $p_{fp}$ values obtained when $C_T$ is 0.9 ($C_T$ is a design parameter to be fine-tuned to trade high false positives for low false negatives), $p_{err} = 0.01$ and the binary grading strategy is being used to assign $c^j$ to state $j$ for random and opportunistic attackers, respectively.

From Table 8.4, we observe that when the random attack probability $p_a$ is high, the attacker
can be easily detected as evidenced by a low false negative probability. Especially when \( p_a = 1 \), a reckless attacker can hardly be missed. On the other hand, as \( p_a \) decreases, a random attacker becomes more hidden and insidious, and the false negative probability increases.

From Table 8.5 we observe that the \( \beta \) values and the resulting \( p_{fn} \) values obtained depend on the aggressiveness of opportunistic attackers. More aggressive opportunistic attackers (those with a smaller \( \varepsilon \)) will have a better detection rate \((1 - p_{fn})\) because of a higher attacker probability \( p_a \) being used.

Here we note that in both Tables 8.4 and 8.5, the false positive probability \( p_{fp} \) remains the same regardless of the random attack probability, because \( p_{fp} \) measures the probability of misidentifying a good node only.

Figure 8.7: Probability of False Negative Versus Compliance Threshold for a Reckless Attacker under Varying \( p_{err} \).

Figure 8.8: Probability of False Negative Versus Compliance Threshold for a Random Attacker under Varying \( p_{err} \).
Figure 8.9: Probability of False Negative Versus Compliance Threshold for an Opportunistic Attacker under Varying $p_{err}$.

Figures 8.7, 8.8 and 8.9 illustrate the effect of $p_{err}$ on $p_{fn}$ of different types of attackers. They show the probability of false negative decreases as $C_T$ increases; this is because as the compliance threshold increases, a bad node is less likely to evade detection. While Figure 8.7 shows that a larger $p_{err}$ will benefit a reckless attacker, a smaller $p_{err}$ will benefit random and opportunistic attackers with a small attack probability $p_a$. This is because $p_{err}$ will obscure some of uniformly hostile behavior of the reckless attacker but will undermine the deceptively compliant behavior of the random and opportunistic attackers with a small $p_a$. Figure 8.8 shows that while a smaller $p_{err}$ will benefit random attackers when $p_a$ is small (0.2), a larger $p_{err}$ will benefit random attackers when $p_a$ is large (0.8).

Figure 8.10: Probability of False Positive Versus Compliance Threshold for a Good Node under Varying $p_{err}$.

Figure 8.10 illustrates the effect of $p_{err}$ on $p_{fp}$ for a good node. It shows the probability of false positive increases as $C_T$ increases; this is because as the compliance threshold increases, a good node is more likely to be incorrectly misidentified as a bad node. Also, it shows a
smaller $p_{\text{err}}$ will benefit a good node; this is because $p_{\text{err}}$ distorts the uniformly compliant behavior of a good node.

By adjusting $C_T$, our specification-based IDS technique can effectively trade higher false positives for lower false negatives to cope with more sophisticated and hidden random attackers. This is especially desirable for ultra safe and secure UACPS applications for which a false negative may have a dire consequence. Figure 8.11 shows a ROC graph of intrusion detection rate $(1 - p_{\text{fn}})$ versus false positive probability ($p_{\text{fp}}$) obtained as a result of adjusting $C_T$ for reckless, random and opportunistic attackers given different $p_{\text{err}}$. As we increase $C_T$, the detection rate increases while the false probability increases. We see that with our specification-based IDS technique, the detection rate of the UACPS node can approach 100% for detecting attackers when using a sufficiently high $C_T$, i.e., an attacker is always detected with probability 1 without false negatives, while bounding the false positive probability to below 0.05% for reckless attackers, below 7% for random attackers and below 7% for opportunistic attackers. Note the ROC surfaces for random and opportunistic attackers cross over roughly along a curve at $p_{\text{err}} = 0.015$, indicating that when $p_{\text{err}} < 0.015$ an opportunistic attacker with $\varepsilon = 0.8$ is more difficult to be detected than a random attacker with $p_a = 0.2$, and vice versa after $p_{\text{err}} > 0.015$. The highlighted dots show the points on the corresponding surface that meet the maximum $p_{\text{fn}}$ requirement (0.01) while minimizing $p_{\text{fp}}$.

The results obtained above can be used by the system to adaptively select the minimum $C_T$ value dynamically to satisfy the imposed $p_{\text{fn}}$ requirement while minimizing $p_{\text{fp}}$ as much as possible in response to the environment condition (e.g., ambient noise) and the suspected attacker type detected at runtime (e.g., a random attacker). Table 8.6 illustrates a scenario in which the maximum $p_{\text{fn}}$ allowable is 1%, which must be satisfied. Given a $p_{\text{err}}$ value and the attacker type as input, there is a $C_T$ value at which $p_{\text{fn}} = 1\%$ as decided from Figures 8.7, 8.8 and 8.9, and $p_{\text{fp}}$ is the resulting $p_{\text{fp}}$ based on the $C_T$ value selected.

Table 8.6 illustrates adaptive IDS design: the system selects the minimum $C_T$ value that
Table 8.6: $C_T$ to Satisfy $p_{fn}$ (1%) while Maximizing $p_{fp}$ Given $p_{err}$ and Attacker Type as Input ($p_a = 0.2$, $\varepsilon = 0.8$)

<table>
<thead>
<tr>
<th>$p_{err}$</th>
<th>Reckless $C_T$</th>
<th>$p_{fn}$</th>
<th>$p_{fp}$</th>
<th>Random $C_T$</th>
<th>$p_{fn}$</th>
<th>$p_{fp}$</th>
<th>Opportunistic $C_T$</th>
<th>$p_{fn}$</th>
<th>$p_{fp}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.01</td>
<td>0.0453</td>
<td>0.01</td>
<td>0.0005</td>
<td>0.9993</td>
<td>0.01</td>
<td>0.0710</td>
<td>0.9981</td>
<td>0.01</td>
<td>0.0614</td>
</tr>
<tr>
<td>0.02</td>
<td>0.0889</td>
<td>0.01</td>
<td>0.0019</td>
<td>0.9992</td>
<td>0.01</td>
<td>0.1359</td>
<td>0.9773</td>
<td>0.01</td>
<td>0.0743</td>
</tr>
<tr>
<td>0.03</td>
<td>0.1311</td>
<td>0.01</td>
<td>0.0043</td>
<td>0.9991</td>
<td>0.01</td>
<td>0.1952</td>
<td>0.9058</td>
<td>0.01</td>
<td>0.0704</td>
</tr>
<tr>
<td>0.04</td>
<td>0.1716</td>
<td>0.01</td>
<td>0.0078</td>
<td>0.9990</td>
<td>0.01</td>
<td>0.2498</td>
<td>0.7493</td>
<td>0.01</td>
<td>0.0560</td>
</tr>
<tr>
<td>0.05</td>
<td>0.2106</td>
<td>0.01</td>
<td>0.0124</td>
<td>0.9989</td>
<td>0.01</td>
<td>0.3001</td>
<td>0.4756</td>
<td>0.01</td>
<td>0.0334</td>
</tr>
</tbody>
</table>

would satisfy the maximum $p_{fn}$ requirement while providing a $p_{fp}$ as small as possible, given knowledge of the attacker type and $p_{err}$. An example, suppose a suspected attacker is of opportunistic type whose attacking behavior is characterized by $\varepsilon = 0.8$ as described in Table 8.6. After determining $p_{err} = 0.01$ (which is detectable), the system would select a high $C_T$ value (i.e., $C_T = 0.9981$) to yield $p_{fn} = 1\%$ while minimizing $p_{fp}$ to 6.146%. If the attacker type is reckless, on the other hand, then the system would select a low $C_T$ value (i.e., $C_T = 0.0453$) to yield $p_{fn} = 1\%$ while minimizing $p_{fp}$ to 0.0468%.

### 8.5 Comparative Analysis

![Figure 8.12: ACCM Receiver Operating Characteristic Graph.](image)

In this section, we compare our IDS design with a multitrust anomaly-based IDS called Multiagent System (MAS) developed by Tsang and Kwong [140] intended for industrial CPSs. MAS includes an analysis function called Ant Colony Clustering Model (ACCM) to reduce the characteristically high false positive rate associated with anomaly-based approaches while minimizing the training period by using an unsupervised approach to machine learning. MAS
uses a standard data set, KDD Cup 1999, for testing. We use it as a benchmark against which our IDS is compared for three reasons: First, there is no existing UACPS IDS available for performance comparison; industrial process control environments of MAS/ACCM are close to UACPS environments with similar safety requirements. Second, MAS/ACCM reported false positive and false negative probability data for ease of comparison. Third, unlike anomaly-based IDS approaches, ACCM generated good false positive rates (reported 1 to 6%). We visualize Tsang and Kwong’s results in Figure 8.12.

Figure 8.13 visually compares the ROC graphs (detection rate or $1 - p_{fn}$ versus $p_{fp}$) for UIDS and ACCM. We set $p_{err} = 0.010$ for UAVs. This is because 1% of mis-monitoring due to ambient noise, temporary system faults and wireless communication faults in UACPS environments is reasonable. This is based on Ho and Shimamoto reporting a $0.2 - 7.5\%$ UACPS packet error rate (depending on altitude, network size and channel access technique) [69] and Palazzi et al. experimenting with UACPS packet error rates between 0.1 and 1.0% [111]. Figure 8.13 shows for reckless and highly aggressive random attackers, UIDS outperforms ACCM across the domain of $p_{fp}$. Also, it shows that UIDS outperforms ACCM for $p_{fp} > 0.02$ for cautious random attackers.

We first note that the coverage area (out of a one by one area) below the ROC curve, referred to as AUC, measures the IDS accuracy. An area of 1 represents a perfect test; an area of 0.5 represents a worthless test.

We clearly see that for UIDS with $p_a = 1$ or 0.8, the detection rate is always higher than ACCM, given the same false positive probability. Therefore the AUC of UIDS is clearly greater than that of ACCM for these configurations. For UIDS with $p_a = 0.4$ (a more cunning attacker), the detection rate is not always higher than ACCM; however, a closer look through Figure 8.14 reveals that the AUC of UIDS is still greater. Moreover, in the experiment conducted by Tsang and Kwong [140], presumably only reckless attackers were considered in which case UIDS with a AUC nearly equal to 1 clearly outperforms ACCM.

![Figure 8.13: UIDS and ACCM ROC Graph Comparison.](image)
8.6 Summary

For UACPSs, being able to detect attackers while limiting the false alarm probability is of the utmost importance to protect the continuity of operation. In this chapter, we proposed an adaptive behavior-rule specification-based IDS technique for intrusion detection of compromised UAVs in a UACPS. We demonstrated that the detection probability approaches one (that is, we can always catch the attacker without false negatives) while bounding the false alarm probability to below 0.05% for reckless attackers, below 7% for random attackers and below 6% for opportunistic attackers. Through a comparative analysis, we demonstrated our behavior-rule specification-based IDS technique outperforms an existing multitrust anomaly-based IDS approach in detection accuracy.
Chapter 9

Conclusions and Future Research

The goal of this dissertation research is to design and validate a resilient, energy-aware and adaptive IDS that can maximize the lifetime of CPSs in the presence of malicious attacks, as well as malicious, erroneous, partly trusted, uncertain and incomplete audit information. We investigated host IDS, system-level IDS and intrusion response designs that help move toward this goal. This chapter summarizes research accomplishments and outlines future research areas.

9.1 Publications Summary

This dissertation work has resulted in two journal papers, five conference papers and five journal submissions.

9.1.1 Journal Publications


Chapter 9. Conclusions and Future Research

9.1.2 Conference Publications


9.1.3 Papers Submitted


In [105], we surveyed the state of the art in wireless network intrusion detection systems. Later in [100], we narrowed our scope, refined our taxonomy and considered attack model to perform a deeper survey of IDSs for CPSs. This benefited our research goal by providing a foundation for the literature search in this dissertation. Mitchell et al. [106] extended the
Signalprints technique to expand the degree of multitrust. In particular, this paper studied the safety of using data from less trustworthy sources. This technique is traffic-based; specifically, it exploits physical layer data. This benefited our research goal by comparing unitrust and multitrust intrusion detection techniques. In [95], we investigated two analysis techniques: enviroconsistency and positional discontinuity. These techniques leveraged application specific sensing and navigation functions of a reference CPS. These techniques are traffic-based; specifically, they exploit application layer data. This benefited our research goal by developing two host intrusion detection techniques. In [96], we studied the optimal audit interval and intrusion detector configurations for a mobile CPS given varying levels of hostility. This benefited our research goal by establishing a basic survivability model for CPS based on Byzantine failure and energy exhaustion. [104] is an expanded version of [96]. In [103], we modeled three attacker behavior models and analyzed the effect of dynamic intrusion and response design on the system reliability. This benefited our research goal by developing the attack and dynamic \( C_T \) control models. In [99], we applied specification-based intrusion detection to the UAS domain. [101] is an expanded version of [99]. In [97], we applied behavior rule-based intrusion detection to the medical CPS domain. [98] is an expanded version of [97]. In [102], we applied behavior rule-based intrusion detection to the smart grid CPS domain.

We modeled mixed attacker scenarios comprising various ratios of reckless, random, insidious and opportunistic attacker behaviors. Chapter 5 contains the highlights of this work. We also modeled Stuxnet attackers. Two important Stuxnet-specific configuration choices are exponential capture strength and insidious attacker behavior. Chapter 5 contains the highlights of this work as well.

### 9.2 Applicability

In this section, we discuss issues and solutions related to the applicability and implementation of the CPS IDS protocols proposed in this dissertation research, taking scalability, energy consumption, operating costs and performance issues into consideration.

First, we can adjust the granularity of the state machine in response to attacker strength without compromising the detection accuracy (in terms of \( p_{fn} \) and \( p_{fp} \)) while reducing energy consumption, operating costs and improving performance. During hostile network conditions, the IDS can use a fine granularity state machine to minimize \( p_{fn} \) and \( p_{fp} \). During placid network conditions, the IDS can use a coarse granularity state machine to minimize energy consumption and operating cost and maximize performance. A coarse granularity state machine can improve performance because more processor and memory are available for business functions instead of being consumed by security overhead.

Also, we can address the scalability of the target system in a number of ways. First, we can use a coarse granularity state machine. Also, we can detune IDS parameters such as sensing
interval ($T_{\text{SENSE}}$), intrusion detection interval ($T_{\text{IDS}}$) and the number of voters in system-level intrusion detection ($m$). Third, we can add a strategy to tune the monitoring degree of each intrusion detector. For example, every intrusion detector audits some percentage of all nodes instead of every node. An interesting problem is how to prioritize an intrusion detector’s nodes for audit. One choice is to choose them randomly. Another choice is to choose the nodes for which the monitor has the best audit data. A third choice is to globally optimize so all nodes are audited with the same frequency.

The techniques in this dissertation are universally applicable to CPS systems. We delivered four case studies instead of one because each of these reference CPSs has unique characteristics that must be modeled differently: For example, MGCPS nodes are limited in channel, energy, memory and processor, an MCPS is heterogeneous and human-integrated, the scale of an SGCP is exceptionally large and UAVs demonstrate high magnitude three dimensional mobility. Our techniques are particularly suited for CPSs because they are lightweight, detect unknown attacks, have a low false positive rate and take advantage of the focused use cases of a purpose-built system by transforming them into specifications. Currently, the techniques in this dissertation are limited to single enclave systems; relaxing this assumption is a future research direction.

9.3 Future Research Directions

There are several future research directions as extensions to this dissertation research:

**Data Mining:** Generally, there are three analysis techniques available: pattern matching, threshold monitoring and data mining [100]. Machine learning, which falls under the data mining research area, is a substantial topic of its own. We used threshold monitoring for analysis in order to avoid the large processing requirement of data mining and the large storage requirement of pattern matching. Processor-efficient data mining is a promising research area. Validating the capture and insider attack behavior classifier sketched in Chapter 3 is an open issue.

**Intrusion Response and Repair:** Researchers should investigate and analyze intrusion response and repair strategies. Possible intrusion responses include evicting individual compromised nodes, isolating compromised segments (microgrid or larger scope) and adjusting IDS parameters (e.g., $T_{\text{IDS}}$, $m$ and $C_T$) to increase detection strength. For example, the IDS can perform better if it retunes its parameters (such as compliance threshold, audit interval and state machine granularity) based on the type and strength of adversary it faces. Possible repair strategies are to identify compromised segments and for each one: stop operating, revert all nodes to certified software loads and configurations, rekey/reset passwords and progressively resume operation from the production side of the network towards the consumers.
Proactivity: Responsiveness is one way IDS research is evolving. The first generation of IDSs used a static or “set and forget” approach to parameterizing intrusion detection. The first evolutionary step forward was for IDSs to self-reconfigure in response to the adversary. Investigators should study proaction because a CPS that is completely dependent on reaction can fall victim to a DoS attack by an adversary that continually provokes a response which impacts the mission. Researchers should pursue countermeasures tailored to attacker behavior, so the next evolutionary step forward will be for IDSs to classify their adversary and reprovision themselves in anticipation of the adversary’s next move. An example of this concept of operation is an IDS that classifies its adversary as an advanced persistent threat intent on exfiltrating data from a certain location. The IDS could stock that location with nonsensitive products and devote more resources to establishing the provenance of the attack. More generally, the best intrusion response is situational: If the adversary is persistent, evicting them is the priority. If the adversary is transient, repairing the system is the priority. If the adversary is ineffective, establishing attribution for the attack is the priority. Analysis techniques that provide early warning of attacks are a potential research area. These techniques would serve as an enabler by providing a trigger for pre-detection responses. A possible research direction is to distinguish early warnings from detections by using the confidence level that accompanies existing analysis techniques.

Closed Loop Control: Closed loop controls are one of the features that distinguish CPSs from cyber systems. IDSs should translate their precise timing constraints into models for specification-based intrusion detection modules; their real-time requirements challenge IDSs to avoid disruption while they afford IDSs opportunities in the form of highly predictable behavior profiles. Also, CPS IDSs should take care not to impact these sensitive business functions by ringfencing their processor, memory and communications usage. IDSs will dynamically adjust their provisioning (for example, compliance threshold, intrusion detector quantity or audit interval) to remain within their resource envelope. There is no CPS IDS research in the literature that considers the closed loop control blocks of a CPS.

Federation: Only the smallest CPSs do not host multiple enclaves. Actors from different enclaves have different charters, privileges and patterns of behavior. Their boundaries may be political (citizens of A, citizens of B, etc.), organizational (for example, Defense Department or Justice Department) or functional (operator enclave, maintainer enclave, administrator enclave, etc.). Trust and reputation management products are good start points for these investigations. IDRS research should be extended to multi-enclave federated CPSs. Researchers should extend the analysis to hierarchically-structured intrusion detection and response system design for a large CPS consisting of multiple enclaves each comprising heterogeneous entities subject to different operational and environment conditions and attack threats. IDSs from different enclaves will struggle to establish trust so they can share audit data and effect trans-enclave sanctions. Multitrust can be a key design factor in building future federated CPS IDSs.
Chapter 9. Conclusions and Future Research

**Multitrust:** Multitrust is unexplored in CPS IDS research but deserves more attention because it increases the dataset available to an IDS. Multitrust is the concept of using reported information (data from witnesses or third parties). *Hearsay* or *gossip* may also be used to refer to reported data. The key problem is guaranteeing the larger data set yields a net gain in key metrics despite the presence of bad-mouthing and ballot stuffing attacks.

**Metrics:** Research is needed to define CPS IDS performance metrics. When numerical results are reported at all, only detection rate, false negative rate and false positive rate are given usually. However, detection latency is a critical metric that researchers rarely report on. A 100% detection rate is a great achievement, but if this IDS takes an hour to detect intruders, the adversary may still have enough time to damage the target system. We have not found detection latency being studied in the literature, but it is clearly a critical metric. Therefore, researchers should develop detection latency as a key IDS metric. Specifically, the SPN modeling technique can be extended to analyze the detection latency performance metric. Another new metric could be mitigation latency, which represents the delay between detection and attack repulsion.

**Application Layer Data Auditing:** Audits should focus on application layer data. The audit of lower layer data that is common to any application has been well-studied, so adversaries expect these defensive measures. A cunning adversary will craft his attack to appear normal in every way possible to avoid widely deployed IDSs. IDSs that audit application layer data focus on detecting the adversary where he must reveal himself to attack the system.

**Implementation Strategies:** CPS intrusion detection has high potential for technology transfer. It is worth considering transition activities like encoding the state machine, host IDS software and system IDS software in a high-level language, cross-compiling for the targets of interest, deployment and tuning the parameterization (e.g., $T_{IDS}$, $m$ and $C_T$) based on desired versus actual false negative and positive rates.

**Modeling and Analysis:** Model-based analysis techniques such as [31, 38, 95, 103, 104] need to be developed and validated to analyze performance of CPS IDS protocols and identify optimal CPS IDS protocol settings to maximize CPS IDS performance based on performance metrics defined. Configuration items (e.g., number of intrusion detectors, audit interval and detection threshold) impact the detection and false positive rates of the IDS and longevity of the CPS as a whole. Researchers should identify parameters that have a local maximum and parameters that are covariant. They should establish heuristics for finding the optimal value for the former set and equations that characterize the tradeoff for the latter set.

**Adversary Modeling and Counter Attacks:** Not all adversaries behave the same, so researchers should deepen the complexity of attacker models. The literature is thin on
Chapter 9. Conclusions and Future Research

adversary modeling [103]. This dissertation research performed a preliminary investigation of adversary modeling of CPSs, i.e., treating the attacker strength as characterized by the attacker behavior type (reckless, random, insidious or opportunistic) and/or the capture attack strength type (constant, logarithmic, linear or exponential). This is not a complete set. The identification of current attacker behavior and/or capture strength is still an unsolved problem and is itself challenging. For example, an oracle attacker could adjust the attacker strength depending on the detection strength to maximize security failure. For countering adversary behavior, more work is called for to apply control theory to design a general $C_T$ control function with several control parameters (such as $C_T^p$, $\delta C_T$, and $P$ in Equations 4.11-4.13) such that the system can adjust the values of these parameters in response to dynamically changing attacker strength detected at runtime.

Automating Specification-based Intrusion Detection: The critical challenge in specification-based designs is to transform a sophisticated system into a formal model; with their well defined actors and functions, CPSs provide an ideal starting point for serious investigation of specification-based designs. Two possible research directions are to create a tool or methodology to transform a system into a formal model and to establish a language or schema for expressing the formal model. Another research direction is to analyze the time complexity of the detection algorithm before and after reducing the size of the state machine, the relationship between time and space complexities and the impact of the state machine reduction on detection accuracy. Possible implementation strategies are to encode the state machine, host IDS software and system IDS software in a high-level language, cross-compile for the targets of interest, deploy and tune the parameterization (e.g., $T_{IDS}$, $m$ and $C_T$) based on desired versus actual false negative and positive rates.

Biology, Immunology and Self-awareness: A potential research area is to pursue IDS approaches inspired by biology, immunology and self-awareness like [18, 44, 115, 120, 121, 129, 140, 157, 158]. In particular, they aid detection of unknown attacks; finding zero-day attacks is a hard problem with great potential for technology transfer. Self-aware detection systems [18, 56] include their own state in deciding whether a suspect is normal or an intruder. One way they may distinguish themselves from classical approaches is by identifying situations where an attacker is provoking them into a DoS scenario with an otherwise benign circumstance. Also, self-aware detection systems may identify situations where the cost of the damage the adversary threatens is greater than stopping the hosted business functions and disconnecting the system.

Alternative Audit Data Approaches: Another future research direction is to investigate other intrusion detection criteria [12, 27, 28] based on accumulation of deviation from good states in addition to the current binary criterion used in this dissertation based on a minimum compliance threshold to further improve the detection rate without compromising the false positive probability.
Feedback Based Voting: A potential research area is to leverage feedback for more effective voting. Specifically, a system IDS can improve performance by assigning higher weights to more trustworthy voters dynamically based on feedback control, or by selecting intrusion detectors intelligently based on machine learning instead of doing so randomly.
Appendix A

Notation

\( \mathcal{C} \)  
Opportunistic attack coefficient

\( C_T \)  
System minimum compliance threshold

\( C_T^p \)  
Minimum threshold set by the system for the persistent attack case

\( C_T(t) \)  
\( C_T \) value set at time \( t \) as a response to \( N_{ab}^a(t) \)

\( c \)  
Compliance degree at some instant

\( c_d \)  
Compliance degree of a node over time for a state machine

\( c_a \)  
Compliance degree of a node over time over multiple state machines

\( c_{d,i} \)  
Compliance degree of state \( i \)

\( c_i \)  
\( i^{th} \) compliance degree output

MTTF  
Mean time to failure

\( m \)  
Number of intrusion detectors in the system

\( N \)  
Starting network size (i.e., the number of nodes)

\( N_b \)  
Number of bad nodes

\( N_{ab}^a(t) \)  
Attacker strength (number of active captured nodes) detected at time \( t \)

\( N_{i}^i \)  
Number of inactive captured nodes

\( N_T^b \)  
Insidious attack threshold

\( N_e \)  
Number of evicted nodes

\( N_g \)  
Number of good nodes

\( N_{IDS} \)  
Maximum IDS cycles before energy exhaustion

\( P_{fn} \)  
System IDS false negative probability

\( P_{fp} \)  
System IDS false positive probability

\( p \)  
Exponential capture strength parameter

\( p_a \)  
Attack probability by an insidious attacker

\( p_{err} \)  
Probability of mis-monitoring

\( p_{ln} \)  
Per-node host IDS false negative probability

\( p_{lp} \)  
Per-node host IDS false positive probability

\( p_{ln}^i \)  
Insidious attacker per-node host IDS false negative probability
### Appendix A. Notation

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_{ij}$</td>
<td>Probability a state machine transitions from $i$ to $j$</td>
</tr>
<tr>
<td>$p_{fn}^R$</td>
<td>Reckless attacker per-node host IDS false negative probability</td>
</tr>
<tr>
<td>$p_{fn}^R$</td>
<td>Random attacker per-node host IDS false negative probability</td>
</tr>
<tr>
<td>$p_{fn}^O$</td>
<td>Opportunistic attacker per-node host IDS false negative probability</td>
</tr>
<tr>
<td>$p_{\text{random}}$</td>
<td>Random attack probability by a random attacker</td>
</tr>
<tr>
<td>$\mathcal{S}$</td>
<td>Set of safe states</td>
</tr>
<tr>
<td>$T_{\text{IDS}}$</td>
<td>Intrusion detection interval</td>
</tr>
<tr>
<td>$X_b$</td>
<td>Compliance degree of a bad node</td>
</tr>
<tr>
<td>$X_g$</td>
<td>Compliance degree of a good node</td>
</tr>
<tr>
<td>$X_i$</td>
<td>Compliance degree of arbitrary node $i$</td>
</tr>
<tr>
<td>$\hat{x}$</td>
<td>Estimate of $x$</td>
</tr>
<tr>
<td>$\beta$</td>
<td>Compliance distribution parameter</td>
</tr>
<tr>
<td>$\delta_{C_T}$</td>
<td>Increment to $C_T$ per active bad node detected</td>
</tr>
<tr>
<td>$\varepsilon$</td>
<td>Opportunistic attack exponent</td>
</tr>
<tr>
<td>$\Lambda_c$</td>
<td>Aggregate capture strength</td>
</tr>
<tr>
<td>$\lambda_c$</td>
<td>Per-node capture strength</td>
</tr>
<tr>
<td>$\lambda_{\text{if}}$</td>
<td>Impairment rate for an attacker to cause severe functional impairment</td>
</tr>
<tr>
<td>$\pi_i$</td>
<td>Probability a stochastic process is in state $i$</td>
</tr>
</tbody>
</table>
Appendix B

Acronyms

APT  Advanced persistent threat  
AUC  Area under the curve  
CD  Cardiac device  
CPS  Cyber physical system  
DAP  Distribution access point/data aggregation point  
DAS  Data acquisition system  
DCS  Distributed control system  
DER  Distributed energy resource  
HE  Head-end  
HIDS  Host-based intrusion detection system  
HMI  Human machine interface  
ICT  Information and communications technology  
IDRS  Intrusion detection and response system  
IDS  Intrusion detection system  
IED  Intelligent electronic device  
LCS  Longest common subsequence  
LOS  Line of sight  
MANET  Mobile ad hoc network  
MCPS  Medical CPS  
MGCP  Mobile group CPS  
MIC  Message integrity code  
MITM  Man-in-the-middle attack  
MTTF  Mean time to failure  
MTU  Master terminal unit  
NAN  Neighborhood area network  
NIDS  Network-based intrusion detection system  
PCA  Patient controlled analgesia  
PLC  Programmable logic controller
### Appendix B. Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PMU</td>
<td>Phasor measurement unit</td>
</tr>
<tr>
<td>R2L</td>
<td>Remote to local attack</td>
</tr>
<tr>
<td>ROC</td>
<td>Receiver operating characteristic</td>
</tr>
<tr>
<td>RTU</td>
<td>Remote terminal unit</td>
</tr>
<tr>
<td>SAN</td>
<td>Sensor actuator network</td>
</tr>
<tr>
<td>SCADA</td>
<td>Supervisory control and data acquisition</td>
</tr>
<tr>
<td>SGCPS</td>
<td>Smart grid CPS</td>
</tr>
<tr>
<td>SM</td>
<td>Smart meter</td>
</tr>
<tr>
<td>SPN</td>
<td>Stochastic Petri net</td>
</tr>
<tr>
<td>U2R</td>
<td>User to root attack</td>
</tr>
<tr>
<td>UACPS</td>
<td>Unmanned aircraft CPS</td>
</tr>
<tr>
<td>UAS</td>
<td>Unmanned aircraft system</td>
</tr>
<tr>
<td>UAV</td>
<td>Unmanned air vehicle</td>
</tr>
<tr>
<td>VANET</td>
<td>Vehicular ad hoc network</td>
</tr>
<tr>
<td>VSM</td>
<td>Vital sign monitor</td>
</tr>
<tr>
<td>WISN</td>
<td>Wireless industrial sensor network</td>
</tr>
<tr>
<td>WMN</td>
<td>Wireless mesh network</td>
</tr>
<tr>
<td>WSAN</td>
<td>Wireless sensor and actor network</td>
</tr>
<tr>
<td>WSN</td>
<td>Wireless sensor network</td>
</tr>
</tbody>
</table>
Bibliography


[67] [http://en.wikipedia.org/wiki/Host-based_intrusion_detection_system](http://en.wikipedia.org/wiki/Host-based_intrusion_detection_system)


