INTRUSION DETECTION AND DETERRENCE FOR CRITICAL INFRASTRUCTURES

by

Muhammad Qasim Ali

A dissertation submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in
Computing and Information Systems

Charlotte

2016

Approved by:

Dr. Ehab Al-Shaer

Dr. Yongge Wang

Dr. Weichao Wang

Dr. Rakesh Bobba

Dr. Yu Wang

ABSTRACT

MUHAMMAD QASIM ALI. Intrusion detection and deterrence for critical infrastructures. (Under the direction of DR. EHAB AL-SHAER)

Critical infrastructures are the systems and networks that are so vital that their unavailability would have a major impact on national security, economy, safety, or any combination thereof [1]. Examples of critical infrastructure are power systems, financial services, emergency services, health care, defense sector, and others [1]. While these infrastructures are readily available, they are inherently vulnerable to attacks. New emerging threats have been highlighted in the recent literature with respect to critical infrastructures [2–5]. Therefore, ensuring both accurate detection and robust deterrence is highly important for protecting these infrastructures from devastating, sophisticated, and evasive cyber and cyber-physical attacks. Deterrence is the ability to make intrusions very unlikely or highly expensive. However, the intrusion detection and deterrence techniques for critical infrastructure face many challenges. First, intrusion detection techniques should be real-time, accurate, robust against stealthy attacks, and economically feasible. Second, intrusion deterrence techniques should be unpredictable, computationally inexpensive, and effective against persistent attackers.

In this thesis, we focus on intrusion detection and deterrence for energy delivery systems (EDS) of smart grids. This thesis has two key goals. The first goal is to develop real-time intrusion detection and robust deterrence techniques to protect EDS against stealthy attacks that can undermine the system's integrity. The

second goal is to identify the limitations of existing intrusion detection techniques that allow for evasive attacks and develop techniques to reduce evasion margin for attackers. We, particularly, investigate advanced metering infrastructure (AMI) and automatic generation control (AGC) in the supervisory control and data acquisition network (SCADA) of smart grids. We show, based on statistical analysis of AMI and AGC operational data, that both AMI and AGC exhibit a predictable behavior that can be exploited to develop accurate and robust intrusion detection and deterrence techniques.

First, we model AMI configuration specification using stochastic temporal properties that can be used to detect anomalous activities. As the AMI exhibits static behavior that can be exploited to launch mimicry and evasive attacks, we developed a new deterrence approach that randomizes the AMI configuration frequently to mislead attackers, without breaking the system operational integrity.

Second, we address the AGC attacks that might result from manipulating sensor measurements that can bypass bad data detection algorithms. We developed a data-driven multi-tier intrusion detection technique for a single and multiple AGC, which exploits the temporal dependence of the measurements to identify potential anomalous behavior at real-time, and then incorporate system-wide knowledge through an offline process to reduce false positives.

Last and third, we investigate the inherent limitations of existing intrusion detection systems against evasive attacks and developed a key-based deterrence approach to reduce the attack evasion margin by introducing a notion of randomized thresholds in intrusion detection systems.

ACKNOWLEDGMENTS

I would first like to give thanks to the Almighty God for all of His blessings that enabled me to complete this dissertation successfully.

Throughout the entire PhD program, my mother, Narjis Khatoon, has sacrificed the most. I cannot thank her enough for leaving her hometown, staying up for my late night meetings, taking care of me, and dealing with my hectic/unscheduled/frenetic life. It is indeed because of her support, help, and prayers that I am able to complete this dissertation.

I am thankful to Dr. Ehab Al-Shaer, Department of Software and Information Systems, University of North Carolina at Charlotte. The experience and keen interest of Dr. Al-Shaer in the field of network and information security encouraged and helped me to conduct this research. His criticism and supervision has definitely helped me in publishing the relevant research at conferences and in journals.

I would also like to express my gratitude to Drs. Weichao Wang, Yongge Wang, Yu Wang, and Rakesh Bobba for serving on my dissertation committee and helping me pursue the degree by offering their suggestions, feedback, and reviews. I must acknowledge the help and support I received from the faculty and staff of the Software and Information Systems Department, College of Computing and Informatics, and the graduate school. They truly supported me whenever I needed them.

Last but not the least, I am thankful to all my friends, both those at the university and others, who boosted my morale when I needed it most. I feel lucky to have such a great group of friends.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

In the recent past years, technology has witnessed an exponential growth. This advancement in technology has made users depend on technology infrastructures in their day-to-day lives. While some of these infrastructures are general purpose, some are critical in nature. Critical infrastructure is the driving force of the nation's economy, security, and safety. Our day-to-day lives are highly dependent on these infrastructures, e.g., the power we use in our homes, the transportation that moves us, and the communication systems we rely on to stay in touch with friends and family. All of these infrastructures have a high impact on the nation's economy, security, and safety. The Department of Homeland Security defines these infrastructures as the assets, systems, and networks, whether physical or virtual, so vital to the United States that their incapacitation or destruction would have a debilitating effect on national security, economy, or safety, or any combination thereof [1]. Therefore, any damage or unavailability of these infrastructures has a high impact on the nation. Examples of these infrastructures are emergency services, energy delivery system (EDS), financial services, nuclear sector, defense sector, etc. Since these infrastructures generally relate to the defense, economy, emergency services, and everyday life, availability and protection of these infrastructures is highly important. Evolving threats targeting these infrastructures have been discussed in the recent literature [2–5]. Therefore, protection of these infrastructures using accurate detection and robust deterrence is

critical [6]. Deterrence is the ability to make intrusions unlikely or computationally expensive.

Some effort has been made to make these infrastructures secure, however, new emerging threats have been witnessed that can compromise these infrastructures. Moreover, these infrastructures are complex as they exhibit a high level of dependency on the underlying physical devices and multiple supporting applications. Therefore, effectively managing the security for these infrastructures is complex, thus it requires further investigation to develop tailored defense mechanisms. These infrastructures still face challenges such as 1) accurate and real-time detection, economically feasible techniques, and robustness against stealthy attacks; 2) unpredictability and effectiveness against persistent attackers, and computationally inexpensive deterrence techniques.

In this thesis, we focus on intrusion detection and deterrence techniques for energy delivery systems of smart grids. The purpose of the work is to: 1) devise tailored intrusion detection and deterrence techniques that can provide accurate and real-time detection against stealthy attacks that can undermine the integrity of power system, and 2) identify the inherent limitations in existing intrusion detection techniques that allow stealthy and evasive attacks, and then devise an approach to reduce evasion margin for attackers.

We particularly focus on advanced metering infrastructure (AMI) and automatic generation control (AGC) of energy delivery systems of smart grids. First, we present an intrusion detection and deterrence technique for AMI. Though efforts have been made to devise intrusion detection techniques for AMI, no robust and practically fea-

sible defense technique exists. For example, the approaches lack practicality as all the contemporary work assumes the availability of sensors and computation power in the field. However, AMI do not provide enough computation power and memory in the field. Therefore, the existing approaches are not cost effective and have been avoided by the utility providers and vendors in the past. To this end, we develop an intrusion detection and deterrence technique for AMI which does not require additional hardware or sensor deployment in the field. The developed approach utilizes the already-collected logs in AMI for modeling the behavior and identifying the presence of anomalous behavior. Furthermore, we introduce key-based randomization to make evasive attacks infeasible for persistent attackers. Second, although bad data detection algorithms exist, recent studies show that the generation control system integrity can be disrupted by proficient attackers. We develop a data-driven multi-tier intrusion detection approach that predicts the system behavior in an online manner and flags deviation. The deviations are then verified in an offline manner by incorporating system-wide knowledge to confirm the presence of intrusion thereby reducing false positives. The approach can work with both single and multiple AGC. Third, we show that existing intrusion detection systems have inherent design limitations that allow attackers to successfully evade these systems. We devise a metric that can measure the evasion margin of these systems and show that some intrusion detection systems are more susceptible to evasive attacks than others. To countermeasure the evasive attacks, we develop a key-based deterrence technique to reduce the evasion margin for attackers by introducing a notion of randomized thresholds in existing intrusion detection systems.

In the later sections, we discuss the background of smart grids. We then discuss the limitations of existing approaches that need to be investigated in depth followed by the work objectives. Based on the limitations and work objectives, we develop intrusion detection and deterrence techniques in the subsequent sections.

1.1 Smart Grids Overview

The paradigm for energy infrastructures is being shifted to a new era. Legacy infrastructures are being replaced with the state-of-the-art smart grids. In the last few years, leading utility providers have taken the initiative of shifting to smart grids in order to efficiently manage the power while providing useful features [7]. The basic premise of moving to smart grids infrastructure is to manage it efficiently while providing uninterrupted low cost energy. The new smart infrastructure supports numerous cyber and physical devices that exhibit a high level dependency. Thus, smart grid is a highly complex and critical cyber-physical system that requires very rigid security considerations.

Different communication networks are connected to the power system for sensing measurements and control commands. These networks are associated with the supervisory control and data acquisition (SCADA) system for its real-time operation. SCADA system connects the generating stations, substations, corporate offices, and control center. SCADA is mainly responsible for monitoring and obtaining data acquisition from the remote equipment; and for controlling the equipment remotely either by the operator or automatically based on the data acquisition. Though it has several advantages along with the readily available communication infrastructure, it

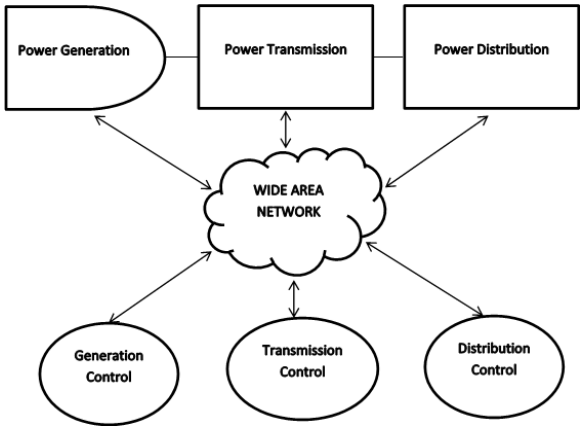


Figure 1: Schematic diagram of a smart grid

makes the system inherently vulnerable to cyber threats.

A basic schematic diagram of a smart grid is shown in Figure 1. It can be noticed that all the three control centers, i.e., generation, transmission, and distribution, are connected to the power infrastructure using cyber infrastructure. In this work we focus on generation and distribution network. Specifically, we focus on Advanced Metering Infrastructure (AMI) in distribution and Automatic Generation Control (AGC) in generation. Control centers take the sensor measurements and other data from the power network in order to analyze and send the control commands using the same infrastructure. This operation is part of SCADA schematic. Substations have the power generation ability for their area and they communicate with the generation control center to adjust the power generation to a required load. On the other hand, AMI is responsible for providing a bi-directional communication between smart meters at customers' premises and headend at utility office for the purpose of efficient energy management and remote management. Detailed background on AMI and AGC is discussed in the later relevant sections.

1.2 Limitations of Existing Intrusion Detection Techniques

In this section we highlight the limitations of existing intrusion detection techniques for the critical infrastructure under consideration, i.e., smart grid.

- Practical Infeasibility of AMI Defense: Contemporary intrusion detection techniques for AMI network are practically infeasible. They assume the hardware deployment in the field for capturing network traffic for detection purposes that is not cost effective, therefore, these approaches have not witnessed a widespread deployment. Moreover, these approaches require higher computational power and memory in the field, which is not available.

- No Standard Protocol Implementation in AMI: In the recent literature, protocol specification based defense mechanism has been proposed since AMI is homogenous in nature and exhibits a deterministic behavior. However, there is no standard implementation of the protocol available and all the vendors use proprietary protocols thereby limiting the applicability of the mechanism.

- Lack of Tailored Defense for Generation Control: No tailored approach exists for intrusion detection in generation control. Since the load behavior changes over the time, no data-driven approach exists that considers the temporal behavior of the power system and system-wide knowledge to identify anomalies. All the contemporary approaches utilize the traditional cyber defense mechanisms, thus the approaches are not well suited for generation control.

- Incapable to Measure Evasion Margin: Although there are several existing in-

trusion detection systems approaches, there is no mechanism or metric that can measure evasion margin of an intrusion detection system that is inherent due to its design regardless of the network under consideration. All the contemporary work focuses on comparing intrusion detection performance in terms of true positive and false positive on a given network.

- Not Effective Against Persistent Attackers: Existing intrusion detection systems employ static features for detection thus making it inherently vulnerable to evasive and stealthy attacks. Persistent attackers exploit this inherent limitation to go undetected thereby leaving the intrusion detection system useless against such attacks.

1.3 Work Objectives

The main objective of this work is to strengthen the security of smart grids by addressing the limitations of existing approaches mentioned in the earlier section.

- Configuration-based Modeling of AMI: Since AMI supports deterministic behavior due to a limited type of supported devices, configuration, and applications, we model the behavior using configurations. We utilize the logs generated in the AMI, thus the approach does not require hardware deployment in the field.

- Robustness Against Evasion in AMI: Since AMI behavior is predictable, evasion is possible. To this end, we mutate the configurations in order to introduce robustness against the evasion attempts.

- Behavior Prediction for Generation Control: Load behavior changes over time,

therefore, we predict the behavior that is used as a metric for anomaly identification in the generation control. Moreover, we introduce the system-wide knowledge in order to verify the presence of anomaly both in single and multiple AGC scenarios.

- Evasion Margin Measurement: We devise a metric that can measure evasion margin of a given intrusion detection technique that is due to its design limitation regardless of the underlying network. The metric is used to compare the performances of the intrusion detection techniques in terms of robustness against evasive attacks.

- Key-based Randomization to Defend Against Persistent Attackers: We devise a key-based randomization mechanism that overcomes the inherent limitation of existing intrusion detection systems due to their design. The mechanism reduces the evasion margin by introducing a notion of randomness for attackers thus making it computationally infeasible or less likely for attackers to go undetected.

CHAPTER 2: RANDOMIZATION-BASED INTRUSION DETECTION SYSTEM FOR ADVANCED METERING INFRASTRUCTURE

An important core network in smart grids is Advanced Metering Infrastructure (AMI). AMI provides bi-directional communication for monitoring and demand-response functions between end devices at customers' premises like smart meters and headend at the utility provider's office. Bi-directional communication in an AMI makes it possible for an end user to use the energy efficiently at low cost. Moreover, headend systems can remotely configure, upgrade, and request meter reading etc. using the AMI. This inherent criticality and availability of the AMI makes it a high potential target for the large-scale attacks that can potentially cause a major regional blackout.

Despite these facts, limited progress has been made so far in order to detect malicious behavior. Recent studies including those by the federal government have shown that AMI is facing immense potential threats [2–5, 8], which could affect the deployment and growth of smart grids. These outline existing vulnerabilities and exploitation, even though secure communication protocols were used. These vulnerabilities were exploited to penetrate in the AMI to gain control of a number of nodes for nefarious purposes. To countermeasure, some efforts have been made to prevent these threats [9–11]. Below we discuss that these proposals lack practical feasibility due to AMI's unique characteristics such as low computational power in the field. Furthermore, details of these works are provided in Section 2.2. Intrusion detection

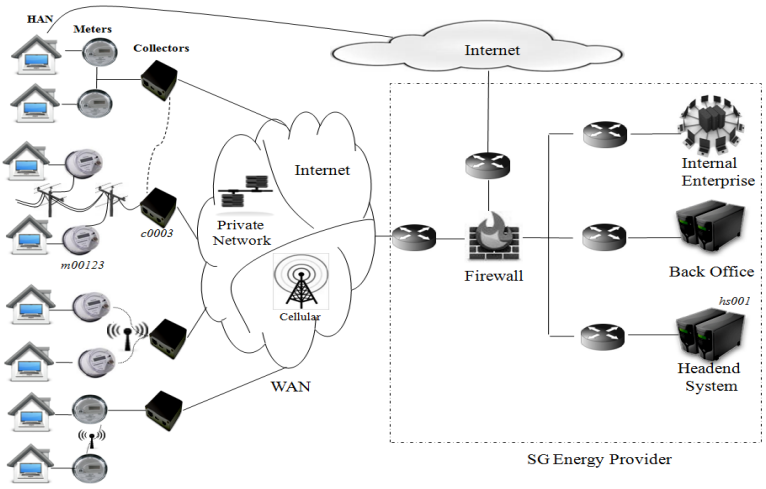


Figure 2: Basic AMI network

systems for networks such as RFID, which is similar to AMI in terms of low computational power in the field, have also been proposed [12]. However, the attack model for such techniques is different as the challenge in these networks is information integrity. Moreover, smart collectors that have already-collected logs may not exist in the RFID field networks and deploying these detection approaches may not be practically feasible. On the contrary, AMI threat model includes network level attacks such as denial of service and mimicry attacks. Therefore, to this end, we developed a technique tailored for AMI and its challenges. Please note that the work presented in this section is published in recent studies [13–16].

Figure 2 presents a typical example of the AMI. This figure shows that the meters communicate with the smart collectors using various mediums and smart collectors communicate with the headend system (and vice versa) using public networks. Unlike traditional networks, AMI has its own requirements which pose significant challenges for monitoring and intrusion detection since it may require capturing network traffic. First, sensor deployment in the meters in today's smart grids network is practically

infeasible due to the limited computational power and space resources at the node [17]. Second, although some researchers have suggested the meter-based sensors [10,11,18], smart grids providers as well as vendors avoid this option for today's smart grids networks due to the prohibitive cost increase associated with the large number of meter deployments in the near future. Therefore, most IDS proposals for the AMI lack practical feasibility. Though modern smart meters replacement is on schedule for some smart grids providers, our goal is to provide a generic solution which does not depend on replacement of modern smart meters.

Deploying detection modules at the smart collector provides the benefit of monitoring both the meter-collector and collector-headend communication. Moreover, AMI communication activity is by default logged at the smart collector thus it does not pose any extra burden. Although device configuration and the log's integrity is protected using headend-collector key pairing, this AMI feature was never exploited for monitoring and characterizing the AMI network communication behavior using the key and configuration parameters. AMI exhibits a predictable behavior in terms of communication and interaction features which can be characterized using device configurations. Therefore, deviations from the configuration-based characterized AMI communication behavior can be used for intrusion detection. Deterrence is introduced by randomizing the AMI behavior for attackers while keeping it deterministic for the system itself. AMI is a special purpose network and its traffic dynamics are often very low since it supports a limited number of protocols and it is configuration-driven. Moreover, similar devices from limited vendors are usually deployed. To exploit the limited behavior, simple specification-based intrusion detection techniques are pro-

Table 1: Some smart grids terminologies and their meaning

Terminology	Meaning
Smart Meter	A meter capable of two-way communication between the meter and the headend system
Smart Collector	An intermediate device that acts as proxy/router between the meter and headend system to enable the communication
Meter-based Sensors	A specific purpose sensor either built-in or deployed next to smart meter
Associated Collector	A smart collector that is responsible for its neighboring meter's communication with the headend system

posed in the recent literature [10] instead of traditional anomaly-based intrusion detectors.

2.1 Background

AMI is a core component of smart grids, which is responsible for bi-directional communication between headend systems and smart meters. Basic architecture of an AMI is shown in Figure 2. AMI has three main communication networks: Home Area Network (HAN), Neighborhood Area Network (NAN), and Wide Area Network (WAN). As shown in Figure 2, HAN can be realized as the customer home network that is connected with the smart meter. Therefore, the smart meter acts as an interface for an HAN as it connects AMI and HAN in the bi-directional communication mode. NAN is responsible for the communication between the smart collectors and the smart meters. NAN scales from hundreds to thousands of nodes including the smart meters and the smart collectors. Lastly, WAN is mainly responsible for the backhaul connectivity of the NAN to the headend system. Basic smart grids terminologies and their meanings are shown in Table 1.

Different communication channels can be used in an AMI network, i.e., HAN,

NAN and WAN. For example, NAN may use wireless or power line communication in order to interact among nodes. Moreover, WAN uses the high range and bandwidth technologies for the purpose of connecting NAN to the headend. It can use cellular technologies or another dedicated communication medium. Usually utility providers utilize the existing network from a third party in order to connect the NAN to the headend.

2.2 Related Work

The work focuses on configuration-randomization for intrusion detection in AMI, which is a component of smart grids. First, we discuss randomization-based defense techniques. Second, we discuss intrusion detection for traditional networks followed by intrusion detection for smart grids. We then discuss the studies that highlight the cyber security issues in AMI and existing intrusion detection systems for AMI. Finally we discuss the novelty of our approach and how it addresses the challenges/unique characteristics of AMI.

Randomization-based defense techniques, also referred as moving target defense, have been presented in the recent past [19, 20]. Several studies show the effectiveness of IP address randomization in traditional networks [21–24]. In addition to traditional networks, randomization techniques for smart grids are also presented [25, 26]. Some studies also presented the randomization for network route and infrastructures for improved performance and resiliency against attacks [27–30]. However, in this work, we randomize AMI configurations for intrusion detection.

Intrusion detection has gained tremendous attention and many anomaly-based IDS

techniques have been proposed for cyber systems [31–36]. For example, n-gram based intrusion detection and markov chain based approaches have been proposed in the past [37–40]. n-gram can be seen as prediction using n-1 markov model. However, a markov chain moves from one state to the next using a weighted list of possible future states. n-gram can calculate the likelihood for next state, however, it only gives the statistical distribution and cannot maintain the temporal order. Since AMI supports limited protocols and configurations, temporal specifications can be defined for markov chain model to devise specification-based intrusion detector. Though n-gram can be used to design anomaly detection systems based on statistical distribution, we believe a specification-based approach is better suited for intrusion detection in AMI due to its nature. Recently proposed techniques have reduced the false positive rate for traditional networks [41–44]. However, we focus our attention on smart grids since these techniques require learning and are computationally extensive. Therefore, they do not fit the application requirements of smart grids.

In [45], an anomaly detection technique for smart grids has been proposed. It considers the temporal anomalies and rank potential intrusion events based on the credibility impact on power system. It also considers the scenario where simultaneous cyber intrusions are launched over multiple substations. The work focuses on the cyber security of substations in smart grids. A survey on supervisory control and data acquisition system (SCADA) specific intrusion detection and prevention is presented in [46]. However, these techniques do not cater for the intrusions on an AMI. An anomaly detection module for industrial control systems is presented in [47]. The work uses self-organizing maps to identify anomalies in the restricted IP network

that exhibits a deterministic behavior. A prototype is shown using self-organizing maps built on top of Bro network security monitor. The work does not take into account the computational resources availability, as it uses machine learning based approach, in the AMI field network hence can not be used at the smart collector level. The work [48] utilizes the theory of the Best Choice problem to identify the best time to launch an attack against process control systems. It considers that the adversary has compromised actuator or sensor signals and needs to identify the best time to maximize the damage in terms of impact. However, it does not deal with the defense mechanisms specific to AMI and its unique characteristics.

In addition to intrusion detection, few studies have been conducted to highlight the cyber security issues in AMI and smart grids [2,6,49–52]. A distributed intrusion detection technique in a multi-layer architecture of smart grids is proposed in [18]. It proposes a three layer architecture for home area network, neighborhood area network and wide area network. It uses trained support vector machines (SVM) for attack classification at each layer. In [9], the requirements and architectural directions are discussed for intrusion detection in an AMI. Based on the requirements and architectural directions highlighted in [9], a specification-based intrusion detection technique has been proposed in [10]. This approach defines a formal model for C12.22 standard protocol, which is used by different meters in the AMI for communication purposes. It verifies the specification at the application layer in order to identify anomalous behavior. The proposed approach is protocol specific. However, not all the deployed smart grids are using this protocol for communication. Therefore, it cannot be applied to those AMI infrastructures. Moreover, there is no standard implementation

of the protocol available and many vendors keep the implementation proprietary [53]. An intrusion detection architecture for the AMI has been proposed in [11]. It uses a data mining based intrusion detection technique for the AMI. It requires deployment at the smart meter, smart collector, and headend system.

All of the proposed approaches for intrusion detection in the AMI assume a computationally expensive intrusion detection module deployment at the meter level. However, this may not be practically feasible since meters do not have enough computation power [17]. If the detection modules are to be deployed as a stand alone unit next to the meter, it requires significantly higher cost. We argue that due to this higher deployment cost, industrial deployments of the intrusion detector for the AMI have not yet been witnessed. In this work, we develop a solution that is cost effective and practical since it can be deployed either in the headend system or in the AMI.

The novelty of our approach lies in characterizing AMI configuration as a baseline for developing a reliable IDS (high detection and low false positive rate), while considering the practical computation and operational constrains of AMI. To the best of our knowledge, no approach exists yet that uses configurations for deriving the LTL properties for markov model. In [14], we developed a proactive AMI configuration-mutation based approach and a reactive AMI configuration-modeling based approach was developed in [13]. However, in this work we combine the proactive and reactive approaches by mutating the AMI configuration parameters and then model them to improve the accuracy of the IDS.

2.3 Challenges

AMI comes with its own challenges that include, but are not limited to: 1) physical environmental conditions since it is deployed in the field, 2) limited computation and memory availability in the field devices like smart collectors and smart meters, 3) deployment costs of any new hardware in the field at thousands of locations. Therefore, the existing intrusion detectors which require higher computation power and memory can not be used for the AMI. Moreover, traditional intrusion detectors require capturing the network traffic. Since capturing the network traffic requires dedicated high computation power hardware, the approach is not practically feasible due to a higher deployment cost associated with it. However, smart collector acts as a gateway for smart meters to communicate with the headend. Moreover, smart collectors log the network activity as a default feature. This can be exploited to build an intrusion detector for the AMI. Therefore, the problem here is *To reduce the deployment cost and make it practical, the developed intrusion detector should be able to work offline in the headend by pulling the smart collector's logs instead of deployment in the field.* The deterministic behavior can be modeled using the network activity information collected at the critical points only.

2.4 Contribution and Approach Overview

In this work, we present a novel stochastic model checking intrusion detection technique. It is designed to meet the requirements and log characteristics of the AMI. We show that AMI behavior is deterministic and can be predicted easily. This allows attackers to evade intrusion detection systems by mimicking the behavior. To make

it robust against evasion, we mutate the behavior using the pre-shared secret key, which keeps the behavior deterministic for the smart collector. We model the AMI infrastructure behavior using the logs generated at the smart collector. A stochastic model based on 4-th order markov chain is used, since it exhibits a low conditional entropy in order to represent the AMI probabilistic behavior. Probabilistic behavior is observed as a result of its configuration and nature of the network. Specifications written in Linear Temporal Logic (LTL) are automatically generated from the a-priori known configurations of the AMI devices (smart meters and collectors), which in turn are then probabilistically verified using the stochastic model generated from the smart collector's logs. The developed technique exhibits high accuracy and it can be easily deployed in the existing AMI of smart grids. For experimentation and evaluation we use a real-world dataset of more than two thousand meters obtained as a result of our collaboration with a leading smart grids based utility provider. Our experimental evaluation shows promising results for the developed model, i.e., accuracy rate of more than 95% with negligible false alarms of 0.1%, i.e., 0.35 to 0.50 false alarms per meter per week.

2.5 Attack Model

Since we work with the logs collected at a smart collector, attacks that do not involve any communication with the smart collector or do not create a log entry would not be detected. This has been highlighted in the limitations of our work. However, the effect of such attacks would be limited to a particular area. In short, our focus is on the large scale attacks, which include compromising a large number of meters

to cause a major blackout in the area. These attacks include, but are not limited to, spoofing, denial of service, distributed denial of service, scanning, penetration, evasion, mimicry, etc. For example, a denial of service attack on the smart collectors or its associated meters will cause service disruption in the area. Similarly, distributed denial of service attacks on the large number of meters or smart collectors can cause a major blackout in a wider area.

Since the infrastructure exhibits a deterministic behavior and is homogenous in nature, spoofing, mimicry, and evasion techniques can inject similar traffic without being detected, thus resulting in the destabilization of the infrastructure. Mimicry attacks can compromise a number of meters and operate as though they were legitimate, thus bypassing detection. In the recent literature [54], it is shown that smart meters send the authentication password in the clear text, which may require a physical access. It is also shown that spoofing and replay attacks are possible. It is intuitive that evasive attacks, which mimics the behavior, would not be detected by the existing AMI detector [10]. The underlying reason is that the detector follows the protocol specification to detect the intrusions and mimicry attacks mimic the behavior to follow the protocol. However, the developed mutation based approach provides defense against such attacks. For example, randomizing the time interval would defend against the replay attacks since the communication will not be expected by the smart collector at the replay time. Similarly, type and size parameters will defend against spoofing and mimicry attacks since the attacker wouldn't know what type and size of communication is expected by the smart collector at a given time. Moreover, randomizing relay nodes will defend against the DoS and reconnaissance

attacks since attackers would not know that at a given time which relay node to use for the communication. Since multiple parameters are randomized, it will be difficult for the attacker to guess/predict all the parameters at a given time in order to stay evasive. Therefore, it makes it more difficult for the attacker to mimic the normal behavior and stay evasive. The basic assumption for key-based mutation here is that the key is stored in a Trusted Platform Module (TPM) which can not be accessed by other processes [55]. In our case study all the deployed smart meters had built-in TPM modules.

Traditional attacks like denial of service will be detected by the property 10, which is defined later, as it will create multiple entries in a time window shorter than $T1$. Similarly, a distributed denial of service will cause the property 12, which is defined later, to be invalid since it will cause multiple sources to create log entries in a time window shorter than $T2$. If a penetration attack is launched against the AMI, it will be detected depending on the attack graph. If the penetration is supposed to go through the smart collector or it tampers with the meter (by changing the configuration), it will be detected since the meter will not behave according to the a-priori known configurations. Similarly, malware that tampers with the meter configuration will be detected as well.

Data injection attacks that only tamper with the usage information will not be detected. This work is not focused on the energy theft by individual users. Intrusion detectors specially designed for detecting an individual user's energy theft have been proposed recently [54, 56].

To evaluate the developed scheme, we generate attacks in a controlled environment

in a smart grid testbed. We use different attack scenarios that include not only different attack types but also different attack locations. Since the home area network has access to the smart meter, we consider two scenarios for this location: 1) try to compromise the meter and tamper with its configuration, 2) using the smart meter as entry point to an AMI, we launch scanning, DoS, evasion, mimicry and data injection traffic. For scanning and DoS, we generate low rate attacks i.e., 0.1pkts/sec to 1pkts/sec. Both the above mentioned scenarios were implemented for these attacks. For evasion, mimicry, and spoofing, we placed a switch in between the meters and collector, and attached an attack machine to it. We wrote a simple program that uses the same configuration as that of a meter and generated similar reading reports in the same format. In order to be evasive, less than 5% of the total generated traffic by the machine was injected traffic and 95% was mimicked traffic. For mimicry and spoofing attacks, all the traffic was mimicked traffic without any injected traffic. Injected traffic includes malicious commands like random file uploads, requesting reports at irregular intervals (requesting report itself is not an attack), administrative commands without proper authorization and failed authorization attempts. These logs were collected from the smart collectors and were labeled as malicious. Malicious logs were mixed into the real-world logs collected at an AMI of the utility provider for the purpose of the accuracy evaluation. It was ensured that the volume of malicious logs did not exceed 10% in the mixed log (benign and malicious).

2.6 Modeling Intrusion Detection and Deterrence Approach

In this section we show that AMI behavior is predictable using a markov model. We first show that log entries exhibit a certain level of temporal dependence, thus these log entries can be modeled using markov chains. To find the order of markov chain that can accurately predict the future behavior or states, we calculated conditional entropy on different markov chain orders. It can be intuitively argued that, as long as the log entries are produced by benign events, i.e., no attacks, the log entries observed should exhibit a certain level of temporal dependence as it is a configuration driven network. In case of malicious behavior, perturbations in this dependence structure can be flagged as anomalous. Therefore, the level of temporal dependence can serve as an important metric for modeling the log entries.

To find the temporal dependence in log entries we use autocorrelation, which measures the on-average temporal dependence between the random variables in a stochastic process at different points in time. For a given time lag k, that represents log entries at time k, the autocorrelation function of a stochastic process X_k is defined as:

$$\rho[k] = \frac{E\{X_0 X_k\} - E\{X_0\}E\{X_k\}}{\sigma_{X_0}\sigma_{X_k}}, \tag{1}$$

where $E\{.\}$ represents the expectation operation, σ_{X_k} is the standard deviation of the random variable at time lag k (log entries at time k). The value of the autocorrelation function lies in the range $[-1, 1]$, where $\rho[k] = 1$ means perfect correlation at lag k

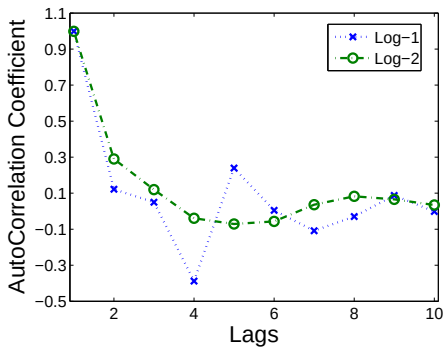


Figure 3: Autocorrelation coefficient trend

(which is obviously true for $k = 0$) and $\rho[k] = 0$ means no correlation at lag k.

Figure 3 shows the autocorrelation function plotted against the log entries at different times represented by lags. For both the logs collected – Log-1 and Log-2 – a certain level of temporal dependence can be easily observed at small lags. This correlation decays in time and eventually drops down to a negligible value. Temporal dependence is present for two reasons: 1) meters respond to the smart collector requests in a short period of time and 2) regular requests and reports are seen, thus justifying the homogeneity.

Since log entries exhibit temporal dependence at small lags, they can be modeled using markov chains. Furthermore, it is well-known that a decaying temporal dependence structure can be accurately modeled using markov chains [57]. Therefore, the concern here is to identify the order of markov chain model that should be used to accurately model the log entries. To determine the markovian order, we conduct analysis on different markov chain orders. The order can be identified by noting the probabilities at different markov chain orders. If the probability is low at a given order, the next log entry can be predicted with high accuracy. For this reason we first

calculate the probabilities of states on different markov chain orders and then use the conditional entropy based measure devised in [57] on different markov chain orders.

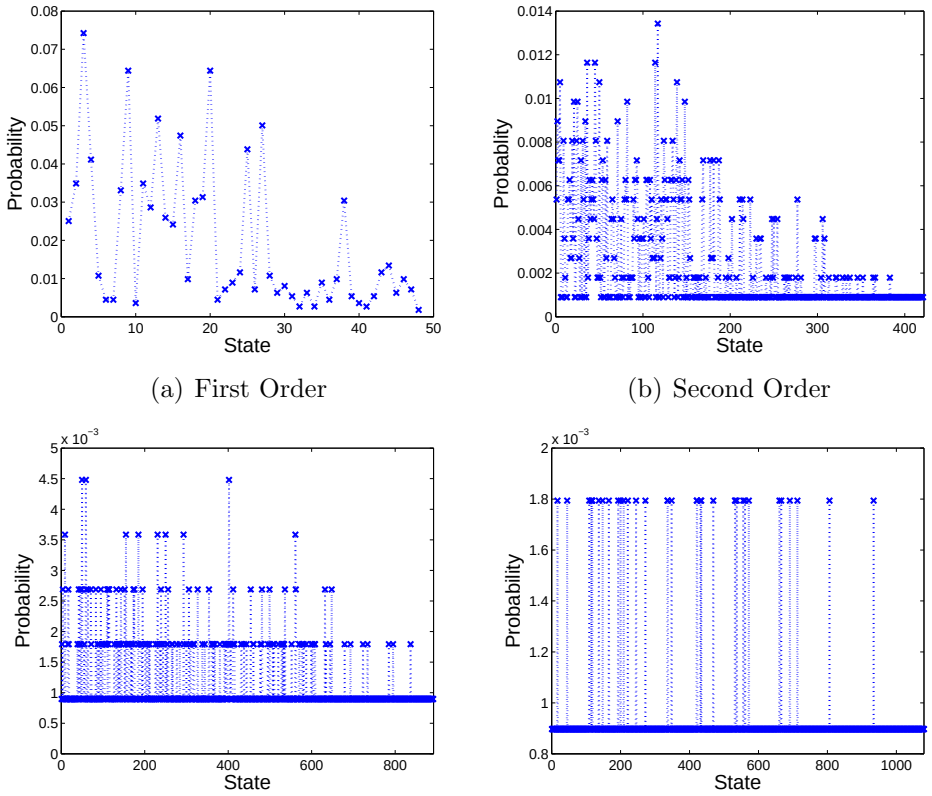
We define a markov chain based stochastic model as follows: Let the log entry tuple at discrete time instance n represents the realization of a random variable derived from a stochastic process X_n. This process is a markov chain if it satisfies the markov property, which is defined as:

$$\begin{aligned} &\Pr\{\mathrm{X}_n = j \,|\mathrm{X}_{n-1} = i, \mathrm{X}_{n-2} = i_{n-2}, \ldots, \mathrm{X}_0 = i_0\} \\ &\qquad = \Pr\{\mathrm{X}_n = j \,|\mathrm{X}_{n-1} = i\} = p_{j|i}. \end{aligned} \tag{2}$$

In other words, the probability of choosing a next state is only dependent on the current state of the markov chain.

In the present context, we can define a markov chain model X_n for an entry by treating each unique log entry individually. Therefore, the number of total states ψ will be dependent on the number of unique log entries. Based on this state representation, we can define a 1-st order markov chain denoted by $X_n^{(1)}$. Similarly, an l-th order markov chain, $X_n^{(l)}$, can be defined in which each state is an l-tuple $\langle i_0, i_1, \ldots, i_{l-1}\rangle$ representing the l log entries. This will increase the size of state space ψ since different combinations of l-log entries are possible.

Figure 4 shows the state probabilities calculated as described above for log entries using different markov chain orders. We calculated it for up to 4-th order markov chain as the possible number of states increase, computational complexity increases and the scalability decreases when markov chain order increases as discussed later in



(a) First Order (b) Second Order

(c) Third Order (d) Fourth Order

Figure 4: State probabilities for higher order markov chain

Section 2.7. The figure shows that the total number of states ψ increases, which is the *x-axis* in Figure 4(a) to (d), when markov chain order increases. Furthermore, it can be noticed that the states get clustered with respect to probability values (*y-axis*), therefore, very few states have a higher probability of occurrence and the rest of the states' probabilities drop to a negligible value. The underlying reason for this behavior is the predictable dependency in AMI network behavior/activities. As AMI is a configuration-driven network, behavioral patterns can be observed in the log activities. For example, reading reports/responses always follow reading requests. Also, meters generate readings at specific intervals. These (probabilistically) recurring patterns due to state dependencies dominate the AMI activities and they are more

detectable using a higher order of markov chain. Therefore, as the order increases, fewer clustered states have higher probabilities. Since the increase of markov chain order also increases the complexity, our aim in this research is to balance between detectability/accuracy and complexity. To explore it further, we use the conditional entropy based measure [57] for the log entries for different markov chain orders. This measure will tell at what order of markov chain most of the information about the next time instance, i.e., log entry.

Conditional entropy, $H(B|A)$, of two discrete random variables A and B characterizes the information remaining in B when A is already known. Phrased differently, conditional entropy is information about B *not* given by A. If A and B are highly correlated, most of the information about B is communicated through A and $H(B|A)$ is small. On the other hand, if p_A and p_B (which respectively represent the probability mass functions of A and B) are quite different then $H(B|A)$ assumes a high value, which means that most of the information about B is not given by A. In the limiting cases, $H(B|A) = 0$ when $A = B$, while $H(B|A) = H(B)$ when A and B are independent.

We can find the conditional probability of the 1-st order markov chain as:

$$H^{(1)} = -\sum_{i \in \psi^{(1)}} \pi_i^{(1)} \sum_{j \in \psi^{(1)}} p_{j|i}^{(1)} \log_2 \left(p_{j|i}^{(1)} \right), \tag{3}$$

where $\pi_i^{(1)}$ is the average probability of being in state i which is computed by counting the total number of times each state is visited and then normalizing the frequency histogram.

The measure $H^{(1)}$ defines how much average information is remaining in log entry

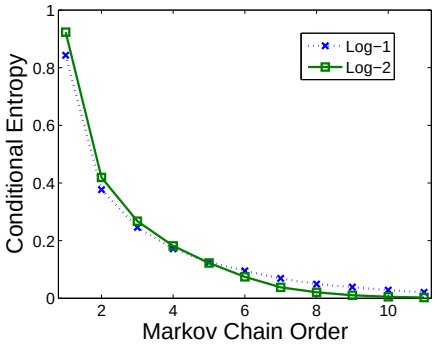


Figure 5: Conditional entropy trend over markov chain orders

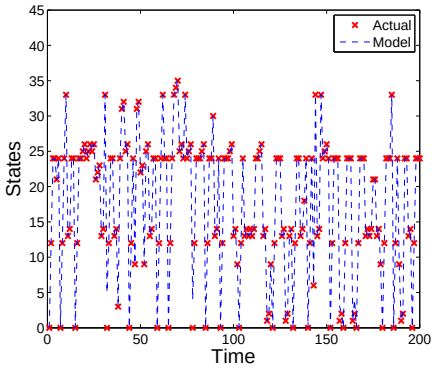


Figure 6: Prediction accuracy using fourth order markov model

X_n when it is calculated using log entry X_{n-1}. If the entry is not highly correlated with entries before X_{n-1}, $H^{(1)}$ will be relatively large implying that information about X_n *not* provided by X_{n-1} is high. In such a case, we use higher l-th order markov chain, $X_n^{(l)}$ as defined previously, in which each state is an l-tuple $\langle i_0, i_1, \ldots, i_{l-1} \rangle$. The conditional entropy of $X_n^{(l)}$ defined on $\psi^{(l)}$ can then be computed using the same method as Equation (3). The number of previous entries required to accurately predict the next entry can then be determined by plotting $H^{(l)}$ as the function of the markov chain order, $l = 1, 2, \ldots$. The order at which the conditional entropy saturates defines the total number of previous entries, which have conveyed as much information of the present entry as possible. It can be clearly seen in Figure 5 that log

entries exhibit a decaying conditional entropy trend over higher order markov chain. It exhibits an exponential decay trend until the 4-th order, which was expected since very few states had higher probability in Figure 4(d) at the same order. It is clear that most of the information about the next instance is already given at the fourth order since conditional entropy drops to a negligible value, i.e., ≤ 0.2. Therefore, we model using the fourth order since it gives enough information and increases predictability. To this end, we show the prediction accuracy of the fourth order markov model. We divide the log into two halves. First half is used to learn the model for prediction and second half is used to test the prediction accuracy. It can be seen in Figure 6 that the model can predict the future log entries with a high accuracy, i.e., $> 95\%$. Therefore, an attacker can easily learn the AMI behavior to devise an evasion technique. To combat this, we present a robust behavior mutation based technique for AMI that minimizes the prediction notion and increases the attack detection.

We also show the results for a third order markov chain that exhibits lower accuracy as compared to fourth order but improves scalability since it exhibits a lesser number of states. In later sections, we show that a higher order markov chain exhibits better accuracy but it increases the number of states as compared to the lower order. Therefore, the choice of markov chain order is a trade-off between scalability (number of states) and accuracy. This is discussed in detail in the later sections, however, we build a model using the fourth order markov chain.

2.6.1 Markov Model for AMI Logs

Since we work with logs of smart collectors, we first look at the format of a log entry. It can be represented as:

$$t, s_{id}, d_{id}, rel, sz, ty$$

where t represents the time stamp of the event. s_{id} and d_{id} refer to the source and destination, respectively. However, rel refers to the relaying/forwarding node for the smart meter. It can either be the neighboring meter or meter itself. sz and ty denote the size and type of communication, respectively.

Since the goal of the work is to model the AMI behavior for intrusion detection we first discuss the AMI modeling approach followed by the properties specification for model checking to verify the presence of intrusions. To counter the evasion and mimicry attacks, we then develop the configuration randomization module.

We model the AMI behavior using smart collector logs that are generated as a result of randomized configurations. We can encode the state of the network with the following characteristic function derived from the log entry format:

$$\sigma : s_{id} \wedge d_{id} \wedge rel \wedge sz \wedge ty \rightarrow \{true, false\} \tag{4}$$

The function σ encodes the state of the network by evaluating to *true* whenever the parameters used as input to the function correspond to the log entry in the smart collector. If the AMI observes 5 different log entries, then exactly 5 different assignments to σ function will result to *true*. Since AMI communication occurs generally at

periodic intervals like meter readings after every X seconds or minutes and reading response comes after reading request, AMI can be modeled using markov chains. Furthermore, state transitions are probabilistic in nature as σ is a conjunction of random variables $s_{id} \wedge d_{id} \wedge rel \wedge sz \wedge ty$ and these variables can observe different values at a given time hence leading to a different state. Since smart collectors function independently of each other in terms of collecting meter readings, we learn the markov model for each smart collector separately as discussed below.

A Labeled Markov Chain (LMC) is a tuple $M = \{Q, \Sigma, \pi, \tau, L\}$ which is a state-transition graph in which states are labeled, where Q is a finite set of states, π is an initial probability distribution τ is the transition probability function and L is a labeling function. Atomic propositions AP are assigned to states by a labeling function using $\Sigma = 2^{AP}$. Each state was assigned a unique label derived from σ, which is used to define the state. A probability distribution for sequence of states can then be defined using a markov chain. LMC M with alphabet Σ induces the probability distribution P_M^{π} over Σ^w through labeling of the states.

Set of states Q is strongly connected if for each state pair (q_i, q_j) there exists a path $q_0, q_1, \ldots, q_n$ such that $q_h \in Q$ for $0 \leq h \leq n$, $\tau(q_h, q_{h+1}) > 0, q_0 = q_i$, and $q_n = q_h$. Therefore, if Q is strongly connected, then M is said to be strongly connected. A distribution π_M^s is a stationary distribution for M if it satisfies

$$\pi_{(q)}^s = \sum_{\acute{q} \in Q} \pi^s(\acute{q}) \tau(\acute{q}, q)$$

Since we are interested in keeping the history of previously visited states, we focus on the probability suffix automata (PSA). A PSA is an extended LMC with a labeling

function $H : Q \rightarrow \Sigma^{\leq N}$, which represents the history of previously visited, at most N, states. However, if the history is fixed to N, i.e., $\Sigma^{=N}$, the LMC will be called $(N+1)th$ order markov chain since it includes the current state also. Therefore, each state q_i will be associated with a string s_i such that $s_i = H(q_i)L(q_i)$.

Since the AMI system under consideration is a real-time system and can not be restarted with different initial states, we argue that the model learning technique should be able to start observing data from the system at any given time and can work with a single long sequence of observations [58]. Suppose we have sequence $S = \sigma_1, \sigma_2, \ldots, \sigma_n$, $\sigma_i \in Q$, where σ represents a state as shown in Equation 4. Since our statistical analysis showed that conditional entropy is negligible at 4-th order, therefore, we use 4-th order markov chain. A finite state machine having directed graph can be learned from the given sequence S. Each state in the graph at time i will be represented by a tuple of 4, i.e., $\langle \sigma_{i-3}, \sigma_{i-2}, \ldots, \sigma_i \rangle$, where σ_i is the $L(q_i)$ and the rest are $H(q_i)$. Therefore, it can be realized as s_i in the finite state machine.

Algorithm 1 explains the learning of a markov model from the given sequence of log entries. It initializes an empty graph representing nodes and edges, and then starts observing the sequence S. It utilizes a sliding window approach where window slides at instance i by one entry, i.e., σ. However, the size of the window to observe $s_i \in S$ is kept to 4, which is the order of model. If s_i already exists in a graph then a directed edge from s_{i-1} to s_i is created, if the directed edge does not exist already. However, if s_i does not exist in graph, then a node is also created for s_i. This process keeps repeating until S is empty. Once the state machine is created, it is easy to calculate the transition probability matrix for that. For each state s_i in

Algorithm 1: Learn markov model

Data: Sequence S
Result: Finite state machine based on 4-th order Markov Model
Initialize empty Graph representing nodes and edges;
$S = \{\sigma_i | \forall \sigma_i \in \Sigma\}$;
$\forall \sigma_i \Pr(\sigma_i) > 0$;
while $S \neq \phi$ **do**
  Slide window by one σ at instance i ;
  Pick $s_i \in S$;
  if $s_i \in Graph$ **then**
    Make directed edge from s_{i-1} *to* s_i ;
  end
  else
    Create node s_i in Graph ;
    Make directed edge from s_{i-1} to s_i ;
  end
end

graph, $\sum_{\forall \sigma_i \in \Sigma} \tau(s_i, \sigma_i) = 1$.

Since a log entry σ is a conjunction of different variables, total possible combinations can exceed and may require a lot of processing power. However, it can be calculated for each network under consideration. In our case study, 10 bits were assigned to s_{id} and d_{id}, 4 bits for relay, 8 bits for sz, and 3 bits for type ty of communication. Therefore, the possible number of σ are $2^{10} \times 2^{10} \times 2^4 \times 2^8 \times 2^3$, which is a relatively large number. Since the model treats each smart collector's log separately, either source or destination of each log entry will be fixed to the ID of that particular smart collector. Moreover, a smart collector can only be connected to its neighboring nodes/meters. In our case study, the smart collector was connected to 8 other devices. Therefore, the number of σ reduces to $1 \times 8 \times 8 \times 2^8 \times 2^3$, which is relatively smaller. Since 4-th order markov model is being used, possible combinations of four σ can yield to a lot of states. To this end, the developed algorithm only takes the combinations

that are observed in the sequence S and only keeps the edges that are observed since all the combinations are not possible due to configuration, thus reducing the size of transition probability matrix by ten(s) to hundred times.

2.6.2 Model Checking for Intrusion Detection

Since the developed AMI model is based on a markov chain and exhibits a temporal dependence, we define properties in Linear time Temporal Logic (LTL) [59]. Unlike traditional model checking, stochastic model checking allows you to check that with what probability the property is satisfied by the model. These probabilities can be thresholded in order to accommodate the unseen behavior up to a certain extent. LTL over the alphabets Σ is defined by the syntax:

$$\varphi ::= true \mid \sigma \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi \mid \bigcirc \varphi \mid \varphi_1 \cup \varphi_2 \quad (\sigma \in \Sigma)$$

The derived additional operators $\Box$ (always) and $\diamond$ (eventually) are also used in the LTL.

Let φ be the LTL formula over Σ. An LTL formula can be satisfied for a sequence of alphabets s, which is a state definition in our case having $s = \sigma_1, \sigma_2, \ldots, \sigma_n$ where $\sigma_i \in \Sigma$. Therefore, the probabilistic LTL can be defined as:

$$\phi ::= P_{\bowtie p}(\varphi), \quad \bowtie \ \in \ \{\geq, >, \leq, <, =\}; \ \ p \in [0,1]; \ \varphi \in \text{LTL}$$

Since the system under consideration is an online system and can not be restarted with a specific initial state, we use the stationary distributions for satisfiability. LTL properties can be verified with the markov chain model built in the earlier section. For example, if a configuration parameter defines the sampling rate and report size,

a property can be written that whenever a report request is received the reply should have this particular size. Temporally it can be stated that given the system is in 'request' state, the next expected state is 'reply_with_size_h'. In the PRISM model checker tool [60], the 'next' state operator is defined using 'X'. For the given state, you can 'filter' the state space to 'current' state only. We wrote a small parser that reads the configuration and generates the properties in LTL format for the tool.

The properties can be defined to make sure that AMI is operating per its configuration. For example, whenever a reading request is generated, it should be followed by a reading report and the reading report should have a size based on its configuration. Furthermore, security related properties can also be defined such as all the meters associated with a collector should not send a report at the same time to flood a smart collector. Therefore, the properties can be derived from the configurations and the security control guidelines such as NISTIR 7628 published by NIST, which defines its operational constraints. Since the configurations shown are related to the reading report, below we show some examples of the properties derived from the configuration. Let γ be the number of meters associated with a smart collector. One basic example is that whenever a report reading request is generated, a meter should respond with a report. It can be formulated as:

$$\phi ::= P_{\geq d_1}(rrep_i | rreq_i), \quad 1 \leq i \leq \gamma \tag{5}$$

where $rreq_i$ and $rrep_i$ represent the reading request and reading report, respectively, for meter i. However, d_1 is used as a probability threshold that this property should be satisfied with the probability greater than or equal to d_1. The property defined

here is in conditional probability syntax, however, it can be represented in LTL as:

$$P_{\geq d_1}(\Box(rreq_i \rightarrow \bigcirc rrep_i)) \tag{6}$$

This is a strict property since it says that the *next* state has to be the $rrep_i$. However, a relaxed property can be defined as:

$$P_{\geq d_1}(\Box(rreq_i \rightarrow \diamond rrep_i)) \tag{7}$$

which says that eventually $rrep_i$ will be seen once $rreq_i$ is observed. However, we use the strict property (Equation 6) in our experiments. Similarly, it can also be defined that the report generated should have a size with in the limits defined since the sampling rate is fixed. It can be formulated as:

$$\phi ::= P_{\geq d_2}(rsz_i | rrep_i), \quad 1 \leq i \leq \gamma \tag{8}$$

where rsz_i denotes the report size for meter i. However, $rsz_i \in sz_i$, which is a valid report size set for meter i. Moreover, Equations 5 and 8 can be combined to show the temporal behavior, i.e., whenever a reading request is generated, it is followed by the reading reply which has a valid size.

Moreover, a meter should not send the reading report twice in the next T_1 consecutive time periods. It can be formulated as:

$$\phi ::= P_{\leq d_3}(rrep_i | rrep_i), \quad 1 \leq i \leq \gamma \tag{9}$$

where d_3 is thresholded with $\leq$ that the probability of seeing such a behavior should be less than d_3. Suppose t_1 is a counter which observe values in the range $\{1, 2, \ldots, T_1\}$.

Equation 9 can be represented in LTL as:

$$P_{\leq d_3}(\Box\ rrep_i \rightarrow (\neg rrep_i \cup t_1 \geq T_1)) \tag{10}$$

To avoid flooding the smart collector with reports from multiple meters at the same time, the associated meters were configured to have different reporting intervals. Therefore, smart collectors will not receive consecutive reports from multiple meters in consecutive T_2 time periods. It can be formulated as:

$$\phi ::= P_{\leq d_4}(rrep_j | rrep_i), \quad i \neq j, \quad 1 \leq (i,j) \leq \gamma \tag{11}$$

This prevents the multiple meters from sending the reports after each other for consecutive T_2 time periods. It can be represented in LTL as:

$$P_{\leq d_4}(\Box\ rrep_i \rightarrow \neg rrep_j \cup t_2 \geq T_2), \forall_{j=1}^{\gamma} \tag{12}$$

To introduce the unpredictability notion for the attacker, a meter can not select the same neighboring meter for relaying the report in consecutive T_3 time periods. It can be represented as:

$$P_{\leq d_5}(\Box\ rel_{i,j} \rightarrow \neg rel_{i,j} \cup t_3 \geq T_3), \forall_{i,j=1}^{\gamma} \tag{13}$$

where $rel_{i,j}$ represents that meter i relayed the report via meter j. Similarly, to avoid conflict and introduce randomness, a meter j can not be selected by multiple neighboring meters, i.e., i and k for relaying the report in a given time window T_4. This also avoids the overwhelming of one meter by its neighboring meters. It can be

represented as:

$$P_{\leq d_6}(\Box\ rel_{i,j} \rightarrow \neg rel_{k,j} \cup t_4 \geq T_4), \forall_{k=1}^{\gamma} \tag{14}$$

Moreover, the thresholds d_x are learned from the model built using the benign logs collected under normal conditions, i.e., without any attack. However, they can also be manually configured based on the requirements of the network under consideration. We show the examples of reading reports' property specification, however, other types of configurations are also specified the same way using LTL. These properties are dependent on the configuration of the network under consideration.

2.6.3 Mutable Configuration Deterrence Approach

In this section we discuss the deterrence approach using configuration randomization for smart meters configurations. Suppose we have γ neighboring meters (8 neighboring meters in our case study) associated with a smart collector c. We use a range of values instead of a fixed value for the configuration parameters. In this work we show it for three configuration parameters of a smart meter – report size, reporting interval, and relaying node. Since sampling rate defines the size of the report, we use size as a parameter in our discussion. Basic operational constraint for our work is that both the smart collector and meter should have a unique pre-shared key. First, we discuss the set of constraints on the assigned ranges followed by the mutation algorithm.

Random Relay Mutation: Meters send the reports to the headend through smart collector. However, to reach the associated smart collector they may use a neighboring meter as a relay. We can apply mutation by selecting a random neighbor meter for

relaying the report to an associated collector. The mutation can defend against reconnaissance and DoS attacks since the meter will use a random neighbor to relay the report.

The following constraints should be satisfied for random relay mutation:

- A unique neighbor should be assigned to every meter in every time window (*Unique Assignment Constraint*).
- A same neighbor should not be selected in two consecutive time windows by a meter for relaying (*Non-repeat Constraint*).
- No meter should get more than U_r relaying requests at any given time window (*Capacity Constraint*).

We can formalize the constraints for T consecutive time windows. We denote set $U = \{1, 2, \ldots, \gamma + 1\}$ to be the index of all meters and the collector, where index i $(1 \leq i \leq \gamma)$ is the index of meter m_i and $\gamma + 1$ is the index of the collector. We use the set U_i $(U_i \subseteq U)$ to denote the set of indices of neighbors of meter m_i.

The *Unique Neighbor Constraint* can be formalized as follows:

$$\sum_{j \in U_i} w_{ij}^t = 1,\ 1 \leq i \leq \gamma, 1 \leq t \leq T \tag{15}$$

$$(j \notin U_i) \Rightarrow (w_{ij}^t = 0),\ 1 \leq i \leq \gamma, 1 \leq t \leq T$$

$$0 \leq w_{ij}^t \leq 1,\ 1 \leq i \leq \gamma, 1 \leq t \leq T, 1 \leq j \leq |U_i|$$

In the above equations, w_{ij}^t is the indication variable. If $w_{ij}^t = 1$ then the jth element in U_i will be assigned to meter m_i in time window t, otherwise the jth element in U_i is not assigned to meter m_i in time window t.

The *Non-repeat Constraint* can be formalized as follows:

$$(w_{ij}^t = 1) \Rightarrow (w_{ij}^{t+1} \neq 1), \tag{16}$$

$$1 \leq i \leq \gamma, 1 \leq j \leq \gamma + 1, 1 \leq t \leq (T-1).$$

The *Capacity Constraint* can be formalized as follows:

$$\sum_{i=1}^{\gamma} w_{ij}^t \leq U_r,\ 1 \leq \text{J} \leq \gamma, 1 \leq t \leq (T-1). \tag{17}$$

We use SMT solver (such as Yices [61]) to find the satisfying neighbor assignment for these constraints. In most cases, there is more than one satisfying assignment. We denote the set of satisfying forwarding neighbors of meter m_i in time window t as U_i^t.

Random Report Size and Interval Mutation: Since we randomly mutate the report size and interval, we denote sz and int to be the set containing all the possible valid configuration values for report size and interval respectively, where sz_i^t and int_i^t is the randomly selected configuration value at time t for meter i.

To guarantee the maximum randomization of choices for report sizes and intervals at any time period, we require that the range size (number of values in a set) for report sizes and intervals should be greater than some threshold. This can be formalized as (*Size Constraint*):

$$|sz_i| \geq \theta \tag{18}$$

$$|int_i| \geq \varepsilon$$

where θ and ε are calculated based on the collision probability in the next t time instances, which can be thresholded by the AMI provider.

To guarantee a certain delay between consecutive reports to avoid flooding, we require that all intervals should be greater than a minimum value (*Flooding Constraint*):

$$\forall int_i^t \in int > sz_i/b_i \tag{19}$$

where b_i and sz_i are the bandwidth channel and report size for meter i.

To avoid local meter memory overflow, a report must be pushed in less than the certain time interval in which memory is not filled by the sampling for report size (*Meter Memory Overfull Constraint*):

$$\forall int_i^t \in int < mem_i/sr_i \tag{20}$$

where mem_i and sr_i are the memory and sampling rate for meter i.

To accurately measure the energy usage, there is a requirement on a minimum sampling rate for meters. This minimum sampling rate differs for different meters, however, a de facto minimum standard value is 128 samples/sec. Since report size depends on the sampling rate, we require that the sampling rate should not fill the buffer in the given reporting interval period. This can be formalized as (*Maximum Sampling Rate Constraint*):

$$\forall sr_i^t < mem_i/int_i \tag{21}$$

To guarantee unpredictability among k consecutive time periods, we require that the assigned report size and interval should not be the same as that in the previous

k time periods. This can be formalized as (*Non-repeat Constraint*):

$$sz_i^t \neq sz_i^{t-j},\ 1 \leq j \leq k \tag{22}$$

$$int_i^t \neq int_i^{t-j},\ 1 \leq j \leq k$$

To avoid the denial of service at the smart collector, we define aggregate constraints. If all the meters associated with a smart collector pick the minimum reporting interval and maximum reporting size (worst case), it should be less than the memory and reporting interval of smart collector to headend.

$$\frac{\sum_{i=1}^{\gamma} sz_i^{max}}{\forall_{i=1}^{\gamma} int_i^{min}} < \frac{mem_{sc}}{int_{sc}} \tag{23}$$

where mem_{sc} and int_{sc} are the memory size and reporting interval of smart collector to headend. However, this is a worst case scenario and the constraint can be relaxed in practical:

$$\sum_{i=1}^{\gamma} \frac{\sum sz_i}{l} < mem_{sc} \tag{24}$$

It calculates the expected report size for each meter and adds it for all the meters to get the aggregate report size.

2.6.3.1 Mutation Algorithm

To make the behavior deterministic for the smart collector, a pre-established hash function H can be used. In order to make the behavior non-deterministic for the attacker and deterministic for smart collector, we use the pre-shared key k_i between smart meter and smart collector along with the time stamp t and current values of rate and interval as the input to the hash function.

Algorithm 2: Mutable configuration

```
Data: Interval range int and report size range sz
Pick a size from range sz /* Equation 25 */
Pick an interval value from int /* Equation 26 */
while timeCounter < int do
    if sample < size then
        KeepSampling();
    end
end
Select relay neighbor from U /* Equation 27 */
SendReport();
Reset timeCounter;
Repeat Process;
```

$$sz(t+1) = H(k_i, t, sz_i^t) mod\ l + 1 \quad (25)$$

$$int(t+1) = H(k_i, t, int_i^t) mod\ j + 1 \quad (26)$$

$$rel(t+1) = H(k_i, t, U_i) mod\ |U_i| + 1 \quad (27)$$

where sz_i^t and int_i^t denotes the current value of report size and interval, respectively, for meter i at time t. We use the modulus function on the result of the hash function to generate a random configuration value within the allowed range. These allowed ranges, i.e., minimum and maximum configuration parameter values, are within the operational constraints of smart grids network based on its architecture, topology, and design, thus not affecting its functionality. A high level working of the algorithm is shown in Algorithm 2. The algorithm takes ranges of interval and size parameters as input. Using the hash function it then selects a value from the ranges for size and interval parameters. It keeps sampling the report until the report size is less than the selected report size parameter value or time counter is less than the the selected report interval parameter value. The algorithm then selects a neighboring meter as relay using the hash function to send the report. It resets the time counter

and repeats the process for next report.

2.7 Experimentation & Evaluation

Before discussing the experimentation and evaluation, we first discuss the robustness against evasion and mimicry.

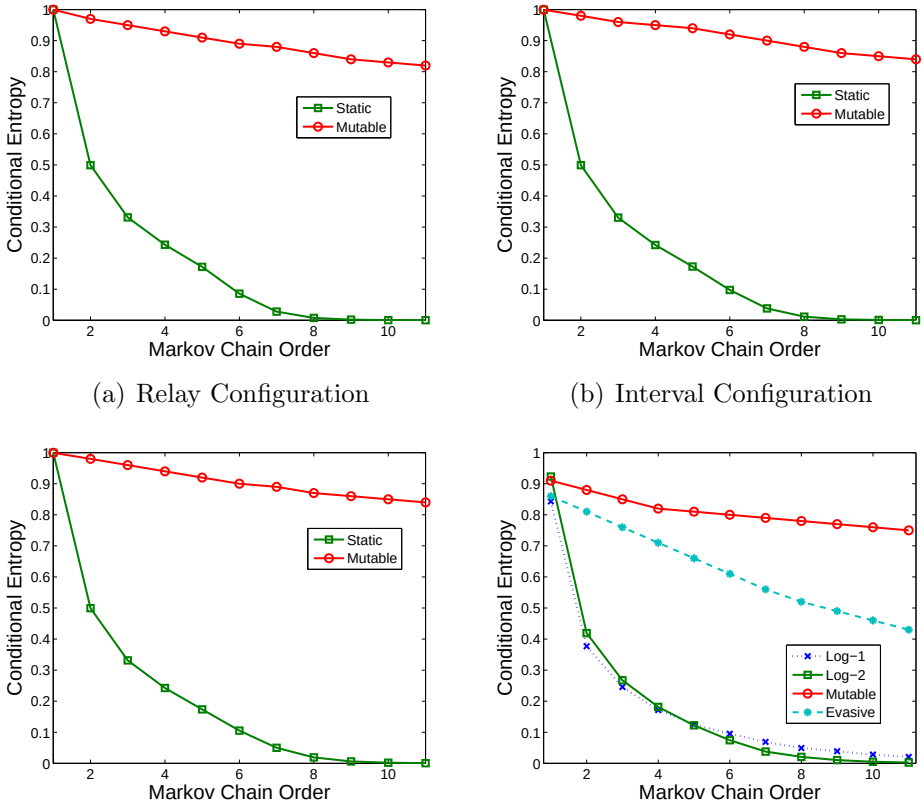
2.7.1 Robustness Against Evasion and Mimicry Attacks

In this section we discuss the effectiveness of randomization module against evasion and mimicry attacks. Since evasion and mimicry attacks leverage the known behavior of the network, they tend to stay below the radar to go undetected. Due to the homogenous and deterministic nature of the network, it is not difficult for an attacker to learn the behavior. Therefore, we randomize configuration parameters, which makes it difficult for the attacker to guess its value. We randomize the reporting interval, size, and relay configuration parameters. These parameters are mutated every time a report is sent. Since it uses the pre-shared key for randomization, the smart collector does the same computation in order to verify if the parameters values are as expected or not. Behavior compliance of these configurations is checked using the properties defined in the same section.

Robustness of the presented approach against evasion depends on two factors: 1) the probability threshold used in the verification properties, and 2) the configuration parameter randomization. Since the configuration parameters, such as reporting interval, is verified by a property that is thresholded, in case of a low threshold evasion is possible to a certain extent because significant deviation can be allowed. For example, if the property threshold is 0.9, it means that 90% of the reports should

be in the correct time and 10% can be deviated to accommodate any unexpected network behavior as learnt from the log training. Thus, attackers can leverage this 10% to launch mimicry and evasion attacks. If we assume that the key is not known to attackers, an attacker has to accurately guess the configuration parameters from a range of values such that success ratio will be above the property threshold in order to evade detection. However, this accuracy (evasion) is highly dependent on the parameter randomization range and how many parameters are randomized. For example, if the range defined as 10 minutes, for reporting interval parameter, in discrete intervals of one second each, then the probability that the attacker can guess the correct time interval (attack probability) is as low as 0.0017. Furthermore, the probability of guessing all the parameters, at a given time, would be much less as it is a product of probability of guessing each parameter accurately, which makes it much harder for the attacker to evade by accurately guessing. In order to evade the detection, the attack probability must be greater than or equal to the property threshold. However, this case is extremely unlikely unless randomization range is poorly selected and/or the property threshold has been immaturely determined based on improper training data set (e.g., unclean or very few samples).

To analyze the normal behavior of the AMI, we use utility provider logs. Mutable log was collected by applying mutable configuration in a smart grid test lab [62]. Multiple smart meters were connected to the smart collector. Network Management System (NMS) was used for configuration management. Smart collector uses a proprietary communication protocol built on top of $C12.22$ protocol. To apply mutable configurations on a smart meter, script was written in C language using the API



(a) Relay Configuration

(b) Interval Configuration

(c) Size Configuration

(d) Combined

Figure 7: Conditional entropy for static and mutable configuration

of NMS. However, the developed mutation methodology should be embedded in the smart meters.

The goal of the devised methodology is to increase the randomness of smart meter's behavior using mutation to make the prediction harder for the attacker. Randomness of a distribution can be characterized using entropy, i.e., higher randomness yield to higher entropy value. To evaluate the approach we calculate the conditional entropy over a different markov chain order for normal logs (utility provider) and mutable logs (testbed).

AMI exhibits a predictable behavior as shown in Figure 6. We apply the same predictor on the mutable logs (testlab). The prediction accuracy was below 10% in

the mutable configuration environment. Therefore, an attacker cannot predict with high accuracy about the next communication instance and its properties like size, interval, and relay. We also calculate the conditional entropy (Equation 3) over different markov chain orders for the static and mutable configuration. It can be clearly observed from Figure 7 that mutable configuration has higher conditional entropy than static configuration. Therefore, it can be stated that the mutable configuration introduces sporadic changes in the AMI behavior as opposed to the static configuration, thus making it more random and hard to predict for the attacker. Figures 7(a), (b), and (c) show the conditional entropy for relay, report interval, and report size, respectively. Figure 7(d) shows the conditional entropy for the entire tuple, i.e., conjunction of the three parameters. 'Log-1' and 'Log-2' denote the real-world logs using static configuration. A steep decaying trend was observed for both the logs, which eventually reveals the predictable nature of the behavior given few previous observations. On the other hand, 'Mutable' log did not show a steep decaying trend. Therefore, it is intuitive that mutation is harder to predict to launch an evasive attack.

To launch an evasive mimicry attack against mutated configuration, we use the predictor to generate traffic similar to mutable configurations in a testbed, i.e., it learns and predicts the future states of mutable logs. We connected an attack machine with the switch between the smart meters and collector. Predicted (mimicry) and actual mutable configuration logs were merged and collected at the smart collector. Merged (predicted and mutation) activities are labeled as 'Evasive' in Figure 7(d). Since prediction accuracy was low, 'Evasive' showed a much steeper curve as compared to the 'Mutable' configuration. This change in behavior ('Mutable' and

'Evasive') helps in differentiating between mutable configuration and prediction for mimicry/evasive attacks. Therefore, it can be concluded that the behavior is hard to learn for an attacker to predict since prediction probability is low.

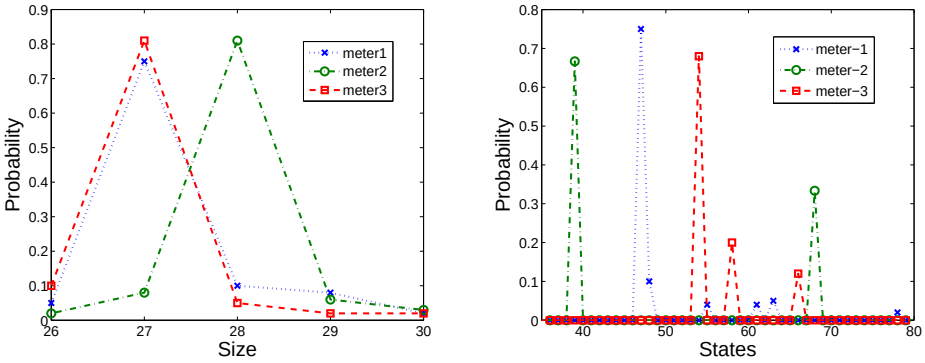
2.7.2 Accuracy Evaluation

The basic premise of our work is to develop a model using the logs of smart collectors and verify the configuration properties written in LTL. If a property is verified against the model with probability $p > d$, where d is set as a threshold, we say that it is normal; otherwise, it is anomalous. Threshold d was learnt separately for each property by noting the property verification probability using the benign logs model (without attack).

Logs were collected from the smart collectors which were connected to the multiple meters. Before discussing the accuracy evaluation, we first show the basic behavior of the three randomly selected meters for two scenarios, i.e., when the mutable configurations are known and not known to the system. It gives us interesting insight about the temporally deterministic behavior. Then we analyze that how well the model represents the system behavior. Please note that this temporal behavioral analysis was done without the attack traffic. The model checking was conducted using a probabilistic symbolic model checker tool PRISM [60].

2.7.2.1 Temporal Behavior of the Model

Since the model preserves the temporal behavior, the properties can be seen as the conditional probability, i.e., $P(A|B)$, where state B has already been observed (given). Therefore, it can be stated as 'what is the probability of seeing state A given



(a) Response for Reading Request

(b) Response for Load Management Request

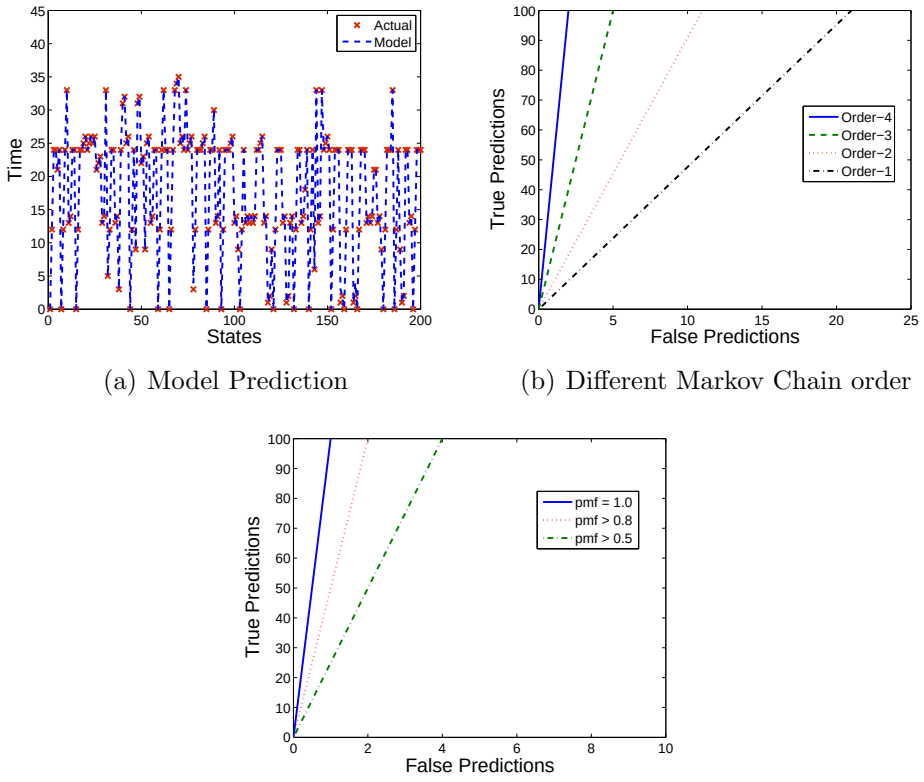
Figure 8: Response probabilities when a request was generated for reading and load management

state B?'. It can be easily written in the LTL using the 'next' or 'eventually' operator for the state A by filtering all the states except the current/given state, i.e., B as discussed in an earlier section.

Figure 8 shows the conditional probabilities given that the request was made for a particular meter for the usage reading and load management as shown in Figures 8(a) and 8(b), respectively. We provide the results for three meters only. However, similar results were observed for the other meters as well. Figure 8(a) shows the probability of response size, given that the system was currently in the reading request state, i.e., a usage/reading request was sent to a meter. It can be seen that all the three meters generated responses within the range of 26 to 30KB with different probabilities. Two meters generated responses of size 27KB with higher probability and the third meter generated response of size 28KB. These responses belong to different states in the model since size was used as a variable in the tuple (σ) used for the state definition. However, none of the meters generated a response size more than 30 or less than

26KB, which was set as a variance boundary. The reading response size depends on the sampling rate of the meter. Since the response size of the reading request was within the range, the property 'whenever a reading request is generated, the next observed state is reading response with the size modeled' (combining Equations 6 and 8) was verified with the probability 1. Here the probability 1 can be realized as the sum of all the probabilities of the states having sizes within the range of 26 to 30KB. Please note that the system can be in one state at a given time, i.e., either request sent to meter 1 or meter 2. These conditional probabilities were calculated separately for the next state given that the system was in the request state.

Similarly, Figure 8(b) shows the next state transition probabilities given that the system was in a load management request state. It can be seen that all the three meters probabilistically responded to the request. Meter 2 (green line with circular marker) showed the two transition probabilities: response of size 9KB with probability close to 0.65 and size 6KB with probability close to 0.35. Similarly, meter-1 responded with 6 different sizes. All of these sizes were within the range of 5 to 10KB. Lastly, meter-3 responded with the three different sizes where one size was most likely and the other two were less likely. It can be observed that whenever the system was in a load management request state, the response state was observed next with the probability of 1. However, response also had multiple states depending on the size of the response that was used as a variable in defining the state s consisting of multiple σ. Therefore, it can be concluded that the model exhibits a temporally deterministic behavior for the system under consideration. It is clear that the temporal probabilities for the properties can be learnt from the model built using the benign logs. Now we analyze



(a) Model Prediction

(b) Different Markov Chain order

(c) Varying pmf on fourth order model

Figure 9: Prediction accuracy for different markov chain order and PMF for fourth order model

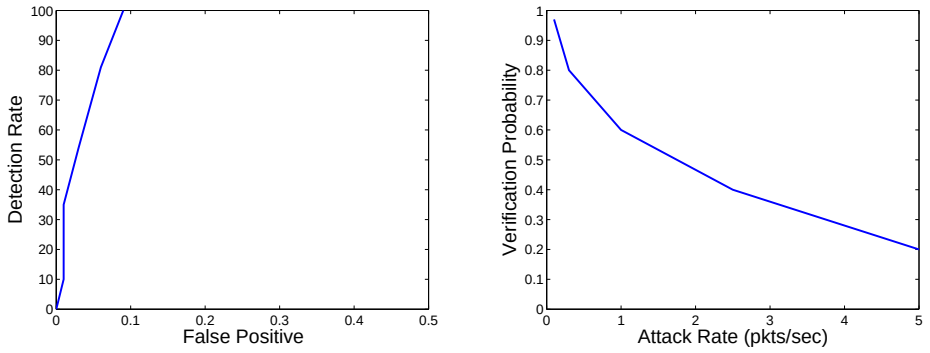
whether the model truly represents the system or, in other words, how well the model represents the logs.

2.7.2.2 Model Accuracy

In order to determine whether the model actually represents the logs or not, we conduct a model accuracy experiment as shown in Figure 9. It does not include the attack traffic or property verification. We divide the benign logs into two halves. The first half was used to build a model using Algorithm 1. The second half was used as the test dataset and the model was applied to it for the prediction. For each step, the current state was learned and based on the current state, the next state was predicted

using the model, i.e., states having higher probabilities in probability mass function (pmf) were predicted. Figure 9(a) shows the prediction using the fourth order model. The red cross marker denotes the actual states observed in the test log. However, blue dashed line represents the prediction of the model. It can be observed that the model predicts the future states with high accuracy though a few false predictions (less than 1%) were encountered as well. These false predictions were observed as a result of the unseen behavior since the benign log was divided into two halves. As a result, few lower probability states were not observed after a certain state s (i.e., tuple history of order four) present in the first half, thus yielding to false prediction. To check the model confidence, we did the prediction using one hour learning to one week learning. In all the cases, the false predictions observed were below 1%.

We used different criteria and show the results in Figures 9(b) and (c). We count the total number of predictions and classify whether they were true or not. In Figure 9(b), we used different markov chain orders, i.e., the current state was defined using one tuple or multiple tuple history. It can be observed in Figure 9(b) that the fourth order markov chain provides the best prediction accuracy as compared to the lower order markov chains. In another experiment, we change the pmf bound used for the prediction. For example, pmf of 0.5 means a minimum number of next transition states having probabilities sum of 0.5. If there are three possible next states based on the current state having probabilities 0.6, 0.2, and 0.2, only states having 0.6 probability will be selected as the 'predicted' state since it has probability greater than or equal to 0.5. However, if none of the states has a probability higher than 0.5, a minimum number of multiple states will be selected whose sum reaches 0.5. Figure



(a) Detection Accuracy (b) Probability vs Attack rate

Figure 10: Detection accuracy and verification probability vs attack rate

9(c) shows the prediction accuracy for the different pmf. It can be seen that pmf of 1, i.e., based on the current state all the next possible states in the model are predicted, provides the best accuracy with a very low false prediction rate. It can be intuitively argued that this false prediction rate will be the bound for the false positive for the attack detection accuracy, since this shows how well the model represents the data. Therefore, it can be concluded that the fourth order markov chain can accurately model the underlying network with minimal loss, hence we use the fourth order for the model checking.

2.7.2.3 Detection Accuracy

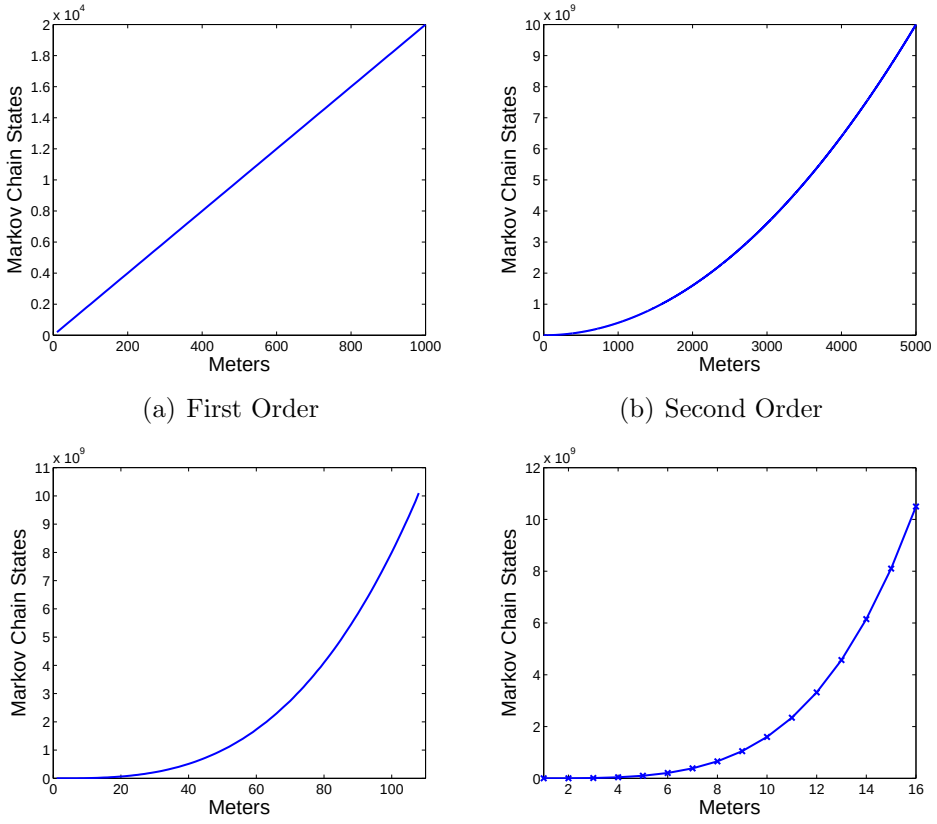
Mixed data was used to calculate the detection accuracy that had both attack and benign logs. Since the attack logs were generated separately in a controlled environment, the time stamps of the attack logs were adjusted by a fixed constant to have the same time window as of normal/benign logs. Allowed communication type ty values were used for the attack logs since other communication types do not exist in the model and can be detected easily. For attack detection, model learning was done in a

continuous online learning fashion using a sliding window approach. The size of the sliding window was kept to one hour and the sliding window interval was set to one minute. The model was learnt separately for each smart collector in the dataset and detection results were averaged. Figure 10(a) shows the detection accuracy achieved by the presented model. It can be seen that high detection rate of more than 95% was achieved with a negligible false alarm rate of approximately 0.1%. Logs were collected from smart collectors for approximately 2000 meters. Average false alarm rate varies from 0.35 to 0.50 false alarms per meter per week, depending upon the threshold used for detection rates $> 70\%$. Please note that these rates are on the entire dataset collected. The utility provider we worked with has hundreds of thousands of smart meters in a state. This might increase to millions of meters in a state for large-scale providers. These could be managed by one or multiple headend systems. The Receiver Operating Characteristics (ROC) curve is generated by changing the verification probability (threshold) of configuration-based LTL properties that were verified against the model built using the mixed data logs. Figure 10(b) shows the effect of verification probability for multiple degree of DoS attacks. It is intuitive that the attack activity does not follow the state transitions as allowed in the temporal properties. It can be observed that the higher attack rate is detectable even with a loose verification probability threshold. The underlying reason is that the higher attack rate generates a larger volume of log entries, which causes the verification properties to not to satisfy even with a lower or loose verification probability threshold. On the other hand, a lower attack rate generates less malicious traffic reflecting in logs, thus a strict or higher verification probability threshold is required to de-

tect such attacks. Please note that the attack rates that are detected by loose/lower verification probability threshold will always be detected by strict/higher verification probability threshold. However, the opposite may not be true.

Complexity of the probabilistic model checker (PRISM) for a markov chain model and LTL property verification is doubly exponential in the size of an LTL formula and polynomial in the size of state space [63]. Algorithm 1 was implemented in Java on a dual core machine to learn the model from the logs. The run-time complexity is in hundreds of milliseconds ($\approx$ 300ms) and the memory size of the model for each collector was a few kilobytes ($\approx$ 20KB). The complexity was measured using an HPROF [64] tool. The model was then verified against the LTL properties using PRISM. The run-time complexity of the properties verification is approximately 1.5 secs. We discuss the complexity of the mutation algorithm in a later section.

The presented model can be used in two fashions: 1) use a centralized approach and build a single giant model for the entire AMI. The model defines smart collector in either source or destination, thus the model can reduce the state space to possible states only for that particular smart collector. In this case, offline model checking can be done in the headend by pulling the logs, 2) if the giant state machine exceeds the scalability limit of the model checker, each smart collector can be modeled separately. In the case of modeling each smart collector separately, the model checking can be done online or offline, depending on the computational power available. Therefore, the developed approach is flexible.



(a) First Order

(b) Second Order

(c) Third Order

(d) Fourth Order

Figure 11: Meters vs number of states

2.7.3 Scalability

PRISM is shown to be scalable up to 10^{10} states [60]. We also investigate the maximum number of meters that can be handled with various markov chain orders as shown in Figure 11. We show this for multiple markov chain orders, i.e., 1 to 4 in Figure 11. For the first order markov chain, it can be observed from Figure 11(a) that the number of states increase linearly with the number of meters. In this case, it can accommodate up to $25,000$ meters per collector, although we show it up to $1,000$ meters in the graph in order to show the linear trend.

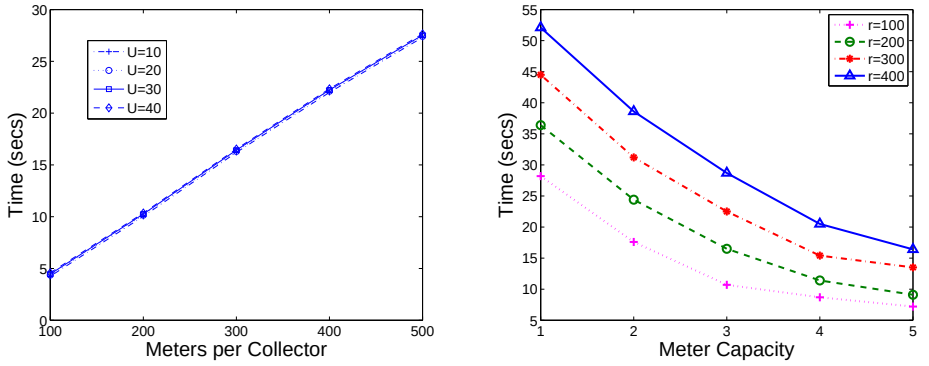
Similarly, the second order markov chain can handle up to $5,000$ meters as shown

in Figure 11(b). It can be observed that the trend tends to be exponential. Scalability for third and fourth order markov chains is shown in Figure 11(c) and (d), respectively. It can be clearly observed that both the markovian orders increase the state space exponentially with the increase in number of meters. More specifically, the third and fourth order markov chains can work up to 108 and 16 meters, respectively.

Since many vendors (including our smart grids provider collaborator) use $8 - 16$ smart meters per collector, we found that using the fourth order markov chain is the optimal choice to maximize accuracy. We also found that many major vendors similarly use limited number of meters per collector due to limited bandwidth or coverage, in power line communication and WiFi, respectively [65], [66]. However, vendors might support different models of the AMI technologies that offer different capacities of handling meters. For example, some smart grids providers/vendors plan to use other technologies that can support hundreds of meters per collector [66].

For this reason, we discuss two approaches that offer much more scalable solutions that sacrifice the accuracy reasonably. First, a third order markov chain can be used since it also incurs a low conditional entropy as shown in Figure 3(b). Moreover, it shows acceptable prediction accuracy, with slightly more false predictions as compared to fourth order as shown in Figure 9(b). Second, we alternatively extend the fourth order markov chain approach to handle more meters by state compression. Moreover, logs can also be divided based on the group of meters (16 in each) to conduct the analysis.

It can be seen from Figure 4(d) that very few (only tens of) states are highly likely (high probability) out of $1,000$. Thus, eventually less than 10% of states are seen more



(a) Randomization Computation vs No. of Meters

(b) Randomization Computation vs Meter Capacity

Figure 12: SMT formalization time for mutation algorithm

frequently than others. Therefore, once the graph is made using Algorithm 1, we run a compression algorithm to remove the nodes having lower probability. We run the detection accuracy using the compressed model and note that it incurs an increase in the false positive rate, i.e., from 0.1% to 0.2% while maintaining the same detection accuracy. However, the compressed state model can easily scale to more than $1,000$ meters per collector. Choosing the appropriate alternative approach depends on the AMI network technology under consideration.

We also evaluate the computational cost and scalability of the mutation algorithm. The Figure 12(a) shows the SMT solving time for the constraints with respect to different numbers of meters per collector. U is the number of average neighbors for every single meter. Solving time is almost linear with the number of meters per collector, and the number of neighbors has only negligible effect. The Figure 12(b) shows the SMT solving time for the combination of constraints such as unique neighbor assignment, non-repeat, and meter capacity, with respect to meter capacity. r is the number of meters for every collector. We can see that the solving time

increases when the number of meters per collector increases, and decreases when meter capacity increases. This is because when the meter capacity increases, it is easier for the solver to find satisfiable solutions.

The developed mutation scheme is shown for the three configuration parameters, however, it can be extended to more. Since the configuration mutation randomly picks a configuration value from a given range, the larger the size of the range, it is easier to satisfy the mutation constraints. It can be seen in Figure 12(b) that as the meter capacity to handle more relay request increases, the computational cost to satisfy constraint decreases. Therefore, the approach is scalable to hundreds of meters per collector. However, in practice, tens of meters per collector are used. In our case we used 8 neighboring meters as relay, 80 different sampling rates ranging from 128 samples per second to 208 samples per second, and 105 different reporting intervals ranging from 15 seconds to 120 seconds.

2.7.4 Limitations

In this section we discuss the limitations of the presented approach

- The original fourth order markov chain based approach only scales up to 16 meters per collector. In actuality, a smart collector may have a higher number of meters associated with it. To combat this, we discussed an alternative in the scalability section that supports up to $1,000$ meters per collector while incurring a slight increase in false alarm rate. Moreover, the original approach can also be used by dividing a smart collector's log into multiple chunks based on the groups of smart meters. Each chunk may have up to 16 meters grouped into it.

This can easily be done by filtering the log based on the meter id.

- Attacks that do not involve communication with a smart collector will not be detected. However, we believe that the effect of such attacks would be limited to a particular area, i.e., not large scale.

- Evasion robustness relies on the key used for mutation. We assume that the key is secured and attacker can not access it. However, if an attacker compromises the key and knows the mutation algorithm along with parameter ranges, the attacker can evade the detection algorithm by generating similar traffic.

- The technique provides robustness against evasion, however, an attacker that has full access to and complete knowledge of the AMI may still be able to evade it.

2.8 Conclusion

In this work, we present a fourth order markov chain based model for intrusion detection since it incurs lower conditional entropy and higher prediction accuracy. Moreover, it fits the state space requirement of the AMI network under consideration. The novelty of the model lies in the configuration-based stochastic modeling of the AMI using the logs collected at smart collector and device configuration. Moreover, considering the challenges of an AMI network, the approach is practical since it does not require high computation power and memory in the field. However, it can also work offline in the substation by pulling the smart collector logs from the area. Therefore, it is flexible and incurs low cost. However, if smart collectors provide

enough computation and memory, it can be deployed in parallel. It also does not need to be trained, unlike traditional intrusion detectors. The model provides acceptable detection accuracy of more than 95% with a false alarm rate close to 0.1%. The model can be easily extended and customized based on the AMI network under consideration using other information in the logs and configurations. Therefore, the scope is not limited to what is presented here.

CHAPTER 3: MULTI-TIER INTRUSION DETECTION FOR AUTOMATIC GENERATION CONTROL

Smart grids have been replacing the legacy power infrastructure as they provide efficient energy management by utilizing bi-directional communications. Bi-directional communications enable the smart grid to take different sensor measurements using cyber infrastructure in order to control the power generation, transmission, and distribution effectively and in real time. The bi-directional communications are associated with the supervisory control and data acquisition system (SCADA). An important task of SCADA is automatic generation control, which is responsible for adjusting the power generation according to the load in the area.

Several threats have been targeted towards the SCADA system due to its dependency on the cyber infrastructure. According to a recent Bipartisan Policy Center report, a Washington D.C. think tank, more than 150 cyber attacks targeted the energy sector in 2013 [67]. There can be several entry points for an attacker to enter the SCADA system and/or control center, including malware attachments in email, malware on a storage device, and WiFi enabled systems in SCADA and/or control center. Moreover, SCADA systems and control centers are connected to the corporate offices using virtual private networks, therefore, anybody having access to the corporate office can access the system. Attacks can be launched by two types of attackers: naive and experienced/knowledged. Naive attackers lack the working knowledge of

the smart grid system. On the contrary, experienced attackers may manipulate the generation control measurements such that it still satisfies the smart grid environment and look benign/normal. Although bad data detection algorithms provide some security to identify data integrity attacks, recent studies have shown that these algorithms can be bypassed by experienced attackers [68]. Moreover, attacks on measurements sent from state estimation, i.e., the initial infection vector or attack entry point is after the state estimation component in the AGC loop (shown in Figure 13), cannot be detected by these detection algorithms [69].

To this end, we developed a data-driven multi-tier intrusion detection approach, which is published in a recent study [70]. The first tier is an online short-term adaptive predictor for both the load and Area Control Error (ACE), which are system variables in AGC. Load measurements are taken by the field sensors. However, ACE is calculated, in AGC, using the frequency and tie-line flow measurements. Generation control takes these measurements every few seconds. The basic hypothesis is that both the load and ACE have different behavior at different times of the day, therefore, at short intervals they exhibit a certain level of temporal dependence, which can be used to predict their future behavior. The developed predictor has the ability to adapt to the change in behavior of the variables. Since load and ACE forms the basis of calculation of set points and lowering/raising the generation, respectively, we use a data-driven multi-tier approach to predict these variables. We show the prediction accuracy of the developed predictor under normal conditions in a well known and widely used two-area power system model. Since the predictor shows high accuracy under normal conditions, therefore, deviations from the prediction can be flagged as

anomalous.

Prediction is done independently and does not take into account the other AGC system variables. Therefore, we build a markov model of AGC using its system variables in the second tier of the intrusion detection system. The model incorporates the system-wide knowledge to detect anomalies. It observes the state transitions, where state is defined using multiple system variables, and calculates the individual variable probability given the system state. If the probability does not fall in the probability range learnt from the normal data, it raises an alarm. The two-tier approach provides the benefit of timeliness, using light-weight online prediction, and accuracy, by reducing the false positives offline using multiple AGC system variables. The first tier does online prediction for timeliness and the output alerts of the first tier are passed on to the second tier, which incorporates the system-wide knowledge to reduce false positives. We also extend the devised framework to an interconnected multi-AGC scenario. In this scenario, the state of one AGC is verified by incorporating the other AGC's knowledge. This helps in identifying which particular AGC is under attack and causing the system de-stabilization overall. We conduct multiple attack case studies, which include tampering with the system frequency and the load in order to mimic the integrity attacks by knowledged attackers. Tampered parameters still satisfy the generation control equations. Our approach detected all of the integrity attacks successfully.

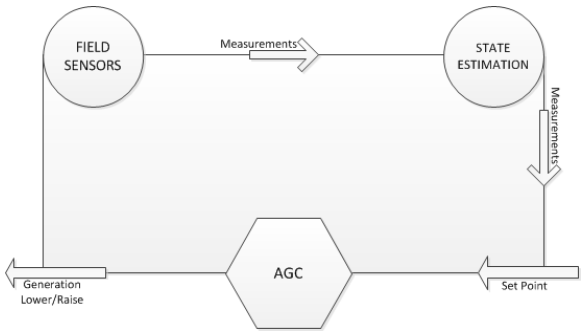


Figure 13: Automatic generation control loop

3.1 Background

AGC is another core component of the smart grid. Figure 13 shows a high level diagram of generation control. AGC takes measurements from the field that also goes to state estimation [69]. However, there are some measurements that are AGC specific and calculated based on different field measurements. The main functionality of AGC is to calculate ACE based on the tie line flow (P_{tie}) and frequency (f) of the tie line. In our case, a tie line is connected to two areas for the inter-area power flow.

$$P_{ACE}^{(t)} = (f^{(t)} - f_{ref}).\beta + (\sum_{i \in \tau} P_{itie} - P_{sch}) \tag{28}$$

where τ, β, f_{ref}, P_{itie} and P_{sch} represent all tie lines, frequency bias factor, base reference frequency, tie line flow for line i, and the scheduled tie line flow, respectively. The first part of Equation 28 considers the system's frequency deviation from standard frequency, and the second part considers the deviation of power flow in each tie line from the schedule flows. Thus, ACE is an error between the scheduled and actual values in the system, which is transmitted to each local generation control in order to adjust the generation accordingly.

Once the composite area-wide error is calculated, it is used along-with the economic dispatch (such as demand-based pricing or other) to define the generation schedule. It is done at intervals of 1 to 15 minutes. In case of continuously changing generation demand, allocation of generation must be made instantly. This task is performed through allocation control logic by means of base points and participation factor algorithms. After calculations, each control action, i.e., either raise or lower the pulse signal, is transmitted to the generation unit at remote generation station for changing the generating unit's load reference point,

$$P_{isch}^{(t)} = P_{ibase}(k) + pf_i \times \Delta P_{total}^{(t)} \tag{29}$$

$$\Delta P_{total}^{(t)} = P_{actual}^{(t)} - \sum_{i \in \xi} P_{ibase}(k) \tag{30}$$

where ξ represents all the generators, P_{ibase} is the base point set by unit commitment, pf_i is participation factor and P_{actual} is actual generation demand. These variables such as P_{ibase} are generation control specific and are input to the system as shown in Figure 13, denoted by 'set point'. The output of the overall AGC system is the raise/lower generation command. Therefore, attacks on these variables or the variables that constitute them such as load, after the state estimation cannot be detected by intrusion detectors working at state estimation or SCADA field measurements.

Generators are generally modeled by swing equation (Equation 31). The equation considers the physical rotor angle and speed as states of the system. Primary controllers, Governor, Automatic Voltage Regulator (AVR) and Power System Stabilizer (PSS) are designed to make these states in their equilibrium values by controlling

mechanical and electrical torques,

$$\Delta P_{mech} - \Delta P_{elec} = M \frac{d\Delta\omega}{dt} \tag{31}$$

where M is the angular momentum of the machine. The generator calculates the difference in change in electrical and mechanical power being generated. These balancing equations (Generator and Scheduling) can be modeled using model-based estimation approaches such as Kalman-Filter based approach [71]. However, the estimation model does not take into account the temporal dependence in the data. We build a hybrid approach in which the first tier is a light-weight online data-driven predictor that leverages the temporal dependence in the data. The second tier builds the system model by incorporating the system-wide knowledge using AGC system variables. This makes the approach practical since the online prediction module (first tier) is light weight and flags any deviation in the data. The output of the first tier is given to the second tier offline verification module, which utilizes the system-wide knowledge to verify the presence of anomaly. Thus, the developed approach benefits from temporal dependence in the data along-with the system model.

Since ACE raises/lower the generation, we change the frequency f that is used to calculate the ACE in order to mimic the attack on ACE. It can be simulated by unbalancing the Equation 31 since that is used to model the generators. To mimic the attack on load we manipulate the active and reactive power by changing R and L of the power system. Since load is used as an input to define the base point and scheduled flow subsequently (Equation 29), and tie-line flow and frequency deviation reflects in ACE, we use these variables for prediction as the first tier of our approach.

In the second tier, we model the system using system variables including scheduled flow, tie line, and frequency along-with load and ACE to determine the AGC state for anomaly verification. Probability of each system variable is verified against all the system variables to verify the anomaly presence. We further extend it to multi-AGC scenario by conditionally comparing the states of one AGC given the state of interconnected AGC in order to mimic a scenario where two different power companies share the power resources.

3.2 Related Work

Since we developed AGC parameters' prediction based intrusion detection approach, we discuss work related to prediction of AGC parameters and intrusion detection. There is no data-based prediction approach that can predict all the AGC parameters. However, there are approaches that can predict short term future load, an AGC parameter, [71, 72]. The work [71] presents a Kalman-Filter based load prediction approach. The approach does not take into account the temporal dependence in AGC data. Furthermore, it assumes that the introduced noise in AGC data is white, which may not be the case in real life. On the contrary, our approach takes into account the temporal dependence in the data and it is specifically designed to classify anomalous behavior in the data. Another approach that uses a hybrid model for adaptive load prediction is presented in [72]. It uses computationally expensive machine learning tools like a support vector machine (SVM) and self organizing maps, and requires supervised learning. Due to its complexity, it does prediction in hourly intervals. Our approach is an online and real-time light weight predictor.

Recent literature discusses attacks on AGC [68, 73–75]. The work [68] presents integrity attacks on AGC by a knowledged attacker. The parameters are manipulated in an acceptable range, thus attack is not detected by embedded data verification modules and causes imbalance in power generation. Cyber attack's impact assessment on a two-area power system is presented in [73]. The work also shows how an attacker can cause undesirable behavior under the conditions established using reachability methods by interrupting the AGC signals. The work is followed by [74] where it is assumed that the attacker has partial information about the system parameters to launch an attack. Protective measures to minimize potential risks were developed using game theoretic framework in [75]. Our work infers the attack model from [68].

Based on the work [71], an anomaly detection technique for AGC is also proposed in [76]. As discussed earlier, the work [71] does not take into account the temporal dependence in predicting the load. In later sections, we show that our developed prediction algorithm outperforms the Kalman-Filter based approach presented in [71]. Although the work [76] introduces temporal dependence by an offline module, which takes the aggregated algebraic sum of estimated and real-time ACE values, it averages out the temporal dependence for short intervals due to aggregation. On the contrary, our work utilizes the temporal dependence in a sliding window manner to predict each AGC parameter value in real time. Furthermore, our work incorporates all the system wide AGC variables (as shown later) to verify the presence of anomalies, however, the approach [76] only works with the ACE parameter. Our approach also considers single and multi AGC scenarios.

Intrusion detection techniques for smart grids have been proposed recently [13, 46,

77–79]. While these techniques address cyber security challenges in smart grids, none of these techniques are designed specifically for AGC. A distributed intrusion detection technique in a multi-layer architecture of smart grids is proposed in [18]. In [13], we presented an intrusion detection approach for advanced metering infrastructure. In [45], an anomaly detection technique for smart grids has been proposed. It focuses on the cyber security of substations and does not take into account the AGC parameters. In [80] an approach for anomaly detection in a SCADA system has been proposed. It considers the SCADA measurements that are used for state estimation in a control center. However, attacks that target control centers, such as AGC that takes state estimation as an input, will not be detected. Since AGC is the last mile to raise or lower the generation, we work with variables related to AGC.

3.3 Challenges

Smart grids are different in nature than traditional networks, therefore, the nature of attacks is also different. In smart grids, the objective of the attacker may not be just to gain unauthorized access, however, an adversary may disrupt the energy system. Power generation collects data at regular intervals, which usually vary from 1 to 15 minutes in order to adjust the generation with respect to the load observed. If an attacker gains access and injects malicious data, the control center will be mislead to a wrong estimate of the load.

While several approaches related to bad data detection have been proposed recently [81], the authors of [69] show that bad data detection approaches can still be evaded by knowledged attackers. Moreover, attacks that target the control cen-

ter after the state estimation component can go undetected by the state estimation based intrusion detection techniques. Therefore, it is possible that generation control receives malicious data injected by an knowledged adversary. To this end, there is a need for another layer of security in the generation control that can model the dependencies present in the data in order to accurately predict the short-term future load that identifies anomalous behavior. We show that load behavior changes over time and exhibits a certain level of temporal dependence at short intervals, which can be exploited to predict the load. We develop an adaptive technique that has the ability to learn and adapt to the change in behavior of the underlying load. Based on the learning, the technique predicts the short term future load, which acts as a threshold in order to find the anomalous behavior in the load. Moreover, ACE is calculated using the frequency and net tie line flow of the area. Since tie line flow is also related to the load behavior in the area, ACE also exhibits a temporal dependence at short intervals. Prediction of both system variables (input and output) helps in identifying if the system is operating as expected or not. If not, intrusion verification is done by incorporating the system-wide knowledge to verify the presence of anomalies.

3.4 Contribution and Approach Overview

The main contribution is to build an anomaly detector for AGC. The approach takes into account system wide variables and also any collaborating AGCs in a multi-AGC scenario. To build an anomaly detector for the generation control, we adaptively predict the short-term future load and ACE behavior for anomaly identification by exploiting temporal dependence. To verify the presence of an anomaly, we incorporate

the system-wide knowledge using markov models. This detector will serve as another layer of security to the existing defense mechanisms, such as bad data detection algorithms, to defend against evasion attacks by experienced attackers.

3.5 Attack Model

Control centers are generally connected to two networks - corporate and control network. Corporate and control networks are separated using added layers of security generally through a firewall. Corporate networks are also connected to the internet through a security layer using firewalls. Therefore, for an attacker to reach a control network, he/she first has to reach corporate network by bypassing the first layer of security. Then the attacker has to gain access to the control network in order to manipulate AGC. Another entry point for an attacker could be a WiFi network in the control center. The intrusions addressed by the work are manipulation of AGC parameters for power imbalance as shown in [68]. We introduce two different types of integrity attacks for two different parameters related to AGC and generation allocation logic. All the parameters used in this work affect the control logic unit. We reiterate that the attack point is the control center, after the state estimation in the control flow, as shown in Figure 13. These attacks will not be detected by SCADA specific intrusion detection systems focusing on SCADA data or state estimation.

3.5.1 Attack on Power Load

First, we change the load of the system by injecting a false load, which satisfies the generation control equations. By manipulating R and L of the loads, we can control the active and reactive power loads. The false load directly impacts the generation

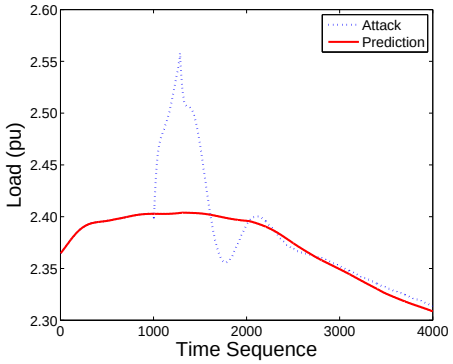


Figure 14: Attack on load

allocation control to generate the power as per the new load observed, since base point is calculated on the load. However, it indirectly affects the AGC parameters as well, i.e., P_{tie} and f. Three cases have been studied in this attack model:

- Case1: $0.15p.u.$ change in total system's load
- Case2: $0.1p.u.$ change in area 1's load
- Case3: $0.1p.u.$ change in area 2's load

As it can be seen, in Figure 14, the load value is increased by $0.15p.u.$ from the normal condition at time $16hr$, which has the max load and minimum frequency. $Psch.tie$ is set to $0.29p.u.$ derived from economic dispatch. Reference for f is $1p.u.$ Detailed normal and attack scenario measurements are provided in Table 2. Moreover, to mimic a sophisticated attacker who has consistent access to the data, we introduce instances of stealthy attacks such that $0.1p.u.$ change commutatively in 100 time instances.

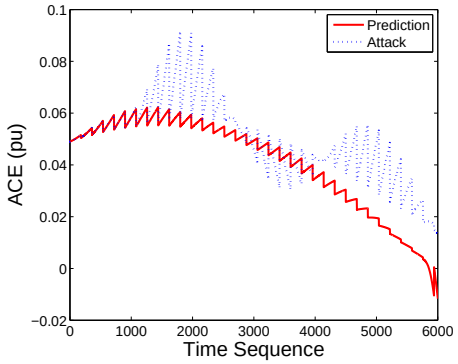


Figure 15: Attack on ACE parameter, frequency

Table 2: Measurements under normal and attack scenarios

Parameter	Normal	Attack Case1	Attack Case2	Attack Case3
P_{load1}	0.844	0.919	0.944	0.844
P_{load2}	1.558	1.633	1.558	1.658
f	0.9902	0.9881	0.9865	0.9867
P_{tie}	0.3578	0.3513	0.3413	0.3849
ACE area 1	0.0492	0.0387	0.0257	0.0696

3.5.2 Attack on Area Control Error

In the second case study, we change frequency f, which affects the value of ACE. This task is simulated by unbalancing the swing equation of generator 3. A sudden $0.2p.u.$ mechanical power change is simulated at 16 hr, which directly causes a frequency deviation from its reference (Equation 31). This gives a false estimate of the frequency f of generator 3 and total system, thus resulting in the disruption of ACE, as shown in Figure 15. It can be observed that the error has been increased, thus destabilizing the system since AGC adjusts according to the error for power generation. We also introduce the stealthy attack instances as described earlier for load. Please note that all the values changed still satisfies the AGC equation.

Since the attack targets measurements and two layers are used in a sequential manner for alert generation, i.e., alerts from the first layer are passed onto second

layer for validation, below we show that the alerts generated by offline validation layer is a subset of alerts generated by the online prediction layer. This essentially means that the attacks that are not detected by the online prediction layer cannot be detected by the offline verification layer.

Lemma 0.1. *Let A and B are detection alert sets of online and offline layers respectively. For all alerts $x \in B$ also implies $x \in A$.*

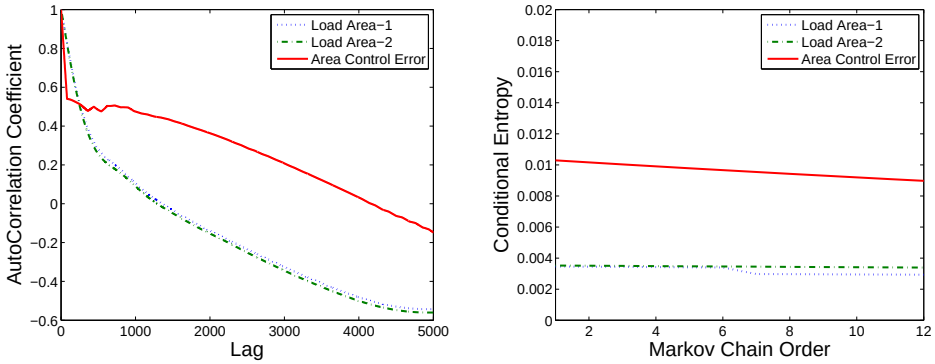
Proof. AGC operates on system variables P_{tie}, f, l, ACE, and P_{sch}. The online layer uses l and ACE for generating detection alerts that are passed onto offline layer for verification. The offline layer uses all the AGC variables P_{tie}, f, l, ACE, and P_{sch} for anomaly verification. Since P_{tie}, f and P_{sch} constitutes ACE, there exists no attack that affects P_{tie}, f, and P_{sch} but does not affect ACE. Therefore, there is no attack that is detectable by offline layer and can not be detected by online layer hence for all alerts $x \in B$ implies $x \in A$. □

3.6 Modeling of Power Load and Area Control Error

In this section we discuss the analysis of load and ACE data followed by the adaptive prediction algorithm.

3.6.1 Analysis of Load and Area Control Error

To reveal the load and ACE behavior, we conduct statistical analysis on the 24-hour data. The data was generated using the two-area Kundur power system model [82], which is well known and widely used in the power community [73–75]. Sample load and ACE data is shown in Table 3, where t_x represents the value realized at time



(a) AutoCorrelation (b) Conditional Entropy

Figure 16: AutoCorrelation and conditional entropy of the load for both areas

Table 3: Sample load and ACE values

	t_0	t_1	t_2	t_3	t_4
ACE	0.059	0.069	0.080	0.091	0.103
Load	0.950	0.952	0.955	0.957	0.959

instance x. It can be intuitively argued that, as long as the load and ACE values are produced by the benign events, the values observed should exhibit a certain level of temporal dependence. In case of an anomalous behavior, perturbations in this dependence can be flagged as anomalies. Therefore, the level of temporal dependence can serve as an important metric for the modeling.

Autocorrelation measures the on-average temporal dependence between the random variables in a stochastic process at different points in time. For a given lag k, data values window shift, the autocorrelation function of a stochastic process X_n (where n is the time index) is defined as:

$$\rho[k] = \frac{E\{X_0 X_k\} - E\{X_0\}E\{X_k\}}{\sigma_{X_0}\sigma_{X_k}}, \tag{32}$$

where $E\{.\}$ represents the expectation operation and σ_{X_k} is the standard deviation of the random variable at time lag k. In our case, load and ACE values are the realization of the random variable X. For example, X_0 represents the ACE or load

value at time 0 as shown in Table 3. The value of the autocorrelation function lies in the range $[-1, 1]$, where $\rho[k] = 1$ means perfect correlation at lag k (which is obviously true for $k = 0$) and $\rho[k] = 0$ means no correlation at lag k.

Figure 16(a) shows the autocorrelation function plotted against the load and ACE values at different lags. For both the load and ACE, a certain level of temporal dependence can be easily observed at small lags. This correlation decays in time and eventually drops down to a negligible value. Temporal dependence is present because load and ACE values are similar in a short time interval. It decreases when the behavior of the load and ACE is changed and it increases when similar values are observed.

It is well-known that a decaying temporal dependence structure can be accurately modeled using markov chains [57]. Therefore, to identify the markov chain order, we conduct analysis on different markov chain orders. We define a markov chain based stochastic model as follows: Let the load and ACE value at discrete time instance n represents the realization of a random variable derived from a stochastic process X_n. This process is a markov chain if it satisfies the markov property, i.e., the probability of choosing a next state is only dependent on the current state.

Each unique realization of X_n for ACE and load is assigned to a unique bin among multiple non-overlapping bins. Therefore, the number of bins will be dependent on the number of unique values. Each bin then represents a state of the markov chain, while the set of all bin indices ψ is its state space. Based on this state representation, we can define a 1-st order markov chain, $X_n^{(1)}$, in which each bin represents a state of the random process. Probability of each state i can be calculated by counting the

number of times state i occurred and dividing it by the total occurrences of all the states in the markov chain model X_n. Similarly, an l-th order markov chain, $X_n^{(l)}$, can be defined in which each state is an l-tuple $\langle i_0, i_1, \ldots, i_{l-1} \rangle$ representing the values taken by the random process in the last l time instances. In this case the occurrences of l-load values will be counted. This will increase the size of state space ψ since different combinations of l-tuple can be observed.

Conditional entropy, $H(B|A)$, of two discrete random variables A and B characterizes the information remaining in B when A is already known. If A and B are highly correlated, most of the information about B is communicated through A and $H(B|A)$ is small. On the other hand, if A and B are quite different then $H(B|A)$ assumes a high value, which means that most of the information about B is not given by A.

The transition probability matrix of the 1-st order markov chain $P^{(1)}$ can be computed by counting the number of times the state i is followed by state j. The resulting $|\psi^{(1)}|$ histograms can be normalized to obtain the state-wise transition probability mass functions as the rows of $P^{(1)}$.

We can find the conditional probability of the 1-st order markov chain as:

$$H^{(1)} = -\sum_{i \in \psi^{(1)}} \pi_i^{(1)} \sum_{j \in \psi^{(1)}} p_{j|i}^{(1)} \log_2 \left(p_{j|i}^{(1)} \right), \tag{33}$$

where $\pi_i^{(1)}$ is the average probability of being in state i, which is computed by counting the total number of times each state is visited and then normalizing the frequency histogram.

It can be clearly seen in Figure 16(b) that load and ACE values exhibit a very

low conditional entropy at all orders. There is a slight decaying trend in conditional entropy, if observed closely; however, the decay is not significant enough. Moreover, conditional entropy at all the markov chain orders is very small for both the parameters, i.e., < 0.012. Therefore, prediction can be done using the first order markov chain.

3.6.2 Prediction Algorithm for Load and Area Control Error

Based on the analysis and results in the previous section, we now develop a simple predictor based on the markov chain. This prediction algorithm is in essence a variant of the adaptive thresholding algorithm presented in [83]. The presented algorithm along-with the prediction is also capable of adapting to the underlying variation observed in the load and ACE behavior. Below we discuss the algorithm and its prediction accuracy under normal conditions, i.e., no attack.

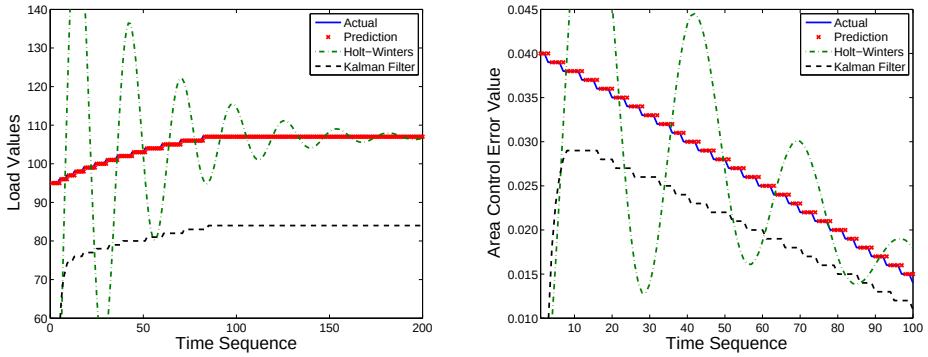
The data was generated using the well known two-area power system, as discussed earlier. Let us subdivide the data values into ψ equal sized bins. Since multiple values will fall into each bin, the prediction will give us the range of expected values for the future time instance. The number of total bins ψ can then be calculated by dividing the minimum and maximum allowed value by the size of each bin. Since allowed values are defined by the capacity of the network, we do not expect to see a smaller value than minimum or greater value than maximum, thus prediction is not needed for those values.

Since the conditional entropy was very low for the load and ACE values at all the markov chain orders, we decide to use the first order markov chain in order to

Algorithm 3: Load prediction algorithm

Data: Observed values l
Result: Prediction for the value
$\varepsilon = |l_p^t - l_o^t|$;
if $\varepsilon > \alpha$ **then**
  $P^{(t+1)}_{l_o^{t-1},l_o^t} = \beta \times P^{(t)}_{l_o^{t-1},l_o^t}$;
end
else
  $P^{(t+1)}_{l_o^{t-1},l_o^t} = \omega \times P^{(t)}_{l_o^{t-1},l_o^t}$;
end
$l_p^{(t+1)} = P^{(t+1)}_{l_o^t,max(l_k)}$; where $l_k \in \psi$

keep the complexity of the algorithm low. Let $P^{(t)}$ denote the transition probability matrix at time t, where $P^{(t)}_{i,j}$ denote the $i-th$ row and $j-th$ column of the matrix. Also, l_p^t and l_o^t denote the predicted and observed values, respectively, at time t. The working of the algorithm is shown in Algorithm 3. The input to the algorithm is the observed value at the current time instance t and the output is the predicted value for the next time instance $t+1$. The algorithm first calculates the error in the prediction and the observed value for time instance t. If the difference is greater than a particular threshold, we update the transition probability matrix, in order to be adaptive, by giving a higher weight (β) to the transition from $l_o^{(t-1)}$ to $l_o^{(t)}$. However, if the difference is not greater than a threshold α, we update the transition probability matrix by a regular weight ω, where $\omega < \beta$ and α, β and ω are tunable parameters. Once the transition probability matrix is updated, it is used to make a prediction for the next time instance $t+1$ based on the current value l_o^t. Since each row is a pmf for a given state, the column with the highest probability, i.e., the bin representing the range of values, is selected as the predicted range of values for the next time instance $t+1$. Since it is an adaptive online prediction algorithm, it adapts and gives



(a) Load Prediction (b) ACE Prediction

Figure 17: Prediction accuracy of the algorithm

higher weight to learn the prediction error greater than the threshold α. This makes the algorithm learn the anomalous behavior as well on the run-time. However, it keeps flagging those instances until the behavior is completely learnt and has higher probability than observing any other state. This helps in reducing the false alarms in case the deviation was caused due to a legitimate change in load behavior.

The prediction accuracy of the developed predictor for load and ACE values is shown in Figures 17(a) and (b), respectively. We show the prediction accuracy comparison with a well-known Kalman-Filter based power system load predictor [71] and general purpose predictor Holt-Winters [84]. Although Kalman-Filter based predictor was originally developed for load prediction only, we also predict the ACE values since our detection approach requires monitoring both the data feeds. We do not compare with the hybrid predictor [72] since it does prediction on hourly basis only, however, we are predicting in an online manner for each data value. The solid blue line denotes the actual values observed while the red cross, dotted green, and black dashed lines denote the prediction using Algorithm 3, Holt-Winters, and Kalman-Filter, respec-

tively. It can be clearly seen that even in the constant load increase and ACE decrease trend, the prediction algorithm was highly accurate and outperformed Kalman-Filter and Holt-Winters. Moreover, the prediction followed the complete trend until the values were stabilized. We show a subset of data for clarity; however, similar results were obtained for the entire 24-hour data. Therefore, this prediction algorithm can be a good measure in order to predict the short term future load and ACE values for the purpose of anomaly detection.

3.7 Anomaly Verification Module for Automatic Generation Control

Prediction serves as the first tier in identifying anomalies. Since prediction is done for short intervals and considers only the temporal dependence in the single variable without complete knowledge of the system, it may yield false positives. To this end, as a second tier, we build a model for the AGC system. AGC can be seen as a control system taking finite inputs and generating finite outputs. The main inputs are P_{tie}, f, and l, where l is for all the areas participating in an AGC. Similarly, the outputs or calculations it does are ACE and P_{sch}, where P_{sch} is for all the participating generators in an AGC. We consider two different cases: 1) single AGC and 2) multiple AGC for two-area power system. Single AGC is the environment where a power company focuses on its own AGC without any collaboration with any other power company's AGC. However, a multi-AGC scenario considers multiple power companies sharing power resources for power generation and control.

3.7.1 Single Automatic Generation Control Model

Since AGC has a finite set of input and output variables, the state of a single AGC can be encoded using the following characteristic function:

$$\sigma : P_{tie} \wedge f \wedge l \wedge ACE \wedge P_{sch} \rightarrow \{true, false\} \tag{34}$$

The function σ encodes the state of the AGC by evaluating to *true* whenever the parameters used as input to the function correspond to the values observed in the system. If the AGC observes x unique combinations, then exact x assignments to σ function will evaluate to *true*. We use these assignments to learn the markov model for the AGC. Since conditional entropy does not show an exponential decay on higher order markov chains, in analysis, we use the first order.

A Labeled Markov Chain (LMC) is a quintuple $M = \{Q, \Sigma, \pi, \tau, L\}$, where Q is a finite set of states, π is an initial probability distribution, τ is the transition probability function, and L is a labeling function. Atomic propositions AP are assigned to states by a labeling function using $\Sigma = 2^{AP}$. Each state is assigned a unique label derived from σ, i.e., s, which is used to define the state. A probability distribution for sequence of states can then be defined using markov chain. Suppose we have sequence $S = s_1, s_2, \ldots, s_n,\ s_i \in Q$. A finite state machine having a directed graph can be learned from the given sequence S.

To learn the markov model, we initialize an empty graph and then start observing the sequence S from AGC. It utilizes a sliding window approach where the window slides at instance i by one entry, i.e., s. If s_i already exists in the graph then a

directed edge from s_{i-1} to s_i is created, if the directed edge does not exist already. However, if s_i does not exist in the graph, then a node is also created for s_i. This process keeps repeating until S is empty. Once the state machine is created, the transition probability matrix is calculated using the frequency of transitions observed while building the finite state machine.

Since the devised model is based on markov chain and exhibits a temporal dependence, we define properties in Linear time Temporal Logic (LTL) [59]. Unlike traditional model checking, stochastic model checking allows you to check that with what probability the property is satisfied by the model. These probabilities can be thresholded in order to accommodate the unseen behavior up to a certain extent. The probabilistic LTL can be defined as:

$$\phi ::= P_{\bowtie p}(\varphi), \quad \bowtie \ \in \ \{\geq, >, \leq, <, =\}; p \in [0, 1]$$

where φ is an LTL formula. Since the states are defined using measurement/calculation variables, properties can be written in the form of conditional probability. For example, given the AGC is in a state having some values for the variables under consideration, what is the probability of seeing the current value of any variable? That will determine the state transition at a discreet time interval,

$$\phi ::= P_{\geq min \& \leq max}(ACE_{i+1} | P_{tie_i}, f_i, l_i, ACE_i, P_{sch_i}); \qquad (35)$$
$$i \in \Upsilon;$$

where Υ is the sequence of measurements in the time domain. The above property checks if the probability of current ACE value observed, given the system was in a

particular state, is less-or-equal- and greater-or-equal-than the minimum and maximum probability thresholds, respectively. These probability thresholds are learnt from the data collected at AGC under normal conditions. This identifies whether the system behaves as expected or not. Moreover, it also identifies which particular variable is causing the anomalous behavior. The thresholds are derived from models built under different operating conditions, like different load levels and hours of day. We define the properties for all the variables, mentioned in Equation 34, in similar fashion.

3.7.2 Multiple Automatic Generation Control Model

One feature of the smart grid is its large-scale distributed generation networks. In interconnected AGCs, power imbalance in one area can be caused by the attack on the interconnected AGC. To this end, we extend the single AGC framework to the case of multiple AGCs, which reflects the scenarios where multiple power companies cooperate for power generation and control. We assume that participating organizations communicate state variable data. The state for each AGC can be calculated in the similar fashion as described earlier.

$$\sigma_{agc_j} : P_{tie} \wedge f \wedge l \wedge ACE \wedge P_{sch} \rightarrow \{true, false\} \tag{36}$$

where σ_{agc_j} represents the characteristic function for AGC j. Note that P_{tie} and f are the same at a given time for both the AGCs. The common state information is shared between two AGCs that are connected to a tie-line. Both the participating AGCs will share the characteristic function output with each other in order to define

the overall state of the entire system, i.e., two AGCs. The shared information allows each AGC to check its own state variables and detect local anomalies. The model will be built using the state transitions for both of the AGCs. Consider the sequence of states is:

$$S = (\sigma_{agc_1}^{t_0}, \sigma_{agc_2}^{t_0}), (\sigma_{agc_1}^{t_1}, \sigma_{agc_2}^{t_1}), \ldots, (\sigma_{agc_1}^{t_n}, \sigma_{agc_2}^{t_n}) \tag{37}$$

where $\sigma_{agc_j}^{t_k}$ represents the state of AGC j at time k. Thus, the finite state graph can be learnt the same way as described previously. However, in this case the state function comes from both AGCs instead of one AGC. Similarly, intrusion can be identified by checking the probability of one AGC characteristic function, given the other AGC's characteristic function instead of individual variables, in order to incorporate the knowledge of both of the AGCs, i.e., the entire system. Thus, it can be written in conditional probability form for the case of two connected AGCs as an example,

$$\phi_1 ::= P_{\geq min \& \leq max}(\sigma_{agc_1}^{t+1} | \sigma_{agc_2}^{t}); \tag{38}$$

where the minimum and maximum probabilities are learnt from the model built for both of the AGCs. Similarly we calculate the probability of AGC 2 state given the state of AGC 1.

$$\phi_2 ::= P_{\geq min \& \leq max}(\sigma_{agc_2}^{t+1} | \sigma_{agc_1}^{t}); \tag{39}$$

The iterative process yielded by Equations 38 and 39 provides a distributed and scalable detection for large scale AGC networks. At each time t, two AGCs exchange their state variables. In order to verify an anomaly in an interconnected AGC scenario, we verify the state of one AGC given the state of the other connected AGC using

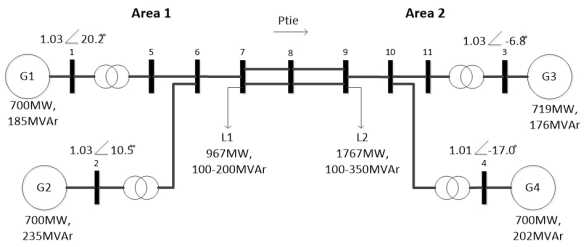


Figure 18: Two-area power system

the above equations. This verifies the overall behavior of both the AGCs together by incorporating system-wide knowledge. This helps in identifying whether both the AGCs behave normally as learnt. It also helps in identifying which AGC deviated away from normal behavior since the property for that AGC would not be validated with respect to the other AGC. This process can be applied to any pair of two connected AGCs.

3.8 Evaluation

In this section we discuss the experimentation and evaluation of the developed approach.

3.8.1 Automatic Generation Control Setup

In our work, we adopt a well-known two-area Kundur's power system model [82]. Figure 18 shows a one-line diagram of the model. This system has been modeled in a power simulation software, PSCAD, which is widely used by power system professionals [85]. The system consists of 2 areas, 11 buses, 4 generation units, and 2 tie lines. All generation units are frequency dependent and the balance is initially restored locally due to the load that varies with frequency. Generator governors change generator output in response to the frequency changes. The two tie lines are identical

in this case study, which can be considered as one equivalent tie line connecting area 1 to area 2. In this simulation, loads are modeled in PSCAD as varying resistance and inductance. Load profile data is derived from September 1, 2010 at NYISO's LONGIL [86]. In this case, the hourly demand curve for NYISO's LONGIL is scaled and used at all the load buses of the test system with time intervals of 5 sec to meet the SCADA data transfer rate as shown in Figure 19. In addition, frequency and tie line power flow (ACE parameters), with respect to time of the day at normal condition, are provided in Figure 20. It can be observed that a power of $0.4p.u.$ flows from area 1 to area 2 under normal conditions (legitimate change in behavior without attack) at maximum load demand, which is hour 16. This high load demand also brings down the system's frequency from $1p.u.$ to $0.99p.u.$ According to our assumption, all the generators respond to change in load and their participation factor is related to their capacity of generation. The value of β, which is the frequency deviation coefficient in Equation 28, is set to 1.9. We show experimental results on a 24-hour dataset. However, the load and ACE can be modeled for a longer duration (e.g., weeks) using their respective base values. The prediction algorithm has been implemented in Java that collects the measurements from PSCAD and predicts accordingly. If an alarm is raised, it is passed on to the anomaly verification module, which has been implemented in PRISM [60]. PRISM supports building the markov model and allows probabilistic property verification.

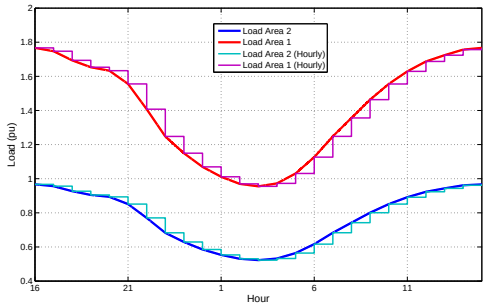


Figure 19: Load profile

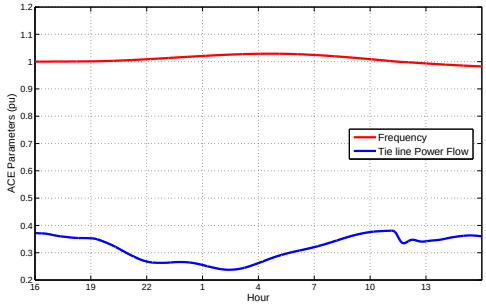
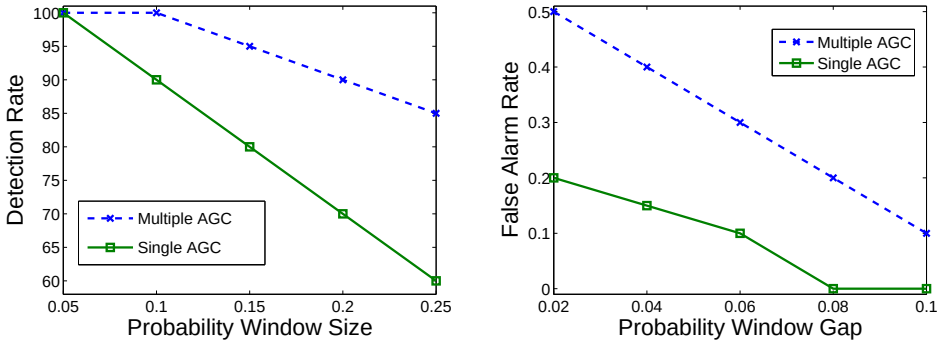


Figure 20: ACE parameters

3.8.2 Detection Accuracy Evaluation for Single and Multiple Automatic Generation Control Model

We evaluate accuracy for both the single and multiple AGC scenarios. Since prediction is done individually for AGC system variables, it is the same in both case studies. Multiple attack instances were introduced and data was collected using the PSCAD tool. The prediction algorithm was employed on the data that included attack instances for both the load and ACE values. It can be clearly seen in Figures 14 and 15 that the prediction follows the trend learnt from the normal data and do not deviate with the sporadic changes observed in the load and ACE values. Perturbations in the ACE values as a result of attack are clear, which ultimately causes the de-stabilization of the power system. All the attack instances were successfully



(a) Detection Rate (b) False Alarm Rate

Figure 21: Comparison of single AGC and multiple AGC scenarios

flagged by the prediction algorithm, i.e., deviated from the prediction; however, one percent false alarms were observed in a 24-hour dataset. These flagged instances were then passed to the anomaly verification module.

The anomaly verification module is evaluated in two fashions – single and multiple AGC. Single AGC is a centralized (or composed) verification of two-area power system as a whole, and the multiple AGC is the distributed verification of the two-area system. In the case of a single AGC, all the attack instances were successfully detected, i.e., 100% detection rate with no false alarm. Similarly, for multiple AGC, we also observed 100% detection rate but with a false alarm rate of 0.1%. Since the properties are probabilistically verified using the model, we change the probability verification threshold and show its effect on the accuracy for both single and multiple AGC in Figure 21. Specifically, we change the minimum and maximum verification threshold, thus increasing or decreasing the window gap between probability verification thresholds. It can be explained by the fact that if the window gap is 1 (maximum gap possible), i.e., the minimum threshold is 0 and the maximum is 1, the property will always be validated successfully. Therefore, it will not detect any intrusions and

hence there is no false alarm. Similar trends can be observed in Figure 21(a) that as the probability window gap increases, detection rate decreases. It can be noticed that the multi-AGC scenario observed a higher detection rate, as compared to single AGC scenario, for the same probability verification window gap. The underlying reason is that the multi-AGC case is more sensitive to intrusions since it checks its local system state given the system state of the other AGC. Intuitively, since it involves all the system variables (in conjunction form) from both the AGCs, all the system variables are verified against all the other variables. Thus, it is a strict property to satisfy and can be thought of as a strict threshold for verification and it is well known that strict thresholds yield higher detection and more false alarm. Hence, it is more sensitive to even slight deviation from normal behavior. On the other hand, in the case of a single AGC, the probability of each system variable is checked only against the entire system state of that particular AGC, thus it is not as sensitive as the multi-AGC scenario. Consequently, multi-AGC observes a higher false alarm rate as shown in Figure 21(b). Although the multi-AGC false alarm rate is higher than the single AGC, it is still negligible and overall higher accuracy was observed in all the cases.

Though the multi-AGC scenario provides higher detection accuracy, it is not an obvious choice. Multiple AGCs requires sharing the system state data among co-operating AGCs, which may be owned by different companies. Thus, it may raise technical and non-technical issues. Moreover, delay in sharing the system state data may cause an incorrect estimate of the system state, thus it requires reliable and efficient communication. Thus, sharing the system state data introduces extra overhead on the system. In such cases, the single AGC scenario is a better suited option.

However, it is not as sensitive to attacks as the multi-AGC scenario. Therefore, the choice depends on the trade off.

3.9 Conclusion

This work presents a two-tier intrusion detection system for AGC. The first tier is an online short-term adaptive predictor for load and ACE variables, which are the key input and output variables in AGC. We show that both load and ACE parameters exhibit a temporal dependence that can be modeled in order to make future short-term predictions accurately. The second tier provides offline probabilistic model checking of the overall system by incorporating system-wide knowledge. Markov models have been used to represent the system state. Two case studies were conducted analyzing single- and multi-AGC scenario. Single AGC verifies the behavior of each system variable with respect to the system state. On the other hand, multi-AGC verifies the state of one AGC given the state of other AGC to verify anomalies. Anomaly verification reduces the overall false alarm rate by incorporating system-wide knowledge. Since measurements are collected every few seconds, model checking at run-time is not computationally feasible. Therefore, the first tier does online prediction for individual variables, and as a result, the flagged instances are then passed to an offline anomaly verification module, which incorporates the complete knowledge of the system in order to verify the anomaly presence. The prediction algorithm exhibits high prediction accuracy ($> 95\%$) under normal conditions. Multiple attack scenarios, inspired from [68], have been implemented. All the malicious measurements have been found to deviate from predicted values, thus being successfully detected. One false

alarm was observed by the prediction on 24-hour data. The second tier successfully verified all the anomalies present with a negligible false alarm rate, i.e., 0% for single AGC and 0.1% for multi-AGC, thus increasing the overall accuracy of the approach.

CHAPTER 4: MEASURING AND DETERRING ATTACK EVASION

In this chapter, we discuss the limitations of existing traditional intrusion defense mechanisms. In previous chapters, we discussed intrusion detection and deterrence for smart grids; however, existing detectors are susceptible to evasion attacks and provide no deterrence against such attacks. While several intrusion detection techniques exist, no effort has solved the problem of comparing the performance of these systems in terms of evasive attacks and deterrence. Critical infrastructure organizations are usually targeted by persistent attackers who have the working knowledge of the targeted system and the capability to evade defense mechanisms. Therefore, we investigate the science of intrusion detection systems and investigate the inherent limitations that allow attackers to evade. Furthermore, to provide deterrence we develop a key-based randomization module that can introduce a notion of randomness for attackers to make evasion more difficult and yet stay deterministic for the intrusion detection system itself.

While the original models for intrusion detection systems were proposed more than two decades ago, intrusion detection still remains an active area of research as the attackers continue to adapt and evade intrusion detection solutions. If an attacker can estimate the normal behavior of the network and the threshold, he/she can easily evade the detection system and send attack packets into the network without detection. Moreover, the possibility of detecting such evasion attacks is bleak due to the

fact that normal network behavior and IDS thresholds are seldom updated.

Advanced persistent threat (APT) attacks can be launched on a network or host. An APT attacker can monitor egress and ingress traffic in a network and launch parameter estimation attacks. While ingress traffic can give information about the normal behavior of the network, egress can provide the information about the alarm. Combining this information can yield the threshold used for detection. Once the attacker develops an estimate of the detection principle and the ADS threshold, it can easily launch evasion attacks that lay below the threshold and hence evade detection. In the later section we show how these parameters can be estimated to paralyze an anomaly detection system. Please note the work focuses on anomaly-based intrusion detection systems, thus we use the terms anomaly detection systems (ADSs) and intrusion detection systems (IDSs) interchangeably.

Numerous studies [87], [88], [89], [90] have focused on the evasion of network intrusion detection systems. As new evasion techniques evolve, new anomaly detection systems are sought to detect and prevent these new evasion attacks. However, to the best of our knowledge, little effort has been invested in studying evasion as a science and developing mathematical models that help future scientists design evasion-resistant anomaly detection systems. Consequently, how to measure the evasion margin for an ADS in a given scenario is still an open research question. Moreover, there are many different categories of evasion techniques depending upon the way they operate. In this work, however, we limit ourselves to the study of parameter estimation attacks only.

Cryptographic systems are robust against estimation attacks due to the extra level

of complexity offered by key-based security. By making the key space exponentially large, it becomes computationally infeasible for an attacker to launch evasion attacks. We use the same principle in our developed key-based randomization technique. The aim is to make it more difficult for the attacker to estimate the parameters and launch a well crafted evasive attack. To this end, we present a case study of the key-based ADS design, which is published in a recent study [91].

4.1 Background

Evasion is one of the most challenging threat in cyber security. Evasion is primarily about staying under the radar and remaining undetected while launching the attack successfully. Attackers who generally launch evasive attacks are sophisticated and persistent. They generally learn or already know the intrusion detection system configuration along with the normal behavior of the underlying network and/or system. The learnt configuration helps attackers carry out nefarious activities without raising an alarm thereby staying undetected.

While some solutions exist for detecting and filtering stealthy attacks, it is still an open challenge in cyber security. Attackers can either initiate a large number of low-volume traffic flows (e.g., from botnets) simultaneously or generate tailored traffic. This helps them stay below the radar, thus avoiding detection. Stealthy evasion attacks take hours, days, or even weeks before they are discovered and mitigated. In turn, they penetrate the network and could consume resources, depriving legitimate users (e.g., provider staff or customers) of access to their information and causing significant loss. Advancing state-of-the-art research requires addressing the following

key research questions: 1) how do they evade existing defense mechanisms, 2) how much can they evade given a defense mechanism and the normal traffic, 3) what changes need to be made in order to detect them or reduce their impact, 4) what valid traffic feature(s) can constitute significant information gain for determining such attacks, and (5) how to measure and evaluate these features in real time.

4.2 Related Work

Science of different systems has been studied in the past to improve systems' performance. The studies [92–95] develop science of firewalls for packet filtering and policy configuration optimization to improve the performance. The study [96] develops adaptive packet filtering of firewall to defend against denial of service attacks. The science of smart grids is developed in study [97] for security and resiliency. In all of these studies, science of systems such as firewalls and smart grids has been developed. However, in this work, we develop a science for intrusion detection systems to improve the performance against evasive attacks.

Intrusion detection systems have been successfully evaded in the past [87, 90, 98–101]. Moreover, new attack techniques were also discovered, which could paralyze the intrusion detection [102–107]. The work [100] discusses practical techniques to evade intrusion detection systems. To countermeasure such evasion techniques, new robust intrusion detection systems have been proposed [41,108–112]. Moreover, quantitative approaches for designing detection algorithms for IDSs have been widely investigated in the past. Methods such as hypothesis testing [32, 113] and machine learning [114, 115] have been applied to provide a scientific basis for IDSs. These

methods aim at designing optimal and effective algorithms to achieve low false negative and false positive rates from the perspective of IDSs. They do not consider the strategic behaviors of the attackers, who intend to evade IDSs by finding countermeasures. Our work develops evasion metrics to evaluate the susceptibility of an IDS to evasion attacks, and designs key-based detection algorithms to defeat attackers' evasion.

One area of related work is the application of game-theoretic tools to model the strategic behaviors of the attackers as in [116–118]. However, due to the lack of appropriate metrics of evasion, few studies have focused on modeling attacker's evasion behavior. Our work aims to fill this gap and provides the basis for game-theoretic analysis of attack evasion.

4.3 Challenges

The performance of intrusion detection techniques is generally compared by comparing the true positive or detection rates on an Receiver Operating Characteristics (ROC) curve. The performance may vary based on the underlying network and/or system used for generating the ROC curve. While this provides a good overview of what type of attacks can be detected by an intrusion detection technique on a given network and/or system, it does not highlight or measure evasion an intrusion detection technique is susceptible to due to its design. There is yet no metric available that can measure evasion susceptibility of an intrusion detection technique regardless of the network and/or system under consideration.

Existing traditional intrusion detection techniques employ static features for the

purpose of anomaly identification. For example, some intrusion detection techniques work on specific fields of a network packet header information or specific characteristics of a system. Intrusion detection technique creates a profile of normal or baseline distribution based on these features, which is then compared with the real-time behavior for the purpose of identifying the presence of anomalous behavior. This inherently limits the intrusion detection technique in identifying the malicious activities that do not cause a deviation in the features employed by the intrusion detection technique. Persistent attackers gain system knowledge, including the features used by the intrusion detection technique and/or the configurations such as threshold to conduct nefarious activities while staying undetected. This is due to the fact that the features employed by an intrusion detection technique are static and may yield a deterministic behavior that can be learnt by persistent attackers to launch attacks. Unlike cryptographic systems, there is yet no approach that can obfuscate or randomize the behavior for attackers to make learning such behavior very unlikely or computationally inexpensive.

4.4 Contribution and Approach Overview

In this work, our contribution is twofold. First, we design a mathematical model to measure evasion for intrusion detection techniques. This enables us to understand the suitability and robustness of an intrusion detection technique against evasion by measuring its evasion margin on different thresholds caused by its inherent design. Second, we develop a key-based randomization technique for existing traditional intrusion detection techniques, that aims to minimize the evasion margin.

We first show that existing intrusion detection techniques can be evaded by persistent attackers. Intrusion detection techniques can be evaded with or without the knowledge of its design and configurations. In this work, we use parameter estimation attacks that monitor the intrusion detection system behavior to infer its design and configuration parameters. We then launch attacks against the intrusion detection techniques by staying under the radar, thus not exceeding the threshold set by intrusion detection technique. This helps us in measuring the evasion margin an intrusion detection technique is inherently susceptible to due to its design.

Evasion margin is a sum of injections that do not cause a deviation in intrusion detection technique's behavior, called the averaging out effect, and the injections that do cause deviations but do not exceed threshold, called the gap between distribution and threshold. To minimize the evasion margin, we then present a mutation approach that can introduce a notion of randomness using key-based randomization for attackers while staying deterministic for the system itself. We use the approach to mutate thresholds and alarms generated by an intrusion detection technique so that the persistent attacker cannot learn the underlying behavior, thus making the intrusion less likely or computationally inexpensive.

4.5 Attack Model

Before we discuss the evasion and how it can be minimized, we first discuss the attack model, datasets, and intrusion detection systems used in this work.

The attack model primarily mimics persistent attackers who are sophisticated. The underlying assumption is that intrusion detection technique's design principle and

Table 4: Endpoint attack traffic for two high- and two low-rate worms

Malware	Avg. Scan Rate(/sec)	Port(s) Used
`Dloader-NY`	46.84	TCP 135,139
`Forbot-FU`	32.53	TCP 445
`MyDoom-A`	0.14	TCP $3127 - 3198$
`Rbot-AQJ`	0.68	TCP 139,769

Table 5: Traffic information for the LBNL dataset

Date	LBNL Hosts	Remote Hosts	Avg. Background Pkt Rate(/sec)	Avg. Attack Pkt Rate(/sec)
10/4/04	4,767	4,342	8.47	0.41
12/15/04	5,761	10,478	3.5	0.061
12/16/04	5,210	7,138	243.83	72

configurations can be estimated or revealed by various methods such as social engineering or observing the underlying network and systems where intrusion detection is deployed. We launch parameter estimation attacks that try to estimate multiple parameters of an intrusion detection technique such as detection thresholds, baseline distribution and real-time distribution by monitoring the system behavior to identify the presence of anomalies. For this purpose, we leverage the temporal dependence in anomaly scores that help estimating the parameters.

4.5.1 Datasets

We use two publicly available and independently collected real-world datasets. The first dataset was collected at the endpoints and the second dataset was collected at the routers of a lab. Thus, both the datasets are different.

4.5.1.1 Endpoint Dataset

This dataset comprises session-level traffic collected at 13 network endpoints. The users of these endpoints included home users, research students, and technical/administrative staff. The endpoints were running different types of applications, including peer-to-peer file sharing software, online multimedia applications, network games, SQL/SAS

clients, etc. Attack traffic in this dataset is mostly comprised of outgoing portscans; see [119] for details. Attack traffic was generated using the following malware: `Zotob.G`, `Forbot-FU`, `Sdbot-AFR`, `Dloader-NY`, `SoBig.E@mm`, `MyDoom.A@mm`, `Blaster`, `Rbot-AQJ`, and `RBOT.CCC` [120]. Table 4 shows statistics of the highest and lowest scan rate worms. For completeness, we also simulated three additional worms, namely `Witty` (worm with fixed source port 4000), `CodeRedv2` (worm with fixed destination port 80), and a low-rate TCP worm (with a fixed and unusual source port 2200). `Witty` and `CodeRedv2` were simulated using the scan rates, pseudocode, and parameters given in [120, 121].

4.5.1.2 LBNL Dataset

This dataset was collected at two international network locations at the Lawrence Berkeley National Laboratory (LBNL), USA. The main applications in internal and external traffic were Web (HTTP), Email, and Name Services. Some other applications like Windows Services, Network File Services, and Backup were also being used by internal hosts. Malicious traffic was mostly comprised of failed incoming TCP SYN requests targeted towards LBNL hosts; see [122] for details. Some pertinent statistics of the LBNL dataset are given in Table 5. Note that the attack rate is significantly lower than the background traffic rate. Thus these attacks can be considered low rate relative to the background traffic rate. We filtered local traffic from the dataset.

4.5.2 Anomaly Detection Systems

We use two well known existing anomaly detection systems. These detectors have shown acceptable accuracy in the recent literature [119]. ADSs used in this work are

diverse in their underlying detection features and principles. Maximum entropy [33] is a self learning system while TRW [32] is sequential hypothesis testing based.

4.5.2.1 Maximum Entropy Anomaly Detector

The Maximum Entropy anomaly detector [33] divides traffic into $2,348$ packet classes and maximum entropy estimation is then used to develop a baseline benign distribution for each packet class. ADS scores of packet class distributions observed in real-time windows are computed by comparing them with their baseline distributions using the Kullback-Leibler (KL) divergence measure. An alarm is raised if a packet class's ADS score repeatedly exceeds a fixed threshold.

4.5.2.2 Threshold Random Walk (TRW) Detector

The TRW algorithm [32] detects incoming portscans by noting that the probability of a connection attempt being a success should be much higher for a benign host than for a scanner. To leverage this observation, TRW computes an ADS score by applying the sequential hypothesis on a remote host's connection attempts. This ADS score is thresholded to determine whether or not a remote host is a scanner.

4.6 Evading Intrusion Detection Systems: A Feasibility Study

Before we discuss the evasion margin measurement for an ADS, we first show how these ADSs can be evaded using parameter estimation attacks. Currently ADSs assume that the underlying detection principle is not known to the attacker. However, it does not hold in the real world where some knowledge about the ADS principle can be obtained through methods like social engineering, fingerprinting, etc. [102]. In fact, ADS can be evaded without knowing the exact design principle. Several

types of attacks (e.g., polymorphic blending attacks, mimicry attacks, etc.) have been proposed in existing literature.

Moreover, ADSs rely significantly on the belief that the attacker does not know the network topology and/or the services running within the network. Hence, it tends to communicate with hosts that do not exist or hosts that do not have the requested service available. Thus the malicious packets from the attacker would considerably alter the real-time traffic characteristics from the baseline distribution. However, we show that this is also a flawed assumption.

For the attacker to estimate the evasion margin, i.e., the number of attack packets that it can send into the network without detection, it has to estimate three ADS parameters [123]: a) baseline distribution, b) realtime distribution, and c) detection threshold.

As a proof of concept, we use two existing, prominent, and diverse statistical ADSs: Maximum Entropy [33] and TRW [32]. These ADSs are briefly described in Section 4.5.2. The rest of the section provides a detailed estimation of the ADS parameters.

The baseline distribution can be estimated by either: a) observing traffic generated from a host within the target network, or b) brute force estimation. For the *realtime observation*, a realistic scenario for estimating the baseline distribution is that the attacker compromises a host X in network A that communicates with the target host [102]. Hence, it can observe the normal traffic from host X to the target entity that can be used to build the normal profile for network A. *Brute force estimation* is the most common modus operandi used in cryptanalysis. Despite its computational complexity, it has been shown that with the current high-performance COTS (multithreaded and

multicore) hardware, it is not difficult for a craft attacker to acquire and exploit hardware parallelism to carry out a bruteforce analysis [102], [89].

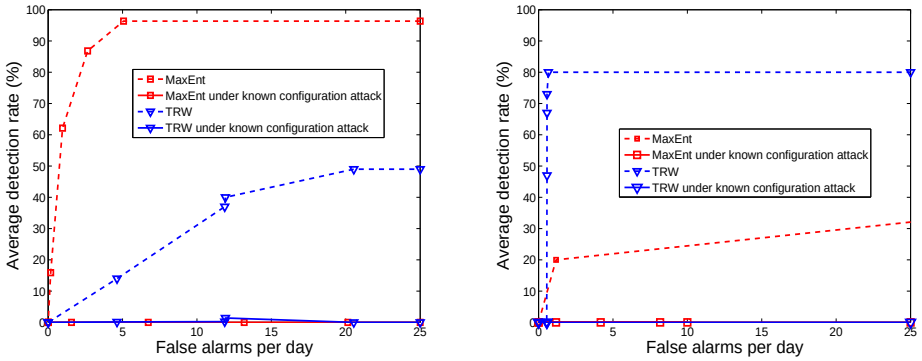
We developed a markovian stochastic model of temporal dependence in an ADS's anomaly scores [83]. While the motivation for the original work was to improve the accuracy and automation of an ADS using threshold estimation, the technique can be adapted to estimate the ADS detection threshold for evasion. However, we restrict the evaluation of conditional entropy to first order markov chains only:

$$H\left(p_j|p_i\right) = -\sum_{\omega} p(p_i, p_j) \log\left(p\left(p_j|p_i\right)\right). \tag{40}$$

The conditional entropy $H(p_j|p_i)$, of two random variables p_i and p_j correspond to the information in p_j not given by p_i. Thus, computing the maximum conditional entropy between baseline distributions in two consecutive time bins, as we slide from bin 1 to bin n, can provide us the minimum information overlap in normal benign data.

Once the baseline distribution and the threshold have been estimated, the detection logic can be used to estimate the realtime distribution. This realtime distribution can then be used by the attacker to generate attack sessions in different feature classes that stay below the threshold and thereby evade ADS detection.

The Maximum Entropy detector [33] employs the Kullback-Leibler (KL) divergence measure for anomaly detection. The measure computes how much the baseline distribution $p(\omega)$ varies from the real-time distribution $q(\omega)$. Traffic is divided into $2,348$ packet classes based on the destination ports and the protocol. The detector uses maximum entropy estimation to develop the baseline distribution for the traffic



(a) Endpoint Dataset (b) LBNL Dataset

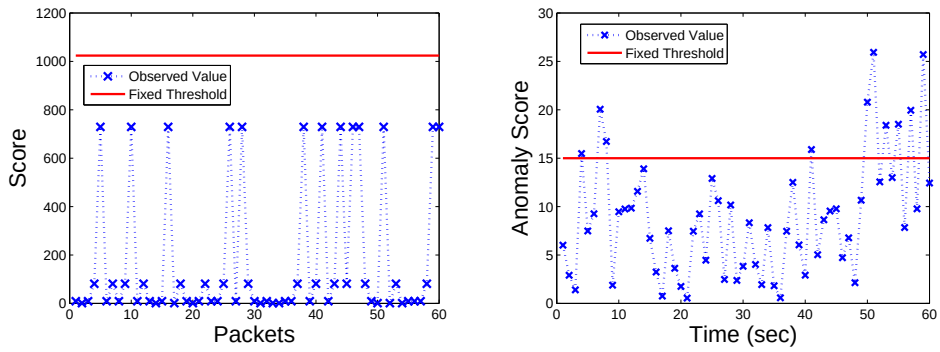
Figure 22: Comparison of ADS performance with and without evasion attack using configuration estimation

classes. If the divergence between the baseline and the real-time distributions for a particular packet class exceeds the threshold τ_{KL} in h of these W windows, an anomaly is flagged by the detector. Maximum entropy has been shown to provide high accuracy dividends in the recent literature [119].

Sequential hypothesis testing based TRW ADSs employ the likelihood ratio test to determine if local/remote hosts are scanners. The TRW classifier [32] detects remote scanners and has been shown to be quite accurate and commercial ADSs also deploy it for portscan detection.

For both Maximum Entropy and TRW ADSs, we launched a stealthy scanning attack by estimating the ADS's parameters as described. It can be observed in Figure 22 that both the ADSs failed to detect the evasive scanning probes (known configuration attack using parameter estimation) on two different datasets. However, it can be seen that both the ADSs were able to achieve acceptable accuracy for traditional scanning attacks (without parameter estimation) discussed in Section 4.5.

Figure 23(a) shows that TRW failed to detect the scan traffic in a malicious time

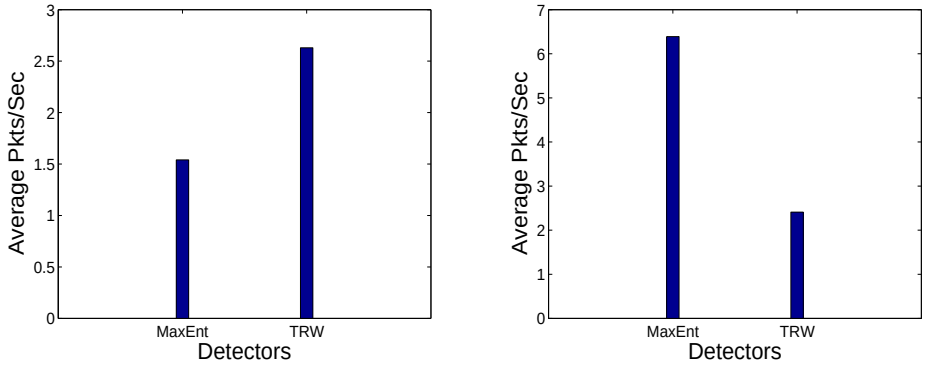


(a) TRW (b) Maximum Entropy

Figure 23: Threshold values observed in stealthy scanning time window for TRW and maximum entropy

window. It can be clearly observed that the likelihood ratio did not exceed the threshold. TRW calculates the likelihood ratio for each connection attempt and classify them as anomalous if the likelihood ratio for any host increases more than the threshold. Since the hosts were also generating benign traffic, the likelihood ratio did not cross the threshold set. Similarly, Figure 23(b) shows the threshold values observed in a 60-second time window for Maximum Entropy. As presented in [33], if the threshold (15) is exceeded 30 times within a 60-second time window, an alarm is raised. Due to stealthy scanning the run-time distribution exceeded the threshold only 10 times. Therefore, just by slowing down (or the effect of averaging out) in the normal traffic, scanning was able to go undetected.

It can be observed that of the three parameters an attacker has to estimate, two of those are dependent on the features employed by the ADS for detection. Since these features are inherently fixed by design, the attacker can easily estimate them to evade the ADSs.



(a) Endpoint (b) LBNL

Figure 24: Evasion margin measurement of MaxEnt and TRW on endpoint and LBNL dataset

4.7 The Science of Intrusion Detection System Evasion

Though several intrusion detection systems have been proposed in the recent past, no science has been developed unlike cryptographic algorithms. Therefore, if the detection principle is known and attacker has some access to the network, he/she can launch evasive attacks. To this end, in this section we discuss the science of intrusion detection. The contribution is two fold. First, we show how evasion can be measured for intrusion detection systems. This helps in measuring the robustness of intrusion detection system against evasion under a given scenario, i.e., network. Second, we introduce a concept of key-based intrusion detection systems, which minimizes the evasion margin and makes it difficult for an attacker to estimate the parameters accurately in order to launch successful evasion attacks. The purpose is to highlight the fact that we need to rethink the design philosophy of an ADS and develop a science.

4.7.1 Evasion Measurement

Evasion occurs when attack traffic does not perturb the run-time distribution more than a threshold. To measure this for an ADS, we inject attack traffic, i.e., SYN flood at rates 0.1pkts/sec and higher. The traffic was injected in each time window with an increasing rate until the alarm was raised by the detector, i.e., the run-time distribution score reached the threshold. The process was repeated for all the time windows to observe how much traffic can go without detection. Figure 24 shows the evasion margin for Maximum Entropy and TRW detectors on endpoint and LBNL datasets. We noticed, for each time window separately, that the amount of traffic that was going undetected was the sum of averaging out and gap between run-time distribution and threshold. Averaging out was the amount of traffic that did not cause any change in the run-time distribution due to its very low volume [124]. In addition, there was some traffic that caused the deviation in run-time distribution but not more than the set threshold. Therefore, the total allowed evasion for each detector was the sum of these two metrics. Below we show the calculation of each separately.

4.7.1.1 Averaging Out

Intrusion detection systems compare the run-time distribution with the baseline distribution to find the deviation between the two. Ideally, there should be a change in the run-time distribution when the attack traffic is inserted into the network. However, if the volume of the normal traffic is overwhelming and the fact that ADS computes distribution using its detection principle, attack traffic might bury inside it

without causing any change in the overall run-time distribution. This phenomenon is known as the averaging out effect. Suppose $\hat{a}$ is the attack traffic launched by the attacker then:

$$\hat{Dr} = Dr + \hat{a} \tag{41}$$

where $\hat{Dr}$ and Dr represents the run-time distribution with and without attack, respectively. To calculate the effect of attack traffic on the normal run-time distribution we compute:

$$Dist^t = diff(\hat{Dr}, Dr)^t \tag{42}$$

where $diff$ is a function that calculates the difference in the two distributions. It can be the intrusion detector's own classification function that is used to compare the run-time distribution with the baseline distribution. However, if the distribution Dr is statistical/probability based, the difference function can be the Kullback-Leibler (KL) divergence measure as used in Maximum Entropy detector. This will calculate the distance between the two distributions, i.e., with and without attack traffic. To calculate how much attack traffic is allowed in the average out scenario,

$$\sigma^t = max(\hat{a})^t s.t. Dist^t = 0 \tag{43}$$

where an attacker wants to maximize the attack traffic $\hat{a}$ such that the distance between the two distributions, i.e., one without attack traffic and with attack traffic remains 0. This implies that both the distributions are the same and attack traffic is successfully averaged out, thus causing no change to the normal run-time distribution.

To calculate it for the t time windows of the network traffic:

$$avg_{ids} = \frac{\sum_{i=1}^{t} \sigma^i}{t} \tag{44}$$

The above equation calculates the averaging out effect for all the time windows and gives us the on-average averaging out in terms of packets allowed by the intrusion detection system. Since the underlying network can change and each intrusion detection system calculates distributions differently, we use ratio or percentage of attack traffic with respect to the normal traffic, which can be calculated using:

$$avg_{ids} = \frac{\sum_{i=1}^{t} (\sigma^i / p^i)}{t} \tag{45}$$

where p^i is the normal background traffic in time window i.

4.7.1.2 Gap between Distribution and Threshold

The second factor to consider in measuring the evasion margin available to the attacker is the gap between run-time distribution score and the threshold. Since threshold is learnt as an average and run-time distribution may change during its course, the gap size between the two will change. Given a fixed threshold and the run-time distribution score, the gap can be calculated as:

$$gap = \triangle = f(\tau, \alpha) \tag{46}$$

where α is basically derived from the run-time distribution $\alpha = f(Dr)$ and $\triangle$ is the number of packets required to make run-time distribution hit the threshold τ. Intrusion detection systems quantify the distance between baseline and run-time distribution in order to compare it with the threshold. If it exceeds a threshold τ, an

alarm is raised, otherwise not.

$$if(\alpha > \tau); anomaly, \quad else; normal$$

Since threshold is mostly fixed for the intrusion detection systems and run-time distribution keeps changing over the period of time. Therefore, percentage or ratio of attack traffic for the gap can be calculated using:

$$gap_{ids} = \frac{\sum_{i=1}^{t}(\triangle^i/p^i)}{t} \tag{47}$$

The above equation calculates the on-average gap available to an attacker. This gap is the resultant of the distance between the threshold and the run-time distribution observed.

Therefore, the total evasion margin available to an attacker is the sum of the averaging-out effect and the gap available. This can be leveraged by the attacker to launch a certain amount of attack traffic while staying undetected

$$\varepsilon = \frac{\sum_{i=1}^{t}(\sigma^i/p^i)}{t} + \frac{\sum_{i=1}^{t}(\triangle^i/p^i)}{t} \tag{48}$$

4.7.2 Evasion Mitigation

We showed in Section 4.6 that ADSs can be evaded. To evade an ADS successfully, an attacker needs to know the detection principle, normal traffic, threshold estimation, and the output, i.e., whether an alarm was raised or not. Since some of these parameters are easily observable or can be estimated, our goal is to deceive the attacker while reducing the evasion margin. For example, detection principle can be revealed using social engineering, thus it is hard to keep it a secret. If an attacker

has access to the network, he/she can infer the normal behavior of the network thus hiding this information is not trivial. We present a key-based ADS design as a case study for key-based adaptive mutation for threshold and the output, i.e., alarm.

Mutating the threshold would deceive the attacker by giving the wrong estimate of the evasion margin. It can be seen in Equation 48 that if the value of τ is changing or not known to the attacker, an accurate estimate of the evasion margin available can not be made. Moreover, to estimate the evasion margin accurately an attacker relies on observing the output, i.e., whether an alarm was raised or not. To this end, we mutate the output using a secret key in order to deceive the attacker by hiding the actual alarm. Therefore, an attacker would not know whether the alarm was actually raised or not. Consequently, he/she would get a false estimate of the evasion margin available. In the subsequent section we discuss the key-based ADS design as a case study and show its effectiveness in reducing the evasion margin.

4.7.2.1 Key-based Threshold Mutation

Several ADSs have been proposed in the recent past. Most of them do not provide any scientific method to calculate the threshold, however, some of them provide a threshold calculation method specific to the detector. In general, thresholds are calculated by observing the ADS's performance on a network using a Receiver Operating Characteristics (ROC) curve. The threshold point providing the acceptable accuracy is selected. Threshold range for generating an ROC curve is selected based on the minimum and maximum deviation observed in the baseline distribution.

$$\tau_{i,n} \rightarrow d^{min}(D_b), \ldots, d^{max}(D_b) \tag{49}$$

where $\tau_{i,n}$ represents the threshold range for ADS i on a network n, D_b is the baseline distribution and d is the deviation function of the baseline distribution. Since the minimum deviation is a strict constraint and yields a higher false positive rate, average deviation is often selected. Therefore, the above equation becomes:

$$\tau_{i,n} \rightarrow d^{avg}(D_b), \ldots, d^{max}(D_b) \tag{50}$$

Moreover, maximum deviation allows the maximum evasion gap, thus reducing the detection rate. Therefore, we replace it with the point providing acceptable accuracy on the ROC curve. Hence, the equation becomes:

$$\tau_{i,n} \rightarrow d^{avg}(D_b), \ldots, d^{roc}(D_b) \tag{51}$$

We mutate the threshold in this range using a secret key and a hash function as follows:

$$\tau_{i,n}^{t+1} = H(k, \tau_{i,n}^{t}, t) mod |\tau_{i,n}| + 1 \tag{52}$$

where $\tau_{i,n}^{t}$ represents the threshold selected at time t, k is the secret key and modulus is taken for the threshold set size to pick a threshold from the given range. Therefore, if the key is not known, an attacker would not know which threshold is being used at a given time.

Since the run-time behavior of the network is ever changing, it may allow more evasion margin at a given time. Therefore, we tend to adaptively mutate the threshold with respect to the network behavior observed. We divide the entire possible range

of thresholds for a network into subranges.

$$\begin{aligned} \tau_{i,n,1} &\rightarrow d^{min}(D_b), \ldots, d^{avg}(D_b) \\ \tau_{i,n,2} &\rightarrow d^{avg}(D_b), \ldots, d^{roc}(D_b) \\ \tau_{i,n,3} &\rightarrow d^{roc}(D_b), \ldots, d^{max}(D_b) \end{aligned} \tag{53}$$

where $\tau_{i,n,j}$ represents the subset range j from the entire range $\tau i, n$. The mutation function becomes:

$$\tau_{i,n}^{t+1} = H(k, \tau_{i,n}^{t}, t) mod \beta + 1 \tag{54}$$

$$\beta = f(D_r, D_b); \quad \beta \in \{\tau_{i,n,1}, \tau_{i,n,3}, \tau_{i,n,3}\}$$

where β is a function of the run-time distribution observed and the baseline distribution. It is selected based on the distance between the two distributions. Therefore, if the distance falls in the subrange $\tau_{i,n,1}$, then this subrange will be selected as β and the threshold will be randomly selected from this range.

4.7.2.2 Accuracy and Performance Analysis of Key-based Threshold Mutation

We use the similar notations as those used in [32]. First we define Y to be the random variable such that $Y = 0$ means a successful attempt by a fixed source host and $Y = 1$ means an unsuccessful event. H_0 and H_1 are the hypotheses that the source host is benign or malicious, respectively. Assume $p(Y = 0|H_0) = \theta_0$ and $p(Y = 0|H_1) = \theta_1$, then we have $p(Y = 1|H_0) = 1 - \theta_0$ and $p(Y = 1|H_1) = 1 - \theta_1$. Suppose the alarm criteria is set to be that, given a source host, if it makes between n_1 and n_2 unsuccessful attempts out of N total attempts, an alarm is raised. Here

the threshold mutation range is between n_1 and n_2.

The probability that a benign user makes n unsuccessful attempts out of N total attempts is:

$$\zeta_0(n) = \sum_{i=n}^{N} \binom{N}{n} (1-\theta_0)^i \theta_0^{N-i} \tag{55}$$

The probability that a malicious user makes n unsuccessful attempts out of N total attempts is:

$$\zeta_1(n) = \sum_{i=n}^{N} \binom{N}{n} (1-\theta_1)^i \theta_1^{N-i} \tag{56}$$

Suppose the ratio of benign users out of all users is u, then given the condition that n unsuccessful attempts out of N attempts, the probability that the source is malicious is:

$$\frac{\zeta_1(n)(1-u)}{\zeta_0(n)u + \zeta_1(n)(1-u)} \tag{57}$$

If the ADS raises an alarm and if there are n unsuccessful attempts out of N attempts, the false positive probability is:

$$\xi_1(n) = \frac{\zeta_0(n)(u)}{\zeta_0(n)u + \zeta_1(n)(1-u)}, \tag{58}$$

and the false negative probability is:

$$\xi_2(n) = \frac{(1-\zeta_1(n))(1-u)}{(1-\zeta_0(n))u + (1-\zeta_1(n))(1-u)} \tag{59}$$

For the mutation range between n_1 and n_2, the false positive probability is between $\xi_1(n_2)$ and $\xi_1(n_1)$, and the false negative probability is between $\xi_2(n_1)$ and $\xi_2(n_2)$.

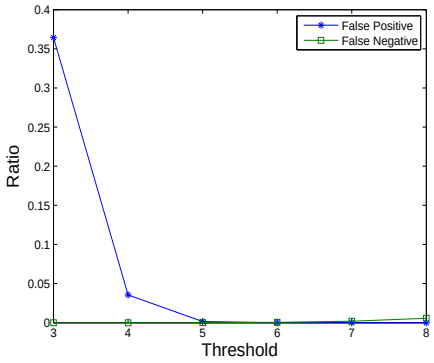


Figure 25: False positive and negative probability

The expected false positive probability and false negative probability are:

$$\sum_{i=n_1}^{n_2} \xi_1(i)/(n_2 - n_1), \text{and} \sum_{i=n_1}^{n_2} \xi_2(i)/(n_2 - n_1), \tag{60}$$

respectively. If $(1 - \theta_0) > n_2/N$, then the expected number of attempts that a malicious source will be identified is between

$$un_1/(1 - \theta_0) + (1 - u)n_1/(1 - \theta_1), \text{and}$$

$$un_2/(1 - \theta_0) + (1 - u)n_2/(1 - \theta_1)$$

Figure 25 shows the false positive and negative probability for $N = 8$, $3 \leq n \leq 8$, $u = 0.999$, $\theta_0 = 0.95$ and $\theta_1 = 0.1$. We can see that under the parameters the false positive and negative are all smaller than 0.1 (except the case of $n = 3$). Figure 26 shows the expected false positive and negative probability with the same parameters except different θ_1. We can see that the false positive is stable for the range of θ_1 in consideration and there is a slight decrease of false negatives with the decrease of θ_1.

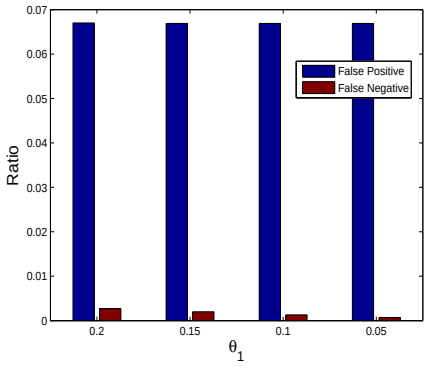


Figure 26: False probability with different θ_1

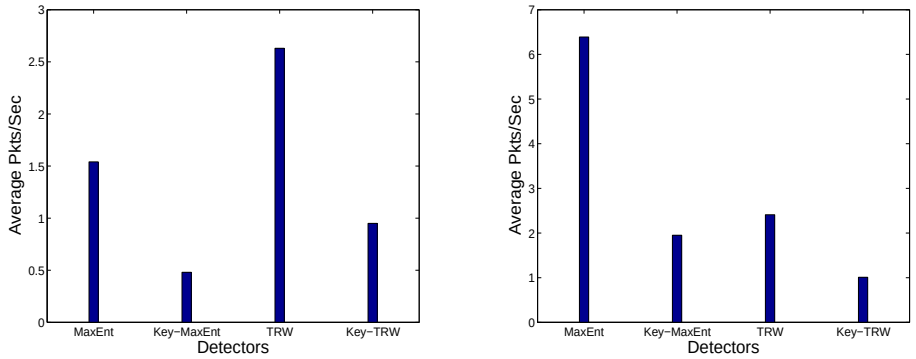
4.7.2.3 Key-based Alarm Mutation

Since an attacker relies on observing the alarm (whether raised or not) to estimate the evasion margin, we mutate the alarm to deceive the attacker. Suppose the alarm is represented by $l \rightarrow true, false$, where the alarm can generally take two values, i.e., $true$ indicates that there was an anomaly and $false$ indicates that there was no anomaly. Instead of sending the original alarm, which may be observable by the attacker, we send a hashed alarm:

$$l^{t+1} = H(k, l^{t+1}, t) mod\ l + 1 \tag{61}$$

where l^{t+1} refers to the alarm raised at time $t+1$. We take the hash of the original alarm with the key and time stamp and send the resultant hashed alarm. The receiver side, security engineer, or administrator, will repeat the same process for both the $true$ and $false$ alarm along-with the time stamp and the key. The one that matches will reveal the original alarm. This helps in deceiving the attacker and gives him/her a wrong estimate.

Since the alarm set has only two possible values, i.e., $true$ and $false$, collision



(a) Endpoint (b) LBNL

Figure 27: Evasion margin measurement of original and key-based MaxEnt and TRW on endpoint and LBNL dataset

probability of the hash function is higher, i.e., both the $true$ and $false$ might yield to the same hashed alarm with the key and time stamp. There are two countermeasures to this. First, increase the alarm set size, i.e., instead of only $true$ and $false$, it may have severity level of the alarm with it. Second, the hash of the alarm can also be sent with the hashed alarm to verify the actual alarm. However, it might give the attacker a chance to recover the key, as it happens in known text attacks in cryptography. To defend this, keys can be changed periodically, thus not providing enough time for the attacker to reveal the key for a given time period. There are several other ways of hiding the original alarm such as cryptographic algorithm. However, the goal here is to deceive the attacker. Therefore, we send the alarm itself but it is randomly selected using a hash function. The purpose of this work is to re-think the design philosophy and we present a case study here, though other techniques can be devised. Therefore, we only work with the resultant hashed alarm, which looks like the original alarm while deceiving the attacker.

To show the effectiveness of the key-based intrusion detection system design, we

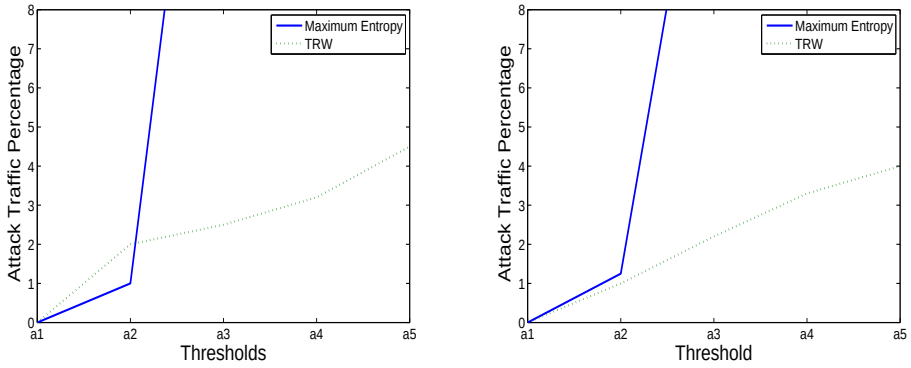
repeat the same experiment of measuring the evasion margin in time windows but with key-based intrusion detection employed. Figure 27 shows the evasion margin comparison of original and key-based MaxEnt and TRW. It can be seen that the evasion margin is reduced significantly, i.e., 60-70% in the case of key-based counterparts of intrusion detectors. Evasion margin can be reduced further if observed anomaly scores are predicted to set the threshold. However, that will not be able to deceive the attacker since thresholds can be estimated. Therefore, we divide the anomaly scores in subranges and randomly select a threshold from the observed subrange. This introduces the notion of randomness to deceive the attacker, which is a goal of the approach.

4.8 Evaluation

In this section we present the evaluation of our developed approach. First, we show the evasion margin measurement for both the ADSs. This measure gives an estimate of how much an ADS is susceptible to evasion attacks, i.e., the evasion margin available to the attacker to launch evasive attacks. Please note that evasion margin is not the opposite of detection. It is rather the susceptibility of attacks to go undetected. Second, we show the effectiveness of key-based ADS design in detecting evasion attacks, i.e., parameter estimation attacks.

4.8.1 Evaluating Evasion Margin Metric

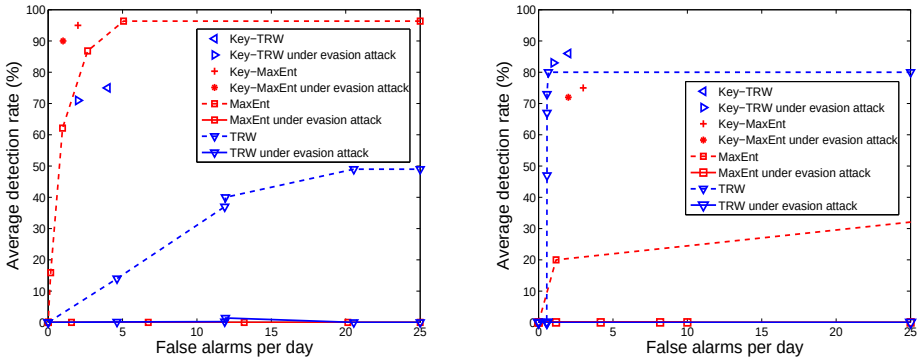
To measure the evasion margin for an ADS, we inject attack traffic, i.e., SYN flood at rates 0.1pkts/sec and higher. The traffic was injected in each time window with an increasing rate until the alarm was raised by the detector, i.e., run-time distribution



(a) Endpoint (b) LBNL

Figure 28: Detectors evasion margin comparison on varying thresholds

score reached the threshold. We repeated the process with different thresholds to discover the susceptibility of ADSs to evasion attacks. Figure 28 shows the results for both the detectors on both the datasets. It can be seen that the evasion trend remains the same for each ADS on both the datasets, i.e., Endpoint and LBNL using different thresholds. This is because evasion margin is a property of ADS's detection principle and to eliminate the effect of the underlying dataset on evasion margin we work with attack traffic percentage with respect to normal traffic. It can be observed in Figure 28 that as the threshold changes from strict to lose bound, maximum entropy evasion margin increases exponentially because it requires the same traffic class to exceed h of W time windows above threshold in order to raise an alarm. On the other hand, TRW evasion margin increases linearly as it works on successful connection ratio. Therefore, if the threshold is configured improperly/lose bounded or attack traffic is distributed evenly across multiple classes, maximum entropy will allow more evasion margin to an attacker as compared to TRW.



(a) Endpoint (b) LBNL

Figure 29: Accuracy comparison of detectors with and without key on regular and evasive attacks

4.8.2 Evaluating Evasion Mitigation

In this section we show how key-based design, when employed with an ADS, improves the detection accuracy against evasion attacks. We conduct experiments on parameter estimation attacks and traditional portscan attacks explained earlier to evaluate the performance of key-based design. Figure 29 shows the evaluation results for both detectors on both datasets. It can be observed that evasion attacks were able to paralyze the performance of an ADS, i.e., almost 0% detection rate, however, key-based ADSs were able to detect most of evasion attacks for both the ADSs on both the datasets. These attacks are labeled as "evasion attacks" in the figure. It can be seen that significant performance improvement is achieved by the key-based design over original ADS in all the cases.

Moreover, key-based ADSs show accuracy improvement for traditional portscan attacks as highlighted in the results. Although traditional portscans were not injected in an evasive manner, some instances of these attacks stayed below the threshold.

Since key-based design adaptively mutates the threshold, it was able to flag some of such undetected instances, thus increasing the accuracy. Although key-based design helps in reducing the evasion margin overall, it does not cater to the averaging out or incapability of an ADS to detect some attacks, which is mainly related to the inherent limitation of an ADS due to its detection principle. Therefore, it does not achieve 100% detection accuracy.

4.9 Conclusion

Anomaly-based intrusion detection systems (ADSs) are capable of detecting zero-day attacks. However, ADSs are inherently susceptible to evasion attacks. As a remedy, we have established a novel design methodology based on cryptographic concepts by making evasion computationally infeasible or harder to estimate for the attackers. To achieve this, we first developed a metric for measuring evasion, and have demonstrated that evasions can be easily achieved under the current ADS scheme. We developed key-based designs that mutate the threshold and alarms to deceive the attacker, and render it difficult for the attacker to estimate the parameters of ADSs. Our results have shown that the key-based mutation strategy was able to successfully detect most of the evasion attacks that were undetected by the original ADS by reducing the evasion margin. Although we show mutation for threshold and the alarm output, the framework can be extended to mutate other parameters of an ADS. The goal of the work is to rethink the design philosophy of ADSs such that their inherent susceptibility to evasion attacks can be addressed. In the future we aim to extend the current work to a game-theoretic framework, which incorporates

the model of attackers' evasion strategies. The extended model will allow us to design optimal mutation strategies for ADSs.

CHAPTER 5: CONCLUSION AND FUTURE WORK

In this work we investigate the intrusion detection and deterrence for critical infrastructure, i.e., smart grids. Critical infrastructures are the infrastructures that have high impact in our day to day life. Unavailability of such infrastructures may cause serious damage to the nation's economy, safety, and/or security. To this end, we investigate two core components in smart grids, i.e., AMI and AGC. We devise intrusion detection and deterrence techniques for these components. Lastly, we investigate the inherent limitations of evasion in existing intrusion detection systems that make them unsuitable for smart grids. We devise an evasion margin measurement metric that can be used to measure an intrusion detection system's performance against evasive attacks. To minimize evasion and make it less likely or computationally inexpensive, deterrence is introduced by randomizing intrusion detection systems' parameters.

First, we show that the current approaches for defense in AMI lack practical feasibility as they require hardware deployment in the field. Moreover, protocol implementation is vendor specific, thus such solutions may not hold true. We show that AMI behavior is static due to limited configurations, devices and application support. To this end, we present an approach that does not require hardware deployment in the field and can work with the logs that are already generated in the current infrastructure. The approach utilizes the configurations in order to derive specifications which in turn can be validated against the model built from logs. We show that the

logs can be modeled using $4-th$ order markov chain and the properties are defined in LTL to detect intrusions. To overcome the static behavior of the network and mitigate against evasion/mimicry attacks, we design a configuration randomization module that introduces a notion of randomness in the AMI behavior for the attacker while staying deterministic for the network itself. We show that the false predictions by the model were less than 1%. Furthermore, the model was able to achieve a more than 95% detection rate with a false positive rate of approximately 0.1%. The run-time complexity of the model was approximately 1.5 secs and it can scale up to approximately $5,000$ to 16 meters per collector depending on the order of markov chain selected, however, as an industry practice we have seen 8 meters per collector.

Second, we investigate AGC in smart grids. Although bad data detection mechanisms exist in smart grids, recent studies show that these algorithms can be bypassed. Moreover, knowledged attackers can manipulate measurements such that they look benign to the AGC environment. We also discuss that there is no tailored defense mechanism for generation control in smart grids that caters to the changing behavior over time. To this end, we develop a two-tier approach for anomaly detection. The first tier utilizes the temporal dependence presence in the key variables to predict short term future behavior. The second tier incorporates system-wide knowledge to verify the alarms raised in the first tier. The developed approach complements existing defense mechanisms such as bad data detection. We show that the approach yields high prediction accuracy of $> 95\%$ under normal conditions. The detection rate was almost 100% with negligible false alarm rates of 0% and 0.1% for single and multi-AGC scenarios, respectively.

Third, since existing defense mechanisms are susceptible to evasion attacks, we present a metric for evasion margin measurement that can help in comparing the performances of intrusion detection systems with and without deterrence. We showed that evasive attacks dropped the performance of intrusion detection rate to 0% thereby paralyzing it. Since evasion susceptibility is due to intrusion detection systems' static design, we present a key-based randomization module that can introduce deterrence and a notion of randomness in an intrusion detection system thereby making evasion more difficult. We showed that the key-based approach reduces the evasion margin of an intrusion detection by 60-70%. We also show that the key based approach improved the accuracy detection by at least 70% against evasive attacks.

In the future, we aim to investigate smart grids as a whole and develop a correlation-based intrusion detection and deterrence system that can identify attacks by incorporating system-wide knowledge of smart grids. We also aim to introduce a key-based randomization mechanism to other parameters used by intrusion detection systems such as features that build the run-time distribution. For example, intrusion detection systems use static features to detect specific types of attacks, thus they are unable to detect other type of attacks. Mutating the feature space would allow for detecting a variety of attacks. Furthermore, we aim to use a game-theoretic framework to model attackers' evasion strategies so that the optimal mutation mechanism can be devised at the run-time.

REFERENCES

[1] Department of homeland security. "hXXp://www.dhs[.]gov/what-critical-infrastructure".

[2] F. M. Cleveland. Cyber security issues for advanced metering infrasttructure (AMI). In *IEEE Power and Energy Society General Meeting - Conversion and Delivery of Electrical Energy in the 21st Century*, 2008.

[3] U.S. Government Accountability office (GAO). information security: TVA needs to address weaknesses in control systems and networks, 2008.

[4] Idaho National Laboratory (INL). NSTB assessments summary report: Common industrial control system cyber security weaknesses, May 2010.

[5] The White House. Homeland Security Presidential Directive 7: Critical infrastructure identification, prioritization and protection, December 2003.

[6] T. Baumeister. Literature review on smart grid cyber security. Technical report, Department of Information and Computer Sciences, University of Hawaii, 2010.

[7] Smart grid news. "hXXp://www.smartgridnews[.]com".

[8] S. McLaughlin, D. Podkuiko, S. Miadzvezhanka, A. Delozier, and P. McDaniel. Multi-vendor penetration testing in the advanced metering infrastructure. In *Proceedings of the 26th Annual Computer Security Applications Conference, (ACSAC)*, 2010.

[9] R. Berthier, W. Sanders, and H. Khurana. Intrusion detection for advanced metering infrastructures: Requirements and architectural directions. In *First IEEE International Conference on Smart Grid Communications (SmartGridComm)*, 2010.

[10] R. Berthier and W. Sanders. Specification-based intrusion detection for advanced metering infrastructures. In *IEEE 17th Pacific Rim International Symposium on Dependable Computing (PRDC)*, 2011.

[11] M. A. Faisal, Z. Aung, J. Williams, and A. Sanchez. Securing advanced metering infrastructure using intrusion detection system with data stream mining. In *Proceedings of Pacific Asia Workshop on Intelligence and Security Informatics (PAISI)*, 2012.

[12] Geethapriya Thamilarasu and Ramalingam Sridhar. Intrusion detection in RFID systems. In *Military Communications Conference (MILCOM)*. IEEE, 2008.

[13] Muhammad Qasim Ali and Ehab Al-Shaer. Configuration-based IDS for advanced metering infrastructure. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, CCS '13. ACM, 2013.

[14] Muhammad Qasim Ali, Ehab Al-Shaer, and Qi Duan. Randomizing AMI configuration for proactive defense in smart grid. In *SmartGridComm*, 2013.

[15] M. Q. Ali and E. Al-Shaer. Probabilistic model checking for AMI intrusion detection. In *SmartGridComm*, 2013.

[16] Muhammad Qasim Ali and Ehab Al-Shaer. Randomization-based intrusion detection system for advanced metering infrastructure. *ACM Transactions on Information and System Security (TISSEC)*, 18(2):7:1–7:30, December 2015.

[17] Smart meter - arm. `"hXXp://www.arm[.]com/markets/embedded/smart-meter.php"`.

[18] Y. Zhang, L. Wang, W. Sun, R. Green, and M. Alam. Distributed intrusion detection system in a multi-layer network architecture of smart grids. *IEEE Transactions on Smart Grid*, 2011.

[19] Ehab Al-Shaer, Qi Duan, and Jafar Haadi Jafarian. *Random Host Mutation for Moving Target Defense*, pages 310–327. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[20] Sushil Jajodia, Anup K. Ghosh, Vipin Swarup, Cliff Wang, and X. Sean Wang. *Moving Target Defense: Creating Asymmetric Uncertainty for Cyber Threats*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[21] J. H. Jafarian, E. Al-Shaer, and Q. Duan. An effective address mutation approach for disrupting reconnaissance attacks. *IEEE Transactions on Information Forensics and Security*, 10(12):2562–2577, Dec 2015.

[22] J. H. Jafarian, E. Al-Shaer, and Q. Duan. Adversary-aware ip address randomization for proactive agility against sophisticated attackers. In *2015 IEEE Conference on Computer Communications (INFOCOM)*, pages 738–746, April 2015.

[23] Jafar Haadi H. Jafarian, Ehab Al-Shaer, and Qi Duan. Spatio-temporal address mutation for proactive cyber agility against sophisticated attackers. In *Proceedings of the First ACM Workshop on Moving Target Defense*, MTD '14, pages 69–78, New York, NY, USA, 2014. ACM.

[24] Jafar Haadi Jafarian, Ehab Al-Shaer, and Qi Duan. Openflow random host mutation: Transparent moving target defense using software defined networking. In *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, HotSDN '12, pages 127–132, New York, NY, USA, 2012. ACM.

[25] S. Groat, M. Dunlop, W. Urbanksi, R. Marchany, and J. Tront. Using an ipv6 moving target defense to protect the smart grid. In *2012 IEEE PES Innovative Smart Grid Technologies (ISGT)*, pages 1–7, Jan 2012.

[26] Mohammad Ashiqur Rahman, Ehab Al-Shaer, and Rakesh B. Bobba. Moving target defense for hardening the security of the power system state estimation. In *Proceedings of the First ACM Workshop on Moving Target Defense*, MTD '14, pages 59–68, New York, NY, USA, 2014. ACM.

[27] Ehab Al-Shaer. *Toward Network Configuration Randomization for Moving Target Defense*, pages 153–159. Springer New York, New York, NY, 2011.

[28] F. Gillani, E. Al-Shaer, S. Lo, Q. Duan, M. Ammar, and E. Zegura. Agile virtualized infrastructure to proactively defend against cyber attacks. In *2015 IEEE Conference on Computer Communications (INFOCOM)*, pages 729–737, April 2015.

[29] Qi Duan, E. Al-Shaer, and H. Jafarian. Efficient random route mutation considering flow and network constraints. In *Communications and Network Security (CNS), 2013 IEEE Conference on*, pages 260–268, Oct 2013.

[30] Jafar Haadi Jafarian, Ehab Al-Shaer, and Qi Duan. *Formal Approach for Route Agility against Persistent Attackers*, pages 237–254. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[31] Pedro Garcia-Teodoro, Jesús E. Díaz-Verdejo, Gabriel Maciá-Fernández, and Enrique Vázquez. Anomaly-based network intrusion detection: Techniques, systems and challenges. *Computers & Security*, 2009.

[32] J. Jung, V. Paxson, A. W. Berger, and H. Balakrishnan. Fast portscan detection using sequential hypothesis testing. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2004.

[33] Y. Gu, A. McCullum, and D. Towsley. Detecting anomalies in network traffic using maximum entropy estimation. In *Proceedings of the ACM SIGCOMM Conference on Internet Measurement (IMC)*, 2005.

[34] Matthew V Mahoney and Philip K Chan. PHAD: Packet header anomaly detection for identifying hostile network traffic. *Proceedings of the 2003 ACM symposium on applied computing*, 2001.

[35] Kingsly Leung and Christopher Leckie. Unsupervised anomaly detection in network intrusion detection using clusters. In *Proceedings of the Twenty-eighth Australasian conference on Computer Science-Volume 38*, pages 333–342. Australian Computer Society, Inc., 2005.

[36] Yinhui Li, Jingbo Xia, Silan Zhang, Jiakai Yan, Xiaochuan Ai, and Kuobin Dai. An efficient intrusion detection system based on support vector machines and

gradually feature removal method. *Expert Systems with Applications*, 39(1):424–430, 2012.

[37] Ke Wang and Salvatore J. Stolfo. Anomalous payload-based network intrusion detection. In *Recent Advances in Intrusion Detection (RAID)*, 2004.

[38] Animesh Patcha and Jung-Min Park. An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Computer networks*, 51(12):3448–3470, 2007.

[39] Rahul Khanna and Huaping Liu. System approach to intrusion detection using hidden markov model. In *Proceedings of the 2006 international conference on Wireless communications and mobile computing*, pages 349–354. ACM, 2006.

[40] Yingbo Song, Angelos D Keromytis, and Salvatore J Stolfo. Spectrogram: A mixture-of-markov-chains model for anomaly detection in web traffic. In *NDSS*, volume 9, pages 1–15. Citeseer, 2009.

[41] Roberto Perdisci, Davide Ariu, Prahlad Fogla, Giorgio Giacinto, and Wenke Lee. McPAD: A multiple classifier system for accurate payload-based anomaly detection. *Computer Networks*, 2009.

[42] Monowar H Bhuyan, Dhruba Kumar Bhattacharyya, and Jugal K Kalita. Network anomaly detection: methods, systems and tools. *IEEE Communications Surveys & Tutorials*, 16(1):303–336, 2014.

[43] Davide Ariu, Roberto Tronci, and Giorgio Giacinto. Hmmpayl: An intrusion detection system based on hidden markov models. *computers & security*, 30(4):221–241, 2011.

[44] Aruna Jamdagni, Zhiyuan Tan, Xiangjian He, Priyadarsi Nanda, and Ren Ping Liu. Repids: A multi tier real-time payload-based intrusion detection system. *Computer Networks*, 57(3):811–824, 2013.

[45] C. Ten, J. Hong, and C. Liu. Anomaly detection for cybersecurity of the substations. *IEEE Transactions on Smart Grid*, 2011.

[46] B. Zhu and S. Sastry. SCADA-specific intrusion detection/prevention systems: A survey and taxonomy. In *First Workshop on Secure Control Systems (SCS)*, 2010.

[47] Matti Mantere, Mirko Sailio, and Sami Noponen. A module for anomaly detection in ICS networks. In *Proceedings of the 3rd International Conference on High Confidence Networked Systems*. ACM, 2014.

[48] Marina Krotofil and Álvaro A. Cárdenas. Is this a good time?: Deciding when to launch attacks on process control systems. In *Proceedings of the 3rd International Conference on High Confidence Networked Systems*, HiCoNS '14. ACM, 2014.

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

[50] Ye Yan, Yi Qian, Hamid Sharif, and David Tipper. A survey on cyber security for smart grid communications. *IEEE Communications Surveys & Tutorials*, 14(4):998–1010, 2012.

[51] Mihui Kim. A survey on guaranteeing availability in smart grid communications. In *Advanced Communication Technology (ICACT), 2012 14th International Conference on*, pages 314–317. IEEE, 2012.

[52] Andreas Berl, Michael Niedermeier, and Hermann de Meer. Smart grid considerations: Energy efficiency vs. security. *Advances in Computers*, 88:159–198, 2013.

[53] Jun Wang and Victor CM Leung. A survey of technical requirements and consumer application standards for IP-based smart grid ami network. In *The International Conference on Information Networking 2011 (ICOIN2011)*, pages 114–119. IEEE, 2011.

[54] Stephen McLaughlin, Dmitry Podkuiko, and Patrick McDaniel. Energy theft in the advanced metering infrastructure. *Critical Information Infrastructures Security*, 2010.

[55] David Carroll Challener, Scott Thomas Elliott, James Patrick Hoff, and James Peter Ward. Storing keys in a cryptology device, 2002. US Patent App. 10/051,495.

[56] Daisuke Mashima and Alvaro A. Cárdenas. Evaluating electricity theft detectors in smart grid networks. In *Research in Attacks, Intrusions, and Defenses*, 2012.

[57] M. Merhav, M. Gutman, and J. Ziv. On the estimation of the order of a markov chain and universal data compression. *IEEE Transactions on Information Theory*, 1989.

[58] Yingke Chen, Hua Mao, Manfred Jaeger, ThomasDyhre Nielsen, Kim Guldstrand Larsen, and Brian Nielsen. Learning markov models for stationary system behaviors. In *NASA Formal Methods*, Lecture Notes in Computer Science. Springer, 2012.

[59] C. Baier and J. P. Katoen. *Principles of Model Checking*. The MIT Press, 2008.

[60] Probabilistic symbolic model checker PRISM. "hXXp://www.prismmodelchecker[.]org/".

[61] Yices: An SMT solver. "hXXp://yices.csl.sri[.]com/".

[62] Smart grid lab. "hXXp://epic.uncc[.]edu/facilities/duke-energy-smart-grid-laboratory.html".

[63] M. Kwiatkowska and D. Parker. Advances in probabilistic model checking. In *Proceedings 2011 Marktoberdorf Summer School: Tools for Analysis and Verification of Software Safety and Security*, 2012.

[64] HPROF: A heap/CPU profiling tool. "hXXp://docs.oracle[.]com/javase/7/docs/technotes/samples/hprof.html".

[65] Ambient communication nodes. "hXXp://www.ambientcorp[.]com/prod-nodes/".

[66] Echelon data concentrator. "hXXp://www.echelon[.]com/products/controllers/meter-data-concentrator/default.htm".

[67] M. Hayden, C. Hebert, and S. Tierney. Cybersecurity and the north american electric grid: New policy approaches to address an evolving threat. *Bipartisan Policy Center*, 2014.

[68] S. Sridhar and G. Manimaran. Data integrity attacks and their impacts on scada control system. In *Power and Energy Society General Meeting, 2010 IEEE*, 2010.

[69] Y. Liu, P. Ning, and M.K. Reiter. False data injection attacks against state estimation in electric power grids. *ACM Transactions on Information and System Security*, 2011.

[70] M. Q. Ali, R. Yousefian, E. Al-Shaer, S. Kamalasadan, and Q. Zhu. Two-tier data-driven intrusion detection for automatic generation control in smart grid. In *Communications and Network Security (CNS), 2014 IEEE Conference on*, pages 292–300, Oct 2014.

[71] Daniel J Trudnowski, Warren L McReynolds, and Jeffery M Johnson. Real-time very short-term load prediction for power-system automatic generation control. *Control Systems Technology, IEEE Transactions on*, 2001.

[72] Shu Fan and Luonan Chen. Short-term load forecasting based on an adaptive hybrid method. *Power Systems, IEEE Transactions on*, 2006.

[73] P Mohajerin Esfahani, Maria Vrakopoulou, Kostas Margellos, John Lygeros, and Göran Andersson. Cyber attack in a two-area power system: Impact identification using reachability. In *American Control Conference (ACC), 2010*, pages 962–967. IEEE, 2010.

[74] Peyman Mohajerin Esfahani, Maria Vrakopoulou, Kostas Margellos, John Lygeros, and Göran Andersson. A robust policy for automatic generation control cyber attack in two area power network. In *Decision and Control (CDC), 2010 49th IEEE Conference on*, pages 5973–5978. IEEE, 2010.

[75] Yee Wei Law, Tansu Alpcan, Marimuthu Palaniswami, and Subhrakanti Dey. Security games and risk minimization for automatic generation control in smart grid. In *Decision and Game Theory for Security*, pages 281–295. Springer, 2012.

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

[77] Mustafa Amir Faisal, Zeyar Aung, John R Williams, and Abel Sanchez. Data-stream-based intrusion detection system for advanced metering infrastructure in smart grid: A feasibility study. *IEEE Systems Journal*, 9(1):31–44, 2015.

[78] Divya M Menon and N Radhika. Anomaly detection in smart grid traffic data for home area network. In *2016 International Conference on Circuit, Power and Computing Technologies (ICCPCT)*, pages 1–4. IEEE, 2016.

[79] Fadwa Abdul Aziz Alseiari and Zeyar Aung. Real-time anomaly-based distributed intrusion detection systems for advanced metering infrastructure utilizing stream data mining. In *2015 International Conference on Smart Grid and Clean Energy Technologies (ICSGCE)*, pages 148–153. IEEE, 2015.

[80] John Bigham, David Gamez, and Ning Lu. Safeguarding SCADA systems with anomaly detection. In *Springer Lecture Notes in Computer Science*, 2003.

[81] Weiyu Xu, Meng Wang, and Ao Tang. On state estimation with bad data detection. In *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on*, 2011.

[82] P. Kundur. *Power System Stability and Control*. McGraw-Hill, 1994.

[83] Muhammad Qasim Ali, Ehab Al-Shaer, Hassan Khan, and Syed Ali Khayam. Automated anomaly detector adaptation using adaptive threshold tuning. *ACM Transactions on Information and System Security (TISSEC)*, 2013.

[84] H. L. V. Trees. *Detection, estimation and modulation theory: part I*. Wiley-Interscience, 2001.

[85] Power system simulation tool. `"hXXps://hvdc[.]ca/pscad/"`.

[86] Meysam Doostizadeh and Hassan Ghasemi. Day-ahead scheduling of an active distribution network considering energy and reserve markets. *European Transactions on Electrical Power*, 2012.

[87] Mark Handley, Vern Paxson, and Christian Kreibich. Network intrusion detection: Evasion, traffic normalization, and end-to-end protocol semantics. In *USENIX Security Symposium*, pages 115–131, 2001.

[88] C. Kruegel. Full system emulation: Achieving successful automated dynamic analysis of evasive malware. In *BlackHat USA Security Conference*, 2014.

[89] C. Smith, A. Matrawy, S. Chow, and B. Abdelaziz. Computer worms: Architectures, evasion strategies, and detection mechanisms. In *Journal of Information Assurance and Security*, 2008.

[90] Thomas H Ptacek and Timothy N Newsham. Insertion, evasion, and denial of service: Eluding network intrusion detection. Technical report, DTIC Document, 1998.

[91] M. Q. Ali, A. B. Ashfaq, E. Al-Shaer, and Q. Duan. Towards a science of anomaly detection system evasion. In *Communications and Network Security (CNS), 2015 IEEE Conference on*, pages 460–468, Sept 2015.

[92] A. El-Atawy, T. Samak, E. Al-Shaer, and H. Li. Using online traffic statistical matching for optimizing packet filtering performance. In *IEEE INFOCOM 2007 - 26th IEEE International Conference on Computer Communications*, pages 866–874, May 2007.

[93] H. Hamed, A. El-Atawy, and E. Al-Shaer. Adaptive statistical optimization techniques for firewall packet filtering. In *Proceedings IEEE INFOCOM 2006. 25TH IEEE International Conference on Computer Communications*, pages 1–12, April 2006.

[94] H. Hamed, A. El-Atawy, and E. Al-Shaer. On dynamic optimization of packet matching in high-speed firewalls. *IEEE Journal on Selected Areas in Communications*, 24(10):1817–1830, Oct 2006.

[95] Hazem Hamed and Ehab Al-Shaer. On autonomic optimization of firewall policy configuration. *Journal of High Speed Networks, Special issue on Security Policy Management*, 13(3):209–227, Aug 2006.

[96] Adel El-Atawy, Ehab Al-Shaer, Tung Tran, and Raouf Boutaba. Adaptive early packet filtering for defending firewalls against dos attacks. In *INFOCOM*, 2009.

[97] Ehab Al-Shaer and Mohammad Ashiqur Rahman. *Security and Resiliency Analytics for Smart Grids*. Springer International Publishing, 2016.

[98] David J Chaboya, Richard A Raines, Rusty O Baldwin, and Barry E Mullins. Network intrusion detection: automated and manual methods prone to attack and evasion. *IEEE Security and Privacy*, 4(6):36–43, 2006.

[99] Tsung-Huan Cheng, Ying-Dar Lin, Yuan-Cheng Lai, and Po-Ching Lin. Evasion techniques: Sneaking through your intrusion detection/prevention systems. *IEEE Communications Surveys & Tutorials*, 14(4):1011–1020, 2012.

[100] Prahlad Fogla and Wenke Lee. Evading network anomaly detection systems: formal reasoning and practical techniques. In *Proceedings of the 13th ACM conference on Computer and communications security*, pages 59–68. ACM, 2006.

[101] Igino Corona, Giorgio Giacinto, and Fabio Roli. Adversarial attacks against intrusion detection systems: Taxonomy, solutions and open issues. *Information Sciences*, 239:201–225, 2013.

[102] D. Wagner and P. Soto. Mimicry attacks on host-based intrusion detection systems. In *ACM Conference on Computer and Communications Security (CCS)*, 2002.

[103] Christopher Kruegel, Engin Kirda, Darren Mutz, William Robertson, and Giovanni Vigna. Automating mimicry attacks using static binary analysis. In *Proceedings of the 14th conference on USENIX Security Symposium*, pages 11–11. USENIX Association, 2005.

[104] Chetan Parampalli, R Sekar, and Rob Johnson. A practical mimicry attack against powerful system-call monitors. In *Proceedings of the 2008 ACM symposium on Information, computer and communications security*, pages 156–167. ACM, 2008.

[105] Weiqin Ma, Pu Duan, Sanmin Liu, Guofei Gu, and Jyh-Charn Liu. Shadow attacks: automatically evading system-call-behavior based malware detection. *Journal in Computer Virology*, 8(1-2):1–13, 2012.

[106] Zhenyu Wu, Steven Gianvecchio, Mengjun Xie, and Haining Wang. Mimimorphism: A new approach to binary code obfuscation. In *Proceedings of the 17th ACM conference on Computer and communications security*, pages 536–546. ACM, 2010.

[107] Hilmi Günes Kayacik and A Nur Zincir-Heywood. Mimicry attacks demystified: What can attackers do to evade detection? In *Privacy, Security and Trust, 2008. PST'08. Sixth Annual Conference on*, pages 213–223. IEEE, 2008.

[108] Yves Younan, Wouter Joosen, and Frank Piessens. Runtime countermeasures for code injection attacks against C and C++ programs. *ACM Computing Surveys (CSUR)*, 44(3):17, 2012.

[109] Clemens Kolbitsch, Paolo Milani Comparetti, Christopher Kruegel, Engin Kirda, Xiao-yong Zhou, and XiaoFeng Wang. Effective and efficient malware detection at the end host. In *USENIX security symposium*, pages 351–366, 2009.

[110] Darren Mutz, William Robertson, Giovanni Vigna, and Richard Kemmerer. Exploiting execution context for the detection of anomalous system calls. In *International Workshop on Recent Advances in Intrusion Detection*, pages 1–20. Springer, 2007.

[111] Manuel Costa, Jon Crowcroft, Miguel Castro, Antony Rowstron, Lidong Zhou, Lintao Zhang, and Paul Barham. Vigilante: End-to-end containment of internet worm epidemics. *ACM Transactions on Computer Systems (TOCS)*, 26(4):9, 2008.

[112] Xuxian Jiang, Helen J Wangz, Dongyan Xu, and Yi-Min Wang. Randsys: Thwarting code injection attacks with system service interface randomization. In *Reliable Distributed Systems, 2007. SRDS 2007. 26th IEEE International Symposium on*, pages 209–218. IEEE, 2007.

[113] Quanyan Zhu, C.J. Fung, R. Boutaba, and T. Basar. A distributed sequential algorithm for collaborative intrusion detection networks. In *2010 IEEE International Conference on Communications (ICC)*, 2010.

[114] S. Jha, K. Tan, and R.A. Maxion. Markov chains, classifiers, and intrusion detection. In *Computer Security Foundations Workshop, 2001. Proceedings. 14th IEEE*, 2001.

[115] Stefano Zanero and Sergio M. Savaresi. Unsupervised learning techniques for an intrusion detection system. In *Proceedings of the 2004 ACM Symposium on Applied Computing*, SAC '04, 2004.

[116] Quanyan Zhu, H. Tembine, and T. Başar. Distributed strategic learning with application to network security. In *American Control Conference (ACC), San Francisco, CA*, 2011.

[117] Q. Zhu and T. Başar. Dynamic policy-based ids configuration. In *48th IEEE Conference on Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference (CDC/CCC 2009)*, 2009.

[118] M. Manshaei, Q. Zhu, T. Alpcan, T. Başar, and J.-P. Hubaux. Game theory meets network security and privacy. *ACM Computing Survey*, 2013.

[119] A. B. Ashfaq, M. Joseph, A. Mumtaz, M. Q. Ali, A. Sajjad, and S. A. Khayam. A comparative evaluation of anomaly detectors under portscan attacks. In *Proceedings of the 11th International Symposium on Recent Advances in Intrusion Detection (RAID '08)*, pages 351–371, Berlin, Heidelberg, 2008. Springer-Verlag.

[120] Symantec security. `"hXXp://securityresponse.symantec[.]com/avcenter"`.

[121] C. Shannon and D. Moore. The spread of the witty worm. *IEEE Security and Privacy (SP '04)*, 2004.

[122] R. Pang, M. Allman, M. Bennett, J. Lee, V. Paxson, and B. Tierney. A first look at modern enterprise traffic. In *Proceedings of the 5th ACM SIGCOMM Conference on Internet Measurement (IMC '05)*, pages 2–2, Berkeley, CA, USA, 2005. USENIX Association.

[123] A.B. Ashfaq, M.Q. Ali, E. Al-Shaer, and S.A. Khayam. Revisiting anomaly detection system design philosophy. In *ACM SIGSAC Conference on Computer & Communications Security*, 2013.

[124] A. B. Ashfaq, S. Rizvi, M. Javed, S. A. Khayam, M. Q. Ali, and E. Al-Shaer. Information theoretic feature space slicing for statistical anomaly detection. *Journal of Network and Computer Applications*, 2014.