AUTOMATED FORMAL ANALYTICS FOR SMART GRID SECURITY AND RESILIENCY

by

Mohammad Ashiqur Rahman

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

2015

ABSTRACT

MOHAMMAD ASHIQUR RAHMAN. Automated formal analytics for smart grid security and resiliency. (Under the direction of DR. EHAB AL-SHAER)

Smart grid is the modernization of the legacy electric power system in which cyber computing and communications are integrated with the physical world of power systems. A smart grid provides efficient and cost-effective management of the grid by allowing real-time monitoring, coordinating, and controlling of the system using communication networks among physical components. A smart grid exhibits complex configurations due to the coexistence of legacy systems with the modern technologies and the interdependency between different cyber and physical components. This inherent complexity significantly increases the vulnerabilities and attack surface in smart grids due to misconfigurations and the lack of security hardening. In a critical infrastructure like smart grid, a security breach can cause devastating damages. Thus, there is a need for formal security analytics to automatically verify smart grid security, provably identify potential attacks and their impacts, and devise cost-effective mitigation plans in a proactive manner. In this dissertation, we focus on achieving these goals through the following three research thrusts.

First, we develop a model to formally verify the compliance of the advanced metering infrastructure (AMI) configurations with the security requirements, and generate remediation plans for potential security violations. The development of this model includes formal modeling of AMI configurations and security control requirements based on NIST guidelines. We develop a similar formal model for verifying the supervisory control and data acquisition (SCADA) security. We primarily verify trusted communications between field devices and the control center along with different resiliency constraints to ensure whether SCADA operations remain reliable in contingencies.

Second, we formally model a framework to automatically synthesize cost-effective, network isolation-based resiliency architecture for the cyber systems in smart grids. The framework considers isolation requirements and usability and cost constraints, and synthesizes resiliency configurations by exploring various isolation patterns for network traffic flows. We devise a hypothesis testing-based refinement mechanism that systematically finds an optimal resilient architecture by investigating alternative designs. We provide another framework to synthesize redundancy-based resilient configurations for AMI, considering operational integrity and robustness requirements.

Third, we develop a formal model for analyzing attack evasions on state estimation, a core control module of SCADA. The model identifies attack vectors (*e.g.*, a set of measurements to be altered) for compromising state estimation, considering (i) a comprehensive set of attack attributes, including access capability, attack resources, and knowledge, (ii) potential attack evasions, and (iii) the interdependency between state estimation and other control modules. The modeling of interdependency enables provable identification of novel stealthy attacks and their escalation to impact on the economic operation of smart grids. For risk mitigation, we provide two techniques: (i) we synthesize proactive security plans that recommend a minimal data integrity of measurements in order to make such attacks infeasible, and (ii) we increase the robustness of the control system against persistent attacks by frequently randomizing state estimation parameters that are critical for launching such attacks.

ACKNOWLEDGMENTS

First I express my heartiest thanks and gratefulness to Almighty God for His divine blessings, which made it possible for me to complete this dissertation successfully.

I feel grateful to and wish to acknowledge my profound indebtedness to Dr. Ehab Al-Shaer, Department of Software and Information Systems, College of Computing and Informatics, University of North Carolina at Charlotte. The deep knowledge and keen interest of Professor Ehab in the field of network and information security and formal analytics influenced me to carry out this research. His endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, and valuable advice at all stage have made it possible to complete this dissertation.

I would like to express my heartiest gratitude Drs. Badrul Chowdhury, Bill Chu, Anita Raja, and Matthew Whelan, University of North Carolina at Charlotte, for serving on my dissertation committee and helping me to pursue my doctoral degree with their precious suggestions. I would like to thank the faculty members and the staff of the Department of Software and Information Systems, University of North Carolina at Charlotte, for their cooperation throughout my doctoral study. I would also like to express my appreciation to Dr. Rajesh Kavasseri, North Dakota State University, and Dr. Rakesh Bobba, Oregon State University, for sharing their deep thoughts on various problems that have been solved in this dissertation.

I must acknowledge the great support and sacrifice that my dearest father and mother have carried out for completing my doctoral degree successfully. I should also admit the constant support and endurance of my loving elder sister, younger sister, brother, nephews, and nieces during my doctoral study. I would like to thank my father-in-law and mother-in-law for their encouragement to achieve my degree. I am grateful to my colleagues in the lab for creating an excellent friendly environment that has allowed me to enjoy the time in the lab and feel motivated and energetic to work. I am also grateful to my friends outside the lab, particularly Farok, Bipul, and Arafat, for their great support toward achieving my doctoral degree. I would also like to thank Susan for her great support while proofreading the dissertation.

Finally, I would like to express my earnest appreciation to my dearly loved wife Nadia who spent sleepless nights with and was always a great support to me, particularly in the moments when there was no one to help me out.

TABLE OF CONTENTS

LIST OF FIGURES xii

LIST OF TABLES xvi

CHAPTER 1: INTRODUCTION 1

1.1. Background 3

1.1.1. Smart Grid Overview 3

1.1.2. AMI 5

1.1.3. SCADA 7

1.2. Motivation: Potential Threats and Impact 9

1.3. Work Objectives 11

1.4. Research Challenges 12

1.5. Contributions 14

1.6. Technical Approach Overview 16

1.6.1. Security Verification for AMI and SCADA 17

1.6.2. Automated Synthesis of Resiliency Architecture 19

1.6.3. Threat Analytics and Security Hardening for Power System State Estimation 21

1.6.4. SMT Overview 23

1.7. Organization 24

CHAPTER 2: SECURITY VERIFICATION FOR AMI and SCADA 26

2.1. Challenges 27

2.1.1. AMI and SCADA Complexity 27

2.1.2. Potential Cyber Threats on AMI and SCADA 31

2.2. Related Work 33

2.3. Contributions 35

2.4. Overview of the Security Analysis Framework 36

2.5. AMI Security Analysis 38

2.5.1. Preliminary 38

2.5.2. Formal Model of AMI Security Verification 39

2.5.3. Implementation 50

2.6. SCADA Security Analysis 56

2.6.1. Preliminary 57

2.6.2. Formal Model of SCADA Security Verification 58

2.6.3. Implementation 70

2.7. Evaluation 73

2.7.1. Accuracy 73

2.7.2. Scalability Evaluation 74

2.7.3. Discussion 78

2.8. Conclusion 79

CHAPTER 3: AUTOMATED SYNTHESIS OF RESILIENCY ARCHITECTURE 80

3.1. Challenges 81

3.2. Related Work 82

3.2.1. Security Verification and Hardening 82

3.2.2. Resiliency of Smart Grids 84

3.3. Contributions 85

3.4. Overview of the Configuration Synthesis Framework 86

3.5. Resiliency Architecture Synthesis Based on Isolation 88

3.5.1. Synthesis Framework 88

3.5.2. Formal Model of Synthesis Framework 90

3.5.3. An Example Case Study 105

3.5.4. Resiliency Architecture Refinement 107

3.5.5. Evaluation 114

3.6. Resiliency Architecture Synthesis Based on Redundancy 124

3.6.1. Synthesis Framework 124

3.6.2. Formal Model of the Synthesis Framework 126

3.6.3. An Example Case Study 139

3.6.4. Evaluation 142

3.7. Conclusion 147

CHAPTER 4: THREAT ANALYTICS AND SECURITY HARDENING FOR POWER SYSTEM STATE ESTIMATION 149

4.1. Background 151

4.1.1. DC Power Flow Model 151

4.1.2. State Estimation 152

4.1.3. Topology Processor 153

4.1.4. Optimal Power Flow 153

4.2. Challenges 154

4.2.1. UFDI Attack 154

4.2.2. Attack Model 155

4.3. Related Work 156

4.4. Contributions 159

4.5. Attack Vector Verification 160

4.5.1. Preliminaries 160

4.5.2. Parameters for Modeling UFDI Attack 162

4.5.3. Parameters for Modeling Topology Poisoning 163

4.5.4. Formalization of Change in State Estimation 164

4.5.5. Formalization of Topology Change 165

4.5.6. Formalization of False Data Injection to Measurements 166

4.5.7. Formalization of Attack Attributes 167

4.5.8. An Example Case Study 170

4.6. Attack Impact Verification 174

4.6.1. Framework 174

4.6.2. Formalization of Impact of UFDI Attacks on OPF 176

4.6.3. An Example Case Study 180

4.7. Security Hardening Against Stealthy Attacks 182

4.7.1. Preface 183

4.7.2. Synthesis Design 184

4.7.3. Formalization of Candidate Architecture Selection 185

4.7.4. An Example Case Study 186

4.8. Proactive Defense Against Persistent Attacks 190

4.8.1. Preface 190

4.8.2. Moving Target Defense Strategy 191

4.8.3. Formal Model for Strategy Selection 194

4.8.4. An Example Case Study 198

4.9. Evaluation 202

4.9.1. Methodology 203

4.9.2. Time Complexity of Verification Model 203

4.9.3. Time Complexity of Impact Analysis 206

4.9.4. Time Complexity of Synthesis Mechanism 209

4.9.5. Time Complexity of MTD Strategy Selection Models 212

4.9.6. Memory Complexity 212

4.10. Conclusion 213

CHAPTER 5: SUMMARY AND FUTURE WORK 215

5.1. Overview of Contributions, Technical Approaches, and Evaluation Results 215

5.1.1. Security Verification for AMI and SCADA 215

5.1.2. Automated Synthesis of Resiliency Architecture 216

5.1.3. Threat Verification and Security Hardening for Power System State Estimation 217

5.2. Future Research 218

REFERENCES 221

LIST OF FIGURES

FIGURE 1: A conceptual model of the smart grid architecture. 4

FIGURE 2: A detailed architecture of a smart grid, which shows the end-to-end power transmission and distribution, AMI and SCADA communication infrastructures, and the utility and control centers. 5

FIGURE 3: A typical AMI network. 27

FIGURE 4: The push-pull mode of data delivery in smart grids. 29

FIGURE 5: A typical SCADA network. 30

FIGURE 6: Root sources of smart grid security threats. 32

FIGURE 7: The architecture of SmartAnalyzer. 36

FIGURE 8: An example data of AMI topology configurations. 38

FIGURE 9: An example data of SCADA configurations. 57

FIGURE 10: An example SCADA topology of a 5-bus power grid. 71

FIGURE 11: Impact of network size on (a) invariant and (b) user-driven constraints verification time. 75

FIGURE 12: Impact of (a) the zone size and (b) the number of collector classes per zone on constraint verification time. 75

FIGURE 13: Impact of (a) the network size and (b) the number of classes on memory requirement. 76

FIGURE 14: Impact of the AMI network size on the analysis time and the memory requirement for cyber bandwidth constraint verification, which show that the analysis of this constraint may fail when the number of collectors increases more than some certain labels. 77

FIGURE 15: The configuration synthesis (ConfigSynth) framework. 87

FIGURE 16: The isolation-based resiliency architecture synthesis framework. 88

FIGURE 17: (a) An example network for synthesizing resiliency configurations and corresponding security device placements. An ID is assigned to each of the hosts, routers, and links. (b) The solution to the example problem, *i.e.*, the placements of necessary security devices. 105

FIGURE 18: The extended architecture of ConfigSynth for the design refinement. 109

FIGURE 19: The placements of necessary security devices after the refinement. 113

FIGURE 20: (a) The maximum possible isolation with respect to the usability constraint considering a fixed cost constraint (*i.e.*, $200K) and (b) the maximum possible isolation with respect to the deployment cost constraint considering a fixed usability constraint (*i.e.*, 5). 115

FIGURE 21: The model synthesis time w.r.t. (a) the number of hosts and (b) the number of routers. 118

FIGURE 22: The impact of (a) the isolation constraint and (b) the deployment cost constraint on the model synthesis time. 119

FIGURE 23: A comparison between the optimized isolation provided by the refinement mechanism and the optimum isolation with respect to (a) various deployment cost and (b) various usability constraints. 121

FIGURE 24: The resiliency architecture refinement mechanism execution time with respect to: (a) the number of hosts and (b) the number of routers. 122

FIGURE 25: The AMI resiliency configurations synthesis framework. 125

FIGURE 26: The execution of the synthesis framework with respect to (a) the number of meters and (b) the average size of each zone. 144

FIGURE 27: Impact of the budget constraint on the execution time, (a) satisfiable cases and (b) unsatisfiable cases. 145

FIGURE 28: Energy control center system security schematic (thanks to Allen J. Wood and Bruce F. Wollenberg, *Power Generation, Operation, and Control*, 2nd Edition [1]). 150

FIGURE 29: The diagram of the IEEE 14-bus test system. Red circles are used for bus numbers, green squares are for transmission line numbers, and round cornered blue squares are for measurement numbers. 170

FIGURE 30: The framework for finding the impact of UFDI attacks on OPF. 175

FIGURE 31: The flow diagram of the security architecture synthesis mechanism for protecting state estimation attack. 184

FIGURE 32: The security architectures (the green squared buses needs to be secured) in different scenarios: (a) incomplete information (the red circled line's admittance is unknown), (b) complete knowledge, and (c) incorporating with topology poisoning attack (the read lines are potential to inclusion or exclusion topology attacks). In all scenarios, bus 1 is the reference bus. 188

FIGURE 33: The moving target defense (MTD) mechanism for hardening the security of state estimation. 194

FIGURE 34: The probability of attack success in the cases of different access capabilities: (a) measurement-based MTD strategy and (b) measurement and line admittance-based MTD strategy. 200

FIGURE 35: The attackability in the cases of different levels of knowledge about line admittances: (a) measurement-based MTD strategy and (b) measurement and line admittance-based MTD strategy. 202

FIGURE 36: The verification model execution time in different experiments: (a) the execution time with respect to the number of buses, (b) the execution time with respect to the number of recorded measurements, (c) the execution time with respect to the attacker's resource limit, and (d) the execution time in unsatisfiable cases with respect to the number of buses. 204

FIGURE 37: The execution time of Impact verification on OPF with respect to the number of buses in (a) UFDI attacks including topology attacks, (b) UFDI attacks without topology attacks, (c) topology attacks without infecting states, and (d) unsatisfiable cases. 208

FIGURE 38: The execution time of the security architecture synthesis mechanism in different experiments: (a) the execution time with respect to the number of buses, (b) the execution time with respect to the number of taken measurements, (c) the execution time with respect to the attacker's resource limit, and (d) the execution time in unsatisfiable cases with respect to the number of buses. 210

FIGURE 39: These two graphs shows the model execution time with respect to the number of buses: (a) the execution time of the observability model and (b) the execution time of the line admittance randomization model. 212

LIST OF TABLES

TABLE 1: Formal definition of a meter and a collector 39

TABLE 2: Modeling of a zone and its relation with AMI devices 43

TABLE 3: Formalizations of reachability and pairing constraints 45

TABLE 4: Formalizations of schedule and resource constraints 47

TABLE 5: Formalizations of different user-driven constraints 48

TABLE 6: An example of resource constraint verification 53

TABLE 7: An example of the diagnosis process 54

TABLE 8: Formal (partial) definition of a remote terminal unit 58

TABLE 9: Formalizations of the secured data delivery constraint for SCADA (partial) 63

TABLE 10: The input for an example of verifying secured observability in a SCADA 72

TABLE 11: Network level isolation patterns 93

TABLE 12: Security devices 94

TABLE 13: Input (partial) of the example 107

TABLE 14: Output of the example: selected isolation patterns for the flows 108

TABLE 15: Selected isolation patterns for the flows in the example corresponding to the refinement mechanism 114

TABLE 16: The memory requirement (MB) w.r.t. problem size 120

TABLE 17: Input of the example in Section 3.6.3.2 140

TABLE 18: Output of the example: meters' configurations 141

TABLE 19: Output of the example: collectors' configurations 142

TABLE 20: Memory requirements (in MB) w.r.t. the problem size 146

TABLE 21: Modeling parameters 160

TABLE 22: Line information of the example in Section 4.5.8 171

TABLE 23: Measurement info of the example in Section 4.5.8 172

TABLE 24: Input of the example in Section 4.6.3 180

TABLE 25: The required memory space (in MB) 213

CHAPTER 1: INTRODUCTION

Driven by the rapid advancement of technology and the growing need of business requirements, cyber communications are embedded in many physical systems. The integration of cyber and physical capabilities leads to the creation of many applications with enormous societal impact and economic benefit. The emerging systems that connect the cyber-world of computing and communications with the physical world are cyber-physical systems (CPS). Operations are monitored, analyzed, and controlled in CPS using cyber systems that interconnect physical components. Many CPS are defined as critical infrastructures due to their national importance. According to the U.S. Department of Homeland Security, "Critical infrastructures are the assets, systems, or networks, whose incapacitation or destruction would have a debilitating effect on security, national economic security, national public health or safety" [2]. Any damage or unavailability of such a critical infrastructure often has a massive and broader impact.

A smart grid is a typical example of a critical cyber-physical system. Smart grids are the modernization of the legacy power systems with the development of a communication infrastructure. To delineate the importance of the safety and reliability of smart grids, we can cite the following from a Schneider Electric report [3]: "The financial impact of power disruption was demonstrated during the August 2003 blackout, which affected 45 million people in eight US states and 10 million people in parts of Canada. Healthcare facilities experienced hundreds of millions of dollars in lost revenue from cancelled services, legal liability, and damaged reputations. Six hospitals were in bankruptcy one year later." This incident clearly illustrates the extent of the impact due to operational interruption in energy networks.

Since communication infrastructures are integrated with the legacy systems, a smart grid exhibits a highly complex configuration comprised of heterogeneous cyber-physical components. This complexity leads to various vulnerabilities and potential security threats. An execution of such a threat can easily cause devastating damages to the grid. Therefore, there is a great need for security analytics to (i) verify the compliance of smart grid configurations with security standards (provided by NIST, NERC, and FERC [4, 5, 6]), (ii) identify potential threats on smart grids, and (iii) design necessary defense mechanisms. This need for smart grid security is reflected by three major security challenges: (i) proactive identification of potential security threats, (ii) resiliency architecture to support the continuity of smart grid operations, and (iii) automated security analytics that offer efficient verification and synthesis of smart grid security properties.

In this dissertation, we address these challenges in three main research thrusts, which will be discussed in detail in the subsequent chapters. We perform the proactive identification of security threats by developing formal security verification techniques that discover cyber and physical security threats on different components of smart grids, including advanced metering infrastructure (AMI), supervisory control and data acquisition (SCADA), and state estimation. We develop formal models that synthesize cost-effective resiliency architectures for smart grids. We focus on cyber systems, AMI, and state estimation in smart grids. A resiliency architecture primarily includes countermeasures (*e.g.*, isolation, data integrity, and redundancy), as well as physical and security devices' configurations and their placements. We also increase the resiliency of the system, particularly for state estimation, by frequently randomizing its critical parameters. We address security automation for each of our research goals, including automated verification of security properties and synthesis of resiliency architecture.

In this chapter, we first describe a brief overview of smart grids. We explore the potential attacks on smart grids and discuss the research challenges. Next, we present our research objectives and contributions, along with the technical approach that we take in this dissertation. Finally, we present the organization of the remainder of the dissertation.

1.1 Background

1.1.1 Smart Grid Overview

Smart grids are modern power systems that represent a perfect example of cyber-physical systems. In the last decade, the paradigm of power grid infrastructures has been shifted to a new age. Legacy infrastructures are being replaced with state-of-the-art smart grids. Leading utility providers in the U.S. have taken different initiatives for deploying smart devices by replacing the existing legacy systems [7]. Smart grids provide intelligent devices with two-way communications among them, which allow efficient management of the power while providing useful features. The basic reason for moving to smart grids is to provide uninterrupted low-cost energy

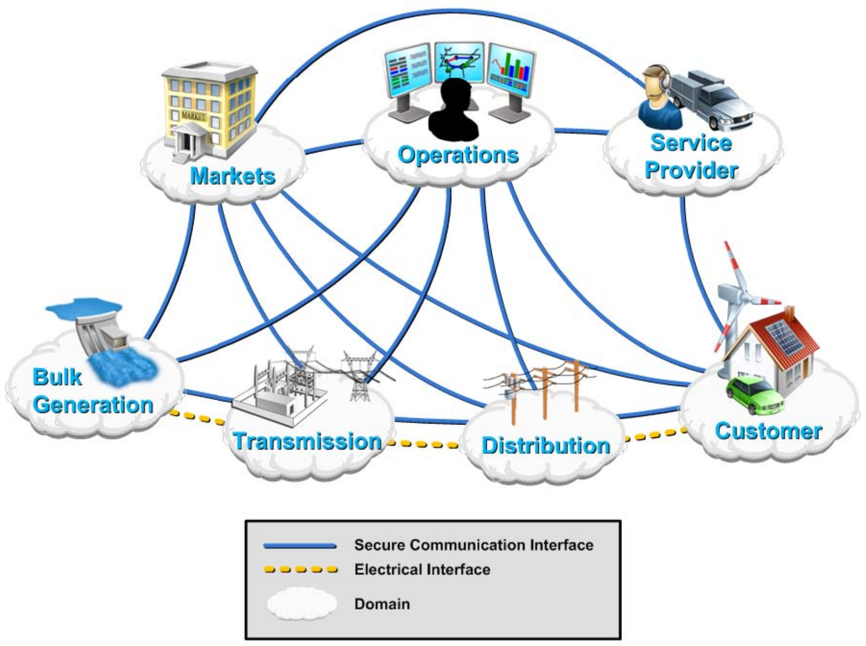


Figure 1: A conceptual model of the smart grid architecture.

considering 21st century demands. However, smart grids usually exhibit a complex infrastructure as they consist of numerous cyber and physical smart devices along with legacy equipment, wherein a device often needs to communicate with other devices. Smart grids are designated as critical infrastructures. Thus, they require very rigid security considerations.

We present a basic conceptual architecture of a smart grid [4] in Figure 1, which shows different components of smart grids and their inter-connections. A detailed diagram of a smart grid is presented in Figure 2, which shows the power generation, transmission, and distribution. The figure also illustrates the major measuring and controlling systems of the grid, namely AMI and SCADA, including the communication infrastructures and the control and utility centers.

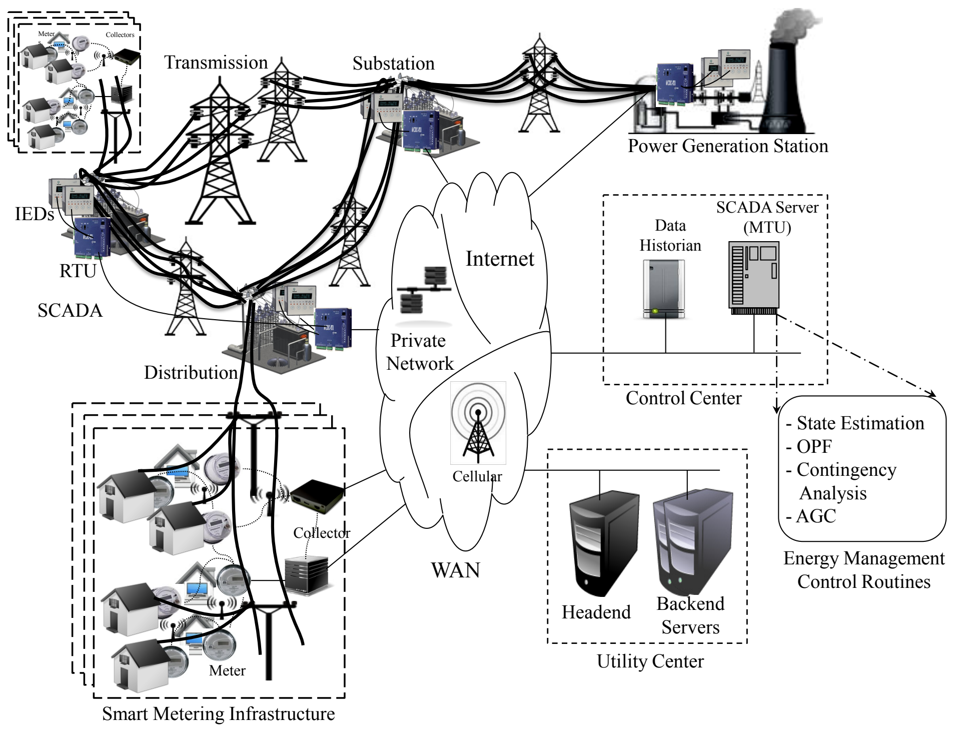


Figure 2: A detailed architecture of a smart grid, which shows the end-to-end power transmission and distribution, AMI and SCADA communication infrastructures, and the utility and control centers.

1.1.2 AMI

Advanced Metering Infrastructure (AMI) is one of the core components of a smart grid. This system measures, collects, and analyzes energy usage of electricity customers. A typical AMI consists of a large number of smart meters and data collectors. Data collectors communicate with smart meters either on request or on a schedule, collect data, and send the report to the utility through the backhaul communication system. In fact, AMI is an extended version of the existing automatic meter reading (AMR) technology by the addition of two-way communication capability, allowing commands to be sent toward the meters for different purposes like demand-response actions or remote service disconnects [8, 9]. The utility center is mainly responsible for billing the energy users for their usage. The scope of the utility center is broadening with the implementation of AMI. Different demand-response utility services are being established in order to efficiently manage the electricity usage by reducing the energy cost as well as the energy loss/theft.

A structure of the AMI network can be found in Figure 2. Usually a meter establishes an authenticated connection with a specific collector to report energy usage data. A collector forwards this data to the utility center over a trusted path. The server at the utility center that receives this data is often named as the headend system. The control commands from the headend to the meters are also transmitted through secure connections. A meter is connected to a collector directly or through another meter. In the latter case, geographically collocated meters form a mesh network. Heterogeneous communication links or communication systems are seen in AMI. For example, there are power lines, wi-fi, ethernet, optical fibers, cellular or serial communication, and proprietary or third-party communication infrastructures. The collectors are usually connected with the utility network through a wide area network (WAN). There are different delivery modes, push or pull, periodic or non-periodic, that are followed by the AMI devices to communicate to each other. There is also a provision to allow an energy user's home area network (HAN) to be connected with the AMI network through the smart meters. There are also different proprietary communication protocols used in AMI other than TCP/IP. LonTalk is an example of such protocols which is often used between a meter and a collector. An AMI network also involves varieties of data stream types like energy usage data, control commands, and software patches.

1.1.3 SCADA

In the energy transmission and distribution side of the smart grid, different communication networks exist for sensing measurements and transmitting control commands. These networks are associated with the SCADA system. SCADA is the major industrial control system (ICS) in smart grids, and connects the generating stations, substations, and control centers. SCADA is mainly responsible for monitoring and controlling the remote equipment by obtaining data from the remote devices, analyzing the received data at the control centers, and executing necessary control commands at the remote devices. We can notice in Figures 1 and 2 that the control centers associated with the generation, transmission, and distribution systems are connected to the physical power system using cyber infrastructure.

There are various kinds of SCADA devices, such as SCADA servers or master terminal units (MTUs), human machine interfaces (HMIs), data historians, and different field devices. Remote terminal units (RTUs), programmable logic controllers (PLCs), and intelligent electronic devices (IEDs) are the typical field devices. IEDs often receive data from sensors, while they actuate the control commands received from the SCADA control server. IEDs and PLCs are also designed to take pre-specified actions automatically when the designated situations occur. RTUs are mainly data collecting and forwarding devices. Similar to AMI, there are different communication links in SCADA. The modem-based serial communication is often seen in SCADA for the communication between RTUs and MTUs. The SCADA communication is done using industrial protocols like Modbus, DNP3, or IEC 61850, which are layered protocol similar to TCP/IP. Some variants of these protocols are built on top of the TCP/IP communication architecture.

In addition to legacy SCADA devices, smart grids are currently being equipped with phasor measurement units (PMUs), also known as synchrophasors, which are capable of measuring both voltage and current phasor measurements with a precisely synchronized timestamp. PMUs are often connected with a global positioning system (GPS), which allows them to be synchronized with microsecond accuracy. There are phasor data concentrators (PDCs) that collect the phasor measurements from connected PMUs, sort the measurements according to the GPS timestamp, and provide this data to the control server.

The control server takes the sensor measurements from field devices through the power network and sends the control commands to them after analyzing the data using the same infrastructure. There are different control modules/routines to manage the grid efficiently and reliably. These modules are specified together as the energy management system (EMS). As we see in Figure 2, state estimation, optimal power flow (OPF), contingency analysis, and automatic generation control (AGC) are the main control modules. These modules are interdependent with each other, and one's outputs are often used as inputs for others. The state estimation is the core component of EMS. Its function is to compute the unknown state variables of the power system from the sensor measurements received through the SCADA system. The output of state estimation is used in OPF and contingency analysis. The objective of the OPF process is to minimize the total generation cost satisfying the load and the operating constraints (*e.g.*, capacities of the transmission lines). Contingency analysis is important for maintaining the system's security in contingencies, such as generation unit failures and transmission line breakages. Outputs of these modules are finally used by AGC, which adjusts the power outputs of generators at different power plants such that the grid operates optimally with respect to the generation cost and the physical safety of the grid.

1.2 Motivation: Potential Threats and Impact

Communication infrastructures are integrated with the legacy systems in a smart grid, which produce a highly complex configuration comprised of heterogeneous cyber-physical components. Moreover, these cyber-physical components are often interconnected through different communication media and protocols, and they are operated in different modes and policies. The inherent complexity of integrating multiple heterogeneous systems in a smart grid significantly increases the potential of security threats, which can cause massive and extremely devastating damage. It is well documented that configuration errors cause 50-80% of vulnerabilities in cyber infrastructures [10]. Since smart grids are still evolving, weak security measures often exist there. In fact, it is possible that there is no security measure present at all. As a result, cyber attacks can happen easily. We discuss the causes and impacts of potential threats to smart grids in the following categories:

- Attacks on connectivity: In order to send or receive necessary measurements, commands, etc., communication must be established between the sender and the receiver. In AMI, there should be a path between the smart meters and the utility servers, so that billing reports can reach the utility from the meters, while control commands (*e.g.*, meter turn on or off) from the utility can reach the designated meters. There are typically intermediate devices for collecting, buffering, and forwarding the data between the meters and the utility. In SCADA, power flow measurements must reach the control centers, often through the intermediate devices, so that state estimation and control routines can be performed. However, a connectivity requirement may fail due to misconfigured devices, link failures, or compromised intermediate nodes.

- Attacks on data integrity: Data integrity is crucial for the smart grid operations. If data (*e.g.*, measurements or control commands) is corrupted, then corresponding applications may fail and serious cascading effects may occur. Due to the lack of data integrity protection, cyber attacks like man-in-the-middle and replay attacks can be launched, and the data can be maliciously modified. In AMI, such false data injection attacks can create incorrect billing, broken service, and inaccurate demand-response [11]. In the case of a SCADA system, incorrect state estimation can cause inefficient or incorrect control decisions that may create serious economical as well as physical damage to the grid. For example, with the corrupted states, the OPF process will generate a non-optimal solution for power generation outputs. This solution may be not only economically expensive but also physically damaging, as it can lead electricity to flow through some transmission lines higher than their critical limits. Similarly, the contingency analysis with corrupted states will have an erroneous perception of security in contingencies.

- Attacks on data or service availability: Smart grids are also subject to denial of service (DoS), link flooding, and wireless link jamming attacks. For example, in the case of a RF mesh network-based smart metering infrastructure, the communication between smart meters and the associated collector can be disrupted by creating radio jamming. More interestingly, incorrect scheduling of usage data reports can also lead to self-generated DoS attacks because it may cause overflowing of a data collector's buffer. DoS attacks can cause data to be lost or delayed in reaching the destination, which consequently can make different utility services fail due to inadequate or out-of-date data. For example, if state estimation is done using delayed data, the computed states can be too old to make a correct control decision.

1.3 Work Objectives

The correct functioning of a smart grid requires consistent and secure data flow and the timely execution of tasks. In order to ensure safe, secure, and reliable functioning of smart grids, our objective in this dissertation is to address the following major problems:

- Proactive security: Reactive security using intrusion response and mitigation is insufficient for critical CPS like smart grids. Unlike traditional IT systems, where the impact of an attack is mostly local, an attack in a critical infrastructure can be devastating in terms of damages and consequences. Thus, it is required to proactively identify the potential threats in order to reduce the attack surface for a smart grid and avoid the detrimental consequences.

- Resiliency: This is the ability to maintain the integrity of smart grid operations and minimize potential damages despite the attack occurrences. Critical infrastructures, such as smart grids, should be resilient to attacks in order to assure the accepted operational state of the system under attacks, even if attack intensity increases.

- Automation: Automating the security analysis is essential for large and complex systems like smart grids, as manual security analysis is quite unrealistic. The smart grid security guidelines developed by NIST [4] are highly detailed and cumbersome to implement due to the complexity of cyber and physical integrations. Therefore, automating security verification and configuration synthesis techniques in a provable and scalable manner is a"holy-grail" challenge for smart grids.

1.4 Research Challenges

Toward fulfilling our research objectives, there are many research challenges that we need to address. The key challenges are as follows:

- Integration of cyber and physical systems: Since cyber and physical systems are included together in AMI and SCADA, along with the coexistence of smart/intelligent devices with legacy systems, modeling this integration is challenging.

- Interdependency modeling: There are interdependencies between different AMI and SCADA physical and functional components. For example, the output from the state estimation process is taken as an input to the OPF process. Thus, it is crucial to investigate such interdependencies and model them accordingly in order to understand the broader space of attack and impact.

- Evasion modeling: There are algorithms to detect bad measurements in SCADA. However, it is possible to corrupt the data without being detected by evading these algorithms. It is important to inspect existing as well as potential evasion techniques, and find defense mechanisms for them.

- Security and resiliency property modeling: There are security guidelines for AMI and SCADA provided by NIST. It is quite challenging to translate these guidelines into logical constraints, considering different AMI and SCADA components and their interactions.

- Generic property modeling: Various energy providers may have different security requirements. Thus, it is important to provide a general framework that allows defining new requirements or constraints with minimum or no change to the existing model.

- Scalability: A smart grid usually consists of a large number of field devices. For example, an AMI typically consists of thousands of smart meters and hundreds of data collectors. It is challenging to model this large number of devices such that the security verification, as well as the resiliency architecture synthesis, can be performed efficiently.

1.5 Contributions

The main contribution of this dissertation is developing formal techniques and tools that can address the security requirements, identify potential threats, and offer remediation plans for smart grids, so that secure and reliable smart grids can be established. In this research, we focus on AMI, SCADA, and EMS, as shown in Figure 2, which are the most critical components of a smart grid. We address the above-mentioned research challenges and contribute to the smart grid security and resiliency through the following three research thrusts:

- Security verification for AMI and SCADA: We develop a scalable and provable formal model to verify the compliance of AMI configurations with NISTIR 7628 security controls [12], and to generate remediation plans for potential security violations. We implement this model as a tool named SmartAnalyzer. The tool enables energy providers to proactively investigate AMI security configurations in order to identify and mitigate potential security threats and to guarantee AMI operational integrity and security requirements. Our main contribution in this work lies in the formal modeling of novel security properties that are critical for the integrity and security of AMI. These security properties include different invariant and user-driven constraints, such as data overflow/overwrite protection, cyber bandwidth limitation, data integrity and confidentiality, assured report delivery, and data freshness. We also develop an identical formal model for the proactive security analysis of SCADA. In this analysis, we focus on the integrity of the SCADA control routines along with the trusted communication between field devices and the control center.

- Automated synthesis of resiliency architectures: We build automated formal models for synthesizing necessary configurations that satisfy resiliency requirements, as well as deployment cost and usability constraints. We first develop a formal model that synthesizes isolation-based resiliency architecture for the cyber systems in smart grids. There is a lack of techniques and tools for automatic security or resiliency architecture design that optimize security and resiliency. Our formal model satisfies this need by automatically synthesizing resiliency configurations, while considering resiliency design alternatives based on network isolation patterns as well as usability and deployment cost. An isolation pattern is defined as a restriction on the network connectivity, such as access denial, authenticated communication, payload encryption, source identity hiding, etc. The model determines the appropriate isolation patterns along with the correct placements of necessary security devices in the network that satisfy the given requirements and constraints. We also devise a hypothesis testing-based mechanism to find optimal isolation configurations by exploring different design alternatives. The novelty of this framework lies in its comprehensiveness as a decision support system as it considers all basic network isolation-based design parameters (both resiliency requirements and business constraints) and provides a complete resiliency architecture (both configurations and device placements). We develop another formal model to synthesize redundancy-based resiliency architecture for AMI, satisfying the operational integrity and robustness requirements. These requirements ensure the reporting of usage data from meters to the utility without overflowing the data collectors, even if some collectors or communication paths fail.

- Threat analytics and security hardening for power system state estimation: We develop formal security analytics for the verification of false data injection attacks against state estimation in power systems. We formally model potential attack evasions on state estimation, interdependency between state estimation and other EMS modules, and a comprehensive list of attack attributes, such as access capability, resource constraint, knowledge limitation, and attack goal. A solution to this model provides the attack vectors (*e.g.*, the set of measurements to be altered) required for compromising state estimation. More importantly, the modeling of interdependent modules together allows us to investigate novel stealthy attacks on state estimation. We also provide a mechanism to devise a security architecture for state estimation, which includes a minimal set of measurements (or associated buses) that require data integrity protection. We also develop a proactive defense mechanism against persistent attacks by randomizing different critical parameters used in state estimation.

1.6 Technical Approach Overview

Our technical approach in this research primarily has five major tasks:

1. We formally model the security and resiliency verification and synthesis as constraint satisfaction problems.

2. We encode the formal models using satisfiability modulo theories (SMT), a powerful tool for constraint logic programming. Later in this section, we present a brief description of SMT logics and their capabilities.

3. We implement the formal models, solve them using suitable SMT solvers, and demonstrate them using examples.

4. We evaluate the formal models in terms of time and space requirements using synthetic configurations and attack scenarios.

5. We increase the scalability of the formal models by applying property-based abstraction during modeling or by providing efficient mechanisms for solving the models.

In the following, we briefly discuss our technical approach with respect to each research thrust:

1.6.1 Security Verification for AMI and SCADA

The correct functioning of AMI and SCADA depends on the security configuration of the AMI/SCADA devices and the secure interactions among them across the network. NIST has developed security guidelines, especially NISTIR 7628 and NIST SP 800-82 for AMI and SCADA security [12, 13, 14]. NERC also provides cyber security requirements for bulk electric power systems [15, 16]. These guidelines consist of numerous security controls that ensure secure and trusted communication and resource availability toward controlling potential security threats on AMI and SCADA. There is a large number of logical relations among the configuration parameters of AMI and

SCADA devices, that must be satisfied in order to comply with these guidelines and ensure the integrity and security of smart grid operations. We develop an automated formal tool, SmartAnalyzer, that can verify smart grid security configurations with standard and organizational security guidelines and identify potential security threats as violations of those security requirements.

In the case of AMI, we formally model AMI configurations and security requirements motivated from NISTIR 7628. The major technical novelty of the tool is its capability to analyze various operational integrity and security constraints on AMI, which are often different than those in conventional IT systems. In this modeling, we apply an SMT-based formal analysis mechanism. We apply property-based abstraction to model the configurations of AMI devices, so that the model can scale efficiently with the large number of devices, particularly considering thousands of smart meters and hundreds of collectors. We solve the model using an efficient SMT solver, Z3 [17], which verifies whether the asserted clauses (*i.e.*, constraints) within the model satisfy each other. A violation of the constraint satisfaction indicates a threat. We systematically analyze the solver's outcome if there is a constraint violation, and identify the potential causes of the threat. In our diagnosis, we also identify remediation strategies to mitigate this threat. The identified threats and remediation plans are presented to smart grid operators to fix the security breaches. We demonstrate SmartAnalyzer, particularly the verification of security constraints and the diagnosis of unsatisfied results, with examples. We evaluate its accuracy with the ground truth using a small AMI testbed established at our university [18]. Next, we evaluate the scalability of the tool using synthetic AMI networks, in terms of time and memory requirements, by executing the tool in different problem sizes and constraint amplitudes.

We apply a similar technical approach for SCADA security analysis considering NIST SP 800-82 security guidelines. Since there is a strong similarity between the security guidelines of NISTIR 7628 and NIST SP 800-82, in this particular work we primarily focus on requirements specific to SCADA. We model the secure data communication from the field devices to the control center in order to verify whether SCADA control routines can operate with trusted data. We consider the resiliency of performing the control routines in contingencies, particularly when some field devices fail due to accidents or attacks. We also illustrate the execution of the formal model with an example.

1.6.2 Automated Synthesis of Resiliency Architecture

We develop a formal model that automatically synthesizes network isolation configurations to construct a resiliency architecture for cyber systems in smart grids. The synthesized configurations satisfy specific resiliency requirements (isolation) and business constraints (usability and deployment cost). We name this formal synthesis framework ConfigSynth. The synthesis framework is designed as a constraint satisfaction problem and is encoded using SMT. The resiliency architecture, generated by ConfigSynth, includes the isolation configurations and the placement of network devices, including firewalls, IPSec gateways, and IDS, in the network topology.

We devise a hypothesis testing-based mechanism for ConfigSynth to find optimal isolation configurations. This mechanism allows us to explore different design alternatives through a feedback loop to ConfigSynth and find an optimal resiliency architecture. Thus, this framework provides a comprehensive decision support system by considering both resiliency requirements and business constraints, generating a satisfiable resiliency architecture including both configurations and device placements, and allowing the exploration of design alternatives to enhance the ultimate solution. It is worth mentioning that this resilient architecture synthesis framework is generic in design. Therefore it is applicable for conventional IT networks as well. We illustrate the executions of both ConfigSynth and the refinement mechanism with examples. We analyze the relationship between isolation requirements and usability and deployment cost constraints. We evaluate the time requirement for the executions of our solutions by varying the problem size as well as magnitudes of isolation requirements and business constraints. We also evaluate the memory requirement for executing the formal model.

We develop a similar formal framework to synthesize redundancy-based resiliency architecture for AMI. We represent resiliency requirements in terms of operational integrity and robustness properties. The robustness property guarantees assured data delivery even in the $n-1$ contingency (*i.e.*, a failure of a link or an intermediate device due to an accident or attack), while the operational integrity focuses on no loss of data due to the limitation of resources (*e.g.*, buffer, bandwidth, etc.) or improper report schedules. We also consider a budget limitation for deploying of intermediate devices and paths. We formally model AMI configurations, their interactions, integrity and robustness requirements, and a redundancy model. The solution to this model automatically synthesizes the required AMI configurations to satisfy resiliency requirements by introducing redundancy within the budget. This redundancy ensures alternative intermediate devices as well as paths toward the utility, such that data loss cannot exceed a limit when a contingency occurs. Since a typical AMI consists of a large number of smart meters and collectors, synthesizing resiliency configurations individually for each device is highly time consuming, given this large number of devices. Thus, we apply the property-based abstraction to model the configurations. We illustrate the execution of this synthesis model with an example, and evaluate its scalability using synthetic AMI networks by varying the problem size and constraints.

1.6.3 Threat Analytics and Security Hardening for Power System State Estimation

We characterize attacks in their most general form so that adversarial capabilities against the power system state estimation can be modeled. We represent an attack in terms of different attributes including the attacker's accessibility, resources, and knowledge. Unlike prior work, where the security of state estimation considers these attack attributes in isolation, we assess the attack feasibility considering these attributes combined in a single model. We formally model attacks on state estimation using SMT. The solution to our model answers whether an attack can be launched stealthily in a particular attack scenario. Our formal framework allows for the exploration of potential threats under different attack scenarios.

We model other EMS modules, particularly the topology processor and OPF, which are interdependent with the state estimation module. The modeling of topology poisoning attacks with regards to the topology processor allow us to explore novel stealthy attacks on state estimation. The modeling of OPF allows us to verify the impact of stealthy attacks on the economic operation of the grid as well as its stability.

We devise a mechanism that automatically synthesizes a security plan, which is a set of measurements that need to be secured, in order to make state estimation immune from stealthy bad data injection while considering the grid operator's resources and assumed attack model.

We also develop a novel idea based on moving target defense (MTD) to provide proactive security to the grid by changing the physical parameters used for state estimation. Our moving target defense technique employs frequent and operationally safe randomization of (i) the set of measurements used for state estimation and (ii) the electrical properties, particularly impedance, of a set of power transmission lines. Due to the frequent randomization of these properties, an adversary cannot gain the perfect knowledge that is required to successfully launch a stealthy attack. As a result, the attack success probability reduces significantly. We also provide necessary formal models to select valid randomization considering different operating constraints of the grid, such as (i) ensuring the observability of the power system from the received measurements and (ii) maintaining the existing optimal power flow solution.

We provide examples to demonstrate our formal models. We also evaluate the scalability of each of the offered models using standard IEEE test power systems along with synthetic attack scenarios. Since our formal model to analyze the impact of stealthy attacks on OPF takes substantial time, we provide a mechanism to increase its efficiency by pruning the search space.

1.6.4 SMT Overview

In the past decade, Boolean formal methods (*e.g.*, SAT [19, 20]) have been used successfully in network security analysis, especially for verifying security policy, which is defined as a sequence of propositional logical constraints [21, 22]. However, due to the increasing complexity of network security and business requirements of CPS, Boolean propositional logic constraints are not suitable to develop security analytics for complex systems like smart grids. SMT has been developed to overcome this shortcoming by offering various "background theories" that efficiently deal with integers, real numbers, arrays, uninterpreted functions, linear arithmetic, etc. In addition, SMT solvers provide a much richer formal modeling platform compared to SAT solvers.

SMT solvers are proved to be powerful tools in solving constraint satisfaction problems that arise in many diverse areas including software and hardware verification, type inference, extended static checking, test-case generation, scheduling, planning, graph problems, etc. [23, 24]. An SMT instance is a formula in first-order logic, where some functions and predicate symbols have additional interpretations according to the background theories. SMT is the problem of determining whether a formula is satisfiable or not. For example, the following simple SMT instance has two constraints:

$$(2x + y < 2) \vee (x - 2y > 0)$$

$$x \leq 1$$

This instance is satisfied with the assignments: $x = 0$ and $y = 0$. There can be other assignments for x and y which can satisfy the constraints, and SMT can provide all of these assignments. SMT solvers are efficiently applied in solving large and complex problems. It has been shown that modern SMT solvers can check formulas with hundreds of thousands of variables, and millions of clauses [23, 24]. In this research, we primarily use theories of integers, real numbers, and linear arithmetic for modeling, and we find that our models are highly efficient in solving complex security problems in smart grids.

1.7 Organization

The rest of the dissertation is organized as follows:

1. Chapter 2 presents formal security analytics for verifying AMI and SCADA configurations against security guidelines. We illustrate the implemented formal verification models with examples and present the evaluation results showing their scalability.

2. Chapter 3 presents automated formal frameworks for synthesizing resiliency configurations. We consider the synthesis of network isolation-based resiliency architectures for cyber systems in smart grids and redundancy-based resiliency architectures for AMI. We demonstrate the developed formal models with examples and evaluate their scalability using simulation experiments.

3. Chapter 4 presents a formal model for the verification of false data injection attacks on power system state estimation. We also model the impact of these attacks on economic operations of the grid. Then, we present mechanisms to mitigate such attacks proactively using static protection as well as moving target

defense techniques. We illustrate these frameworks with examples and present evaluation results by running experiments on different IEEE test systems.

4. Chapter 5 presents the summary of this dissertation, focusing on contributions and evaluation results. We conclude this chapter, and thus the dissertation, with a number of potential future research directions.

CHAPTER 2: SECURITY VERIFICATION FOR AMI AND SCADA

Smart grids provide innovative and efficient energy management services that offer operational reliability and value-added advantages to both customers and energy providers. The potential market for smart grids projects that it will be the most widely deployed critical infrastructure in the 21st century. AMI and SCADA are the major components in a smart grid. Unlike the traditional cyber networks, these components consist of heterogeneous devices, such as smart meters, intelligent data collectors, intelligent electronic devices (IEDs), remote terminal units (RTUs), head-end systems, hosts, routers, firewalls, etc. AMI and SCADA devices communicate with one other through various communication protocols, physical media, and secure tunnels. These devices transfer measurement data following different modes of data delivery, which are controlled by alternative security policies. Security attacks on such networks due to misconfigurations or constraint violations have the potential to cause critical damages including power outages and destruction of equipment [25, 26, 27].

In this work, our objective is to develop an automated security analysis tool for smart grid networks, particularly AMI and SCADA. This tool takes smart grid configurations and organizational security requirements as inputs, formally models configurations and various security constraints, and verifies the compliances of the configurations with the constraints using satisfaction checking. The tool generates a comprehensive threat report that includes the traces and reasoning behind various

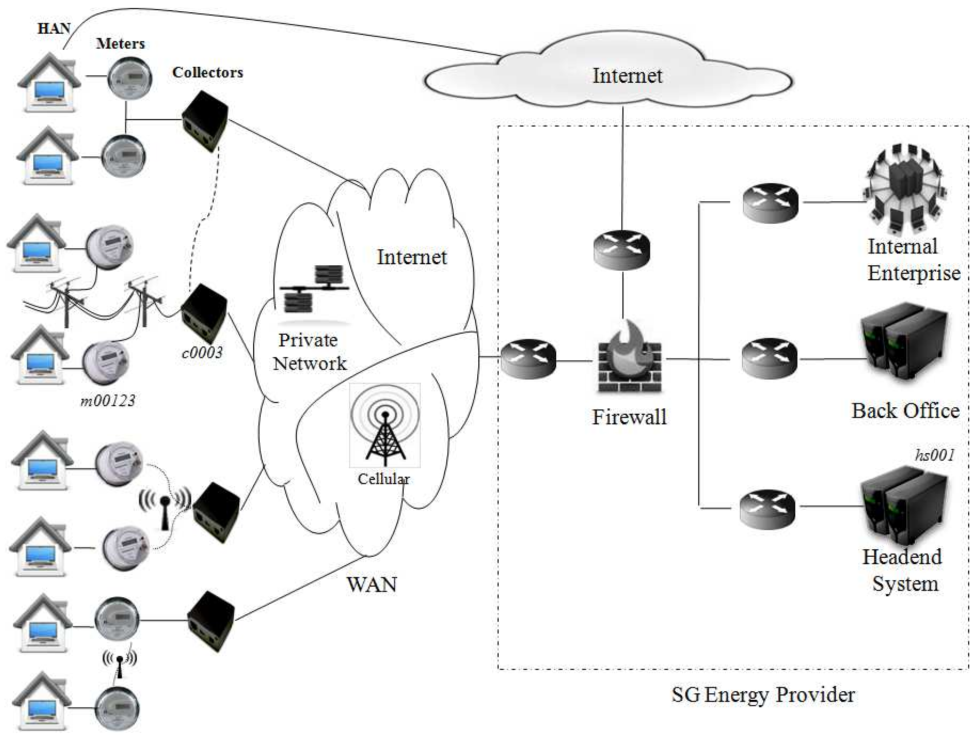


Figure 3: A typical AMI network.

constraint violations and potential reconfiguration plans. The performance of the tool is analyzed using various real and synthetic data.

2.1 Challenges

In this section, we first briefly discuss smart grids and potential threats on them. Then, we discuss the state of the art with respect to the security analysis of smart grids. Lastly, we briefly write our contributions in this particular work.

2.1.1 AMI and SCADA Complexity

We discuss the complexity of a smart grid considering its two major components: AMI and SCADA.

2.1.1.1 AMI Complexity

The general structure of an AMI network is shown in Figure 3, which usually consists of millions of smart meters, thousands of intelligent data collectors, and one or more headend systems as the main components. Although in some AMI architectures, a meter directly reports energy usage data to the headend system, often data collectors are used to collect and store meter data, and later to send the stored data to the headend system when it is required [28, 29]. This collector based AMI design gives better manageability by allowing scalable infrastructure design, flexible protocol use, and efficient networking. A meter often establishes a secure connection with a specific collector and reports energy usage data periodically. A collector forwards the data received from a group of meters to a headend over a secure connection. It also forwards control commands and patches from the headend to the meters. A meter may be connected to a collector directly or through another meter. The latter case occurs in a mesh network of meters, where intermediate meters relay the data to the collector. A large number of collectors are connected with a headend, usually through a proprietary but third party network. There are one or more firewalls for restricting the access between AMI and the energy provider's network. There are two data delivery modes, which can be used between meter and collector, and between collector and headend: (i) *push-driven mode* (simply, push mode) in which a meter or a collector reports data periodically based on a pre-configured delivery schedule, and (ii) *pull-driven mode* (simply, pull mode) in which a meter or a collector reports data only upon receiving a request. In the pull mode, requests are usually sent periodically

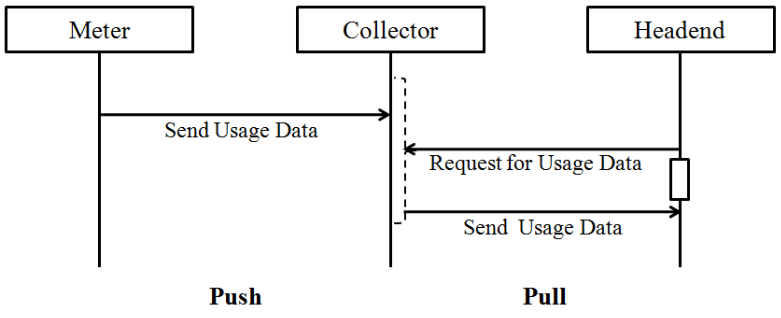


Figure 4: The push-pull mode of data delivery in smart grids.

following a schedule. In practice, the push mode is used between meter and collector, while the pull mode is used between collector and headend (Figure 4).

AMI networks are more complex than traditional networks mainly due to the following reasons. First, AMI is a hybrid network consisting of (i) heterogeneous devices (*e.g.*, meters, collectors, firewalls, routers, IPSec gateways, etc.), (ii) varieties of links (*e.g.*, power lines, wired, and wireless), and (iii) different protocols (*e.g.*, LonTalk protocol [30] between meter and collector, and TCP/IP protocol between collector and headend). Second, an AMI network involves varieties of data stream types (*e.g.*, power usage data, control commands, and software patches), which exhibit different priorities and resource requirements. Third, unlike the policy-based Internet forwarding, data delivery in AMI is either time-driven or request-driven and it follows specific schedules. For the purpose of successful delivery of data, AMI must be configured carefully to synchronize the data delivery without overflowing the network or its devices. Moreover, an AMI network must be accessible from the utility network for different purposes like control and patch management. Energy users from Home Area Networks (HANs) can also access the AMI network via the Internet or smart meters.

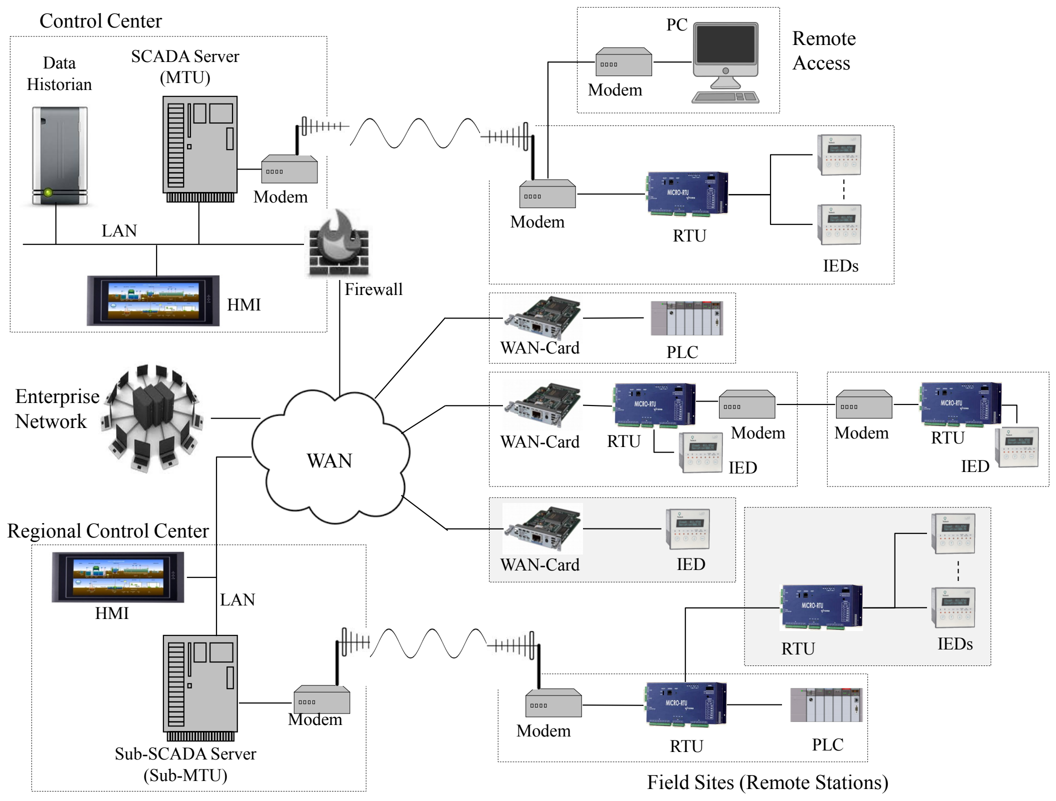


Figure 5: A typical SCADA network.

2.1.1.2 SCADA Complexity

Industrial Control Systems (ICS) are often found in industries, such as electric, water, oil, natural gas, chemical, transportation, etc. SCADA is the most important type of ICS, which is responsible for monitoring and controlling dispersed assets by gathering and analyzing real-time data. A typical topology of SCADA is shown in Figure 5. Typical SCADA operations are automatic and human control loops and remote diagnostics and maintenance utilities. There are various kinds of control components, such as SCADA servers or Master Terminal Units (MTUs), Remote Terminal Units (RTUs), Programmable Logic Controllers (PLCs), Intelligent Electronic Devices (IED), Human Machine Interfaces (HMI), data historian, etc. In addition to these control components, there are different network components, such as communications routers, modems, and remote access points. These components usually use ICS protocols like Modbus, DNP3, or IEC 61850 variants for communicating with one another. SCADA networks are also complex like AMI networks due to possessing heterogeneous field devices, various communication links and protocol standards, and different data delivery modes and requirements, as well as integrating different systems like renewable energies, vehicle-to-grid, and microgrids with the grid.

2.1.2 Potential Cyber Threats on AMI and SCADA

In order to promote connectivity and remote access capabilities among corporate business systems, information technology (IT) is now increasingly used in AMI and SCADA, which escalates the possibility of cyber security vulnerabilities and incidents, as these systems have not been built taking security into consideration from the first place. The inherent complexity associated with integrating multiple heterogeneous systems in smart grids significantly increases the potential of security threats, which can cause massive and devastating damage. The root causes of security threats on a smart grid have been shown in Figure 6. There are two main causes of threats [12]. The first is the *misconfiguration* that might cause inconsistency, unreachability, broken secure tunnels, and many other security breaches. It is well documented that configuration errors cause 50-80% of vulnerabilities in cyber infrastructure [10]. The second is the *weakness or absence of security controls* that can cause cascaded cyber attacks, such as scanning, man-in-the-middle, denial of service (DoS), and jamming. Moreover, there are threats due to *operational errors*. For example, if the data deliv-

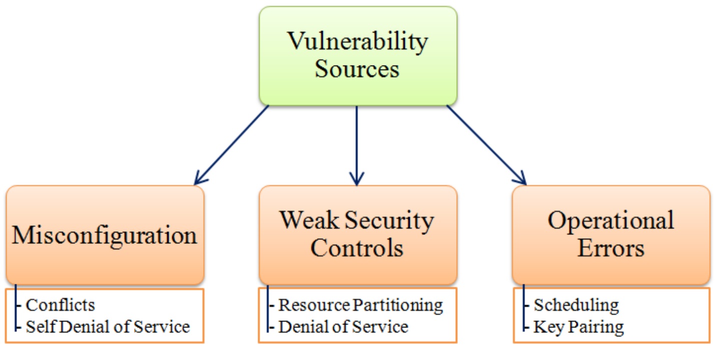


Figure 6: Root sources of smart grid security threats.

ery is request-driven rather than time-driven, an operational error, such as requesting data from a large number of collectors at the same time, may lead to cyber breakdown.

To achieve successful data delivery, reachability must hold between the sender and the receiver. Inconsistencies in communication protocols or authentication/encryption parameters of the communicating devices may cause failed data transmission leading to service disruptions. In addition, data should be delivered such that it satisfies end-to-end integrity. The violation of this requirement not only can cause incorrect billing but may also launch malicious control commands toward physical devices. In the case of AMI, improper scheduling of data delivery between meters and collectors can lead to buffer overflow and data loss in the collector side. This can cause delay in data delivery and even data loss at the endpoints due to limited link bandwidth. For example, if UDP protocol is used between a collector and a headend, improper scheduling may allow a large number of nodes to transfer data to a headend, which can flood links on the path and consequently cause data loss. A SCADA network can face similar threats. The main purpose of AMI and SCADA is to deliver measurement data from the field/physical devices (meters/sensors) to the provider's side (control center/utility), while delivering control commands from the provider's side to the field/physical devices. Hence, the resource unavailability threats can be dangerous. Other than these threats, there are typical cyber threats on AMI and SCADA communication networks, such as endpoint DoS, link flooding, and wireless link jamming.

2.2 Related Work

Throughout the last decade, the security policy misconfiguration and its verification have been studied extensively [31, 32, 21, 22]. In these approaches, the formal definition of configuration anomalies and safe deployment of single or multiple security devices are proposed and algorithms are presented to discover configuration inconsistency. There are also a number of works on risk-based security configuration analysis. Risk analysis using attack graphs is proposed in several works [33, 34]. Noel and Jajodia [33] present attack graphs to predict the various possible ways of penetrating a network to reach critical assets. Dewri et al. [34] model the problem of selecting a set of security hardening measures to minimize the residual damages in a predefined attack graph within a budget. There are some existing works [35, 36] that propose solutions to find optimal deployment of security devices using attack graphs in order to block all attack scenarios. However, all these above mentioned security analysis tools are proposed for analyzing misconfiguration problems in traditional networks. These tools do not model time-driven data forwarding and different operational and security controls specific to a smart grid.

During the last few years, a distinctive number of works [12, 13, 37, 38, 39] have been initiated to describe the interoperability among heterogeneous smart grid components, including security issues based on different attack scenarios. These works also describe the operational functionalities of smart grid components and the corresponding energy provider's internal system with guidelines for secure communication between them. They advise that the utilities should not be trusted to ensure that proper security is implemented. McDaniel et al. [26, 40] present the security and privacy challenges in smart grid networks. The works report that customers work closely with utilities to manage energy usage in smart grids, requiring that they share more information about how they use energy, thus exposing them to privacy invasions. Energy use information stored at the meter and distributed thereafter acts as an information-rich side channel, exposing customer behaviors.

Wang et al. [41] present an artificial intelligence-based approach for analyzing risks in smart grid networks. However, in their analysis they do not consider network link capacity, bandwidth, or different modes of communication between the smart grid components. Anwar et al. propose a couple of frameworks [42, 43] for modeling power grids and their control elements using first order logic. These frameworks are capable of evaluating power flows and overloading violations in smart grids. Liu et. al. [44] present a study on false data injection attacks in power grids. McLaughlin et. al. [27] present an approach for penetration testing on AMI systems. They develop archetypal and concrete attack trees for energy fraud, denial of service, and targeted disconnect attacks. However, these works do not analyze various misconfiguration problems and security controls on power grid networks.

The survey reveals that no significant research has been done on formal modeling of the complex AMI and SCADA configurations for analyzing various security constraints. Therefore, it is necessary to build a novel and useful tool for provably analyzing operational consistency and security controls in smart grids.

2.3 Contributions

The correct functioning of AMI and SCADA stands on consistent and secure execution of tasks in time. The safe security configuration depends not only on the local device parameters but also on the secure interactions and flows of these parameters across the network. There is a significant number of logical constraints on configuration parameters of many AMI/SCADA devices, which need to be satisfied to ensure safe and secure communications among AMI/SCADA components. NIST has developed security guidelines (*e.g.*, NISTIR 7628 and NIST SP 800-82 [12, 13]) consisting of hundreds of security controls for ensuring trusted path, resource availability, boundary security protection, etc., toward controlling different security threats on smart grids. Implementing these security controls in a scalable manner is one of the major challenges in smart grid security modeling. In addition, there is no such formal framework to support energy providers by analyzing these security properties and organizational business requirements.

The main contribution of this work is that we develop an automated tool, SmartAnalyzer, that allows energy providers to objectively assess and investigate smart grid security configurations to identify and mitigate potential security threats, and to enforce smart grid operational and organizational security requirements. In this work, we consider two core components of a smart grid: AMI and SCADA. In the first part of this work, we develop SmartAnalyzer considering AMI configurations and

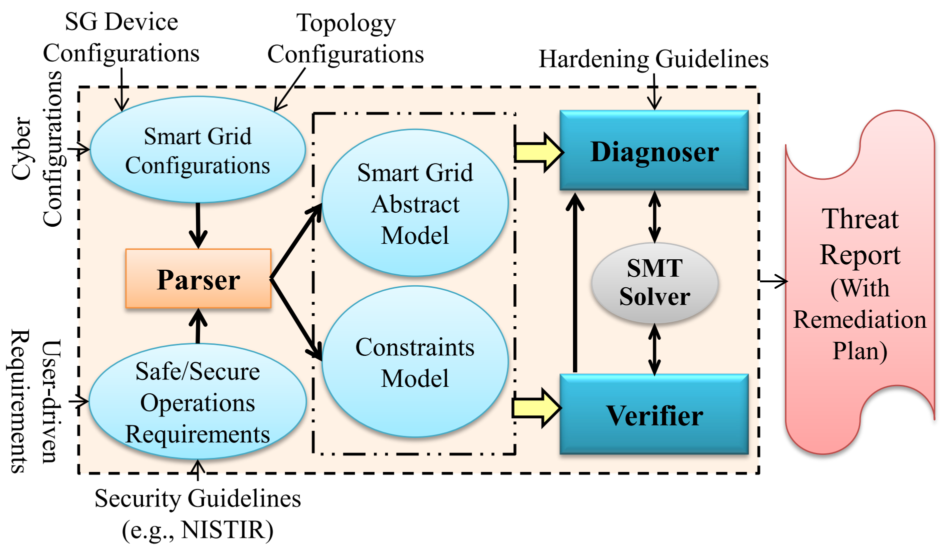


Figure 7: The architecture of SmartAnalyzer.

its operational integrity and security requirements. The major technical novelty of the tool lies in its capability of analyzing various operational integrity and security critical constraints on AMI, such as data overwrite protection, device scheduling and cyber bandwidth constraints, assured data delivery, data freshness, etc. Apart from these, the tool is capable of verifying various basic security properties, such as trusted path, data integrity, confidentiality, etc. Most importantly, the tool uses an SMT-based formal analysis engine as the core and provides a proof-based threat report as the outcome, which can be comprehensively used for fixing the errors. Second, we extend SmartAnalyzer for SCADA security analysis. In this extension, we primarily model the trusted/secure data communication from the field devices, so that SCADA control routines can operate with valid/correct data.

2.4 Overview of the Security Analysis Framework

The smart grid security analyzer, SmartAnalyzer, is an automated security analysis tool for smart grids that has the following functionalities:

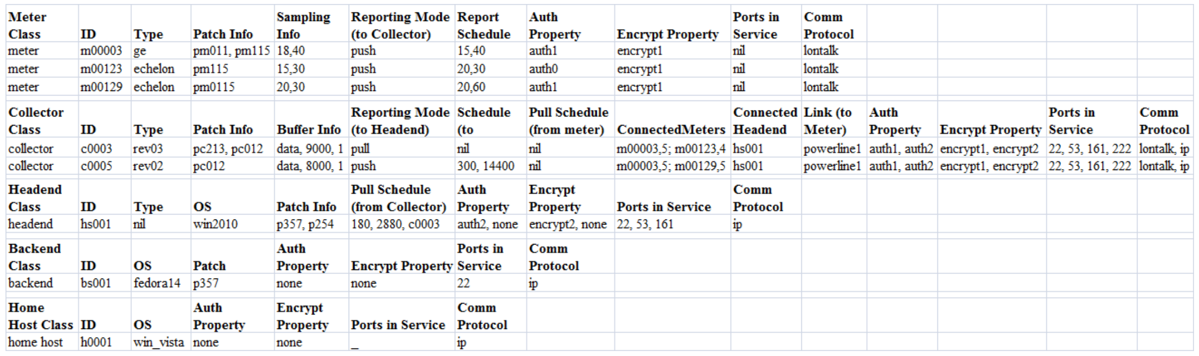
- Provides an extensible global model abstraction capable of representing millions of smart grid (*e.g.*, smart meters in AMI) configurations.

- Formally models and encodes of various constraints into SMT logic.

- Verifies the satisfaction of the constraints with smart grid configurations using an SMT solver.

- Identifies security threats from the constraint violations and provides remediation plans for security hardening by analyzing the verification results.

The SmartAnalyzer architecture is shown in Figure 7. First, the tool parses given smart grid configurations. The input regarding smart grid configurations is given following a template/format (often a CSV file). It consists of the device configurations, topology, communication between the devices, etc. SmartAnalyzer formally models the basic organizational requirements and various security guidelines as invariant and user-driven constraints, and encodes these constraints into SMT logic with respect to the given smart grid configurations. Then, the Verifier module of the tool uses the Z3 SMT solver [45] to verify these constraints with the configurations. A comprehensive threat report is generated based on the verification results. These threats correspond to those constraints whose verifications are unsatisfied. Finally, the tool's Hardener module creates a remediation plan by systematically analyzing the traces of unsatisfied results, which helps the administrators to reconfigure the corresponding smart grid system by directly fixing the configuration values or further incorporating new security alternatives.



Meter Class	**ID**	**Type**	**Patch Info**	**Sampling Info**	**Reporting Mode (to Collector)**	**Report Schedule**	**Auth Property**	**Encrypt Property**	**Ports in Service**	**Comm Protocol**				
meter	m00003	ge	pm011, pm115	18,40	push	15,40	auth1	encrypt1	nil	lontalk				
meter	m00123	echelon	pm115	15,30	push	20,30	auth0	encrypt1	nil	lontalk				
meter	m00129	echelon	pm0115	20,30	push	20,60	auth1	encrypt1	nil	lontalk				
Collector Class	**ID**	**Type**	**Patch Info**	**Buffer Info**	**Reporting Mode (to Headend)**	**Schedule (to**	**Pull Schedule (from meter)**	**ConnectedMeters**	**Connected Headend**	**Link (to Meter)**	**Auth Property**	**Encrypt Property**	**Ports in Service**	**Comm Protocol**
collector	c0003	rev03	pc213, pc012	data, 9000, 1	pull	nil	nil	m00003,5; m00123,4	hs001	powerline1	auth1, auth2	encrypt1, encrypt2	22, 53, 161, 222	lontalk, ip
collector	c0005	rev02	pc012	data, 8000, 1	push	300, 14400	nil	m00003,5; m00129,5	hs001	powerline1	auth1, auth2	encrypt1, encrypt2	22, 53, 161, 222	lontalk, ip
Headend Class	**ID**	**Type**	**OS**	**Patch Info**	**Pull Schedule (from Collector)**	**Auth Property**	**Encrypt Property**	**Ports in Service**	**Comm Protocol**					
headend	hs001	nil	win2010	p357, p254	180, 2880, c0003	auth2, none	encrypt2, none	22, 53, 161	ip					
Backend Class	**ID**	**OS**	**Patch**	**Auth Property**	**Encrypt Property**	**Ports in Service**	**Comm Protocol**							
backend	bs001	fedora14	p357	none	none	22	ip							
Home Host Class	**ID**	**OS**	**Auth Property**	**Encrypt Property**	**Ports in Service**	**Comm Protocol**								
home host	h0001	win_vista	none	none	_	ip								

Figure 8: An example data of AMI topology configurations.

2.5 AMI Security Analysis

We start this section with a brief preliminary discussion about an AMI and its security. Then, we present the formal model corresponding to the AMI security verification. Lastly, we describe the implementation of this formal model along with necessary demonstrations using examples.

2.5.1 Preliminary

A partial example of an AMI configuration template is shown in Figure 8. It consists of the device configurations, topology, communication between the devices, data delivery schedules in the network, etc. The device configurations are modeled using an abstraction model. The abstraction is done by exploiting the correlation between the configuration parameters of different AMI devices, as there is often a very large number of smart meters in a smart grid. The organizational requirements and various security guidelines (*e.g.*, NISTIR) are modeled as AMI invariant and user-driven constraints.

Table 1: Formal definition of a meter and a collector

Smart Meter: $SM_i \rightarrow Type_i \wedge Patch_i \wedge SRate_i \wedge Mode_i \wedge RSche_i \wedge$ $Auth_i \wedge Encr_i \wedge Serv_i \wedge CommProto_i \wedge TRate_i$ $Patch_i \rightarrow \bigwedge_{j=0\ldots} Patch_{i,j}$ $SRate_i \rightarrow SSize_i \wedge STime_i$ $RSche_i \rightarrow RScheBase_i \wedge RScheInt_i$ $Auth_i \rightarrow \bigwedge_{j=0\ldots}(AAlgo_{i,j} \wedge AKey_{i,j})$ $Encr_i \rightarrow \bigwedge_{j=0\ldots}(EAlgo_{i,j} \wedge EKey_{i,j})$ $Serv_i \rightarrow \bigwedge_{j=0\ldots} SPort_{i,j}$ $CommProto_i \rightarrow \bigwedge_{j=0\ldots} CommProto_{i,j}$
Intelligent Data Collector: $IC_i \rightarrow Type_i \wedge Patch_i \wedge BufSize_i \wedge Mode_i \wedge RSche_i \wedge$ $PRSche_i \wedge Auth_i \wedge Encr_i \wedge AttachSM_i \wedge LinkToSM_i \wedge$ $AttachHS_i \wedge Serv_i \wedge CommProto_i \wedge TRate_i$ $\cdots\cdots\cdots$ $PRSche_i \rightarrow \bigwedge_{j=0\ldots}(PScheBase_{i,j} \wedge PScheInt_{i,j} \wedge RDev_{i,j})$ $ConnSM_i \rightarrow \bigwedge_{j=0\ldots}(CSMId_{i,j} \wedge CSMNum_{i,j})$

2.5.2 Formal Model of AMI Security Verification

First, we discuss the modeling of AMI configurations. Next, we present the threat analysis model.

2.5.2.1 AMI Configuration Model

An AMI smart grid network consists of different types of devices, heterogeneous communication links, and sets of traffic control rules as well as security properties. Here, we describe these components along with their formalization.

2.5.2.1.1 Modeling of AMI Physical Components

Here, we present the formalizations of different AMI physical device components including meters, collectors, and hosts. We first briefly describe the idea of logical abstraction that we apply for modeling AMI configurations.

Configuration level abstraction: An enterprise smart grid network typically consists of millions of smart meters and thousands of collectors distributed over different geographical regions. In order to perform data delivery, these devices communicate with each other based on device configurations and communication properties. For the purpose of achieving better scalability, we apply the idea of configuration level abstraction that leverages the similarities between configurations of the devices. In our model, according to this abstraction we define classes for each kind of devices. A particular class of devices shares the same (physical and logical) configuration properties. We are not limited with this class-based concept within meters and collectors. In order to have a common design model, the same concept is applied for all kinds of AMI devices. However, the cyber-physical devices (*e.g.*, routers, firewalls, etc.) are not modeled as classes. Since these classes do not associated with network identities (*i.e.*, IP addresses), we introduce the concept of *zone* as a collection of similar (but not only limited to same class) AMI devices and provide network identity (usually subnet-based) to the zone.

Smart meter: A meter class is identified by an ID. Its profile *SM* is represented as a conjunction ($\wedge$) of different parameters as shown in Table 1. The vendor type (*i.e.*, *Echelon*, *GE*, etc.) is represented by parameter *Type*. We represent the sampling information of a meter using *SInfo* that consists of two components: sampling size (*SSize* in KB) and sampling time (*STime*). A meter can deliver data to a collector in two different modes: pull and push. In pull mode, the meter reports data based on the request from the collector that follows a specific pull schedule of the collector. On the other hand, in push mode, the meter reports data to the collector (without waiting for a request) based on its own report schedule. This reporting mode is captured by *Mode*. The reporting time schedule of a meter (in push mode) is modeled using *RSche* that consists of *RScheBase* and *RScheInt*. This indicates that the meter will report periodically in a regular interval of *RScheInt* starting from *RScheBase* after the base time. To achieve end-to-end security, the communicating devices must agree in their authentication and encryption properties. We model the authentication properties of a meter using parameter *Auth* as conjunction of algorithm (*AAlgo*) and key length (*AKey*). A meter may support multiple authentication properties. The encryption property is modeled similarly as *Encr*. The running services and communication protocols associated with a meter are represented by *Serv* and *CommProto*, respectively. Parameter *Patch* denotes the patches that are installed in the meter. The maximum transmission rate (in Mbps) of a meter is denoted by *TRate*. The formalization of a meter class is shown in Table 1.

Intelligent data collector: A collector class profile *IC* is represented as a conjunction of different parameters, which include all those of the meter class profile except the sampling information. In addition, each collector may have a pull schedule that is represented by parameter *PRSche*. It has three components: *PSBTime*, *PInt*, and

RDev, which denote that the collector periodically pulls data from a reporting device (*RDev*, a meter) starting at *PScheBase* with interval *PScheInt*. A collector has a buffer for storing the report data from different meters. *BufSize* represents the buffer size (in KB). The parameter *ConnSM* is a conjunction of meter classes (*CSMId*) and their numbers (*CSMNum*), and they represent the meters connected to the collector. *LinkToSM* represents the ID of the link that connects the collector to the meter. Parameter *AttachHS* represents the headend system to which data is reported by the collector.

Headend system: A headend system class profile *HS* is a conjunction of various parameters: *Type*, *OS*, *Mode*, *TRate*, *Patch*, *PRSche*, *Auth*, *Encr*, *Serv*, and *CommProto*. These properties are modeled as similar to those of meter/collector.

Host devices: An AMI network contains different type of hosts, such as (i) hosts of home area network (enterprise clients), (ii) enterprise internal hosts, (iii) enterprise application servers (backend systems), and (iv) external hosts from the Internet. Hosts have considerably less parameters. For example, an enterprise client host class profile has *OS*, *Auth*, *Encr*, *Serv*, *CommProto*, and *TRate* parameters only.

2.5.2.1.2 Modeling of AMI Topology

AMI topology defines the physical and logical connectivity between different AMI and network devices.

Router, firewall and IPSec devices: We model router (R), firewall (F), and IPSec (IS) devices similar to the work by Ehab et al. [21]. We also introduce the traffic limiting capability of a firewall in the model using parameter *FwLim* along with its

Table 2: Modeling of a zone and its relation with AMI devices

Zone:
$Zone_i \rightarrow ZSn_i \wedge ZMem_i \wedge ZGw_i$
$ZSn_i \rightarrow Ip_{i,j} \wedge Mask_{i,j}$
$ZMem_{i,j} \rightarrow ZMId_{i,j} \wedge ZMNum_{i,j}$
....
$ZMem_i \rightarrow \bigwedge_{j=0\ldots} ZMem_{i,j}$

Representation of a source:
$(S \rightarrow Id \wedge ZId) \rightarrow (Id = ZMId)$

action (*FwAct*) in its policy (*FwPolicy*). A router selects the next-hop (*RNext*) for a particular traffic based on its forwarding policy (*RPolicy*).

Link: A link is identified by an ID (*LId*). Its profile is a conjunction of *NodePair* (*i.e.*, the node-pair connected by the link) and *LinkStatus* (*i.e.*, up or down). *LId* binds the specified link type to parameter *LinkProp* that represents the properties of that link including *MediaType* (*i.e.*, wireless, ethernet, etc.), *SharedStatus* (*i.e.*, shared or not), *CommMode* (*i.e.*, half-duplex, full-duplex, etc.), and *LinkBw* (in Mbps).

Zone: We model a zone as a collection of similar AMI devices. Each zone has an ID. The profile of a zone (*Zone*) is comprised of three parameters: *ZSn*, *ZMem*, and *ZGw*. The parameter *ZSn* denotes an IP-address (with subnet Mask) that covers all devices in that zone. *ZMem* represents the IDs of different device classes and the number of devices (that belong to the zone) under each class. *ZGw* denotes the gateway router ID for that zone. The formalization of a zone, $Zone_i$ is represented in Table 2. The source or destination node of a traffic flow is represented as a conjunction of its ID (*Id*) and its zone's ID (*ZId*). The number of traffic sources/destinations depends on the number of zones and the number of classes in the zones. For example, if there are 50 zones and 4 collector classes per zone on average, then there are 200 possible source/destination collectors. In the rest of the paper, we refer to a source/destination (especially in traffic) as a node associated with its zone. Note that a group of meters is directly associated with a collector, thus addressable through this collector.

2.5.2.2 AMI Threat Verification Model

Appropriate modeling of the constraints is required to identify security threats on AMI. We classify these constraints into invariant and user-driven constraints, many of which are mapped to the NISTIR [12] guidelines.

2.5.2.2.1 Modeling of Invariant Constraints

There are various invariant constraints based on connectivity, data delivery schedule, resource availability, etc. between AMI components. These constraints must be satisfied for any successful communication.

Reachability constraint: Reachability must hold between a pair of devices, if data is required to be transmitted between them. For example, a meter should be able to reach a collector to deliver the report to the collector. Similarly, there should be reachability from collector to headend, so that the collector can deliver the report to the headend. This constraint intuitively verifies the links between a pair of devices for the satisfaction of routing and security device policies. The formalization of general reachability constraint is shown in Table 3. We first define *Forward* that checks whether a specific traffic $Tr_{S,D}$ (*i.e.*, from S to D) can be transferred from a node (X) to another node (Y) (like state transition). Then, we define *Reachable* and

Table 3: Formalizations of reachability and pairing constraints

Reachability Constraint:

$$\begin{aligned}
&Forward_{X,Y,Tr_{S,D},TrR} \rightarrow \\
&\quad Link_{X,Y} \wedge \\
&\quad (((X = S) \wedge (ZGw_S = Y)) \vee (Y = D) \vee \\
&\qquad (R_X \wedge RPolicy_{X,Tr_{S,D}} \wedge (RNext_X = Y)) \wedge \\
&\quad ((F_X \rightarrow FwPolicy_{X,Tr_{S,D}} \wedge FwAct_X \wedge \\
&\qquad (FwLim \rightarrow (TrR = min(LimVal, LinkBw_{X,Y})))) \vee \\
&\qquad (\neg F_X \rightarrow (TrR = LinkBw_{X,Y})))
\end{aligned}$$

$$\begin{aligned}
&Reachable_{A,B,Tr_{S,D},TrR} \rightarrow \\
&\quad Forward_{A,B,Tr_{S,D},TrR} \vee \\
&\quad (\exists C, Forward_{A,C,Tr_{S,D},TrR1} \wedge Reachable_{C,B,Tr_{S,D},TrR2} \wedge \\
&\qquad (TrR = min(TrR1, TrR2))
\end{aligned}$$

$$ReachabilityConstr_{Tr_{S,D},TrR} \rightarrow Reachable_{S,D,Tr_{S,D},TrR}$$

Pairing Constraint:

$$\begin{aligned}
&AuthPairing_{S,D} \rightarrow \\
&\quad (AAlgo_S = AAlgo_D) \wedge (AKey_S = AKey_D) \\
&EncrPairing_{S,D} \rightarrow \\
&\quad (EAlgo_S = EAlgo_D) \wedge (EKey_S = EKey_{m,D}) \\
&ProtoPairing_{S,D} \rightarrow \\
&\quad (Proto \in CommProto_S) \wedge (Proto \in CommProto_D) \\
&PairingConstr_{S,D} \rightarrow \\
&\quad AuthPairing_{S,D} \wedge EncrPairing_{S,D} \wedge ProtoPairing_{S,D}
\end{aligned}$$

the reachability constraint (*ReachabilityConstr*) on top of this. In the constraint formalization, we also model the maximum possible transmission rate (TrR) by taking the minimum bandwidth of the links across the path along with the limits that may be imposed by a firewall.

Connectivity pairing constraint: Consistent pairing between a meter and a collector is required over reachability for successful communication. This constraint is considered as a conjunction of security pairing and protocol pairing. In other words, it states that the authentication and confidentiality properties of the communicating devices should match and they have a common protocol to communicate. For example, in Figure 8, although there are 4 meters of class $m000123$ connected with collector $c0003$, they are not allowed to communicate as the security pairing will be violated due to a mismatch in their authentication properties (*i.e.*, $auth0$ and $auth1$). Similarly, a host from HAN will not be able to communicate with a meter if that host does not support the LonTalk protocol, which is the only protocol supported by a meter. *PairingConstr* in Table 3 checks these issues.

Schedule constraint: The schedule constraints (Table 4) ensure the basic correctness of the report or pull schedule configuration. The *MeterSampConstr* constraint states that the sampling time and the reporting base-start time of a meter must be less than or equal to its reporting interval, such that no reporting is done without new data. It also verifies that the sampling rate cannot be more than its maximum transmission rate. If a meter is in push mode ($Mode$ is true), then it should have a reporting schedule. A similar constraint (*CollectorPullScheConstr*) is true for a collector. If a collector is connected with some meters, whose reporting modes are pull ($Mode$ is false), then the collector should have a pull schedule for them.

Resource constraint: There are different resource constraints (Table 4), which are often related to report/pull schedules. The *CollectorBufConsrt* constraint states that the buffer size of a collector should be greater than or equal to the cumulative sampled data size of all the meters connected to that collector. Otherwise, data loss must occur in the collector buffer under any report schedule. Similarly, the *CollectorTrRateConstr* constraint states that the cumulative sampling rate of the connected meters cannot be more than the transmission rate of the collector. The *CollectorB-*

Table 4: Formalizations of schedule and resource constraints

Schedule Constraints:

$$\begin{aligned}
&MeterSampConstr_M \rightarrow \\
&\quad SM_M \wedge (Mode_M \rightarrow ((STime_M \leq RScheInt_M) \wedge \\
&\quad\; (RScheBase_M \leq RScheInt_M))) \wedge \\
&\quad ((SSize_M / STime_M) \leq TRate_M)
\end{aligned}$$

$$\begin{aligned}
&CollectorPullScheConstr_C \rightarrow \\
&\quad IC_C \wedge (((M = CSMId_C) \wedge \neg Mode_M) \rightarrow PRSche_C)
\end{aligned}$$

Resource Constraints:

$$\begin{aligned}
&(TotalSData_C = \textstyle\sum_M SData_M) \rightarrow \\
&\quad (M = CSMId_C) \rightarrow \\
&\quad\quad (SData_M = (SSize_M \times CSMNum_C))) \\
&CollectorBufConstr_C \rightarrow \\
&\quad IC_C \wedge (BufSize_C \geq TotalSData_C)
\end{aligned}$$

$$\begin{aligned}
&(TotalSRate_C = \textstyle\sum_M SRate_M) \rightarrow \\
&\quad (M = CSMId_C) \wedge \\
&\quad (SRate_M = ((SSize_M / STime_M) \times CSMNum_C))) \\
&CollectorTrRConstr_C \rightarrow \\
&\quad IC_C \wedge (TrR_C \geq TotalSRate_C) \\
&CollectorBwOutConstr_C \rightarrow \\
&\quad IC_C \wedge (TotalSRate_C \leq LinkBw_{C,ZGw_C})
\end{aligned}$$

wConstr constraint states that the bandwidth of the link from the collector to its gateway must be greater than or equal to the accumulated sampling rate of all the meters connected to it. Otherwise, no schedule will be possible without data loss.

2.5.2.2.2 Modeling of User-driven Constraints

To achieve correct and secure functioning of the AMI network, there can exist different user-driven constraints. We focus on AMI-specific constraints. Formalizations of these constraints are shown in Table 5.

Table 5: Formalizations of different user-driven constraints

Data Overwrite Protection Constraint: $(TotalRData_C = TotalSRate_C \times Period) \rightarrow$ $((Mode_C \rightarrow (Period = RSInt_C)) \vee$ $(\neg Mode_C \rightarrow (H = AttachHS_C) \wedge (Period = PRSInt_H)))$ $OverwriteProtectConstr_C \rightarrow$ $IC_C \wedge (BufSize_C \geq TotalRData_C)$
Cyber Bandwidth Constraint: $(Num_C = \sum_Z ZMNum_Z) \rightarrow (MId_Z = C)$ $(TotalRRate_{H,Sche} = \sum_C (TotalSRate_C \times Num_C)) \rightarrow$ $(H = AttachHS_C) \wedge Mode_C \wedge (RSche_C = Sche)$ $LinkBwConstr_{H,X,Y} \rightarrow$ $HS_H \wedge (LinkBw_{X,Y} \geq TotalRRate_H)$
Assured Data Delivery Constraint: $AssuredDelivery_{M,C,H} \rightarrow$ $SM_M \wedge IC_C \wedge HS_H \wedge$ $Pairing_{M,C} \wedge (M = CSMId_C) \wedge Reachable_{M,C} \wedge$ $Pairing_{C,H} \wedge (H = AttacheHS_C) \wedge Reachable_{C,H} \wedge$ $ResourceConstr_{M,C,H} \wedge CyberConstr_{M,C,H}$
Availability Protection Constraint (Limit DoS Attack): $(MaxTrR_{H,X,Y} = \sum_C TrR_C \times Num_C) \rightarrow$ $Compromise_C \wedge (AttachHS_C = H) \wedge Forward_{X,Y,Tr_{C,H},TrR}$ $AvailProtectionConstr_{H,X,Y} \rightarrow$ $IC_C \wedge (LinkBw_{X,Y} \geq MaxTrR_{H,X,Y})$

Data overwrite protection constraint: This constraint states that the aggregate report data of all the meters connected to a specific collector must not flood the collector buffer within the reporting interval [46, 47]. For example, in Figure 8, collector $c0005$ receives reports from 5 meters of class $m00129$ (sampling rate: 20KB/30secs) and 5 meters of the $m0003$ class (sampling rate: 18KB/40secs). Therefore, $c0005$ will receive 335KB (average) data every 60 seconds, which is to be stored in its buffer. Based on the report schedule, the collector pushes the data to the headend every

$1,440$ seconds. Thus, during this period, an aggregate of $8,040$KB data will be sent to the collector by these meters. This amount of data will flood the collector buffer (size $8,000$ KB), which will in turn cause data loss (*i.e.*, initial 40KB report data will be overwritten).

Cyber bandwidth constraint: The *LinkBwConstr* constraint is to confirm that the aggregate report rate of collectors reporting simultaneously due to matching report schedule should not exceed the bandwidth limitation of the network path (considering a link from X to Y) connecting to the headend (H). A violation of this constraint will cause link congestion/DoS.

Assured data delivery constraint: This constraint requires checking the end-to-end data delivery (from a meter to a headend through a collector) to satisfy the AMI global functionality. This constraint intuitively implies the satisfaction of the following constraints: (i) reachability, (ii) successful security pairing, (iii) availability of resources (conjunction of all resource constraints including data overwrite constraint as *ResourceConstr*), and (iv) synchronous reporting without flooding the cyber toward the headend. A violation of these constraints can create failure in data delivery.

Quality-of-delivery constraint: There are user-driven constraints for ensuring the quality of delivery. For example, the report freshness constraint (*FreshnessConstr*) restricts the delivery of data within a specific time window along with assured data delivery. A user can have a quality requirement on trusted paths. For example, this requirement can be defined as the satisfaction of (i) end-to-end encryption level based on key length (*e.g.*, 256 bits); and (ii) specific single or nested tunnels (*e.g.*, 2-level nested tunnels) requirements.

Availability Protection Constraint: This constraint (*AvailProtectionConstr*) ensures that if there are X number (or portion) of AMI devices being compromised, the assured data delivery constraint is still preserved. It intuitively verifies that DoS attack is not possible on links or endpoints, when the number of compromised nodes is no more than X (say, 5% collectors).

2.5.3 Implementation

2.5.3.1 SMT Encoding and Constraint Verification

We use Boolean terms to encode the Boolean configuration parameters. We also use Boolean terms to encode some of the integer configuration parameters, which usually take a very small range of values. The remaining parameters are modeled as integer terms. We normalize the parameters into integers that may take real values (*e.g.*, bandwidth). We use bit-vector terms for encoding IP addresses. In some of the computations, we require multiplying/dividing two variables. But, Yices SMT solver [48] does not support such non-linear operations (multiplication or division of two variables). Due to this shortcoming of Yices, we choose Z3, another well-known SMT solver [45]. This tool supports non-linear operations. We apply Z3 NET API to implement SmartAnalyzer. The tool (the Parser module) reads the configuration from the input template file and directly builds the model using the API.

After defining the configuration parameters as SMT variables, we model the configurations associated with the AMI network topology and the AMI components. We encode each constraint under the same formalism. Then SmartAnalyzer creates a verification query that checks the satisfaction of a constraint with the configuration. If M_{Conf} and M_{Constr} are the AMI configuration and constraint models respectively, the verification query Q is encoded as the following clause:

$$Q \rightarrow M_{Conf} \wedge M_{Constr}$$

The verification engine of SmartAnalyzer generates the verification results, which are either *sat* (satisfiable) or *unsat* (unsatisfiable). In the case of an *unsat*, the verification engine provides an *unsat-core* that basically represents the traces of constraint violations in the configuration. Then, SmartAnalyzer (through its *Diagnoser* module) systematically examines these violation traces and generates a comprehensive threat report. This report includes threat sources, targets, violating rules and reasonings, and a remediation plan showing possible reconfigurations for hardening the system with respect to the identified problems.

2.5.3.2 Methodology of Unsatisfied-core Generation

If the SMT solver gives an *unsat* result, we get the *unsat-core*, which describes the unsatisfied constraints and the corresponding configurations that the constraint does not support. In order to get the *unsat-core*, we use the concept of *hard* and *soft* clauses (a verification of assumptions in Z3). We separate configurations and constraints into these two groups. We often take fixed properties as hard clauses, while the rest as assumptions or soft clauses. If the model verification fails, the *unsat-core* shows the list of assumptions, *i.e.*, the constraints and the configurations, which are not satisfied. From the list of the unsatisfied configurations, it is possible to trace the reasoning of a constraint failure.

2.5.3.3 Methodology of Remediation Plan Synthesis

In order to get a remediation plan, we consider a policy for the reconfigurations. The policy shows feasible or preferred invariant and user-driven guidelines for possible reconfiguration candidates. An invariant guideline represents the configurations that are practically infeasible to modify. The vendor specific device configurations (such as the buffer size of a collector) are usually constant for a device. Hence, changing this property requires replacing the device with a different or newer product that has the required configuration property. The user-driven guidelines represent the organizational priorities or capabilities for performing reconfigurations. For example, the organization may be fine with deploying many collectors, but a minimum number of meters must be connected to each collector.

In the process of exploring the reconfiguration plan, the diagnoser module continuously checks the satisfaction of the model by releasing the assumptions (soft clauses) of the configurations systematically according to the remediation guidelines until the model verification gives a *sat* result. Releasing an assumption lets the solver choose the configuration values associated with the assumption that satisfy the hard clauses along with the remaining assumptions. This process is an implementation of *max-sat* [20]. Then, a remediation plan is generated from the *max-sat* output. It is worth mentioning that we use quantifiers for the purpose of verifying some constraints. In such cases, Z3 may return *unknown* instead of *sat*. This implies that there is no constraint violation found by the solver. Hence, if the result is not *unsat*, we consider

Table 6: An example of resource constraint verification

```
(assert (M 0)) ;; Meter 1 (Id 0)
(assert (= (Id 0) 0))
(assert (= (SSize 0) 25))
(assert (= (SInt 0) 45))

(assert (M 1)) ;; Meter 2 (Id 1)
(assert (= (MId 1) 1))
(assert (= (SSize 1) 15))
(assert (= (SInt 1) 30))

(assert (IC 10)) ;; Collector 1 (Id 10)
(assert (= (Id 10) 10))
(assert (= (BufSize 10) 200))
(assert (=> P0 (= (CSMId 10 0) 1)))
(assert (=> P1 (= (CSMId 10 1) 0)))
(assert (=> P2 (= (CSMNum 10 0) 8)))
(assert (=> P3 (= (CSMNum 10 1) 8)))

(assert (=> PC
  (=> (CollectorBufConstr 10)
    (and (M (CSMId 10 0)) (M (CSMId 10 1))
     (= (SData 10 0) (* (CSMNum 10 0) (SSize (CSMId 10 0))))
     (= (SData 10 1) (* (CSMNum 10 1) (SSize (CSMId 10 1))))
     (>= (BufSize 10) (+ (SData 10 0) (SData 10 1)))))))

(assert (CollectorBufConstr 1))

(check-sat PC P0 P1 P2 P3) ;; Sat
(get-model)
(get-unsat-core) ;; Unsuccessful
```

that the model is satisfied with the given constraints.

2.5.3.4 Verification Trace Analysis: An Example

This section describes how verification traces (results) from the SMT solver are analyzed to find the causes of constraint violations, along with the remediation plans for them. We explain the procedure with the example of a constraint verification.

In our example, we demonstrate the collector resource (buffer) constraint (Table 3).

Table 7: An example of the diagnosis process

```
Modified Model:
.........
(assert (= (BufSize 10) 100))
.........
(check-sat PC P0 P1 P2 P3) ;; Unsat
(get-unsat-core)

Solver Output:
unsat
(P0 P1 P2 P3 PC)

Max-SAT Implementation:
.........
(assert (forall ((c Int) (x Int))
   (=> (and (>= c 10) (<= c 10) (>= x 0) (<= x 1))
     (and (>= (CSMId c x) 0) (<= (CSMId c x) 1)))))

(assert (forall ((c Int) (x Int))
   (=> (and (>= c 10) (<= c 10))
     (>= (+ (CSMNum c 0) (CSMNum c 1)) 6)
     (>= (CSMNum c 0) 0) (>= (CSMNum c 1) 0))))
.........
(check-sat P0 P1 P2 PC) ;; Unsat
(get-model) ;; Unsuccessful
(get-unsat-core) ;; (P0 P1 P2 PC)
.........
(check-sat P0 P1 PC) ;; Sat
(get-model) ;; Successful
(get-unsat-core) ;; Unsuccessful

Satisfied Model:
.........
CSMNum − > {
   10 1 − > 6 ;; The number of type 1 meter is 6
     else − > 0} ;; The number other type (type 0) meter is 0
.........
```

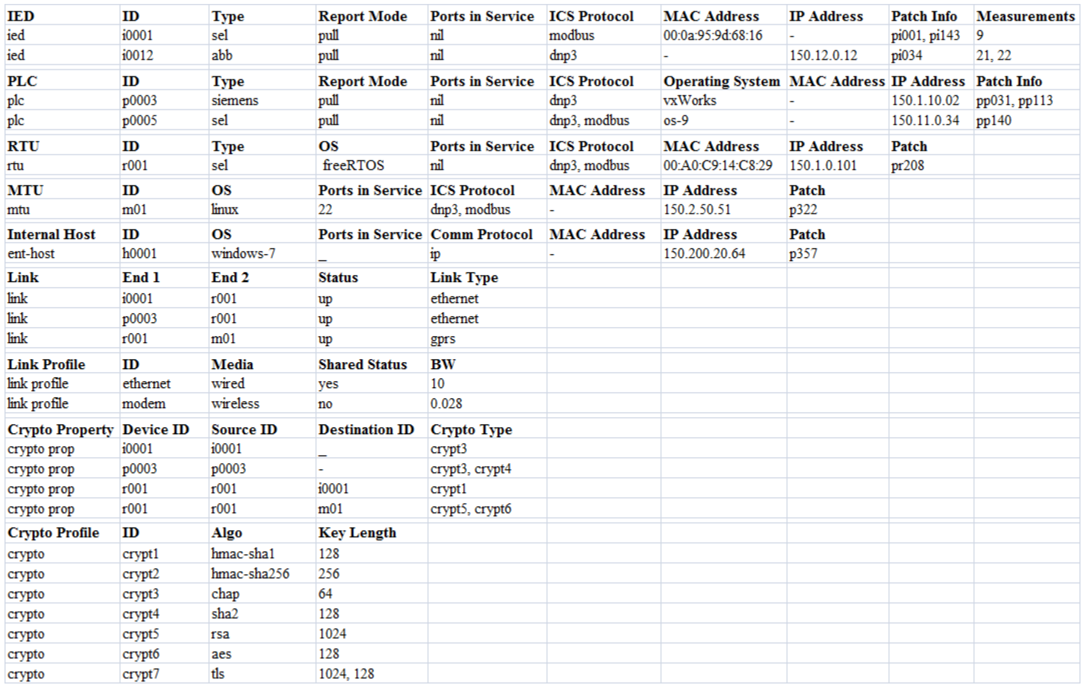
Table 6 shows the SMT-LIB encoding of the AMI configuration (required segment only) and the collector resource constraint. To comprehend the verification trace, we consider a tiny AMI configuration with two meters, one collector, and one headend. Here, the constraint verification gives a *sat* result. This signifies that the AMI configuration satisfies the resource constraint. Thus, there is no *unsat-core* in this evaluation. However, we get an *unsat* result when the AMI configuration model is modified by setting the buffer size to a reduced value, 100 KB.

The experiment's result is presented in Table 7. It shows that there is no model that satisfies the collector resource constraint. It also shows the *unsat-core*, *i.e.*, the unsatisfied constraints (assumptions). To find a *sat* result, we next run the *max-sat* implementation on the configuration model sequentially by intuitively weakening the configuration constraint following the *unsat-core*. This is done by removing one predicate (among the predicates of the *unsat-core*) from the configuration constraint each time and running the model verification until the verification result converges to *sat*. The resultant satisfiable model indicates the configurations that satisfy the resource constraint. Then, we use the immediately preceding *unsat* trace as the potential cause of the constraint violation. For example, Table 7 shows that the resource constraint (PC) is satisfied with the configuration predicates $P0$ and $P1$. In this case, the number of the type 0 meters and that of the type 1 meters are respectively 6 and 0. The immediately preceding *unsat-core* is "PC $P0$ $P1$ $P2$", which indicates that the predicate $P2$ leads the violation. It can be observed that the predicate $P2$ in the configuration (as shown in Table 6) asserts the number of each type of meters as 8, which leads to the unsatisfiability of the resource constraint when the buffer size is 100 KB.

From the *unsat-core*, it is found that the remediation to the collector resource constraint violation is possible by applying one of the following measures: (i) changing the collector's buffer size, (ii) changing the sampling size or rate of the meter(s), and (iii) changing the number of meters to transmit data to the collector. The buffer size of a collector is basically vendor-specific and this is not configurable except by replacing the collector with a different one (having a larger buffer). This is an example of invariant guidelines, which specifies that the collector's buffer size cannot be considered in the remediation plan. The sampling rate of a meter is also vendor-specific. However, it might be possible to replace a meter with a different one (chosen from the available meters) that has a smaller sampling size or rate. Hence, in our example, we consider the meters connected to the collector as assumptions (the predicates $P0$ and $P1$). It is easy to change the number of meters connected to the collector. We take the assumptions $P2$ and $P3$ corresponding to the number of meters. However, in the diagnosis process, we could assume that the organization is using only one vendor-specific type of meter (*e.g.*, the type 0 meter) and presently the organization is not willing to try different vendors. This is an example of user-driven guidelines, when we would not consider the assumptions $P2$ and $P3$.

2.6 SCADA Security Analysis

In this section, we first briefly discuss the SCADA configurations and their security. Next, we present the formal model corresponding to SCADA security requirements. We conclude this section with an example illustrating the verification of a requirement using our formal model.



IED	**ID**	**Type**	**Report Mode**	**Ports in Service**	**ICS Protocol**	**MAC Address**	**IP Address**	**Patch Info**	**Measurements**
ied	i0001	sel	pull	nil	modbus	00:0a:95:9d:68:16	-	pi001, pi143	9
ied	i0012	abb	pull	nil	dnp3	-	150.12.0.12	pi034	21, 22
PLC	**ID**	**Type**	**Report Mode**	**Ports in Service**	**ICS Protocol**	**Operating System**	**MAC Address**	**IP Address**	**Patch Info**
plc	p0003	siemens	pull	nil	dnp3	vxWorks	-	150.1.10.02	pp031, pp113
plc	p0005	sel	pull	nil	dnp3, modbus	os-9	-	150.11.0.34	pp140
RTU	**ID**	**Type**	**OS**	**Ports in Service**	**ICS Protocol**	**MAC Address**	**IP Address**	**Patch**	
rtu	r001	sel	freeRTOS	nil	dnp3, modbus	00:A0:C9:14:C8:29	150.1.0.101	pr208	
MTU	**ID**	**OS**	**Ports in Service**	**ICS Protocol**	**MAC Address**	**IP Address**	**Patch**		
mtu	m01	linux	22	dnp3, modbus	-	150.2.50.51	p322		
Internal Host	**ID**	**OS**	**Ports in Service**	**Comm Protocol**	**MAC Address**	**IP Address**	**Patch**		
ent-host	h0001	windows-7	_	ip	-	150.200.20.64	p357		
Link	**End 1**	**End 2**	**Status**	**Link Type**					
link	i0001	r001	up	ethernet					
link	p0003	r001	up	ethernet					
link	r001	m01	up	gprs					
Link Profile	**ID**	**Media**	**Shared Status**	**BW**					
link profile	ethernet	wired	yes	10					
link profile	modem	wireless	no	0.028					
Crypto Property	**Device ID**	**Source ID**	**Destination ID**	**Crypto Type**					
crypto prop	i0001	i0001	_	crypt3					
crypto prop	p0003	p0003	-	crypt3, crypt4					
crypto prop	r001	r001	i0001	crypt1					
crypto prop	r001	r001	m01	crypt5, crypt6					
Crypto Profile	**ID**	**Algo**	**Key Length**						
crypto	crypt1	hmac-sha1	128						
crypto	crypt2	hmac-sha256	256						
crypto	crypt3	chap	64						
crypto	crypt4	sha2	128						
crypto	crypt5	rsa	1024						
crypto	crypt6	aes	128						
crypto	crypt7	tls	1024, 128						

Figure 9: An example data of SCADA configurations.

2.6.1 Preliminary

SmartAnalyzer takes necessary SCADA configurations from a template file. Figure 9 shows an example of partial SCADA configurations, particularly with regards to the physical components, the topology, and the communication and security properties. It is worth mentioning that, unlike SmartAnalyzer for AMI, we do not use abstraction to model SCADA components (*i.e.*, the data communicating field devices like IEDs, RTUs, and PLCs), because they are limited in number compared to smart meters in AMI. SmartAnalyzer formally models SCADA configurations and security guidelines as constraints and encodes these constraints into SMT logics. Then, these constraints are verified using the Z3 SMT solver [45] and a violation of these constraints is identified as a threat. In this particular extension of SmartAnalyzer, we do not present the generation process of necessary remediation plans for identified

Table 8: Formal (partial) definition of a remote terminal unit

Remote Terminal Unit (RTU):

$RTU_i \rightarrow Type_i \wedge Patch_i \wedge Mode_i \wedge Crypt_i \wedge OS_i \wedge Serv_i \wedge$
$\quad ICommProto_i \wedge MAC_i \wedge Ip_i$

$Patch_i \rightarrow \bigwedge_{j=0\ldots} Patch_{i,j}$

$Crypt_i \rightarrow \bigwedge_{j=0\ldots} (CAlgo_{i,j} \wedge CKey_{i,j})$

$Serv_i \rightarrow \bigwedge_{j=0\ldots} SPort_{i,j}$

$ICommProto_i \rightarrow \bigwedge_{j=0\ldots} ICommProto_{i,j}$

threats, since the approach is the same as we have presented in Section 2.5.3.4.

2.6.2 Formal Model of SCADA Security Verification

The formal modeling for the SCADA security verification has two parts: the modeling of SCADA configurations and that of the security requirements.

2.6.2.1 SCADA Configuration Model

A SCADA network consists of different types of devices, heterogeneous communication links, and various access control and security policies. The formalizations of these SCADA configurations are similar to those of AMI configurations. Therefore, we present some selective formalizations of these configurations.

2.6.2.1.1 Modeling of SCADA Physical Components

SCADA consists of different physical device components, among which IEDs, PLCs, RTUs, and MTUs are important. Usually IEDs, PLCS, RTUs, and PMUs are associated with substations, while an MTU is associated with a control center. IEDs, PLCs, and RTUs are referred to as field devices. We model each SCADA physical device based on its parameters, as shown in Figure 9. The modeling of physical devices are similar to that of AMI physical devices, except we do not use the class-based abstraction in modeling the field devices. Here we present the formalization of an RTU, while the others have similar modeling.

The formalization of an RTU is shown in Table 8. An RTU is identified by an ID. Its profile *Rtu* is represented as a conjunction of different parameters. Parameter *Type* represents the vendor type (*e.g.*, ABB, SEL, and Siemens) of the RTU. Although an RTU typically delivers data (measurements) to the control server (*i.e.*, MTU, residing at the control center) upon receiving a request from the server, we consider both of the possible reporting modes: pull and push, which is captured by *Mode*. The reporting time schedule of an RTU is modeled similarly to a smart meter or a collector. To achieve end-to-end security, the communicating devices must agree in their cryptographic (authentication and encryption) properties. We model the cryptographic properties of a meter using *Crypt* as a conjunction of algorithm (*CAlgo*) and key length (*CKey*). A device often supports multiple cryptographic properties. The running services associated with an RTU are represented by *Serv*. The communication protocols are specified using *CommProto*. Typically, there are ICS-specific protocols (*e.g.*, modbus, DNP3, etc.) for communication. Since we do not use the class-based abstraction in formalizing an RTU, each RTU has a unique address. The MAC address and the IP address of an RTU are specified using *MacAddr* and *IpAddr*, respectively. Parameter *Patch* denotes the patches that are installed in the RTU.

2.6.2.1.2 Modeling of SCADA Topology

Typically, multiple IEDs/PLCs are connected with an RTU, while all or some RTUs are connected to an MTU directly or through some intermediate RTUs and/or WAN. There can be more than a single MTU, in which case one of them works as the main MTU (corresponding to the main control center), while the rest of the MTUs are connected to the main one. The measurements and control commands flow through this communication topology between the devices. If PMUs exist in SCADA, there is usually a separate network from PMUs to MTUs through WAN, where multiple PMUs are connected to one or more intermediate PDCs and these PDCs feed data to MTUs. Usually communications to control centers (MTUs) are restricted by firewalls. Here, we present the formalizations of communication links.

A link in SCADA is formalized similarly to that in AMI. Identified by an ID, a link is a conjunction of *NodePair* (*i.e.*, the nodes connected by the link), *LinkStatus* (*i.e.*, up or down), and *LType* (*i.e.*, ethernet, modem, etc.). *LType* binds this link with another profile corresponding to the link type. This profile consists of *LinkProp* that represents the properties of that link, including *MediaType* (*i.e.*, wireless, ethernet, etc.), *SharedStatus* (*i.e.*, shared or not), and *LinkBw* (in Mbps). In this topology formalization, we do not use any abstraction, like zone, which we use in the AMI topology modeling (Section 2.5.2.1). Thus, each of the end nodes for a link is a single node, with a specific MAC or IP address.

2.6.2.2 SCADA Threat Verification Model

The potential threats to SCADA are very similar to the threats that we see in the case of AMI. The security controls specified in NIST SP 800-82 [13] also comply with those specified in NISTIR [12] guidelines. As they share similar logics, most of the invariant and user-driven constraints modeled in Section 2.5.2 are also applicable to SCADA with no or minor modification in logics. Therefore, in this particular work we consider those security constraints that are specific for SCADA. These constraints are mainly user-driven requirements, which ensure if a SCADA control process has sufficient secure (particularly, authenticated and integrity protected) data to provide correct results in normal cases or contingencies. In this case, we consider the observability analysis, a prior and crucial requirement for performing the power system state estimation control routine [1, 49]. We consider the following user-driven security constraints: (i) secured observability, (ii) k-resilient secured observability, and (iii) bad data detectability. These constraints inherently use different invariant constraints like reachability and security pairing within their formalizations. To model these constraints, we also define another constraint named secured data delivery, similar to assured data delivery.

2.6.2.2.1 Secured Data Delivery Constraint

The assured data delivery constraint verifies whether data can reach from the source to the destination, particularly a field device to the MTU through zero, one, or more intermediate devices, without ensuring whether the transmission has occurred under necessary security measures. Although this constraint checks security pairing between the communicating parties, it is only to ensure necessary handshaking for communication. In the secured data delivery constraint (*SecuredDelivery*), we verify whether data is sent under proper security measures, particularly authentication and integrity protection including the assured data delivery. That is, the communicating nodes, *e.g.*, an RTU and the MTU, may have correct security pairing, as they are using the same security protocol, *e.g.*, Challenge-Handshake Authentication Protocol (CHAP). However, this security pairing on CHAP only ensures secure authentication. In this case, the transmission will not be data integrity protected. Moreover, we need to consider the vulnerabilities of the security measures in use. For example, if DES (Data Encryption Standard) is used for data encryption, the transmitted data cannot be considered as protected, as a good number of vulnerabilities of DES have already been found. We take expert and widely accepted knowledge about the security/cryptographic measures applied to the communication in order to realize whether the data is authenticated and integrity protected. Table 9 shows the formalization of this constraint, in which we also define two sub-constraints *Authenticated* and *IntegrityProtected*, which ensure the authentication of the communicating parties and the integrity of the transmitted data, respectively.

2.6.2.2.2 Secured Observability Constraint

The power system is observable when the measurements can solve a list of unknown variables. Each of these variables stands for a state. Typically, each measurement represents a power equation. Therefore, we need to know each equation regarding a particular measurement, where the equation specifies the variables that produce

Table 9: Formalizations of the secured data delivery constraint for SCADA (partial)

Secured Data Delivery Constraint:

$Authenticated_{S,D} \rightarrow$
  $CryptProps_{S,D} \wedge$
  $\forall_K (K = CryptType_{S,D} \wedge$
    $((Algo_K = hmac \wedge KeyLength_K \geq 128) \vee \ldots))$

$IntegrityProtected_{S,D} \rightarrow$
  $CryptProps_{S,D} \wedge$
  $\forall_K (K = CryptType_{S,D} \wedge$
    $((Algo_K = sha2 \wedge KeyLength_K \geq 128) \vee \ldots))$

# An IED is directly connected to the MTU

$SecuredDelivery_{I,M} \rightarrow$
  $Ied_I \wedge MTU_M \wedge$
  $AssuredDelivery_{I,M} \wedge Authenticated_{I,R} \wedge IntegrityProtected_{I,M}$

# An IED is connected to the MTU through an RTU

$SecuredDelivery_{I,R,M} \rightarrow$
  $Ied_I \wedge Rtu_R \wedge Mtu_M \wedge$
  $AssuredDelivery_{I,R,M} \wedge Authenticated_{I,R} \wedge Authenticated_{R,M}$
  $IntegrityProtected_{I,R} \wedge IntegrityProtected_{R,M}$

# Other possible communication paths from field devices toward the MTU

$SecuredDelivery_{I,R,R',M} \rightarrow$

....

this measurement. As we have already mentioned, we consider the state estimation control routine in this formalization. In state estimation, there is a Jacobian Matrix that represents the relationships between the measurements and the unknown variables [49]. The secured observability constraint ensures two conditions: (i) the authenticated and data integrity protected distinct measurements can cover all the variables (*i.e.*, unknown states), and (ii) the number of these measurements is greater than or equal to the number of variables. These two conditions are minimal requirements to ensure that there is a secure as well as a unique solution. There are often more than one measurement that actually represents the same electrical component. For example, the power flow through a line can be measured at both ends of the line [49]. Therefore, these two measurements (forward line power flow and backward line power flow) represents the same electrical component. In the following we describe the formalizations of this constraint.

Each row of the Jacobian matrix has a set of entries (column values), where each entry is associated with a state/variable:

$$\begin{bmatrix} h_{1,1} & h_{1,2} & \cdots & h_{1,n} \\ h_{2,1} & h_{2,2} & \cdots & h_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ h_{m,1} & h_{m,2} & \cdots & h_{m,n} \end{bmatrix}$$

In the matrix, $h_{i,j}$ is an entry where i is the row number associated with measurement i and j is the column number associated with variable j. This entry is often zero, *i.e.*, the variable corresponding to this entry does not influence the measurement value associated with this row. Therefore, the variables corresponding to the nonzero entries only have impact on the measurement. Let X define a state/variable while Z defines a measurement. Let $1 \leq X \leq n$ and $1 \leq Z \leq m$. We define $StateSet_Z$ as the set of states that constitute measurement Z. $StateSet_Z$ is built as follows:

$$\forall_X X \in StateSet_Z \rightarrow h_{Z,X} \neq 0$$

If two measurements represent the same electrical components, their corresponding rows should have non-zero entries on the same columns, and they must be the same values, although the direction (sign) can be the opposite (*e.g.*, forward and backward line power flows). We define $UMsrSet_E$ as the set of measurements that represent the same electrical component, E. These sets are built using a clustering process, where each pair of sets, $UMsrSet_E$ and $UMsrSet_{E'}$, satisfy the following property:

$$\forall_{Z \in UMsrSet_E} \forall_{Z' \in UMsrSet_{E'}} \exists_X \ (h_{Z,X} \neq h_{Z',X}) \wedge (h_{Z,X} \neq -h_{Z',X})$$

Similarly, there are power consumption measurements at buses, each of which corresponds to a particular bus. The power consumption at a bus is the summation of the all power flows incident to that bus. Thus, if all of these power flows are received as measurements, then the bus consumption measurement is redundant (*i.e.*, not unique). We also consider this condition to get the unique bus consumption measurements.

A field device can be responsible for delivering one or more measurements. Therefore, from the mappings between communicating field devices and measurements, we can logically identify which measurements are secure, while from the mappings between the measurements and the states, we can find out whether the secure measurements can observe the system. Let us assume that $IedSet$ is the set of IEDs that are responsible for taking the necessary measurements (meters/sensor data) and sending them to the MTU through an RTU. Let $MsrSet_I$ be the set of measurements transmitted by IED I, while S_Z is a Boolean variable denoting whether measurement Z is secure. A measurement is secure if the following two conditions hold:

$$\forall_{I \in IedSet} \ \forall_{Z \in MsrSet_I} \ (\exists_R \ SecuredDelivery_{I,R,M}) \rightarrow S_Z$$

$$\forall_Z \ \ S_Z \rightarrow \exists_{I \in IedSet} \ (Z \in MsrSet_I) \wedge (\exists_R \ SecuredDelivery_{I,R,M})$$

If a measurement is secured, then the variables corresponding to this measurement can be securely estimated. Let SE_X denote whether state X can be securely estimated or not. Then, the following formalization specify when a state is securely estimated:

$$\forall_Z \ \ \forall_{X \in StateSet_Z} \ \ S_Z \rightarrow SE_X$$

$$\forall_X \ \ SE_X \rightarrow \exists_Z \ S_Z \wedge (X \in StateSet_Z)$$

We define $SecUMsr_E$ to denote whether one or more measurements within $UMsrSet_E$ are secured:

$$\forall_E \ \ \exists_{Z \in UMsrSet_E} \ \ SecUMsr_E \rightarrow S_Z$$

Now, we formalize the secured observability constraint (*SecuredObservability*), ensuring that each state/variable is covered by the secure measurements and the minimum number (*i.e.*, at least m) of secure measurements (*i.e.*, equations):

$$\mathit{SecuredObservability} \rightarrow (\forall_X \ SE_X) \wedge (\sum_E \ SecUMsr_E \geq m)$$

2.6.2.2.3 k-Resilient Secured Observability Constraint

This constraint verifies whether secured observability is ensured even if k field devices (*i.e.*, IEDs and RTUs) are unavailable. A device can be unavailable because of its failure to communicate with the MTU or the next device toward the MTU, due to its technical failure or remote attacks (*e.g.*, DoS) on it or the path toward the destination. In this constraint modeling, we assume RTU failures, although formalizations are the same for IEDs or others.

In order to model this constraint, we first define a parameter for each field device to denote whether that device is unavailable. Let $URtu_R$ be denote whether RTU R

is unavailable. Therefore, the following relation holds:

$$\forall_R \ \ URtu_R \rightarrow \neg Rtu_R$$

We formalize the k-resilient secured observability constraint (*ResilientSecuredObservability*) as follows:

$$ResilientSecuredObservability \rightarrow \forall_{\sum_R URtu_R \leq k} \ SecuredObservability$$

The above formalization of the k-resilient secured observability constraint needs to execute all possible combinations of RTU failures up to the number k, which is not efficient. Therefore, we devise an efficient but heuristic-based modeling of this constraint as follows. Let $SRtuMsr_{R,Z}$ denote whether measurement Z is securely transmitted from the IED associated with this measurement to the MTU through RTU R. Then, we define $SRtuMsr_{R,Z}$ as follows when there is only one intermediate RTU in the transmission path:

$$SRtuMsr_{R,Z} \rightarrow \exists_I \ (Z \in MsrSet_I) \wedge SecuredDelivery(I, R, M)$$

When, there are two or more intermediate RTUs in this transmission path, the RTU at the top of the hierarchy is the most critical one with respect to the availability of these RTUs. Therefore, we only consider that RTU in defining $SRtuMsr_{R,Z}$, as shown in the below for two intermediate RTUs:

$$\forall_R \, \forall_Z \ \ SRtuMsr_{R,Z} \rightarrow \ \exists_I \ (Z \in MsrSet_I) \wedge \exists_{R'} \ SecuredDelivery(I, R', R, M)$$

Let $SRtuState_{R,X}$ denote whether RTU R securely transmits a measurement that is influenced by state X. Then, the following constraint holds:

$$\forall_R \, \forall_X \;\; SRtuState_{R,X} \rightarrow \exists_Z \; SRtuMsr_{R,Z} \wedge (X \in StateSet_Z)$$

A state can be securely estimated even after k failures of RTUs, considering the worst case when all of these unavailable RTUs are responsible for transmitting different measurements associated with this state, if there is at least one available RTU that also transmits one or more measurements corresponding to that state. Moreover, we also need to ensure that there are sufficient unique measurements to observe the system. We must consider the number of missing unique measurements when k RTUs are unavailable. The most conservative calculation of this number is taking the maximum possible missing unique measurements for k unavailable RTUs. Since it needs union operations of unique measurement sets for all combinations of k RTUs, we take a weak heuristic to calculate this number by considering the average number of unique measurements that an RTU transmits. However, this heuristic approach cannot ensure the observability. A sound and complete, as well as time efficient, modeling of this constraint remains as a topic for future research. Let us define $SecUMsrRtu_k$ to denote the number of missing unique measurements when k RTUs are unavailable. Now, k-resilient secured observability constraint is formalized as follows:

$$\begin{aligned} ResilientSecuredObservability \rightarrow \forall_X \; (\sum_R SRtuState_{R,X} \; \geq k+1) \\ \wedge \; ((\sum_E \; SecUMsr_E) - SecUMsrRtu_{E,k} \geq m) \end{aligned}$$

2.6.2.2.4 Bad Data Detectability Constraint

The obtained measurements for observability must be able to detect bad data. Note that a measurement can be secured or trusted, but the data itself can be an outlier due to containing noise (random variations) and other inaccuracies at the censor/meter corresponding to this measurement, or the censor/meter being compromised. If there is a single measurement associated with a state, then the measurement is a critical one and it is not possible to detect if that measurement is bad. Therefore, in order to detect bad data it is required to have at least two measurements corresponding to each state, if we assume no more than one measurement among them can be bad at a time. We can generalize the bad data detectability constraint as k, r-resilient bad data detectability, where if k RTUs (IEDs) are unavailable, the bad data is detectable even if r measurements are available. It is worth mentioning that we only rely on secured measurements for detecting the bad data, since non-secured measurements cannot be trusted [44]. We model this constraint by extending the formalization of the previous constraint, where we assume the failures of RTUs only:

$$ResilientBadDataDetectability \rightarrow \forall_X \ (\sum_R SRtuState_{R,X} \ \geq k + r + 1)$$

In this modeling, we consider a worst case scenario where an RTU, corresponding to a state, may be involved with transmitting only a single measurement associated with that state.

2.6.3 Implementation

In this section, we briefly discuss the implementation of the model and illustrate the model's execution with an example.

2.6.3.1 SMT Encoding

Similar to the case of implementing AMI security verification formalizations, we use SMT to encode the formalizations for SCADA security verification in SmartAnalyzer. In this case, we use the same SMT solver, Z3, but a python-based API for encoding the formalizations. The solution to the model gives the result *sat* or *unsat*. In the case of *sat*, from the results we can find out the detailed scenario that makes the constraints/requirements satisfied. For example, in our particular modeling it will show us the measurements that are secure (authenticated and integrity protected), as a result of which the observability is secure. In the case of *unsat*, taking the *unsat-core* we can trace the constraint violations, *i.e.*, the potential threat points.

2.6.3.2 An Example

In our example, we demonstrate the k-resilient secured observability constraint. We consider a small 5-bus SCADA system, as shown in Figure 10. The corresponding input is partially presented in Table 10. The input includes primarily the Jacobian matrix corresponding to the SCADA system, the topology (connectivity between the communicating devices and the association of the measurements with the IEDs), and security profiles of each communicating host pair. We assume that the measurements are taken/collected by different IEDs only and these measurements are sent to an

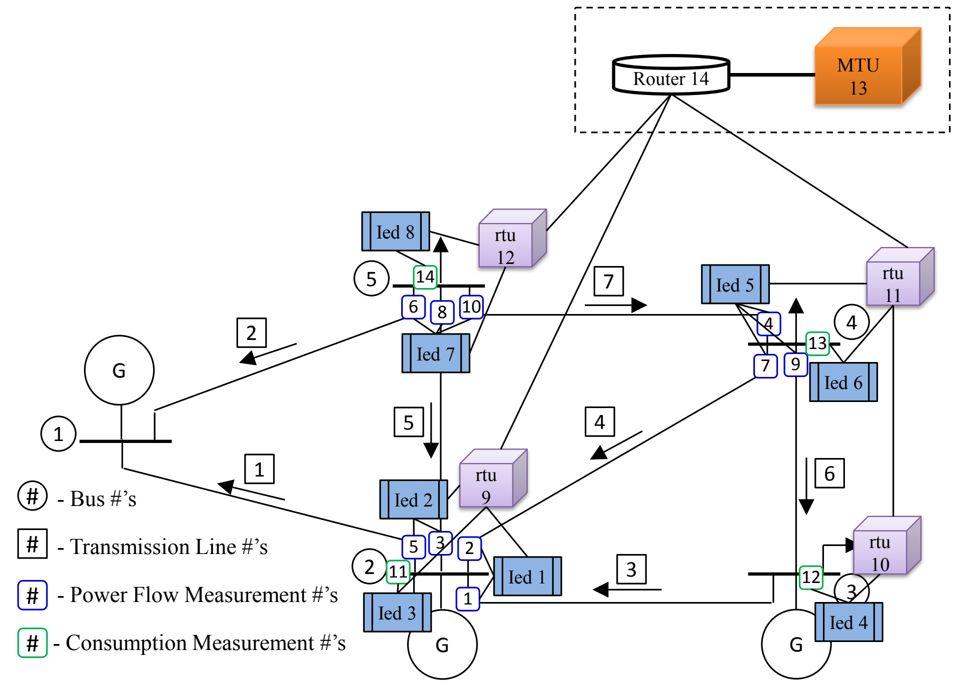


Figure 10: An example SCADA topology of a 5-bus power grid.

MTU (*i.e.*, the SCADA server at the control center) through RTUs. Each row of the Jacobian matrix corresponds to a measurement (first row corresponds to measurement 1, and so on). Each row has 5 entries (columns) for each state (corresponding to each bus). The resiliency requirements specify that the secured observability must be satisfied even if an IED or an RTU is unavailable. The solution to the formal model corresponding to this example returns a satisfiable answer, *i.e.*, the given SCADA system is securely observable. From the assignment to the variables, it is found that all of the measurements, except measurements 1, 2, and 12, are transmitted securely to the MTU, and these measurements are sufficient to observe the system even an IED or an RTU is unavailable. However, if we lower the security properties of the communication from RTU 9 to the MTU to authentication-only (*e.g.*, hmac [50]), then there is no solution even if we increase those of RTU 10 to both authentication

Table 10: The input for an example of verifying secured observability in a SCADA

```
# Number of states and measurements
5 14
# Jacobian matrix (the relation between the states and the measurements)
0 -5.05 5.05 0 0
0 -5.67 0 5.67 0
0 -5.75 0 0 5.75
0 0 0 -23.75 23.75
16.9 -16.9 0 0 0
4.48 0 0 0 -4.48
0 5.67 0 -5.67 0
0 5.75 0 0 -5.75
0 0 5.85 -5.85 0
0 0 0 23.75 -23.75
-16.9 33.37 -5.05 -5. 67 -5.75
0 -5.05 10.9 -5.85 0
.........
# Number of each type of devices in the topology
# IEDs (Id 1 to 8), RTUs (Id 9-12), MTU (Id 13), Router (Id 14), and Firewall (Id 15)
8 4 1 1 1
# Connectivity (among the IEDs, the RTUs, and the MTU)
.........
# Measurements corresponding to IEDs
1 1 2
2 3 5
3 11
4 12
5 4 7 9
6 13
7 6 8 10
8 14
# Security profile (source, destination, applied cryptographic algorithms/keys)
11 # Number entries of security profiles
1 9 hmac 128
2 9 chap 64 sha2 128
3 9 chap 64 sha2 128
5 11 chap 64 sha2 256
6 11 chap 64 sha2 256
7 12 chap 64 sha2 128
8 12 chap 64 sha2 128
9 13 rsa 2048 aes 256
10 13 hmac 128
.........
# k-resiliency requirements (IED, RTU)
1 1
```

and data integrity protection (rsa and aes [51, 52]). This is because RTU 9 is more critical than RTU 10, as the former node is responsible for transmitting a greater number of measurements than the latter. In this case, if we reduce the resiliency requirement to unavailability of a single IED only, while no unavailability of RTUs, then there is again a satisfiable solution.

2.7 Evaluation

We first evaluate the accuracy of SmartAnalyzer. Then, we evaluate the scalability of the tool. We analyze the tool by evaluating different constraints under synthetic configuration data.

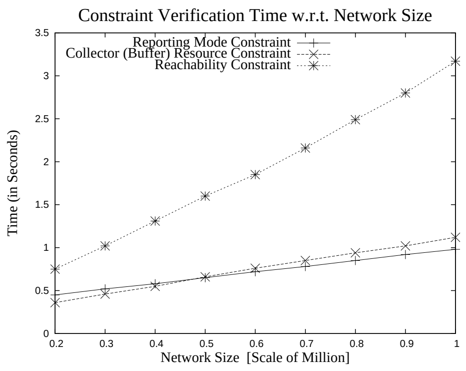
2.7.1 Accuracy

We evaluate our tool, particularly the AMI security verification model, with ground truth scenarios by deploying it in a small AMI testbed created at our university [18]. The testbed setup typically represents a small subset of the network as shown in Figure 3. We analyze some of the security constraints, especially data overwrite protection and cyber bandwidth constraints. The results of our tool are cross-validated with the real scenario. For the purpose of analyzing the constraints, we slide the values of different configuration parameters, such as taking very low and high pull schedule intervals for the headend, and changing the bandwidth of the links from high to very low. We find some constraint violations that lead to link flooding and data loss. In addition, we inject a high amount of data through the simulation (by adding multiple simulated collectors in the testbed) to observe the effect on the cyber bandwidth constraint. After observing the constraint violations, we reconfigure the setup according to the remediation plan and reevaluate the constraint to see the effect. For example, in the case of a cyber bandwidth constraint violation, we add traffic limits to the firewall rules and observe the mitigation of link flooding. These tests significantly help us in verifying the accuracy of the tool.

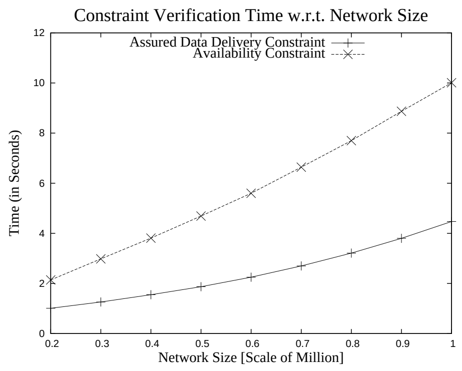
2.7.2 Scalability Evaluation

2.7.2.1 Methodology

We evaluate the scalability of SmartAnalyzer by analyzing the time and memory required in constraint verification by varying the smart grid network size. In this scalability evaluation, we again focus on AMI, since the number of devices, particularly the smart meters, in an AMI system is much larger than the number of field devices in a SCADA system. We consider the size of AMI as the total number of collectors in AMI (the number of meters are proportional to the number of collectors). The number of collectors depends on the number of collector zones and their sizes. We consider only a single headend zone (10 headends of two headend classes) in the network. We take 100 and 50 meter and collector classes respectively, while each collector is connected with 10 meters (of 2 random meter classes) on average. Each collector zone consists of around $1,000$ collectors (of 5 random classes). We keep the values of these parameters fixed in most of the experiments, except in those cases where their impacts on the scalability are analyzed. We use the Z3 SMT solver [17] to execute our formal model. We run the experiments in an Intel Core i3 processor with 4 GB memory and 32 bit operating system.

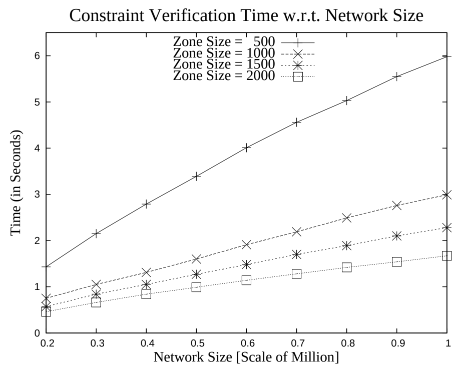


(a)

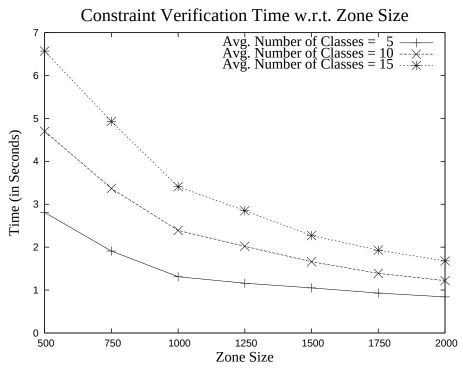


(b)

Figure 11: Impact of network size on (a) invariant and (b) user-driven constraints verification time.



(a)

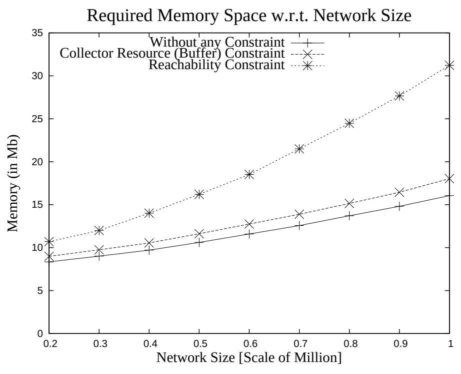


(b)

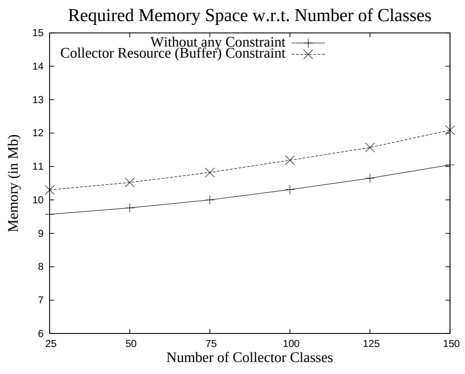
Figure 12: Impact of (a) the zone size and (b) the number of collector classes per zone on constraint verification time.

2.7.2.2 Evaluation Results

Impact of the network size: Figure 11(a) and Figure 11(b) show constraint verification time with respect to network size. We show the verification time for different invariant constraints (*i.e.*, reporting mode, collector resource, and reachability) and user-driven constraints (*i.e.*, assured data delivery and availability protection constraint). A significant part of the constraint analysis time is covered by the modeling



(a)

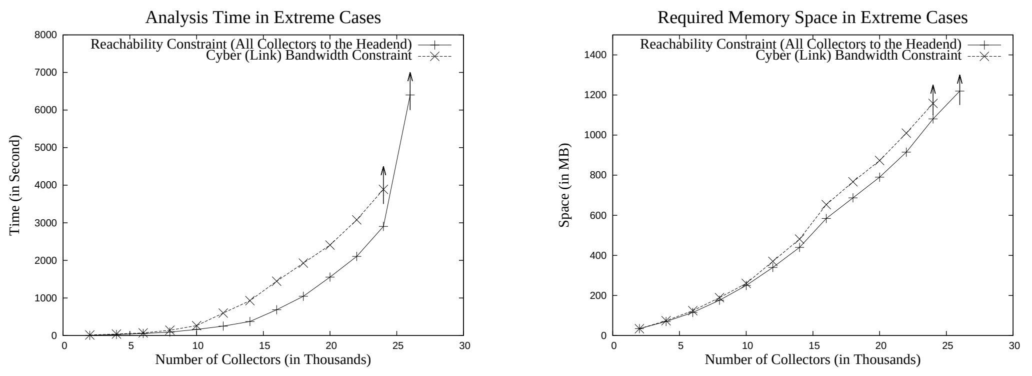


(b)

Figure 13: Impact of (a) the network size and (b) the number of classes on memory requirement.

time, which is almost linearly dependent on the network size that varies with the number of zones. Verifications of some constraints (*e.g.*, reachability) involves all (or a large portion of) possible potential source/target nodes that implicitly increase with the number of zones. Thus, the verification time of such constraints increases more with the size of the network than that of the constraints (*e.g.*, collector resource), which are involved with the class size only. Usually, the user-driven constraint analysis time is more than the invariant constraint analysis time (as shown in Figure 11(b)), since most of the former type of constraints subsume the later type of constraints.

Impact of the zone size and the member classes: We evaluate constraint verification time with respect to the network size considering different network zone sizes. This analysis is shown in Figure 12(a) with respect to the reachability constraint. We observe that the analysis time significantly reduces with the increase in the number of collectors in the zone. This is due to the fact that the number of zones decreases as the zone size increases, which in turn decreases the overall model size and the



(a) (b)

Figure 14: Impact of the AMI network size on the analysis time and the memory requirement for cyber bandwidth constraint verification, which show that the analysis of this constraint may fail when the number of collectors increases more than some certain labels.

potential sources/targets. Figure 12(b) shows the constraint verification time taking a fixed zone size and varying the number of average classes per zone. We find that the time increases if the variation among the classes increases.

SMT memory requirement: The memory requirement of the SMT solver [17] is evaluated by changing the network size (*i.e.*, number of zones) and the number of classes. Such analysis results are shown in Figure 13(a) and Figure 13(b). We observe almost linear growth of the memory with the network size. Similar to the analysis time, the memory for constraint verification is the sum of the memory for modeling of AMI configurations and that for modeling a constraint. The figures justify this by showing that less memory is required when no constraint is verified. The constraints involving more quantifiers require larger memory memory for encoding. Figure 13(a) shows such a comparison between collector resource and reachability constraints.

Extreme cases when the SMT solver fails : If the model size increases significantly, SmartAnalyzer may fail to give a solution. An increase of the model size depends not only on the problem size but also on the constraint type. We present this event in Figure 14(a) and Figure 14(b). In the figures, we show the time and memory requirements of verifying the all reachability constraint (*ReachableConstr* for all collectors to the headend) as well as the cyber bandwidth constraint (*LinkBwConstr*). The modeling of *LinkBwConstr* also requires knowing all the traffics (and traffic size) from collectors to the headend passing through a link at a particular time (according to the reporting schedules). Due to the modeling of all possible traffics between the collectors and the headend, the model size becomes very large. The figures show that if the number of collectors increases more than 23 thousand (arrow sign in the figure), the cyber-bandwidth constraint verification fails. Similarly, the reachability satisfaction constraint fails if the number of collectors is over 26 thousand. These failures happen due to the *out-of-memory-exception* given by the SMT solver. Figure 14(b) shows the memory consumed by the model. However, the 64 bit implementation of our model overcomes this memory explosion problem.

2.7.3 Discussion

SmartAnalyzer can successfully identify possible threats on AMI and SCADA by constraint satisfaction checking. It is highly scalable with the network size. However, there are a couple of limitations of the tool. First, we use device and property level abstraction, particularly in the case of threat verification for AMI, to achieve scalability under large scale smart grid configurations, which in turn may not provide fine-grain attack paths. Second, since an SMT solver is used as the core analysis engine, different normalizations are considered for real valued calculations. Moreover, the tool does not provide the functionality for analyzing some of the inherent smart grid security properties, such as LonTalk, DNP3, or Modbus protocol configurations.

2.8 Conclusion

Automated analysis of smart grid configurations is an important but challenging problem. A smart grid contains a large number of cyber and physical devices that exhibit highly dependent configuration parameters, which makes potential misconfigurations and security vulnerabilities likely. In this chapter, we present automated smart grid configuration verification, diagnosis, and repair techniques, that are implemented in a tool called SmartAnalyzer. We particularly focus on AMI and SCADA, two major components of a smart grid. We define various AMI and SCADA specific constraints and requirements that are important for protecting smart grids from various security threats. According to these constraints and smart grid configurations, we create logic-based threat verifying models and we use SMT to solve these models as constraint satisfaction problems. Our implemented tool performs static configuration analysis in order to determine potential threats due to violations to the smart grid security requirements. We evaluate the accuracy and scalability of our presented tool using different test configurations and we achieve significantly high scalability by applying the property level abstractions in the model. The constraint verification time lies within few seconds for a network of thousands of collectors.

CHAPTER 3: AUTOMATED SYNTHESIS OF RESILIENCY ARCHITECTURE

Today, organizational security requirements are complex due to the extensive use of various network services and newly evolving security threats. In addition, most organizations are emphasizing not only on the security enforcement but also on being resilient against attacks, while, at the same time, requiring the satisfaction of different business constraints, particularly on usability and deployment cost. The problem of providing strong resiliency, which includes security as well, in a network by exploring different design alternatives, as well as resolving the contention between the resiliency requirements and the business constraints, is important. In this chapter, we address this research problem for smart grids.

We first develop a framework for the automated design of isolation-based resiliency architectures for cyber systems in smart grids. The literature review shows that this problem has not been addressed comprehensively even for traditional IT networks. Therefore, we discuss this framework from a general network's point of view. Next, we provide another framework for the automated design of redundancy-based resiliency architectures for smart grids, particularly AMI. With the first framework we achieve pre-attack resiliency through resistance against potential attacks, while with the second framework we receive during-attack resiliency through robustness.

3.1 Challenges

A system is resilient if it can run its mission (services) under attacks. The resiliency of a system is a combination of pre-attack, during-attack, and post-attack resiliency. The pre-attack resiliency is the resistance against attacks, which is the result of the deployed security configurations. The during-attack resiliency is the robustness of the system, *i.e.*, alternative ways to run the mission if some hosts or links fail due to contingencies or attacks. The post-attack resiliency is the capability of the system to recover/restore swiftly such that damage is minimal due to the disruption of services.

The pre-attack resiliency of a host is ensured by isolating it from others in order to restrict unnecessary access to this host. The resiliency requirements of such kind can be indicated by isolation measures among the hosts. The isolation patterns are defined based on different security devices and their capabilities. An isolation pattern signifies the type of security resistance, *e.g.*, traffic filtering (firewall), trusted communication (IPSec), and payload traffic inspection (IDS). However, a resiliency design has to satisfy the business constraints of the organization, which are mainly represented in terms of usability and deployment cost. The connectivity requirements influence usability, which define the essential traffic/service flows between various network devices. The implementation of isolation measures significantly affects these constraints. For example, the use of an IPSec-based isolation pattern might reduce the usability by causing some applications to be inaccessible for a host, while the use of a firewall-based access denial would give no usability. Therefore, it is necessary to determine resiliency configurations by exploring different isolation design alternatives that maintain the resiliency and usability within an expected level. Moreover, since there is a cost to deploy a security device, it is also required to find an isolation design at an affordable cost. Thus, the pre-attack resiliency architecture design is a challenging task.

We have similar challenges to design during-attack resiliency measures. For example, in order to make a system robust we need to implement redundant communication paths, backup/alternative collecting/forwarding devices (*i.e.*, collectors), etc., each of which requires additional cost. The usability can also be affected due to implementing the plan for robustness. For example, although a mesh network of smart meters provides robustness from communication path failures [53], it offers delayed data transmission compared to direct communication from smart meters to the collector. Designing a strong resiliency architecture manually is burdensome, even for a security expert. Again, misconfigurations are common in manual design. Therefore, the automated synthesis of a strong resiliency architecture is a crucial research problem.

3.2 Related Work

In this section, we first present previous research works that focus on security management of cyber systems in general. Then, we describe prior research on resiliency of smart grids in particular.

3.2.1 Security Verification and Hardening

Throughout the last decade, security policy misconfigurations have been studied extensively [31, 54, 55, 56, 21]. These works formally define configuration anomalies and propose algorithms to discover configuration inconsistency as well as the safe deployment of single or multiple security firewalls. These works follow the traditional bottom-up approach of analyzing existing security policies, which cannot be used to automatically synthesize policies based on business requirements.

Several studies have been done on attack graph based security configuration analysis. For instance, Sheyner et al. [57] present techniques for automated generation and analysis of attack graphs. Noel et al. [58] present an attack graph-based security hardening technique that considers costs of the security measures and minimizes the overall hardening cost. Noel and Jajodia [33] propose a technique to place IDS sensors and prioritize IDS alarms using attack graph analysis. The IDS sensors are placed to cover all attack paths. Dewri et al. [34] have modeled the selection of security hardening measures to minimize the residual damage in a predefined attack graph within a certain budget. Several works propose to find optimal deployment of security devices using attack graphs in order to block all attack scenarios [59, 35]. Several risk analysis-based techniques have also proposed for security management [60, 61, 62, 63]. However, these works cannot automatically find optimal security or resiliency configurations nor the security device placements within the deployment budget and usability constraint.

The research on the automated security and resiliency configuration synthesis is in a premature stage. ConfigAssure [64] is a requirement solver, which is close to this synthesis idea. The tool takes security requirements and configuration variables as inputs and produces the values of the configuration variables as outputs that make the requirements true. ConfigAssure requires complete and well-defined properties and it cannot satisfy optimal security requirements based on isolation, usability, and deployment cost. Zhang and Ehab [65, 66] present procedural approaches for generating firewall configurations considering the device deployment cost. However, they address only the generation of firewall policy configurations without considering different isolation measures (*i.e.*, firewalls, IPSec, IDS, etc.) in the context of resiliency requirements and usability and deployment cost constraints. In addition, their work cannot find the optimal placements of security devices in the network and it does not relate security device placements (according to the topology) with the computation of residual risks. Hence, although the above described works can generate some sorts of security design architectures, they do not consider the security or resiliency requirements and business constraints together and cannot explore various resiliency design alternatives to determine satisfiable resiliency configurations.

3.2.2 Resiliency of Smart Grids

The above-mentioned existing research works are often not suitable for the security or resiliency analysis of smart grids as those works do not consider the relations between the cyber and physical components and their interactions. As we have seen in our description about the related research in smart grid security in Chapter 2, a significant number of works (*e.g.*, [38, 12]) have been initiated over the last several years that describe the interoperability among heterogeneous smart grid components, including security issues based on different attack scenarios. McDaniel et al. [26, 40] discuss the security and privacy challenges in smart grid networks with case studies, while Wang et al. [41] present an artificial intelligent based approach for analyzing risks in smart grid networks. Anwar et al. [42] propose a framework for modeling power grids and their control elements using first order logic. A study on false data injection attacks in power grids is performed by Liu et al. [44]. To understand the security of an AMI system, McLaughlin et al. [27] present a penetration testing-based mechanism. SmartAnalyzer, which we have presented in Chapter 2, is a formal model-based tool for provably analyzing operational consistency and security controls in AMI systems. This tool provides possible remediation plans for constraint violations, which are useful for reconfiguration planning toward security hardening. However, all these works follow the traditional bottom-up approach of analyzing existing or deployed security policies.

With regards to automated security or resiliency architecture design for CPS, particularly smart grids, to our knowledge, no substantial research has been done. Since there is an important need to solve the problem of synthesizing smart grid's configurations to ensure resiliency of the system within the grid operator's capability, our work focuses to meet this need.

3.3 Contributions

In this work, we develop two frameworks to solve the resilience architecture synthesis problem. First, we develop a formal framework that automatically synthesize pre-attack resiliency architecture. It is worth mentioning that, the pre-attack resiliency primarily refers to the traditional proactive security against attacks. The synthesized resiliency architecture includes isolation-based network security configurations and physical placements of security devices. The framework is named *Configuration Synthesizer* or *ConfigSynth*. The framework takes the network's topology, isolation-based resiliency requirements, and business constraints as inputs, formulates the resiliency architecture synthesis model, and solves the model to get the configuration values. ConfigSynth is a novel framework that incorporates the security device placements in the network topology within the design in order to model the deployment cost, which is crucial for a resiliency architecture. The framework can be used as a decision support system to find optimal resilience configurations for a network by exploring different design alternatives.

Second, we develop a framework that automatically synthesizes AMI configurations that satisfy during-attack resiliency requirements within resource constraints. Here, our focus is to add necessary redundancy to the system and configure corresponding parameters in order to satisfy resiliency requirements in terms of operational integrity and robustness properties. These properties mainly ensure the successful delivery of data both in normal (no failure) and partially failed (link or device failures due to accidents or attacks) environments. We also consider other organizational requirements like data freshness. The deployment budget is the main resource constraint. This framework is easily extensible for modeling further resiliency requirements.

3.4 Overview of the Configuration Synthesis Framework

The security or resiliency configurations synthesizer, ConfigSynth, follows a top-down design automation approach instead of the traditional bottom-up security analysis approach. The major contributions of ConfigSynth are as follows:

- It formally models the network topology, resiliency requirements, and business (organizational/user-driven) constraints.

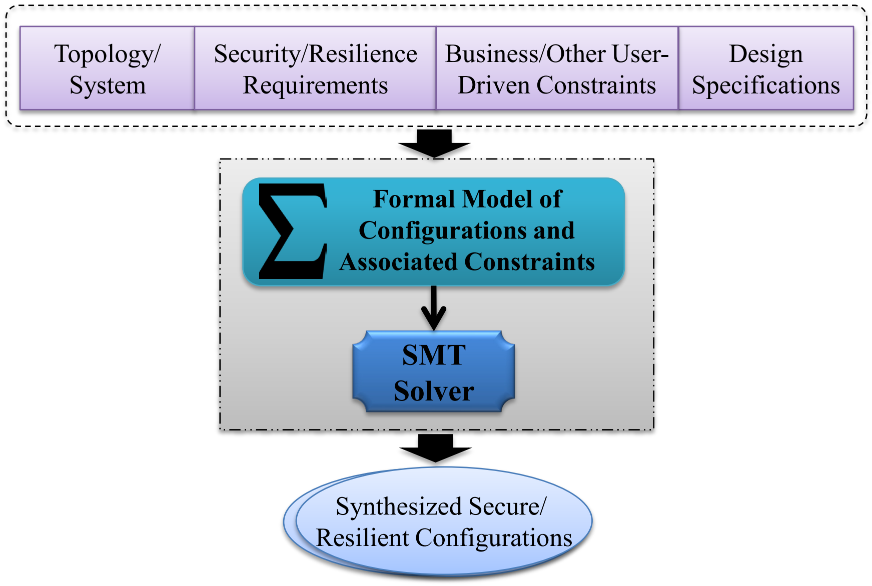

Topology/ System
Security/Resilience Requirements
Business/Other User-Driven Constraints
Design Specifications
Formal Model of Configurations and Associated Constraints
SMT Solver
Synthesized Secure/ Resilient Configurations

Figure 15: The configuration synthesis (ConfigSynth) framework.

- It formalizes the resilience architecture synthesis problem as the determination of appropriate resiliency measures satisfying the given requirements/constraints.
- It encodes the synthesis problem using SMT logics and solves the model using an SMT solver.

The ConfigSynth framework is shown in Figure 15. ConfigSynth takes the following as its main inputs: (i) the current/existing system (*e.g.*, network topology), (ii) resiliency requirements, and (iii) business constraints, and (iv) design specification or guidelines. We model the relations/interdependency among the security or resiliency measures with the necessary system configurations, their impacts on operations, as well as the expense regarding their deployment/implementation. ConfigSynth formalizes the resiliency architecture synthesis as the conjunction of the constraints that guide the design of the resiliency configurations. In ConfigSynth, the formalizations

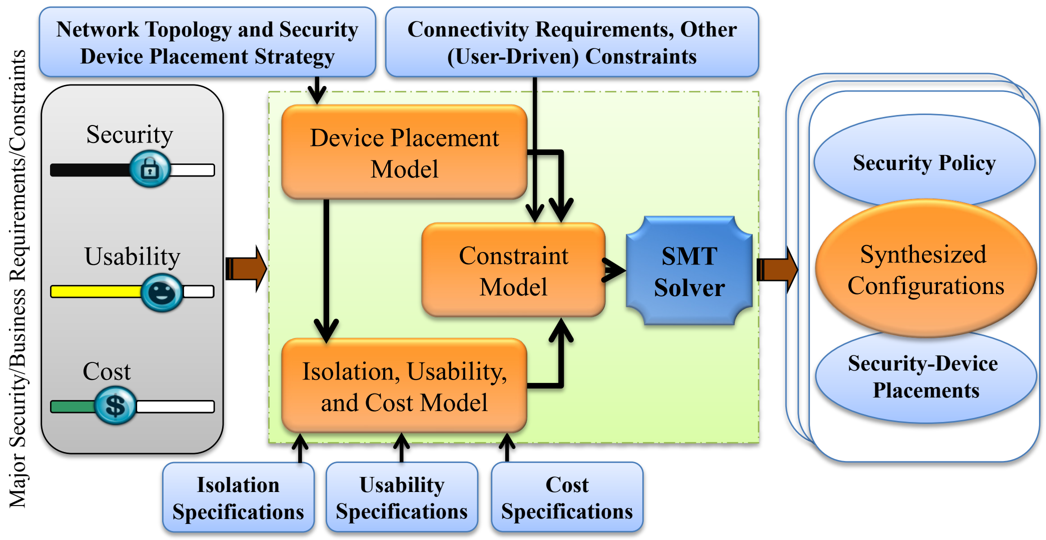

Network Topology and Security Device Placement Strategy
Connectivity Requirements, Other (User-Driven) Constraints
Major Security/Business Requirements/Constraints
Security
Usability
Cost
Device Placement Model
Constraint Model
SMT Solver
Isolation, Usability, and Cost Model
Security Policy
Synthesized Configurations
Security-Device Placements
Isolation Specifications
Usability Specifications
Cost Specifications

Figure 16: The isolation-based resiliency architecture synthesis framework.

are encoded using SMT logics. The formal model is solved using Z3, a powerful SMT solver [45]. We specialize this general architecture of ConfigSynth with respect to the problem context.

3.5 Resiliency Architecture Synthesis Based on Isolation

In this section, we first describe the architecture of ConfigSynth. Then, we present the formal model of the isolation-based resiliency architecture synthesis and a brief description of its implementation, and an illustrative example. We also present a hypothesis testing-based mechanism using ConfigSynth in order to have an improved security or resiliency architecture. Finally, we present the evaluation results regarding ConfigSynth and the refinement mechanism.

3.5.1 Synthesis Framework

ConfigSynth, the resiliency design synthesizer, follows a top-down security or resiliency design automation approach instead of the traditional bottom-up approach.

The major contributions of ConfigSynth are as follows:

- It formally models the network topology, resiliency requirements and business constraints.
- It formalizes the resiliency architecture synthesis problem as the determination of appropriate isolation patterns along with the correct placements of necessary security devices in the network that satisfy the given requirements/constraints.
- It encodes the synthesis problem using SMT logics and solves it using an SMT solver.

The ConfigSynth framework is shown in Figure 16. ConfigSynth takes the following as its main inputs: (i) the network topology, (ii) resiliency requirements in terms of isolation, and (iii) business constraints in terms of usability and deployment cost. The tool provides its users with three sliders in order to select the constraints on the isolation measure taken in the network, the usability of the system, and the cost for deploying necessary security devices. The sliders are scaled from 0 to some upper limit. As inputs, the tool also takes partial or complete specifications about the qualities of isolation patterns, the demands of different traffic flows, and the deployment costs of different security devices. The isolation requirements are conditioned on the specifications of different primitive and composite isolation patterns along with their relative order based on their capabilities. ConfigSynth models the functional mapping from each flow to an isolation decision variable. Finally, it calculates the overall isolation of the network by accumulating isolation measures between different host pairs under various services.

The usability is modeled based on the connectivity requirements and the ranks of the services and service flows as provided in the specifications. The connectivity requirements are modeled as a set of rules, where each rule functionally maps a flow to a decision variable. We model impacts of different isolation patterns on the usability. The deployment of isolation measures is associated with a cost. The cost depends on the security devices required for implementing the isolation patterns. The number of security devices depends on the network topology. ConfigSynth also models different invariant and user-defined constraints on selecting the resiliency design. ConfigSynth formalizes the resiliency architecture synthesis as the conjunction of all of the isolation, usability, and cost constraints. Therefore, the tool determines the isolation patterns between each service flow in the network such that the overall isolation in the network and the usability of the system satisfy the associated constraints, while the cost for isolation measures deployment does not exceed the organization's given budget. ConfigSynth solves the model using Z3, and the resiliency configurations are synthesized as an output.

3.5.2 Formal Model of Synthesis Framework

ConfigSynth models the network topology as a graph. The network model is defined as $\langle \mathbb{N}, \mathbb{L} \rangle$, where,

- $\mathbb{N}$ defines a finite set of network nodes including hosts and routers. Thus, $\mathbb{N}$ is a union of two sets: $\mathbb{H}$ and $\mathbb{R}$. $\mathbb{H}$ denotes a finite set of hosts. $\mathbb{R}$ denotes a finite set of routers. Each host is identified by an ID (*e.g.*, IP address). A host may execute one or more services, which are accessed by different hosts. A service is denoted using $g \in \mathbb{G}$, where $\mathbb{G}$ is the set of all services. The term $g(i, j)$ defines the traffic/service flow between a pair of hosts $\{i, j\}$ under a service g, where i is the source and j is the destination of the flow.

- $\mathbb{L} \subseteq \mathbb{N} \times \mathbb{N}$ is a finite set of links that define the interconnections between the network hosts.

ConfigSynth formalizes different requirements and constraints which are the building-blocks for formulating the configuration synthesis problem. The requirements can be classified into two categories: (i) isolation-based resiliency requirements, and (ii) business constraints. There are also invariant and user-defined constraints on resiliency implementations.

3.5.2.1 Modeling Isolation

Usually, the more a host is isolated from other hosts in the network, the lower the security threat to it. We define *isolation* as the resistance or restriction on the connectivity, *i.e.*, network communication. The communication between two hosts can be restricted by applying different security devices or systems, such as firewall, IPSec, IDS, NAT, etc. For example, a firewall can be placed to simply block some traffic flows (*i.e.*, complete isolation), while IPSec can be placed to ensure authenticated transmission for the allowed flows (*i.e.*, restriction based on authorization). Both of these devices are required to ensure authenticated and controlled traffic flow.

In order to formalize isolation, it is required to define different isolation patterns, considering different kinds of security devices, the levels of restrictions they can enforce on the flows (*i.e.*, their effectiveness on the isolation), and their impact on the usability. The objective of isolation requirements is to have fine-grained security or resiliency measures in the network. Therefore, it is necessary to devise an appropriate combination of security devices for providing fine-grained resiliency controls.

3.5.2.1.1 Isolation Patterns

Isolation patterns can be network level, host level, or application level. In this research, we consider the network level isolation. The following patterns are examples of the network level isolation:

- *Access deny.* This is naturally enforced by a firewall.
- *Trusted communication*, *i.e.*, authenticated and encrypted communication. IPSec devices are used to build a trusted path (a.k.a. tunnel).
- *Payload inspection.* This is done by an intrusion detection system (IDS).
- *Source identity hiding.* A network address translation (NAT) device is applied in order to use a different address (typically a real IP address) instead of the original address. NAT gives a kind of security by ensuring one way communication (*i.e.*, internal hosts to external hosts).
- *Traffic forwarding through Proxy.* For example, a reverse proxy gives a layer of security in terms of traffic filtering or implementing access control rules (ACLs) in the proxy instead of the server.

ConfigSynth allows network administrators to define isolation patterns considering different security devices (*primitive isolation*) and their combinations (*composite isolation*), along with their relative order based on the capabilities and functionalities of

Table 11: Network level isolation patterns

Isolation (k)	Isolation Pattern	Decision	Isolation Score
1	Access Deny	$y^1_{i,j}(g)$	4
2	Trusted Communication	$y^2_{i,j}(g)$	2
3	Payload Inspection	$y^3_{i,j}(g)$	1
4	Traffic Forwarding through Proxy	$y^4_{i,j}(g)$	1
5	Traffic Forwarding through Proxy with Trusted Communication	$y^5_{i,j}(g)$	3

the devices. A set of primitive isolation patterns is shown in Table 11. Each pattern is represented using an ID, k. As shown in the table, $k = 1$ denotes "access deny" and $k = 2$ indicates "trusted communication", and so on. We formalize the isolation measures as a set of rules, where each isolation rule is defined as follows:

$$y^k_{i,j}(g), \text{where, } i, j \in \mathbb{H} \text{ and } g \in \mathbb{G}$$

The term $y^k_{i,j}(g)$ indicates that corresponding kth isolation pattern must be deployed between the host pair $\{i, j\}$ for service g. Note that a host can represent a group of hosts that have the same properties (*e.g.*, OS, services, etc.), the same level of users, and reside in the same subnet.

3.5.2.1.2 Isolation Patterns and Security Devices

An application of an isolation pattern requires the deployment of one or more security devices. Usually, an isolation pattern is related to a particular type of security device. This one-to-one matching is true for primitive isolation patterns. In the case of a composite isolation pattern, it is necessary to deploy more than one security device. The following equation models the relationship between an isolation pattern

Table 12: Security devices

Id (d)	Device Name	Primitive Isolation Pattern
1	Firewall	Access Deny
2	IPSec	Trusted Communication
3	IDS	Payload Inspection
4	Proxy	Traffic Forwarding through Proxy

and associated security device(s):

$$\forall_{i,j,g}, y^k_{i,j}(g) \rightarrow x^d_{i,j}(g) \tag{1}$$

Equation (1) specifies that if the kth isolation is selected for $g(i,j)$ flow, the dth (type of) security device is required to be deployed between the host pair $\{i,j\}$ (*i.e.*, on the route of the flow). A particular value of d denotes a particular type of security device. For example, as shown in Table 12, $d = 1$ represents a firewall security device. If the kth isolation pattern is a composite one, multiple security devices are required to implement the pattern. Hence, in this case, multiple $x^d_{i,j}(g)$s are true. Usually, a security device deployment depends on the isolation pattern only, not on the flows (*i.e.*, i, j, or g). Equation (1) considers this. Table 12 shows a list of network security devices and the associated primitive isolation patterns.

3.5.2.1.3 Score of an Isolation Pattern

We define the isolation *score* (can also be named *rank*) of the kth isolation pattern between a pair of hosts $\{i,j\}$ under the network service g by the parameter $L^k_{i,j}(g)$. The score of an isolation pattern denotes its isolation capability compared to others. The scores are computed based on the relative order of the isolation patterns according to their isolation capabilities. An administrator can provide the relative order explicitly or provide partial information about the order. A simple formal model is developed based on the given partial order between different isolation patterns. The model generates a complete relative order by assigning a value to each isolation pattern. The value assigned to a pattern denotes its (relative) isolation score. The highest value specifies the maximum isolation score. We assume the same score (L^k) for a particular isolation pattern irrespective of hosts and services. Table 11 shows an example of relative isolation scores from the following partial information:

$$\forall_{k \neq 1}, L^k < L^1$$

$$(L^2 > L^3) \wedge (L^2 > L^4) \wedge (L^5 > L^2)$$

The isolation scores are normalized according to a specified range (*e.g.*, a scale of $0-1$). Note that this scoring of isolation patterns is relative, and resiliency requirements based on this scoring system reflect the same relative meaning.

3.5.2.1.4 Isolation of a Host

The decision variables $y^k_{i,j}(g)$ for all k, represent isolation patterns between a pair of hosts $\{i,j\}$ for the flow $g(i,j)$. These decision variables and associated isolation weights $L^k_{i,j}(g)$ are used to formally define the total isolation ($\bar{I}_{i,j}$) of j with respect to the incoming traffic from i. $\bar{I}_{i,j}$ is formalized as follows:

$$\bar{I}_{i,j} = \frac{\sum_g \sum_k y^k_{i,j}(g) \times L^k_{i,j}(g)}{\sum_g \sum_k y^k_{i,j}(g) \times 1}$$

The equation indicates that the isolation between a pair of hosts $\{i,j\}$ is the sum of the isolation measures taken for different services between these hosts. The equation also indicates that the isolation is normalized by dividing the sum by the maximum possible isolation (*i.e.*, the maximum isolation for a flow $g(i,j)$ is 1 in the scale of $0-1$). We use a similar normalization throughout the model. For the ease of presenting the equations, we do not show the normalization factors (*i.e.*, the denominators at the right-hand side of the equations) for the rest of this chapter.

The isolation of a host depends not only on the hosts that can connect to it, but also on the hosts to which it can connect. For example, if a host can connect to the Internet, the host can download malicious content from the Internet and can get infected. However, the threat due to the outgoing communication is expected to be less compared to the incoming communication. Since the outgoing traffic from j to i is the incoming traffic for i from j, the total isolation $I_{i,j}$ considering both the incoming and the outgoing traffic with respect to j for the pair of hosts $\{i,j\}$ is defined as follows:

$$I_{i,j} = \alpha \bar{I}_{i,j} + (1-\alpha)\bar{I}_{j,i} \tag{2}$$

Here, α ($0 \leq \alpha \leq 1$) is the weight for the isolation due to the incoming traffic, while $1-\alpha$ is the weight for the isolation due to the outgoing traffic. The total isolation score of a host j is defined in Equation (3).

$$I_j = \sum_{i \neq j} I_{i,j} \tag{3}$$

Equation (4) represents the overall isolation in the network (*i.e.*, the network isolation) considering all of the hosts.

$$I = \sum_{i} I_i \tag{4}$$

3.5.2.2 Modeling Business Constraints: Usability

Business constraints (*i.e.*, usability and deployment cost constraints) play a significant role in synthesizing usable and cost-effective resiliency configurations in a network. For example, a higher network isolation provides strong defense, but the network usability may be reduced to a level which is unacceptable to the organization. Resolving the contention between resiliency requirements and business constraints is a challenge. In ConfigSynth, we formalize the synthesis problem under two business constraints: (i) usability and (ii) deployment cost. In this subsection, we discuss the formalization of the usability.

3.5.2.2.1 Connectivity Requirements

Every organization usually has a number of service flows, which are essential for its successful operation. Each of these *connectivity requirements* represents a flow that must be able to reach the destination from its source. Connectivity requirements are formalized as a set of rules, where each connectivity rule defines the mapping from a flow (*i.e.*, a tuple of source, destination, and service) to a decision variable c that represents whether the flow is required to be allowed. The formal definition of each rule is as follows:

$$c_{i,j}(g), \text{where, } i, j \in \mathbb{H} \text{ and } g \in \mathbb{G}$$

Here, $c_{i,j}(g)$ is a binary variable. When it is true, it represents that the service flow g must be allowed from i to j. If it is false, then nothing has been specified for this flow, *i.e.*, the flow can either be allowed or denied. CR represents the conjunction of all connectivity requirements.

$$CR \rightarrow \bigwedge_{i,j,g} c_{i,j}(g) \tag{5}$$

3.5.2.2.2 Usability Calculation

The usability of the network depends on the ranks of the service flows between the hosts in the network. The rank of a service flow denotes the demand of the flow. Each service flow $g(i,j)$ is associated with a rank, $a_{i,j}(g)$. These ranks are expected to be given to the synthesis model in the form of a relative order based on the organizational requirements. Partial information can be given, from which a complete relative order can be derived, as it has been shown in the case of the isolation patterns. If no specification is given about the demand of different flows, all flows receive the same rank. The ranks are normalized between 0 and 1. The usability of a service g running on a host j is formalized as follows:

$$S_j(g) = \sum_i \sum_k y^k_{i,j}(g) \times b^k_{i,j}(g) \times a_{i,j}(g)$$

The application of an isolation pattern to a flow can affect its usability. The parameter $b^k_{i,j}(g)$ represents the usability of the flow $g(i,j)$ due to applying the k isolation pattern between $\{i,j\}$. We assume that the usability depends on the isolation pattern, not on the host-pair ($i.e.$, $b^k_{i,j}(g) = b^k(g)$). The value of $b^k(g)$ can be determined based on the knowledge of network security experts by considering the time or effort required to access a service under an isolation measure. The valuation of the parameter $b^k(g)$, in the simplest form, can be as follows: the ‘access deny’ isolation pattern reduces the usability to zero (*i.e.*, $\forall_g, b^1(g) = 0$), while other isolation patterns maintain the same usability (*i.e.*, $\forall_{g,k\neq 1}, b^k(g) = 1$). The usability S_j with respect to a host j represents the accumulated usability considering all of the services running in the host.

$$S_j = \sum_g S_j(g)$$

The overall usability of the network (*i.e.*, the network usability) is represented by Equation (6).

$$U = \sum_j S_j \tag{6}$$

3.5.2.3 Modeling Business Constraints: Deployment Cost

The deployment of a security device incurs costs and an organization often has a limited budget for deploying security measures. The deployment cost is the sum of the prices of the security devices that are required to be deployed in different segments of the network in order to implement necessary isolation patterns between different host-pairs. The number of security devices depends not only on the isolation measures, but also on the topology. The cost cannot be calculated from the isolation measures alone. This is because there are usually similar types of isolation between multiple host-pairs, and these host-pairs can share one or more links for communication. In this case, placing a single security device at one of the shared links may ensure the desired isolation. Moreover, if there is more than one routing path between a host-pair, we have to secure all of the alternative paths. Therefore, modeling correct and optimal placements of the security devices is challenging, considering the network topology, the isolation patterns, and the budget.

3.5.2.3.1 Flow Routes

ConfigSynth requires the flow routes between the hosts for the purpose of determining the placements of the security devices satisfying the isolation measures. A flow route, $F_{i,j}^{z}$ is defined as a set of links $\{l_{i,j,z,1}, l_{i,j,z,2}, ...\} \subseteq \mathbb{L}$, that form a path from a source i to a destination j. As multiple routes are possible between a pair of hosts, z indicates the index of a flow route (*i.e.*, the zth route), between the host-pair $\{i, j\}$. The term $|F_{i,j}^{z}|$ denotes the path length, *i.e.*, the number of hops or links in the path. $F_{i,j}$ denotes all of the flow routes possible from i to j.

$$F_{i,j} \to \bigwedge_{z} F_{i,j}^{z}$$

ConfigSynth finds the flow routes for a host pair by applying a path searching algorithm on the network topology.

3.5.2.3.2 Device Placements

Equation (1) specifies the security devices which are required to employ an isolation pattern. The placements of the security devices on the flow routes are modeled from these specifications. If an isolation pattern (*e.g.*, "access deny") is selected for the traffic from a host i to a host j, then it is required to block the traffic through all possible flow routes between $\{i, j\}$. Equation (1) specifies a firewall to be deployed for implementing an "access deny" isolation pattern. Hence, there should be a firewall deployed on a link of each flow route. We formalize the placement of a security device d for a particular pair of hosts as follows:

$$x_{i,j}^{d}(g) \to \forall_{z} \exists_{t} l_{i,j,z,t}^{d} \tag{7}$$

In the equation, $l^d_{i,j,z,t}$ represents that a security device of type d is deployed on link $l_{i,j,z,t}$. Here, t is the index of a link on the zth routing path from host i to host j). Note that if there a security device, *e.g.*, firewall, is deployed on the flow route of a host pair, this deployment does not imply that the flow access between the pair is denied. The firewall will deny the access only if the "access deny" isolation pattern is specified for the host pair.

The placement of an IPSec device requires special modeling which is different from that of security devices like firewall and IDS. The "trusted communication" isolation pattern usually requires encrypted communication (*i.e.*, tunnel) to take place throughout the unsecured or untrusted part of the network, which is likely to be host to host. Moreover, to ensure an encrypted tunnel between a host pair, it is necessary to place two IPSec devices, one at the source side (start of the tunnel) and another at the destination side (end of the tunnel). A network administrator needs to specify the guidelines for placing the IPSec gateways. The administrator can specify the maximum number of hops (*i.e.*, the number of links) from the end-hosts that can be outside of the tunnel. For example, it can be specified that the source-gateway and the destination-gateway should be deployed within two hops from the source and the destination respectively. We model this as follows:

$$\begin{aligned} x^2_{i,j}(g) \rightarrow & \forall_z, (|F^z_{i,j}| \geq (2 \times T)) \wedge \\ & (\exists_t (l^2_{i,j,z,t} \wedge (t \leq T)) \wedge \\ & \exists_{t'}, (l^2_{i,j,z,t'} \wedge ((|F^z_{i,j}| - t') \leq T))) \end{aligned}$$

Here, $|F^z_{i,j}|$ represents the length of the flow route $F^z_{i,j}$, and T denotes the maximum number of hops that can be outside of the tunnel. According to this approach, if the flow route between the source and the destination has only a few hops (*e.g.*, less than $2T + 1$ hops), then "trusted communication" is not possible to be deployed between this pair of hosts.

For the deployment of the security devices, the deployment cost is computed as the summation of the costs of all of the devices deployed in different links. We define C_d as the average deployment cost of the security device d. Now, if l^d denotes whether a security device d is deployed on the link $l \in \mathbb{L}$, the total deployment cost C is computed as follows:

$$C = \sum_{l \in \mathbb{L}} \sum_{d} l^d \times C_d, \text{where } l^d \rightarrow \exists_{i,j,z,t}, l^d_{i,j,z,t} \tag{8}$$

3.5.2.4 Modeling Constraints

ConfigSynth synthesizes resiliency configurations by solving a number of constraints. In the following, we discuss these constraints in different categories.

3.5.2.4.1 Threshold Constraints

In ConfigSynth, we have three generic threshold based constraints in selecting the isolation measures for the network flows:

$$TC : (I \geq Th_I) \wedge (U \geq Th_U) \wedge (C \leq Th_C) \tag{9}$$

In the equation, Th_I, Th_U and Th_C represent the slider values, the constraints on the network isolation, usability, and deployment cost, respectively. The network isolation and the network usability must be greater than or equal to their respective threshold values, Th_I and Th_U. The deployment cost must also be within the budget, Th_C.

3.5.2.4.2 Invariant Constraints on Isolation Selections

In ConfigSynth, there are different invariant constraints. These constraints ensure the consistency between functional behaviors of the isolation patterns and the business requirements.

$$IIC_1 : y^k_{i,j}(g) \rightarrow \forall_{\bar{k} \neq k} \neg y^{\bar{k}}_{i,j}(g)$$

$$IIC_2 : c_{i,j}(g) \rightarrow \neg y^1_{i,j}(g)$$

$$IIC \rightarrow \bigwedge_c IIC_c \tag{10}$$

The constraint IIC_1 states that only one isolation pattern can be selected for a flow. The constraint IIC_2 ensures that when "access deny" is chosen as the isolation pattern for the flow from i to j, there should be no connectivity requirement for that flow. Equation (10) combines all invariant constraints.

3.5.2.4.3 User-defined Isolation Policy Constraints

User-defined constraints represent organizational requirements. The following are examples of some user-defined constraints:

$$UIC_1 : (g = \text{SSH}) \rightarrow \neg y^2_{i,j}(g)$$

$$UIC_2 : \neg y^1_{i,\hat{j}}(g) \rightarrow \neg y^1_{\bar{i},i}(g) \wedge \neg(\bar{i} = \text{Internet})$$

$$UIC_3 : (g = \text{WEB}) \rightarrow \neg y^2_{i,j}(g)$$

$$UIC \rightarrow \bigwedge_c UIC_c \tag{11}$$

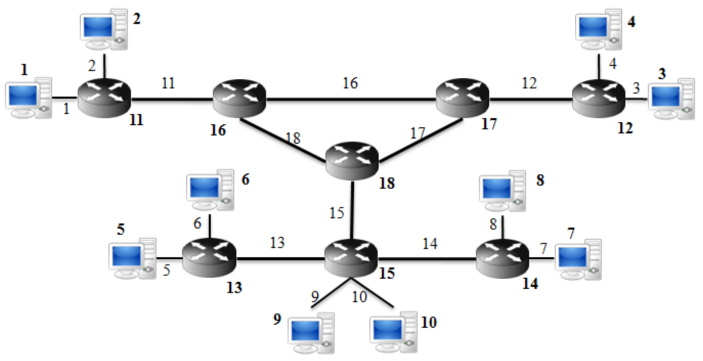
An organizational policy (UIC_1) may state that IPSec should not be deployed for a pair of hosts in the case of Secure Shell (SSH)-based communication. The isolation requirement for a particular type of flow can be defined by stating that access will be allowed from a source i to a specific destination $\hat{j}$ under the service g, if the Internet is not allowed to connect to i. This is modeled in UIC_2. The organizational policy may require that no web service should be protected by the "trusted communication" isolation pattern (UIC_3), while the flow is already specified to be allowed as a connectivity requirement. Equation (11) represents all user-defined constraints.

3.5.2.5 The Complete Synthesis Model

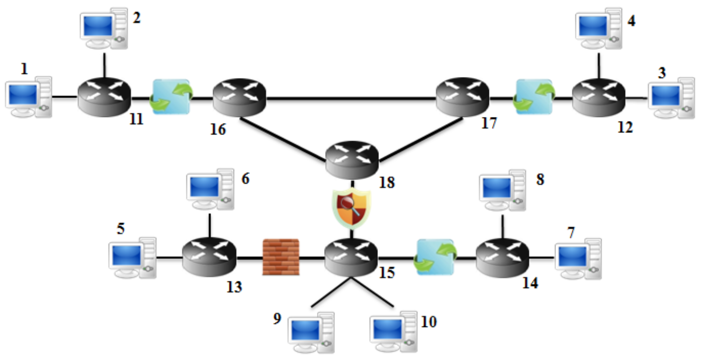
The main objective of our configuration synthesis problem is to maximize the resiliency in the network by satisfying various resiliency requirements as well as the organization's business constraints. Thus, the synthesis problem is formalized as the satisfaction of the constraint ($Constr$), which is the conjunction of all of the constraints as follows:

$$Constr \rightarrow CR \wedge TC \wedge IIC \wedge UIC \tag{12}$$

The equation specifies that the solution to the synthesis problem produces resiliency configurations, *i.e.*, isolation patterns between different host pairs ($y^k_{i,j}(g)$s), along with the placements of necessary security devices (l^ds), by ensuring the fulfillment of all the constraints.



(a)



(b)

Figure 17: (a) An example network for synthesizing resiliency configurations and corresponding security device placements. An ID is assigned to each of the hosts, routers, and links. (b) The solution to the example problem, *i.e.*, the placements of necessary security devices.

3.5.3 An Example Case Study

3.5.3.1 SMT Encoding

We implement our model by encoding the system configuration and the constraints using SMT logics [23]. In this encoding purpose we use the Z3 Dot Net API [45]. For encoding the formalizations of the network topology, device configurations, traffic modeling, and the resiliency and business properties, we use mainly two types of terms: Boolean and integer. We use Boolean terms for encoding the Boolean configuration parameters and decision variables, such as isolation patterns and device placements. The remaining parameters are modeled as integer terms. The parameter values that may take real numbers (*e.g.*, the isolation and usability scores) are normalized into integer values. In our modeling, we represent a host using an integer ID, which is not necessarily in IP address format, since no IP address-based computation is required in this model. Each service is also encoded as an integer value (as an ID specifying a protocol-port pair). ConfigSynth takes the system configurations, requirements and constraints from a input file. ConfigSynth also provides a graphical interface so that its user can select or edit the inputs.

3.5.3.2 Example

Figure 17(a) shows a small network for which an optimal resiliency design will be synthesized based on the given input file as shown in Table 13. In this example, the connectivity requirements are considered as a list of allowed services between different hosts. In order to keep the example simple, we consider only three primitive isolation patterns (*i.e.*, “access deny”, “trusted communication”, and “payload inspection”). We also assume a single flow type (*i.e.*, a single service) between each pair of hosts. ConfigSynth gives a satisfiable result for this example. From the resultant satisfiable instance, we find the necessary isolation patterns along with the necessary device placements. Figure 17(b) shows the placements of the security devices. Table 14 shows the isolation patterns. For example, the first row of the table specifies the isolation patterns that are selected on the traffic flows toward host 1.

Table 13: Input (partial) of the example

```
# Number of Security Devices
3 # 1 for Firewall, 2 for IPSec, and 3 for IDS, while 0 for None
# Isolation Specifications (partial orders)
2 # Device, Device, Comparison (1 for =, 2 for >, and 3 for >=)
1 2 2
2 3 2
# Usability if an isolation pattern is applied
0 2 3
# Cost of each isolation device (in thousand dollars)
20 18 15
# Number of Hosts and Routers
10 8
# Links
18
1 11
2 11
.........
# Connectivity Requirements (each row for a host, which ends with 0)
3 0 # The flow from Host 1 to Host 3 must be allowed
4 0
1 2 0
2 0
3 4 0
3 4 0
1 2 0
1 0
0
1 0
# Sliders' Values (Isolation 0-10, Usability 0-10, Cost in thousand dollars)
6 5 90
```

3.5.4 Resiliency Architecture Refinement

ConfigSynth provides an isolation-based resiliency architecture satisfying the pre-attack resiliency requirements and business constraints, as we have seen in the last section. We can compute the optimal solution by running the ConfigSynth framework many times by increasing the isolation requirement slowly or by running a binary search algorithm within a range of maximum and minimum isolation requirements.

Table 14: Output of the example: selected isolation patterns for the flows

Destination	Sources Classified according to Selected Isolation Patterns			
Host	Access Deny	Trusted Communication	Payload Inspection	No Isolation
1	5, 6	3, 4, 7, 8	9, 10	2
2	5, 6	3, 4, 7, 8	9, 10	1
3	–	1, 2, 7, 8	6, 9, 10	4, 5
4	–	1, 2, 7, 8	9, 10	3, 5, 6
5	1, 2, 3, 4, 7, 8, 9, 10	–	–	6
6	1, 2, 3, 4, 7, 8, 9, 10	–	–	5
7	5, 6	1, 2, 3, 4, 9, 10	–	8
8	5, 6	1, 2, 4, 9, 10	3	7
9	5, 6	7, 8	1, 2, 3, 4	10
10	5, 6	7, 8	1, 2, 3, 4	9

Unfortunately, from evaluation results (Section 3.5.5), we find that if the isolation requirement is close to the optimal, above or below the optimal value, with respect to the business constraints, the time for executing the ConfigSynth framework turns out to be significantly high. This time is found to be extremely high when the number of hosts is more than 100 hosts. Moreover, if the network is in a clean state, *i.e.*, there is no given isolation measure as well as security device placement, the time for finding a satisfiable resiliency architecture by ConfigSynth becomes long, as the search space is large. On the other side, if there is already a given resiliency architecture, the time to improve the architecture by adding more isolation measures or security devices within the constraints is much shorter. Considering the above mentioned learning, we devise a statistical hypothesis testing-based resiliency architecture refinement mechanism for tuning or improving the resiliency architecture. This improved architecture essentially may not be the optimal one, but it will be a close one, which is synthesized in a plausible time period. The corresponding extended architecture of ConfigSynth is shown in Figure 18.

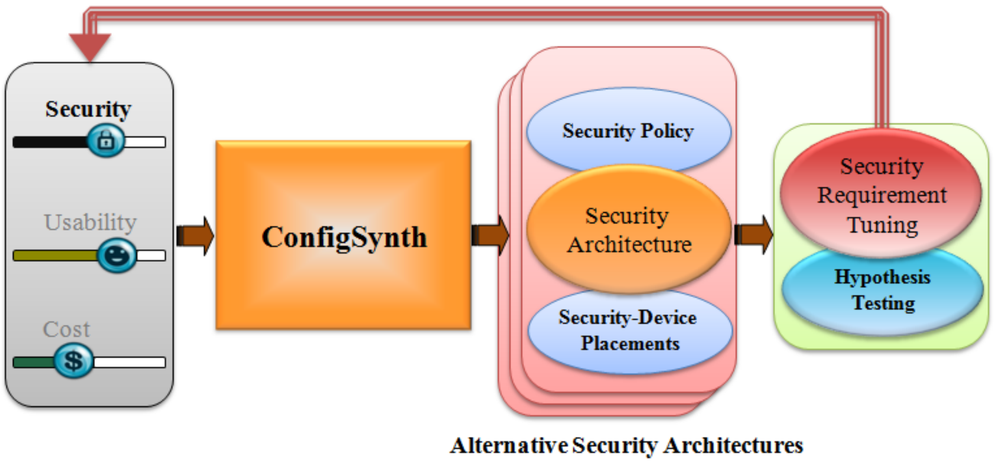

Security
Usability
Cost
ConfigSynth
Security Policy
Security Architecture
Security-Device Placements
Alternative Security Architectures
Security Requirement Tuning
Hypothesis Testing

Figure 18: The extended architecture of ConfigSynth for the design refinement.

In the following, we present a mechanism for the resiliency architecture refinement. We also present a case study demonstrating the refinement process.

3.5.4.1 Hypothesis Testing based Refinement

In this refinement process, our objective is to disprove the null hypothesis, which we take as "there is no better resiliency architecture within the given business constraints other than the known best resiliency architecture." Thus, in order to reject this null hypothesis, we need to prove the alternative hypothesis, *i.e.*, there are resiliency architectures significantly better than the known best resiliency architecture, which satisfies the given business constraints as well. Let us define N as the number of alternative resiliency architectures that will be used to verify our null hypothesis. We would like to reject the null hypothesis if we find just one better resiliency architecture from the N number of architectures. The reason for this idea is to increase the refinement efficiency because each search for a resiliency architecture consumes a substantial time.

In this hypothesis testing, we consider standard values for necessary parameters. For example, the standard deviation is considered to be 5% of the isolation provided by the known best resiliency architecture. We find that, when we consider the significant level as 5% and the standard deviation as the 5% of the maximum isolation (in the scale of 0-10), then if N is taken as 25, the null hypothesis can be rejected if and only if one resiliency architecture is found that offers an isolation significantly better than the known best isolation. If the known best isolation is 5, then the increase needs to be around 10% of the known best isolation. However, it is worth mentioning that for a larger known best isolation value, the increase in the isolation requirement can be smaller, while for a smaller known best isolation, we need to have a larger increase in the requirement, in order to reject the null hypothesis when a single resiliency architecture is found providing that increased isolation irrespective of the others in the sample set. The potential of a type II error is unknown when we have N number of alternative resiliency architectures, as we do not know the maximum number of alternative resiliency architectures. However, if we do have architectures less than N, then there is no Type II error.

We adopt the one-sided test in our hypothesis testing mechanism. As we look for better resiliency architectures in the refinement process, we consider the right-sided test, in which the alternative hypothesis specifies the set of values strictly greater than the critical value. In the following, we present an example process of hypothesis testing with respect to a particular scenario. This example illustrates our choices for N in the hypothesis testing.

3.5.4.1.1 Computation of Parameter Values with Regards to Hypothesis Testing

According to the isolation scale of 0 to 10, the known best isolation I is 5. The sample size (*i.e.*, the number of alternative resiliency architectures) N is 25. The standard deviation D is 0.05. Then, the standard error E is calculated as follows:

$$E = D/\sqrt(N) = 0.05/5 = 0.01$$

Assume that each of the $N-$ 1 resiliency architectures provides isolation I in average, while the remaining one provides I' isolation significantly greater than I. Let $I' = I$ + 10% of I. Then, I' = (5 + 0.5) = 5.5.

Thus, the average isolation $\bar{I}$ provided by the sample resiliency architectures is (5 $\times$ 24 + 5.5) / 25 = 5.02. Then, we calculate test statistic z as follows:

$$z = \bar{I} - I)/E = (5.02 - 5)/0.01 = 0.02/0.01 = 2$$

The significance level α is considered as 5%. At this significance level, from the table of z-scores, the critical value CV is found as 1.645. Since $z > CV$, the null hypothesis is rejected.

3.5.4.2 Refinement Algorithm

According to the hypothesis testing based refinement idea discussed in the previous subsection, we devise a mechanism as presented in Algorithm 1. The mechanism starts with a null hypothesis specifying an isolation value (I^{mean}) which is the isolation provided by the known best resiliency architecture. If the null hypothesis is rejected, then we update the null hypothesis with the isolation of the resiliency architecture

Algorithm 1 Systematic Refinement of Resiliency Architecture

```
Require: $Synth$ is the synthesis model (i.e., ConfigSynth) as of Equation 12.
Require: $I^{mean}$ is the mean isolation.
Require: Null Hypothesis ($H_0$): There is no resiliency architecture possible providing higher isolation score than $I^{mean}$.
Require: $N$ is the number of alternative resiliency architectures, i.e., the sample size.
Require: $I^{min}$ is the minimum isolation requirement for a resiliency architecture to be a member of the sample set. $I^{min} \leq I^{mean}$.
Require: $Th_I$ is initialized with $I^{min}$.
Require: $X$ (e.g., 10% of $I^{mean}$) is the minimum increase in the isolation score provided by a resiliency architecture in order to reject the null hypothesis.
Require: $I^{max}$ is the maximum isolation provided by a resiliency architecture after improvement.
Require: $\mathcal{Z}^{Best}$ is the known best resiliency architecture.
 1: for $i = 1$ to $N$ do
 2:     if Solver returns $sat$ then
 3:         Get the model, $\mathcal{M}$.
 4:         From $\mathcal{M}$ fetch the complete resiliency architecture, $\mathcal{Z}$ (it includes the isolation measures for all the flows and security device placements for all the links).
 5:         From $\mathcal{M}$ fetch the applied resiliency, $\bar{\mathcal{Z}}$ (it specifies the flows with positive isolation measures and the links with one or more device placements).
 6:         Update $Synth$ by adding $\neg\mathcal{Z}$.
 7:         Push or save $Synth$ in the memory.
 8:         Set $I^{max}$ to $I$ as it is obtained from $\mathcal{M}$.
 9:         Update $Synth$ by adding $\bar{\mathcal{Z}}$.
10:         Increase $Th_I$ with a small value $Y$, such that $Y \leq X$,
11:         while Solver returns $sat$ do
12:             Get the model, $\mathcal{M}'$.
13:             From $\mathcal{M}'$ fetch the complete resiliency architecture, $\mathcal{Z}'$.
14:             From $\mathcal{M}'$ fetch the applied resiliency, $\bar{\mathcal{Z}}'$.
15:             Pop or retrieve $Synth$ from the memory, that we saved in Steps 7 or 17.
16:             Update $Synth$ by adding $\neg\mathcal{Z}$.
17:             Push $Synth$ in the memory.
18:             Update $Synth$ by adding $\bar{\mathcal{Z}}$.
19:             Set $I^{max}$ to $I$ as it is obtained from $\mathcal{M}$.
20:             Update $Synth$ by adding $\mathcal{Z}$.
21:             $Th_I = Th_I + Y$.
22:         end while
23:         if $I^{max} \geq I^{mean} + X$ then
24:             Reject $H_0$.
25:             Update $I^{mean}$ with $I^{max}$.
26:             Update $\mathcal{Z}^{Best}$ with $\mathcal{Z}'$.
27:             Reinitialize $i$ with 1.
28:         end if
29:         Pop $Synth$ from the memory, that we saved in Steps 7 or 17.
30:     end if
31:     Return $\mathcal{Z}^{Best}$.
32: end for
```

that beats the null hypothesis and start the hypothesis testing from the beginning according to the updated hypothesis. The refinement process continues until we fail to reject the null hypothesis.

3.5.4.3 An Example Case Study

In this case study, we use the same network that we have used in Section 3.5.3.2. The corresponding inputs are similar to Table 13. Therefore, the objective of the

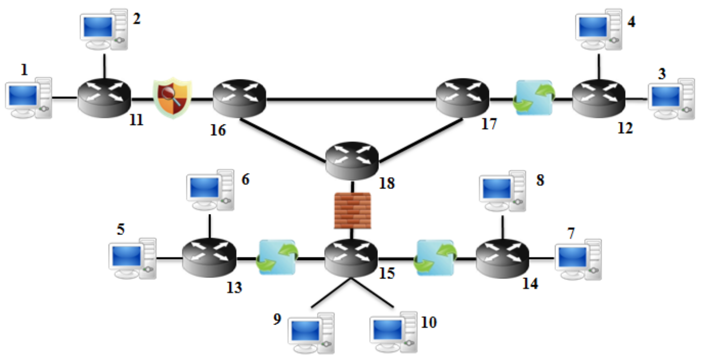


Figure 19: The placements of necessary security devices after the refinement.

refinement process is to find the optimal resiliency architecture that provides the best isolation, considering isolation 6 as the minimum resiliency requirement and satisfying the business constraints. After the execution of Algorithm 1, we receive the best resiliency architecture, which provides isolation 6.63. The placement of security devices is shown in Figure 19 and the resiliency policy (*i.e.*, the isolation measures) is presented in Table 15.

As the algorithm executes, we observe 4 updates of the null hypothesis. The null hypothesis starts with the known best isolation 6. That is, the hypothesis specifies that there is no resiliency architecture that provides isolation greater than 6, as well as satisfies the business constraints. This null hypothesis is rejected by the first alternative resiliency architecture as this architecture provides isolation 6.07. Thus, the null hypothesis is updated with this isolation 6.07. That means the hypothesis states that there is no resiliency architecture which provides isolation greater than 6.07, as well as satisfies the business constraints. Based on this updated null hypothesis, the algorithm starts from the beginning. This null hypothesis is rejected by the 2nd alternative resiliency architecture. We receive isolation around 6.40 according to this

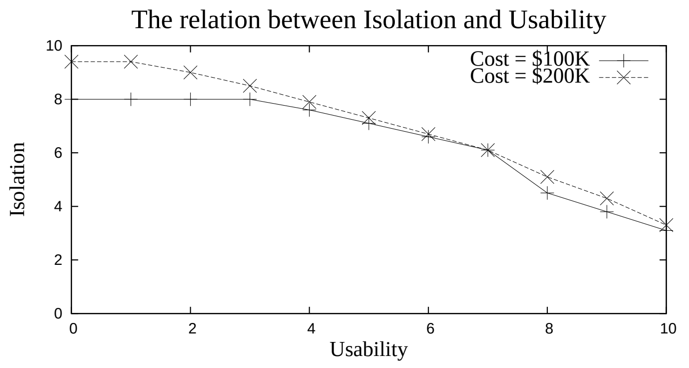
Table 15: Selected isolation patterns for the flows in the example corresponding to the refinement mechanism

Destination	Sources Classified according to Selected Isolation Patterns			
Host	Access Deny	Trusted Communication	Payload Inspection	No Isolation
1	5, 6, 9	–	3, 4, 7, 8, 10	2
2	5, 6, 8, 9	–	3, 4, 7, 9, 10	1
3	7, 8, 9, 10	5, 6	1, 2	4
4	7, 8, 9, 10	5, 6	1, 2	3
5	1, 2, 3, 4	7, 8, 9, 10	–	6
6	1, 2, 4	3, 7, 8, 9, 10	–	5
7	1, 3, 4	5, 6, 9, 10	2	8
8	1, 2, 3, 4	5, 6, 9, 10	–	7
9	1, 2, 3, 4	5, 6, 7, 8	–	10
10	1, 3, 4	5, 6, 7, 8	2	9

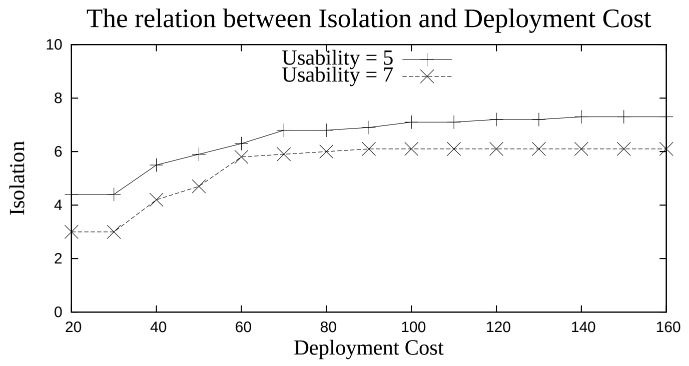
architecture. Therefore, the current null hypothesis is rejected, while the null hypothesis is renewed with isolation 6.40. This null hypothesis is rejected, as we find that the 16th resiliency architecture provides isolation 6.44. The null hypothesis is updated according to this new isolation. This null hypothesis is rejected again. It is rejected by the 16th alternative resiliency architecture as it provides isolation 6.63. Thus, the null hypothesis is updated again with isolation 6.63. This time, we cannot reject this updated null hypothesis, as none of the 25 consecutive resiliency architectures can provide isolation greater than 6.63.

3.5.5 Evaluation

Here, we first present the analysis on the relationships between the isolation, usability, and deployment cost. Then, we present the scalability performance analysis of the tool. We run our experiments on different synthetic test networks.



(a)



(b)

Figure 20: (a) The maximum possible isolation with respect to the usability constraint considering a fixed cost constraint (*i.e.*, \$200K) and (b) the maximum possible isolation with respect to the deployment cost constraint considering a fixed usability constraint (*i.e.*, 5).

3.5.5.1 Analysis of the Relation Between Isolation, Usability, and Deployment Cost

In this analysis, we run a number of experiments considering the same network topology as is shown in Figure 17(a). The impact of the network usability constraint on the network isolation is shown in Figure 20(a) under two different deployment cost constraints. We find that with the increase of the usability constraint, the maximum possible isolation reduces. However, due to the connectivity requirements, the isolation cannot be more than a particular point, though the usability constraint is very low. At the lower values of the usability constraint, the rate of the isolation decrease is less in comparison to the rate at the higher values of the usability constraint.

The deployment cost constraint has an impact on the isolation. Figure 20(a) shows that in the case of the higher cost constraint (*i.e.*, \$200K), a higher isolation is achieved compared to the case of the lower cost constraint (*i.e.*, \$100K). A higher cost allows ConfigSynth to deploy more security devices, particularly IPSec devices, in these experiments, which helps in increasing the isolation. We also find that with the increase of the usability constraint (*i.e.*, from 0 to 7), the difference between the maximum possible isolation values in both of the cases reduces. At the usability value 7, we find that the isolation difference sharply increases, then slowly reduces. The reasons behind this behavior are that different security devices have different prices and different impacts on the usability. Even different deployment aspects influence the deployment cost. For example, IPSec-based security usually requires deployment of two IPSec gateways close to the end hosts, which are at the boundary of the core network. This does not allow many hosts to share these gateways for implementing the "trusted communication" isolation pattern. As a result, IPSec-based security incurs a higher deployment cost compared to the firewall or IDS based security.

Figure 20(b) shows the relationship between the isolation and the deployment cost more adequately, considering two different usability constraints. We change the cost constraint and observed the maximum possible isolation. As expected, in the case of the lower usability constraint (*i.e.*, 5), the isolation is higher compared to the case of the higher usability constraint (*i.e.*, 7). We also find that after a certain level, it is not possible to increase the isolation, despite increasing the deployment budget. This is due to the usability constraint. To increase the isolation after a certain point, it is required to use the highly scored isolation patterns (*e.g.*, “access deny”), which at the same time reduce the usability.

3.5.5.2 Scalability of ConfigSynth Framework

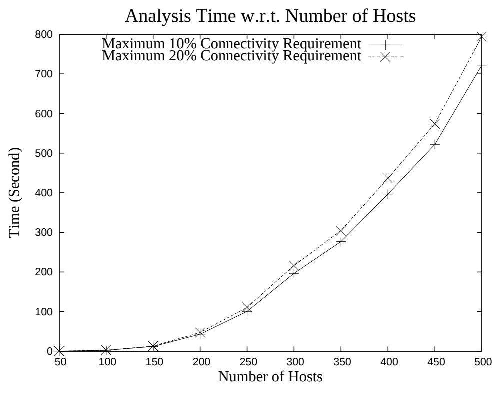
We evaluate the scalability of ConfigSynth by analyzing the time and space required in synthesizing the configurations (*i.e.*, satisfying the constraints) by varying the problem size and the constraints. The problem size depends mainly on the number of flows, since the synthesis problem considers the isolation pattern for each flow. The number of flows mostly depends on the number of hosts. We also evaluate the performance of the resiliency architecture refinement mechanism.

3.5.5.2.1 Methodology

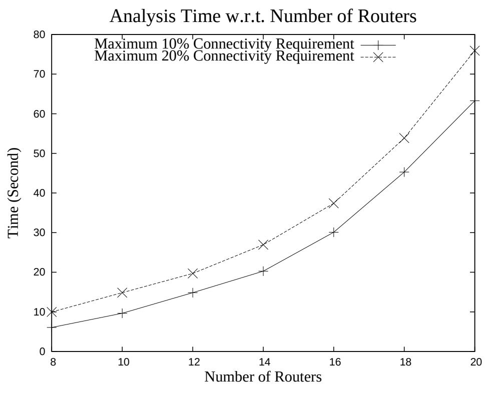
We run ConfigSynth in a machine running Windows 7 OS. The machine was equipped with an Intel Core i3 processor and a 4 GB memory. We generate the test networks, taking hosts within the range of 50−500 and the routers within the range of 8−20. In the test networks, we randomly choose 1−3 services (*i.e.*, maximally 3 flows) between a pair of hosts. The isolation and usability constraints were chosen from normalized scales (sliders) of 0−10 (0 for no isolation/usability, while 10 for complete isolation/usability).

3.5.5.2.2 Impact of the Problem Size

Figure 21(a) and Figure 21(b) show the model synthesis time with respect to the problem size. In the first analysis, we consider two different scenarios. In one scenario, the volume of the connectivity requirements is 10% of all the flows possible between the hosts. In the other scenario, the percentage is 20%. In this analysis, we vary the



(a)

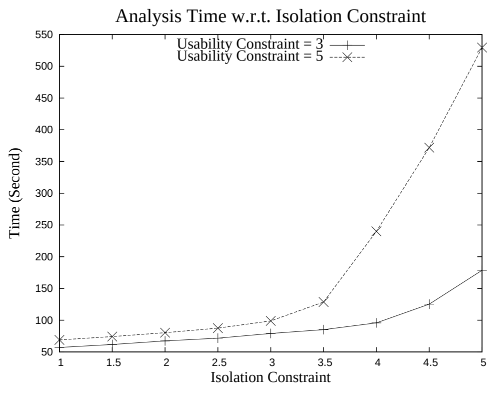


(b)

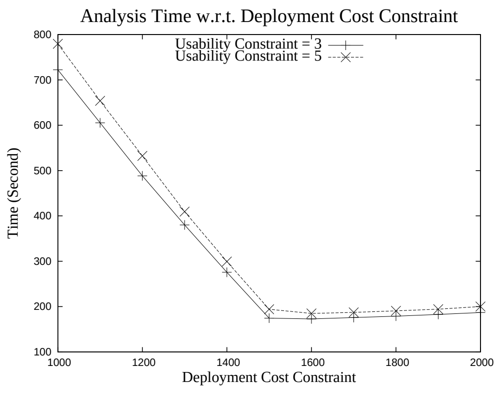
Figure 21: The model synthesis time w.r.t. (a) the number of hosts and (b) the number of routers.

problem size with respect to the number of hosts, and the corresponding results are shown in Figure 21(a). We observe that the analysis time increases rapidly with the number of hosts. This is because the problem size depends on the number of possible flows in the network. The number of flows is $O(N^2)$, where N is the number of hosts and the number of services is constant. The volume of the connectivity requirements also increases with the increase in the number of flows. As a result, the model size increases, which requires the verification of more constraints. All of these increase the running time over $O(N^2)$.

In the second analysis, we vary the core network by changing the number of routers in two different connectivity requirements. The results are presented in Figure 21(b). In this case, since the number of hosts in the network remains the same, there is no increase in the number of flows. However, due to the increase in the number of routers, the core network becomes larger, such that the hosts are more distributed and more links are found to bes candidates for security device placements. As a result,



(a)



(b)

Figure 22: The impact of (a) the isolation constraint and (b) the deployment cost constraint on the model synthesis time.

more search is required to find a satisfiable model, which increases the synthesis time. We observe quadratic increase in the synthesis time.

3.5.5.2.3 Impact of the Constraints

We analyze the impact of the tight or relaxed constraints on the model synthesis time. Tightening (relaxing) the network isolation or usability constraint requires increasing (decreasing) the associated constraint value. On the other hand, tightening (relaxing) the deployment cost constraint requires decreasing (increasing) the constraint value. The analysis results are shown in Figures 22(a) and 22(b), varying the isolation constraint and the deployment cost constraint, respectively. In these analyses, we consider a fixed number of hosts (300) and a fixed volume of connectivity requirements (10% of all flows) in two different network usability constraints (3 and 5 in a scale of 10).

We observe that the execution time increases significantly with the increase of the network isolation constraint (see Figure 22(a)). This is because increasing the

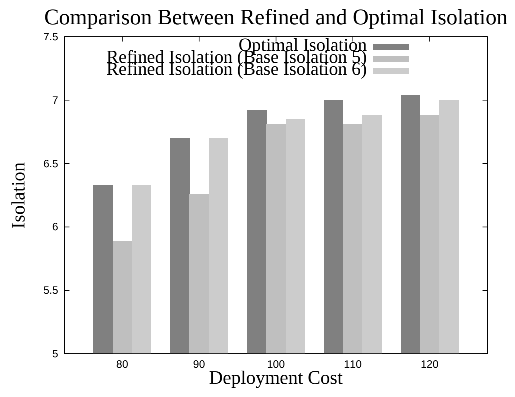
Table 16: The memory requirement (MB) w.r.t. problem size

Hosts	Scenario 1	Scenario 2
200	6.71	6.59
400	30.48	41.72
600	113.99	160.70
800	376.21	532.89
1000	818.92	1158.54

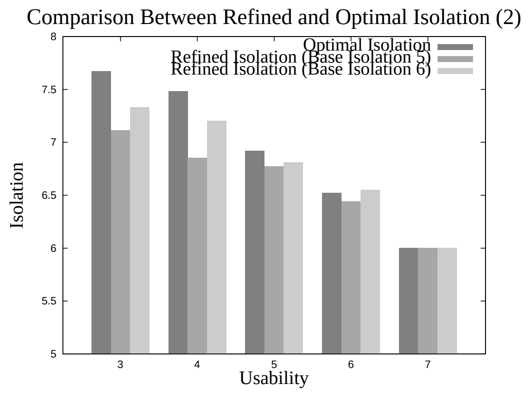
isolation constraint reduces the number of possible solutions to the model with respect to a particular usability constraint and a specific deployment budget. As a result, usually more search (*i.e.*, a longer time) is required to find a solution. After a certain value of the isolation constraint (*i.e.*, 3-4), a small increase in the constraint increases the synthesis time sharply. We observe almost similar behavior in the case of the deployment cost constraint, as shown in Figure 22(b). In this case, the higher the budget, the more satisfiable options are available. Hence, the synthesis time decreases with the increase of the budget. We observe that after a certain increase in the budget ($1,500K), the synthesis time does not decrease further because the number of potential satisfiable models does not increase any more despite increasing the budget.

3.5.5.2.4 Memory Requirement

We evaluate the space (memory) requirement for executing the formal model corresponding to ConfigSynth with respect to the SMT solver [45]. We vary the number of hosts to understand the impact of the problem size on the space requirement. The memory requirement actually specifies the memory required for modeling the synthesis problem, which is the sum of the memory for modeling the system parameters and that for modeling the constraints. The analysis results are shown in Table 20 in two different scenarios of the network isolation constraint. In the first scenario,



(a)



(b)

Figure 23: A comparison between the optimized isolation provided by the refinement mechanism and the optimum isolation with respect to (a) various deployment cost and (b) various usability constraints.

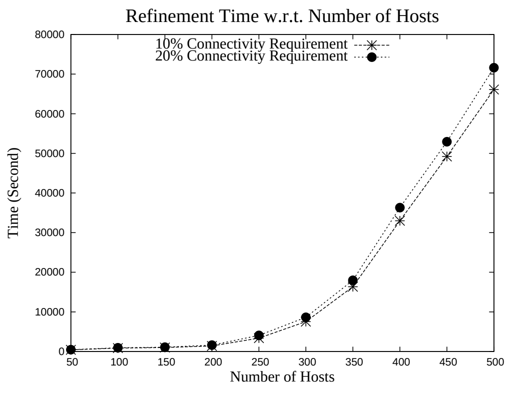
the isolation constraint is 3 (in a scale of 10), while in the second scenario, this is 5. We observe that the memory requirement increases quadratically ($O(N^2)$) with the increase in the number of hosts. The table shows that the memory requirement in the second scenario is larger than the memory requirement in the first scenario. If the isolation constraint is high, the solver needs to search more options for a satisfiable solution, which incurs larger space.

3.5.5.3 Performance of ConfigSynth Refinement Mechanism

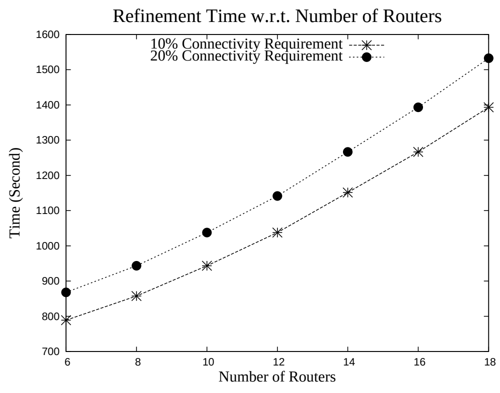
We first evaluate the capability of the our refinement mechanism in optimizing the resiliency architecture in terms of isolation. Next, we also evaluate the time-efficiency of the mechanism.

3.5.5.3.1 Isolation Optimization Capability

We compare the optimized isolation provided by the refinement mechanism with the optimum isolation found by applying the brute-force method. We consider a random



(a)



(b)

Figure 24: The resiliency architecture refinement mechanism execution time with respect to: (a) the number of hosts and (b) the number of routers.

network of 50 hosts and 10 routers, execute the refinement process (Algorithm 1) to get the optimized isolation, and compare it with the optimum value. The results are shown in Figures 23(a) and 23(b), where we vary the deployment cost and usability, respectively for two different values of the base isolation requirement, 5 and 6. In the first case, the usability constraint is kept at 5, while in the second case the cost constraint is fixed at \$100. In Figure 23(a), we observe that the improved isolation received after the refinement process is close ($< 10\%$) to the optimum value, and when the base isolation requirement is high, the refined isolation becomes more close. However, with the increase in the base isolation requirement, the execution time also increases rapidly, which ultimately increases the running time of the refinement process. We observe almost 5 times higher time consumption on average in the case of base isolation requirement 6 compared to the lower base. We observe similar characteristics in Figure 23(a).

3.5.5.3.2 Time Efficiency

Figures 24(a) and 24(b) show the execution time of the ConfigSynth refinement mechanism with respect to the problem size: the number of hosts and the number of routers, respectively. In the first case, where we vary the problem size with respect to the number of hosts, we consider two different sizes of connectivity requirements. In one scenario, the volume of the connectivity requirements is 10% of all possible flows, while in the second scenario, it is 20%. The execution times are shown in Figure 24(a). We observe that the analysis time increases quadratically with the number of hosts. This is due to various reasons. The first reason is the quadratic time complexity of ConfigSynth (Section 3.5.5.2.2). Second, the refinement process needs to execute the framework many times (often hundreds of executions) although the time requirement is not simply the multiplication of the number of times the ConfigSynth framework is executed. This is because the resiliency architecture is often improved based on the initial resiliency architecture. Therefore, the increase is within a reasonable amount of time. It is worth mentioning that the isolation, which we receive after the refinement process, most often cannot be achieved within the execution time of the refinement process if that isolation is given as the initial isolation requirement.

In the second case, the core network is varied by changing the number of routers. We again considered the same two different connectivity requirements. The results are presented in Figure 24(b). Due to the increase in the number of routers, the core network becomes larger, and more links are found to be candidates for security device placements. Therefore, both the synthesis time and the execution time of the refinement mechanism increase.

3.5.5.4 Discussion

Our evaluation results show that the time and memory requirements of ConfigSynth increases quadratically with the problem size. However, a synthesis problem with 500 hosts (*i.e.*, several thousands of flows) needs 800 seconds and 100 MB memory. It may seem that this number of hosts is small compared to large enterprise networks. However, in most of the large networks, it is common for many hosts to exhibit similar properties. They are often running the same OS and services, and are operated by the same level of users (*e.g.*, a student lab in a university or a customer service center in an organization). They usually reside under the same subnet. The resiliency configurations required for such a group is expected to be the same. Therefore, this group can easily be assumed as a single host.

3.6 Resiliency Architecture Synthesis Based on Redundancy

We start this section with a description of the architecture of ConfigSynth in the case of redundancy-based resiliency design, particularly with respect to the operational integrity and robustness requirements. Then, we discuss the corresponding formal model and illustrate the model with an example. Finally, we present the evaluation results for this redundancy-based resiliency synthesis.

3.6.1 Synthesis Framework

We formally model the problem of synthesizing AMI configurations with respect to several during-attack resiliency requirements, which particularly include a set of operational integrity and robustness constraints. The framework architecture, as

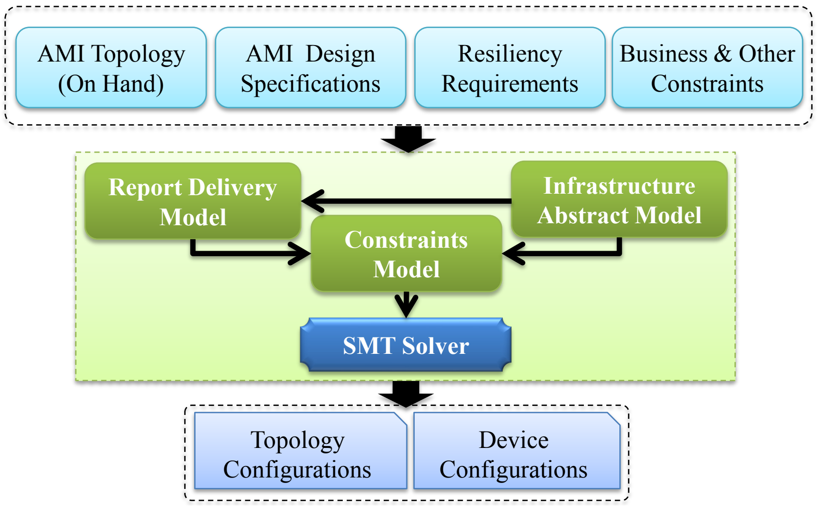


Figure 25: The AMI resiliency configurations synthesis framework.

shown in Figure 25, follows a top-down resiliency design automation approach. The framework includes the following tasks:

- Formal modeling of the AMI system, *i.e.*, the topology, devices, and interactions (*e.g.*, data deliveries) among the devices.
- Formal modeling of operational integrity and robustness requirements on top of the AMI system model, satisfaction of which determines necessary AMI configurations, which include an appropriate deployment of AMI devices (particularly collectors) and their report schedules.
- Implementing or encoding of the model using satisfiability modulo theories (SMT) and solving it using an SMT solver.

The synthesis framework takes different inputs as shown in Figure 25: (i-ii) specifications about the AMI topology and devices (smart meters and collectors), (iii) AMI topological and system invariants which are required for the operational integrity, and (iv) user-driven constraints including resiliency/robustness requirements along with deployment budget constraints. Our main resiliency requirement specifies the maximum tolerable damage (*i.e.*, the loss of reported usage data) in the case of the failure of a collector or a communication path. For a particular AMI topology, the smart meters are considered as already deployed, while the collectors are required to be deployed satisfying the requirements. With respect to the inputs, the framework models the deployment of collectors, reachability among the devices, report schedules, topological and operational integrity invariants, and robustness and other user-driven requirements. We follow a group based abstraction in modeling meters and collectors by considering the similarity among the properties to cope with the big number of AMI devices. The solution to the model provides the deployment design of AMI including topology and devices' configurations, satisfying invariant and user-driven constraints that provide operational integrity and robustness to AMI. Topology configurations specify the placements of collectors, while devices' configurations include report schedules of the meters and the deployed collectors.

3.6.2 Formal Model of the Synthesis Framework

In this section, we first define the parameters that define AMI configurations. Then, we present the formalizations of the synthesis framework as a constraint satisfaction problem. In our notations, variables start with small alphabetic letters, while constants start with capital letters.

3.6.2.1 AMI Configurations Parameters

We define different parameters to denote AMI configurations that include AMI devices and topology properties.

3.6.2.2 Configuration Level Abstraction

An enterprise AMI network typically consists of thousands of smart meters distributed over different geographical regions. These devices communicate to collectors for delivering data based on device configurations and communication properties. For the purpose of achieving better scalability, we apply the concept of abstraction in terms of groups based on the similarities between the configurations of the meters. A particular group or class of devices shares the same (physical and logical) configuration properties. Collectors are modeled as individual devices. Moreover, we use the idea of *zone* (similar to Section 2.5.2.1) to denote a collection of meters residing at the same geographical area. The meters within a zone form a mesh network to communicate to one or more collectors deployed in that zone. This collection of meters often forms a number of meter groups. Therefore, a meter group is identified or localized with the help of a zone.

3.6.2.3 AMI Device Configurations

A meter group is identified by $m_{k,i}$, where k is the zone index and i is the meter group index. There can be one or more meter groups in a zone, while the number of groups in the zone cannot be more than a threshold value. A meter group exists when $m_{k,i}$ is true. The objective of our synthesis framework is to synthesize properties of each existing meter group. Each group consists of a particular type of meters. Other properties of a meter group include the number of meters (*i.e.*, group size), the reporting schedule for them, and the collector to which the meters of this group need to report the usage data. We use $mType_{k,i}$ for denoting the meter type and $mSize_{k,i}$ for representing the group size. A particular type of meter is mainly specific to a vendor and it has a specific data sampling rate (*i.e.*, the number of samples per time slot), as well as a specific size for each sample. Since each meter in a group has the same type, they have the same property values. That is, the sampling rate and the sample size of each meter of a group are the same and they are denoted by $mSamplingR_{k,i}$ and $mSampleS_{k,i}$, respectively. The report schedule is represented by two parameters, the base (starting) time of reporting ($mReportB_{k,i}$) and the reporting interval ($mReportI_{k,i}$), which indicate that the meters of this group report periodically at each interval starting from the base time with respect to a specific time period, *e.g.*, during a day. We use $mC_{k,j}$ to identify the collector that is associated to the meter group. We assume minute as the unit for time slots and kilo bytes (KB) for the data or storage size.

A collector is represented by $c_{k,j}$, where k is the zone index and j is the collector index. Like a finite number of meter groups in a zone, we consider a finite number of (maximally) possible collectors in a zone. If $c_{k,j}$ is true, then the associated collector is deployed in the system. The objective of our synthesis framework includes synthesizing the configurations (*i.e.*, properties) of each collector that is deployed, and the correspondence between the meter groups and the collectors in a particular zone. We model a collector's profile with two properties: the type of the collector and its reporting schedule. A collector often follows the pull mode to forward the stored report data to the headend system. Therefore, the reporting schedule of a collector represents the report requesting schedule of the headend system specific to the collector. We use $cType_{k,j}$ to denote the collector type. A particular type of collector has a specific buffer size and a specific cost (price) to deploy this particular type of collector. The buffer size is denoted by $cBufferS_{k,j}$ and the deployment cost by $cCost_{k,j}$. Similar to a meter, the report schedule of a collector is also represented by two parameters: the base time $cReportB_{k,j}$ and the reporting interval $cReportI_{k,j}$.

3.6.2.4 AMI Topology Configurations

An AMI topology mainly defines the connectivity (*i.e.*, communication paths) between the AMI devices. As shown in Figure 3 in Chapter 2, the AMI topology, *i.e.*, the connectivity between the AMI devices, is well defined. The meters in a particular zone are considered to be connected to one or more collectors by forming a mesh network between them. Each collector is considered to be individually connected to the headend system through WAN communication, although all of the collectors share the path in the energy provider's network after the border router. We consider $MeshBW$ as the bandwidth of the mesh network communication, $IndividualBW$ as the bandwidth for each individual collector to the border router, and $SharedBW$ as the shared link bandwidth. The bandwidths of these communication paths play an important role for choosing the report schedules and the redundancy policy for robustness. In this particular work, we do not consider other details of the topology, because those details are not necessary for our synthesis objective.

3.6.2.5 Modeling of AMI Operational Integrity and Robustness Constraints

Ensuring the constraints associated with the system invariants (operational integrity constraints) and robustness requirement for an AMI system mitigates the threats due to the potential violations of these constraints.

3.6.2.6 Meter Groups and their Properties

The meters in a group have the same meter type. A valid meter type (between 1 to *MTypes* number of available types) for group i in zone k is ensured as follows:

$$m_{k,i} \rightarrow (mType_{k,i} \geq 1) \land (mType_{k,i} \leq MTypes)$$

We assume that the meters are already deployed. That is, the number of a particular type of meters in a zone is given. Since a meter group in a zone has a specific type, the size of the group must be within the number of meters of that particular type residing in the zone. The following equation ensures this constraint:

$$\begin{aligned} m_{k,i} \rightarrow & (mType_{k,i} = t) \rightarrow \\ & \quad ((mSize_{k,i} \geq 1) \land (mSize_{k,i} \leq MSize_{k,t})) \end{aligned}$$

In the above constraint formulation, $MSize_{k,t}$ denotes the number of meters of type t residing in zone k. Moreover, if we sum up the sizes of all meter groups in a zone having the same meter type, then the summation must be equal to the total of this particular type of meters in the zone. Therefore:

$$m_{k,i} \wedge (mType_{k,i} = t) \rightarrow (mSize_{k,i,t} = mSize_{k,i})$$

$$\neg(m_{k,i} \wedge (mType_{k,i} = t)) \rightarrow (mSize_{k,i,t} = 0)$$

$$MSize_{k,t} = \sum_{i} mSize_{k,i,t}$$

The sampling rate and the sample size of each meter of a meter group in a zone depend on its type. If $MSamplingR_t$ and $MSampleS_t$ are the sampling rate and sample size of a meter of type t, then the following is true:

$$\begin{aligned}(mType_{k,i} = t) \rightarrow (&mSamplingR_{k,i} = MSamplingR_t) \\ &\wedge (mSampleS_{k,i} = MSampleS_t)\end{aligned}$$

The meters of a meter group in a zone send their sampled data to a specific collector deployed in the same zone. If $CNum$ is the maximum number of potential collectors in a zone, then:

$$m_{k,i} \rightarrow (mC_{k,i} \geq 1) \wedge (mC_{k,i} \leq CNum)$$

In a particular zone, no two meter groups can have the same values for all properties. That is:

$$\begin{aligned}m_{k,i} \wedge m_{k,\hat{i}} \wedge i \neq \hat{i} \rightarrow \neg((&mType_{k,i} = mType_{k,\hat{i}}) \wedge (mC_{k,i} = mC_{k,\hat{i}}) \\ &\wedge (mReportB_{k,i} = mReportB_{k,\hat{i}}) \wedge \\ &(mReportI_{k,i} = mReportI_{k,\hat{i}}))\end{aligned}$$

3.6.2.7 Collectors and their Properties

There is a finite number of collector types. Let this number be $CTypes$. A collector's type must be within this set of types. Therefore, if collector j exists in zone k, then its type should satisfy the following constraint:

$$c_{k,j} \rightarrow (cType_{k,j} \geq 1) \wedge (cType_{k,j} \leq CTypes)$$

The buffer size and deployment cost of each collector in a zone depend on its type. Therefore, if $CBufferS_t$ and $CCost_t$ are the buffer size and the deployment cost of a collector of type t, then:

$$\begin{aligned}(cType_{k,j} = t) \rightarrow &(cBufferS_{k,j} = CBufferS_t) \wedge \\ &(cCost_{k,j} = CCost_t)\end{aligned}$$

If a collector is selected as the designated collector for a meter group in a zone for reporting, that indicates that the particular collector is deployed. Therefore:

$$\bigvee_{i} (mC_{k,i} = j) \rightarrow c_{k,j}$$

3.6.2.8 Report Schedule Constraints

We consider a finite number of potential values for the base time of the report schedule for meters as well as collectors. That is, if $\mathcal{B}_M$ and $\mathcal{B}_C$ are the set of potential base times for meters and collectors respectively, then $mReportB_{k,i} \in \mathcal{B}_M$ and $cReportB_{k,j} \in \mathcal{B}_C$. Similarly, we consider a finite set of potential values for reporting intervals. We have a number of invariant constraints to choose the reporting schedules. First, the base time of a report schedule must be lower than its interval. Second, a collector should report less frequently than its associated meters, considering that the latest data sampled by the meters can reach the headend system. The following equations ensure these constraints:

$$m_{k,i} \rightarrow mReportB_{k,i} < mReportI_{k,i}$$

$$c_{k,j} \rightarrow cReportB_{k,j} < cReportI_{k,j}$$

$$(mC_{k,i} = j) \rightarrow mReportI_{k,i} \leq cReportI_{k,j}$$

A collector should forward its stored usage data to the headend system in a timely manner so that no part of this data is overwritten with any new incoming report data. That is, the total incoming data from the meters within the report interval of the collector should not exceed its buffer. Moreover, the reporting data should not exceed the communication bandwidth. In order to formalize these constraints, we use *mReportAt* and *mReportS* to denote whether a meter reports at a particular time slot and the size of the reported data (on average), respectively. Similarly, *cReportAt* and *cReportS* are used for collectors. We find *mReportAt* and *cReportAt* as follows:

$$mReportAt_{k,i,s} \rightarrow m_{k,i} \rightarrow \\ ((s - mReportB_{k,i}) \% mReportI_{k,i} = 0)$$

$$cReportAt_{k,j,s} \rightarrow c_{k,j} \rightarrow \\ ((s - cReportB_{k,j}) \% cReportI_{k,j} = 0)$$

We compute the report size of a meter considering the average number of times a meter sends data to the associated collector within the reporting interval of the meter. We do the same for the case of reporting from a collector to the headend system. That is:

$$m_{k,i} \rightarrow (mReportS_{k,i} = mSize_{k,i} \times mSampleS_{k,i} \times mReportI_{k,i} / mSamplingR_{k,i})$$

$$c_{k,j} \rightarrow (cReportS_{k,j} = \sum_{\{i|(mC_{k,i}=j)\}} mReportS_{k,i} \times cReportI_{k,j} / mReportI_{k,i})$$

With the above formations of *cReportS*, which is ultimately the total usage data sent to the collector by the associated meters, the following equation ensures no overwrite on the stored data in the collector's buffer:

$$c_{k,j} \rightarrow cReportS_{k,j} \leq cBufferS_{k,j}$$

We assume that the bandwidth in a mesh network is shared by the participating nodes in the network. That is, all the meters (often hundreds in number) and the collectors (one or more in number) share a particular data transmission bandwidth (*i.e.*, they have a shared data throughput). Therefore, to ensure the successful delivery of usage data to a collector from the associated meters, the accumulated rate of data transmission by the meters must be within the bandwidth. The following equation ensures this communication bandwidth constraint:

$$\sum_{\{i|m_{k,i}\}} mReportS_{k,i} \leq MeshBW \times mReportI_{k,i}$$

Each collector is connected to the border router of the energy provider's network individually. Then, the rest of the communication path is shared by all of the collectors. Thus, the communication bandwidth constraints for the communication between the collectors and the headend system are as follows:

$$\mathit{cReportS}_{k,j} \leq \mathit{IndividualBW} \times \mathit{cReportI}_{k,j}$$

$$\sum_{\{k,j|c_{k,j}\}} \mathit{cReportS}_{k,j} \leq \mathit{SharedBW} \times \mathit{cReportI}_{k,j}$$

The grid operators may have constraints (often known as organizational or user-driven constraints) on the quality of the data delivery, especially with respect to the reporting delay. For example, a meter should report its usage data to the associated collector within a particular time interval, while a collector should not delay in forwarding the data to the headend system more than a threshold time. All these constraints are reflected in choosing the set of potential values for the report intervals for meters and collectors (*i.e.*, $\mathcal{B}_{\mathcal{M}}$ and $\mathcal{B}_{\mathcal{C}}$). Different organizational constraints can limit the data transmission delay (latency). For example, the data transmission delay from a meter to a collector should reach within a threshold time ($\mathit{MaxDelayMtoC}$). In the case of simultaneous reporting, *i.e.*, data delivery at the same time slot (*e.g.*, at slot s), the total data should be delivered within the threshold time. A similar threshold limit ($\mathit{MaxDelayCtoH}$) on the data transmission delay can be applied for collectors to the headend system. The following equations establish the above constraints:

$$\Big(\sum_{\{i|m_{k,i} \wedge \mathit{mReportAt}_{k,i,s}\}} \mathit{mReportS}_{k,i}\Big) \leq \mathit{MeshBW} \times \mathit{MaxDelayMtoC}$$

$$\mathit{cReportS}_{k,j} \leq \mathit{IndividualBW} \times \mathit{MaxDelayCtoH}$$

$$\left(\sum_{\{k,j | c_{k,j} \wedge cReportAt_{k,j,s}\}} cReportS_{k,j} \right)$$

$$\leq SharedBW \times MaxDelayCtoH$$

3.6.2.9 Robustness (Fault-tolerance) Constraints

The energy provider can have a robustness or fault-tolerance policy such that when one or more intermediate devices (*i.e.*, collectors) or one or more communicating links fail, the system still can operate without any damage (*e.g.*, data loss). In this work, we consider the standard $n-1$ contingency verification in smart electric grids, which is the 1-fault tolerance. More specifically, we assume a single node (device or communication link) failure maximally in a zone. As we consider mesh networks of smart meters for connectivity between meters and collectors, these networks are self-healing because there are alternative paths in the cases of intermediate meter or link failures toward the associated collector. Therefore, we focus on the collector failures and communication link failures from collectors to the headend system. We define robustness as one minus damage, where the damage is the number of meters (with respect to the meter groups) whose usage data is not ensured to be delivered to the headend system in the case of a collector or communication path (the path from a collector to the border router of the energy provider's network) failure.

A system is robust when there are alternatives to perform necessary operations. These alternatives can be found only if there is sufficient redundancy. For example, if one collector is sufficient for the meters in a zone, another collector is required for a single collector failure. Since the meters are connected with each other in a mesh network, and the collectors are connected to this mesh network, a meter can report data to a different collector through the same network. Therefore, the deployment of redundant collectors is only considered as a step to be applied for ensuring the robustness.

There are a number of constraints which must be satisfied to ensure the robustness. If a collector fails, the rest of the collectors in a zone must have enough buffer space to store the data reported by the meters of the zone. Since different collectors often have different reporting intervals (and base times), we can describe the same constraint in different words: the total data reported by the meters during the general cycle period (*e.g.*, a day) must be less than or equal to the maximum possible data that the rest of the collectors can store (*i.e.*, total buffer sizes) throughout the period (considering the number of times each collector reports to the headend system) without any overwrite. The following equation formalizes this constraint:

$$\begin{aligned} cRobust_{k,j} \rightarrow \sum_{\{j \mid c_{k,j}\}} cReportS_{k,j} \times Period/cReportI_{k,j} \leq \\ \sum_{\{j \mid (j \neq \hat{j}) \wedge c_{k,j}\}} cBufferS_{k,j} \times Period/cReportI_{k,j} \end{aligned}$$

Here, *Period* is the cycle period, often a day. This constraint needs to be ensured for each of the collectors' failures.

We need to consider the bandwidth limit in the contingency of a communication path failure. Although communication paths between collectors of neighboring zones could be deployed to be used to cover-up a link failure, redundant collectors deployed in zones can solve the problem. The collectors will provide alternative links to the headend system, as we assume without loss of any practicality that the collectors in a zone are connected to each other (often through the mesh access points) and each of them is connected to the utility (up to the border router) through a separate communication path. The following equation ensures enough extra bandwidth to forward the stored usage data from the collectors to the headend system in the case of a communication path failure:

$$\begin{aligned} cRobust_{k,j} \rightarrow & \sum_{\{j|c_{k,j}\}} cReportS_{k,j}/cReportI_{k,j} \\ & \leq \sum_{\{j|(j \neq \hat{j}) \wedge c_{k,j}\}} IndividualBW \end{aligned}$$

The above constraint must be ensured for the communication path failure for each of the collectors in the zone. A meter group is robust with respect to its collector failure or the associated communication path failure if, despite the failure of that collector, the system remains in operation. That is:

$$mRobust_{k,i} \rightarrow m_{k,i} \wedge \exists_j \; ((mC_{k,i} = j) \wedge cRobust_{k,j})$$

The robustness constraint, *i.e.*, the maximum possible damage (*MaxDamage*) in the case of a single node failure is formalized as follows:

$$\sum_{\{i|\neg mRobust_{k,i}\}} mSize_{k,i} \leq MTotal \times MaxDamage$$

3.6.3 An Example Case Study

3.6.3.1 SMT Encoding

We encode the system configurations and the constraints into SMT [23]. We write a program leveraging the Z3 Dot Net API [17] for encoding the formal model. We encode our formalizations mainly using Boolean (*i.e.*, for logical variables, such as $m_{k,i}$s, $c_{k,j}$s, etc.) and integer (*e.g.*, for the property values like $\mathit{mSampleS}_{k,i}$s, $\mathit{cBufferS}_{k,j}$s, etc.) terms. We use real terms for some of the variables (*e.g.*, $\mathit{mReportS}_{k,i}$s, $\mathit{cReportS}_{k,j}$s, etc.), where either they take real values or they are used in division, which can generate fractions. In our formalizations, presented earlier, we can see a number of division operations. Unfortunately, Z3 does not scale well in the case of divisions. That is why, in our encoding, we apply each division by taking the inverse of the corresponding denominator as a (real) variable. By executing the model (in Z3), we obtain the verification result as either satisfiable (*sat*) or unsatisfiable (*i.e.*, no solution exists). In the case of *sat*, we get the necessary AMI configurations from the assignments of the variables.

3.6.3.2 Example

We illustrate the execution of our formal synthesis model with a synthetic example. In this example, we consider an arbitrary AMI system of 1,000 smart meters distributed in 5 zones. It is required to find safe and reliable configurations of the AMI system, including the deployment of collectors and the report schedules (for both meters and collectors) within the given budget and robustness constraints. The input of the example is shown in Table 17. There are 2 types of meters. The number

Table 17: Input of the example in Section 3.6.3.2

```
# Number of meters and Zones
1000 5
# Distribution of meters in each zone based on meter types
100 1 100 2 0
250 1 50 2 200
120 1 60 2 60
230 1 100 2 130
300 1 200 2 100
# Number of Meter Types
2
# Meter Properties (Sampling interval (minute) starting from 0, sampling size (KB))
5 2
10 3
# Reporting Schedule (Potential base time (minute))
4
0 10 30 60
# Potential interval time (minute)
3
30 60 120
# Potential Maximum Meter Groups and minimum number of meters in a group
6 20
# Number of Collector types
2
# Collector Properties (Buffer size (KB) and deployment cost (k$))
20000 12
30000 16
# Reporting Schedule (Potential base time (minute))
4
0 60 120 240
# Potential interval time (minute) 3
120 240 360
# Potential Maximum Number of Collectors per Zone
4
# Link Bandwidth (kbps) (meter to collector, collector to headend- individual/shared)
40 100 200
# Freshness Constraint: Max report transmission delay
5 15 80 # meter to collector, collector to headend), % of data satisfying freshness
# Maximum Data Loss in Contingency
5
# Budget (Cost constraint in k$)
200
```

of each type of meters in a specific zone is given. Each type of meter has a particular set of properties (*i.e.*, sampling rate and size of each sample). A type 1 meter takes a sample (of size 2 KB) at each 5 minutes, while a type 2 meter takes a sample (of size 3 KB) at each 10 minutes. Two lists of potential values for the base time and the interval of the report schedule of a meter are also given. According to these values, we can see that the minimum reporting interval is 30 minutes, while the maximum is

Table 18: Output of the example: meters' configurations

Zone	Group Id	Meter Type	Group Size	Associated Collector Id	Reporting Base Time	Reporting Interval
1	1	1	20	2	60	120
1	2	1	20	2	10	120
1	3	1	20	2	0	30
1	5	1	20	1	0	30
1	6	1	20	1	0	120
2	1	1	30	1	30	60
2	3	2	200	1	10	30
2	5	1	20	3	60	120
3	1	2	60	1	0	30
...	...	...	...	...	...	...

120 minutes. The maximum number of meter groups expected in a zone is 6, while each group should have at least 20 meters. A collector can be either of 2 types, while each type has a different buffer size (*e.g.*, type 1 has 20,000 KB buffer, while type 2 has 30,000 KB buffer) and deployment cost. Similar to meters, there are two given lists of potential values of the reporting base time and the reporting interval. The maximum number of collectors that can be deployed in a zone is 4. The communication bandwidth between meters and collectors (*i.e.*, the mesh network) is 40 kbps. The individual link from a collector to the utility's border router is 100 kbps, while the shared link after the utility border router toward the headend system is 200 kbps. The organizational requirements specify the data freshness constraint and the robustness constraint, as well as the collector deployment budget. According to the freshness constraint, at least 80% of the data should reach (*i.e.*, the transmission delay) from a meter to a collector in 5 minutes, while from a collector to the headend system in 15 minutes. The robustness constraint specifies that the maximum data loss in a contingency (*i.e.*, in the case of a single node or link failure) is no more than 5%.

Our framework corresponding to this example returns a satisfiable result along with

Table 19: Output of the example: collectors' configurations

Zone	Collector Id	Collector Type	Reporting Base Time	Reporting Interval
1	1	1	240	360
1	2	1	120	360
2	1	1	60	120
2	3	2	60	360
3	1	2	0	240
3	2	1	60	120
4	1	1	60	120
4	4	1	120	240
5	1	2	60	120
5	2	1	60	120
5	3	2	120	360
5	4	1	60	120

the synthesis of necessary configuration parameters. The configurations associated with the meters are shown in Table 18, where we see that 5 meters groups are selected in each of zones 1 and 5, while 4, 3, and 2 meter groups are selected in zones 4, 2, and 3, respectively. The collector's id associated to each meter group is also shown. Similarly, in Table 19, we can see that 4 collectors are selected to be deployed in zone 5, while 2 collectors are selected for each of the remaining zones. The report schedules are selected in such a way that the collectors do the reporting in distributed time slots, which also consider the limited shared bandwidth. Moreover, reporting intervals are chosen such that report sizes satisfy the associated collector's buffer limit as well as the limited bandwidth. We can see the same in the case of meters. Note that the framework only synthesizes a satisfiable set of configurations that may not be the optimal solution.

3.6.4 Evaluation

We evaluate our AMI synthesis framework mainly in terms of scalability. We also verify the accuracy of the framework.

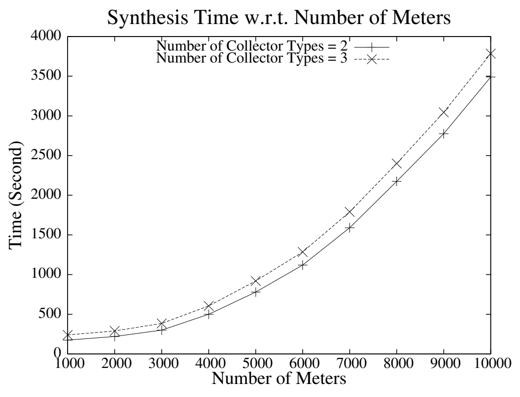
3.6.4.1 Accuracy

Although the formal modeling of the AMI resiliency architecture synthesis ensures the provability of the accuracy of the synthesized configurations, we still verify the accuracy of this model by executing the synthesized configurations with the help of SmartAnalyzer (presented in Chapter 2). We mainly verify the data overwrite protection constraint and the bandwidth limit constraint and find that the synthesized configurations correctly satisfy both of these constraints. In the case of the robustness constraint, we pick a random collector to be out of the AMI network, and see whether the data overwrite protection and bandwidth limit constraints still hold. Similarly, we also arbitrarily consider a communication link from a collector to the border router to be in a failure state (*i.e.*, excluding the link from the topology) and see whether the robustness constraints (*e.g.*, the bandwidth limit constraint) associated with this contingency still hold. In these cases, we find that the configurations synthesized by the framework satisfy the robustness constraints.

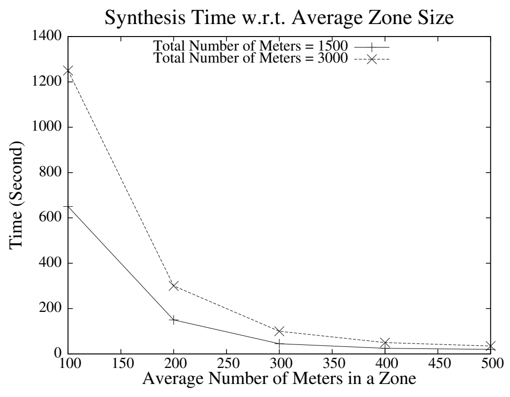
3.6.4.2 Scalability

3.6.4.2.1 Methodology

We evaluate the scalability of the AMI resiliency architecture synthesis model by analyzing the time and memory required in constraint verification by varying the AMI network size. We consider the network size as the total number of smart meters in the AMI system, which are distributed in different sizes of zones. We consider only a single headend system in the network. The size of each zone is considered between 200 and 500 meters. Since an organization usually is limited within the choice of a



(a)



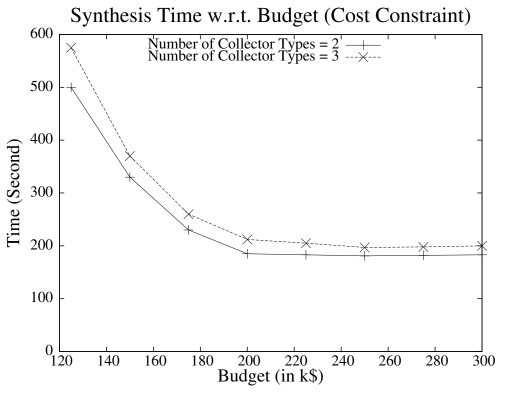
(b)

Figure 26: The execution of the synthesis framework with respect to (a) the number of meters and (b) the average size of each zone.

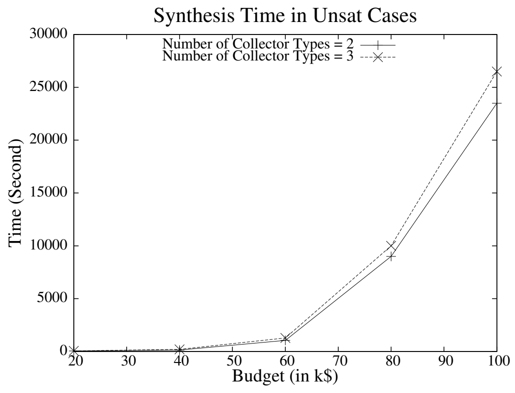
few types of meters and collectors, we consider up to 3 types of meters or collectors in our experiments. The deployment cost of a particular type of collector is taken arbitrarily. The number of potential values of the reporting base time as well as the interval is kept less than or equal to 10. We run our experiments on an Intel Core i5 machine with 4 GB memory.

3.6.4.2.2 Impact of the Problem Size

Figure 26(a) shows the execution time of our synthesis framework with respect to the AMI size, *i.e.*, the number of smart meters. We show the execution time in two different scenarios of the number of collector types. The graphs in the figure show that with the number of meters, the increase in the execution time lies between linear and quadratic growths. Although the number of parameters seems to be increased exponentially (as does the execution time), we observe complexity less than that. This is due to the application of the property-based abstraction (*i.e.*, the grouping of the meters when they share the same properties). The evaluation results with respect



(a)



(b)

Figure 27: Impact of the budget constraint on the execution time, (a) satisfiable cases and (b) unsatisfiable cases.

to the average size of each zone are shown in Figure 26(b). From the graphs we can see that the execution time decreases with the zone size. If the zone size increases, the number of zones reduces in a particular AMI network, which ultimately reduces the effective problem size.

3.6.4.2.3 Impact of the Constraints

The synthesis of AMI configurations depends on the given constraints, *e.g.*, the budget, freshness, and robustness requirements. However, the tighter the constraint, the more time is required to synthesize the configurations. We analyze the impact of this budget (*i.e.*, the deployment cost limit), on the execution time. The analysis results are shown in Figure 27(a). The graphs show that the execution time increases rapidly with the decrease of the budget. This is because the lower the budget, the more space is required by the solver to search for a satisfiable set of configurations, and thus the execution time increases. If the budget becomes much lower, there may be no solution. In the unsatisfiable cases, the execution time is often high, as

Table 20: Memory requirements (in MB) w.r.t. the problem size

Hosts	Scenario 1	Scenario 2
1000	43.20	46.20
2000	105.30	110.40
3000	168.10	175.00
4000	330.50	340.90
5000	465.90	478.60

whole of the search space needs to be traversed to conclude that there is no solution. However, if a constraint is too tight (*e.g.*, the budget is too low), the solver takes a much shorter time to conclude with unsatisfiability. In such cases, the potential space is small due to the highly tight constraints. Figure 27(b) shows the evaluation results in the unsatisfiability cases.

3.6.4.2.4 Memory Requirement

We evaluate the space (memory) requirement for executing our model in the SMT solver [23] by changing the number of meters. The memory requirement mainly includes the memory required for the variables that we use in modeling, and the intermediate variables that the solver uses to implement the theories applied in our constraint modeling. The analysis results are shown in Table 20 for two different scenarios. In the first scenario, the number of collector types is 2, while in the second scenario, the number is 3. We observe that the memory requirement lies between the linear and quadratic orders with respect to the number of meters. The table shows that the memory requirement in the second scenario is larger than the memory requirement in the first because, due to a larger number of collector types, there are more options (and so more variables) to design the deployment of collectors.

3.7 Conclusion

In this chapter, we present automated frameworks for synthesizing resiliency architectures in order to provide the secure and robust operation of the system. Both of these models are implemented using SMT. First, we present a formal framework for isolation-based resiliency design for cyber systems in smart grids. The framework synthesizes correct and cost-effective network isolation configurations. It formally models the network topology, pre-attack resiliency requirements in terms of network-based isolation, and the organizational business constraints in terms of usability and deployment cost, along with different invariant and user-defined constraints. Then, the framework formalizes the resiliency architecture synthesis as the conjunction of all the requirements and constraints. It solves the problem and results in a resiliency design along with optimal placements of security devices. We also develop a refinement mechanism that adds a feedback loop to ConfigSynth and applies hypothesis testing to find an improved resiliency architecture in a scalable manner. We evaluate ConfigSynth as well as the refinement mechanism in different synthetic networks and find that our solutions scale reasonably well with the problem size.

Next, we present a formal framework for the automated synthesis of AMI configurations that satisfy redundancy-based resiliency in terms of operational integrity and robustness properties. We model various constraints that are crucial for safe and robust data delivery in AMI systems. We model the robustness with respect to a single node (or link) failure, which is extendable for further robustness requirements. The execution of the formal model synthesizes necessary configurations satisfying the constraints. The accuracy of our framework is evaluated using an existing AMI threat analyzer. We evaluate the scalability of our framework in different synthetic AMI networks and requirements, and observe that its execution time is around an hour for a network of 10,000 smart meters in our particular computing environment. We achieve significantly high scalability by applying the group level abstractions to the model. In our future work, we would like to address the post-attack resiliency, which is the after-attack recoverability aiming to keep the attack damage minimum.

CHAPTER 4: THREAT ANALYTICS AND SECURITY HARDENING FOR POWER SYSTEM STATE ESTIMATION

In modern energy control centers, the energy management system (EMS) refers to a set of computational tools which are employed for system wide monitoring, analysis, control, and operation. A schematic diagram of EMS and its modules are shown in Figure 28. State estimation is the core module in EMS that estimates the system state variables from a set of real-time telemetered measurements (from meters) and topology statuses (from breakers and switches). The term "states" denotes bus voltages, from which power flows through transmission lines can be computed. As seen in Figure 28, the output of state estimation is required by several other modules, *i.e.*, optimal power flow (OPF), contingency analysis, and automatic generation control (AGC), for economic dispatch calculations and security assessment.

Cyber technologies are increasingly used in smart power grids with the promise of providing larger capacity, higher efficiency, and more reliability [67]. While this integration helps energy providers to offer smarter services, real time demand responses, and economic advantages, power grids also become vulnerable to cyber attacks. Particularly, cyber intrusions and false data injections can be launched against power grids, which can cause improper controls leading to serious damages, including power outages and destruction of critical equipment [26, 68].

In the case of state estimation, an attacker can compromise meters or commu-

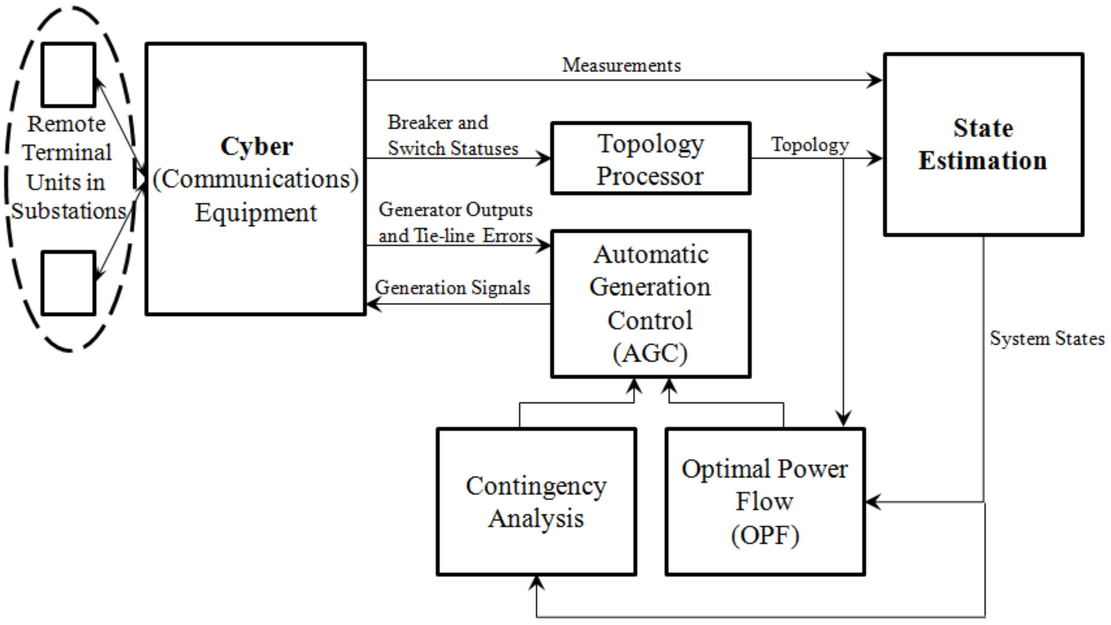

Remote Terminal Units in Substations
Cyber (Communications) Equipment
Measurements
Breaker and Switch Statuses
Topology Processor
Topology
State Estimation
Generator Outputs and Tie-line Errors
Automatic Generation Control (AGC)
Generation Signals
System States
Contingency Analysis
Optimal Power Flow (OPF)

Figure 28: Energy control center system security schematic (thanks to Allen J. Wood and Bruce F. Wollenberg, *Power Generation, Operation, and Control*, 2nd Edition [1]).

nication media to introduce malicious measurements, which can lead to incorrect state estimation. There are bad data detection algorithms [49, 69], which detect bad measurements principally based on the square of differences between observed and estimated measurements with some threshold values. It has been shown that an attacker can generate bad measurements with the knowledge of the grid, which can bypass the bad data detection [44]. As a result, states are estimated incorrectly, which can easily lead the system to a non-optimal and vulnerable situation. Stealthy attacks of this kind are known as undetected false data injection (UFDI) attacks. It is crucial to develop a threat analytics framework which can identify potential UFDI attacks considering different attack models, as well as the interdependency among different EMS modules.

4.1 Background

We start this section with a brief overview of the DC power flow model, which has been widely used to analyze stealthy attacks on state estimation (*e.g.*, [44, 70]). This DC model is simplistic, yet useful in preliminary analytical power systems studies.

4.1.1 DC Power Flow Model

The DC power flow model describes the power balance equations in a lossless power system [1]. With voltage magnitudes at all buses fixed at 1 per unit (p.u.), the only variables are phase angles. Therefore, the voltage phasor at bus i is given by $1\angle\theta_i$. Denoting the admittance of the transmission line between buses i and j by Y_{ij}, the real power flow (P_{ij}) across a transmission line is given by: $P_{ij} = Y_{ij}(\theta_i - \theta_j)$ where Y_{ij} is the reciprocal of the reactance. The model expresses the power-balance constraint which equates the algebraic sum of powers incident at every bus to zero. This yields a linear system of equations of the form: $[\mathbf{B}][\theta] = [\mathbf{P}]$. One of the buses is designated as the reference bus (also known as the slack bus), where $\theta_i = 0$. Assuming n buses, $[\mathbf{B}]$ is an $n - 1$ dimensional square matrix, and $\mathbf{P}$ is an $n - 1$ dimensional column vector whose elements denote the net power demand (*i.e.*, load minus generation) at a bus and $[\theta]$ is a column vector of unknown phases corresponding to the bus voltage phasors. The model solves unknown bus voltages, given admittances of the lines and net power demands at the buses. This linear model provides the basis for DC state estimation which is described next.

4.1.2 State Estimation

The state estimation problem based on the DC model is to estimate the bus voltages given several measurements of transmission line power flows. Specifically, one needs to estimate n number of the state variables $\mathbf{x} = (x_1, x_2, \cdots, x_n)^T$ based on m number of meter measurements $\mathbf{z} = (z_1, z_2, \cdots, z_m)^T$ [49]. Under the DC power flow assumptions, the measurement model is linear (*i.e.*, the measured power flows are linear functions of the bus voltages) and hence the measurement model reduces to:

$$\mathbf{z} = \mathbf{H}\mathbf{x} + \mathbf{e}, \text{ where } \mathbf{H} = (h_{i,j})_{m \times n}$$

The measurement set has redundant elements (*i.e.*, $m > n$), which are used for creating an over-determined set of linear equations. The redundancy enables the detection, elimination, and smoothing of unavoidable gross measurement errors. When the measurement error distribution is Gaussian with zero mean, state estimate $\hat{\mathbf{x}}$ is expressed by the following equation:

$$\hat{\mathbf{x}} = (\mathbf{H}^T\mathbf{W}\mathbf{H})^{-1}\mathbf{H}^T\mathbf{W}\mathbf{z} \tag{13}$$

Here, $\mathbf{W}$ is a diagonal "weighting" matrix whose elements are reciprocals of meter error variances. Thus, estimated measurements are calculated as $\mathbf{H}\hat{\mathbf{x}}$. The measurement residual $||\mathbf{z} - \mathbf{H}\hat{\mathbf{x}}||$ is used to determine bad data. If $||\mathbf{z} - \mathbf{H}\hat{\mathbf{x}}||$ is greater than τ, a selected threshold value, it indicates bad data.

4.1.3 Topology Processor

EMS uses a *topology processor* (refer to Figure 28) to map the grid topology. This processor receives telemetered statuses of various switches and circuit breakers in the system to determine network connectivity. When the connectivity matrix $\mathbf{A}$ and the branch admittance matrix $\mathbf{D}$ are known, the measurement matrix $\mathbf{H}$ is computed as follows (as in [71]):

$$\mathbf{H} = \begin{bmatrix} \mathbf{DA} \\ -\mathbf{DA} \\ \mathbf{A}^T\mathbf{DA} \end{bmatrix} \tag{14}$$

Matrices $\mathbf{DA}$ (*i.e.*, multiplication of $\mathbf{D}$ and $\mathbf{A}$) and $-\mathbf{DA}$ represent the line power flows in forward and backward directions, respectively. The matrix $\mathbf{A}^T\mathbf{DA}$ represents power consumption at the buses.

4.1.4 Optimal Power Flow

The state estimated solution (from Equation (13)) estimates bus voltages from which the system power flows are computed. Summing up the net power flows incident on a bus then yields the estimated power (or load) at that bus. System conditions determined from state estimation are then used in the OPF module (see Fig 28).

The OPF problem aims to minimize the total cost of *generation* subject to the following constraints: (i) the total system load is served and (ii) equipment ratings, transmission line limits, and control variables are satisfied [1]. Denoting the generation cost of generator k by $C_k(P_k)$, where C_k depends on the nature of the plant (*e.g.*, fossil fired, combined cycle, etc.), the OPF routine with respect to the DC power flow model is described by:

$$min \sum_{i} C_k(P_k) \text{ s.t.} \tag{15}$$

$$[\mathbf{B}][\theta] = [\mathbf{P}] \tag{16}$$

$$|P_{ij}| \leq P_{ij}^{max} \tag{17}$$

$$P_k^{min} \leq P_k \leq P_k^{max} \tag{18}$$

Here, Equation (15) describes the objective function of minimizing the total cost of generation, subject to power flow constraints in Equation (16), transmission line capacities in Equation (17), and generation capacities in Equation (18).

4.2 Challenges

Here we describe the idea behind the stealthy attacks on state estimation. We also define the attack attributes that must be considered to identify potential threats.

4.2.1 UFDI Attack

Liu et al. have introduced a interesting kind of attack, named UFDI attack, against state estimation [44]. They have shown that it is possible to generate a stealthy attack vector that can bypass the bad data detection process. The idea is briefly explained here. Consider an attacker who injects arbitrary false data $\mathbf{a}$ to the original measurements $\mathbf{z}$ such that $\mathbf{a} = \mathbf{Hc}$, *i.e.*, a linear combination of the column vectors of $\mathbf{H}$. Here, $\mathbf{c}$ is added to the original state estimate $\hat{\mathbf{x}}$ due to the injection of $\mathbf{a}$. Since $\mathbf{z}+\mathbf{a} = \mathbf{H}(\hat{\mathbf{x}}+\mathbf{c})$, the residual $||(\mathbf{z}+\mathbf{a})-\mathbf{H}(\hat{\mathbf{x}}+\mathbf{c})||$ still remains the same as $||\mathbf{z} - \mathbf{H}\hat{\mathbf{x}}||$. Thus, the bad data detection is evaded. Note that this requires knowledge of $\mathbf{H}$, *i.e.*, the system topology, electrical properties of the transmission lines, and measurement configurations. Moreover, the attacker's capability should be considered to identify the potential threats on state estimation.

4.2.2 Attack Model

Our goal is to describe attacks in their most general form so that adversarial capabilities can be modeled. Thus, we characterize attacks in terms of different attributes. The attributes that represent the attack model include mainly the attacker's accessibility, resources, and knowledge of the system, which are described below.

Accessibility: An attacker may not have access to all of the measurements when physical or remote access to substations is restricted, or when certain measurements are already secured. For example, in order to inject false data into the measurements taken at a substation (*i.e.*, bus), an attacker needs to have the access to that substation (or to the corresponding remote terminal unit) [72].

Resource constraint: An adversary may be constrained in cost or effort to mount attacks on vastly distributed measurements. In such cases, an adversary is constrained to compromising or altering a limited subset of measurements at a time. It is useful to represent this resource limitation with respect to buses. In order to launch an attack, if it requires false data injections to a set of measurements distributed in many substations (*i.e.*, buses), then it would be harder for the attacker to inject false data into those measurements compared to the set of measurements distributed in a small number of substations.

Grid topology and knowledge: State estimation of a power system is done based on the given topology (*i.e.*, connectivity among the buses) of the grid. This topology is mapped by the topology processor. For a successful UFDI attack, an attacker needs to know the grid topology and the electrical parameters of the transmission lines, which is not trivial [44]. In the case of partial knowledge, the attacker's capabilities become restricted. On the other hand, an attacker can inflict novel UFDI attacks against state estimation by conveying false status information at the transmitting devices or media, such that the topology generated by the processor includes one or more open lines (*i.e.*, non-existing in the true topology), or excludes one or more closed lines (*i.e.*, existing in the true topology).

Attack goal: An attacker may have the aim of corrupting a chosen set of states or a specific portion of the system, which often requires to perform false data injections to a certain set of measurements.

Modeling the state estimation, its interdependency with other EMS modules, and the consideration of all these attack attributes in a single model is crucial to identify potential stealthy threats and their impact on the power grid. We address this challenge in this work.

4.3 Related Work

The concept of an undetected false data injection attack is presented by Liu et al. for the first time in [44], and this concept is extended later in [73]. The authors discuss UFDI attacks considering different scenarios, such as limited access to meters and limited resources to compromise meters, under arbitrary or specific targets, assuming that the adversary has complete information about the grid. In the general case, the attack vector computation problem is NP-complete. Therefore, the authors present few heuristic approaches that can find attack vectors. Bobba et al. [74] show that for detecting UFDI attacks it is necessary and sufficient to protect a set of basic measurements, which is actually a minimum set of measurements ensuring observability. Kim and Poor [75] propose a greedy suboptimal algorithm, which selects a subset of measurements that can be made immune from false data injection for the protection against UFDI attacks. Kosut et al. [76, 77] propose a mechanism based on the generalized likelihood ratio test to detect UFDI attacks. A similar approach is found in [78] with the help of adaptive cumulative sum control chart test.

Vukovic et al. [72] propose a number of security metrics to quantify the importance of individual buses and the cost of attacking individual measurements considering the vulnerability of the communication infrastructure. Sou et al. [79] claim that an l_1 relaxation-based technique provides an exact optimal solution of the data attack construction problem. UFDI attacks with incomplete or partial information are discussed in [70, 80]. These works mathematically show the impact of incomplete knowledge on the potentiality of UFDI attacks. Recently, Kim and Tong present algebraic conditions of undetected topology attacks in power grids [81]. However, none of the works discussed above provides a comprehensive model of UFDI attacks considering different attack attributes together. It is also necessary to investigate the other forms of attacks, *e.g.*, topology poisoning attacks along with the measurement alterations, in order to introduce novel UFDI attacks on states. In addition, we need to devise an automated security architecture synthesis mechanism that can consider the grid operator's resource constraints with respect to an attack model. A grid may not require the absolute security which needs expensive deployment of security measures, or the grid operator may choose to use its available but limited resources to secure the system up to a desired level.

There is a few works that show the impact of the stealthy attacks on the economic operation of the power grid. For example, Esmalifalak et al. [82] present a game-theoretic analysis showing the impact of UFDI attacks on energy markets considering the locational marginal pricing (LMP). However, we still need to investigate other economic operation modules, *e.g.*, optimal power flow. Moreover, we need to consider multiple modules together to understand the conjoint impact.

As we have seen that UFDI attacks can be defended by securing a strategically selected number of measurements, the associated cost can be beyond the capability of the stakeholders. Therefore, an easier and cheaper security solution like moving target defense mechanisms can be a good option. MTD techniques have been presented for traditional enterprise networks in recent literature. Antonatos et al. propose a network address space randomization scheme to offer an IP hopping approach that can defend against hitlist worms [83]. Duan et al. [84, 85] present proactive random route mutation techniques, which enables the random and simultaneous changes of the routes of the multiple flows in a network. However, to the best of our knowledge, moving target based defenses have not received as much attention in SCADA and other control networks. Mo and Sinopoli [86] propose perturbing the input signal to a control system in order to detect replay attacks. Controlled perturbation of line admittances to detect UFDI is proposed in [87, 88]. Line admittance perturbation and parameter estimation are used to enhance the detectability of UFDI attacks on non-linear state estimation in [89]. However, none of these works studies the security of the grid applying the idea of MTD. Therefore, it is crucial to explore the potentiality of MTD to defend UFDI attacks against state estimation.

4.4 Contributions

The prior work addresses UFDI attacks considering different attack attributes in isolation. Therefore, we take the challenge to assess the attack feasibility when all the attack attributes are considered simultaneously. The interrelation among the attack variables has an integral impact on the attack feasibility. We model the UFDI attack on state estimation as a constraint satisfaction problem, the solution to which answers whether a UFDI attack can be launched in a particular scenario with respect to a given set of constraints.

Our formal model framework allows a grid operator to analyze and explore potential threats under different attack scenarios and initiate appropriate security measures. The framework is also extended to consider the impact of UFDI attacks on other interdependent modules, particularly OPF, to understand the economic loss. In order to make the state estimation secure against stealthy attacks, we propose an automated mechanism to synthesize a security architecture (*i.e.*, measurements that need to be secured) within the grid operator's resources. This architecture satisfies the security requirements that actually specify the protection of state estimation from UFDI attacks with respect to a given attack model. In addition, we also devise a mechanism which provides proactive security to the grid by introducing agility in the system with the help of moving target defense strategies.

Table 21: Modeling parameters

Notation	Definition
b	The number of buses in the grid.
l	The number of lines in the grid topology.
f_i	The *from-bus* of line i.
e_i	The *to-bus* of line i.
d_i	The admittance of line i.
g_i	Whether the admittance of line i is known.
P_i^L	The power flow through line i.
P_j^B	The power consumption at bus j.
θ_j	The state value, *i.e.*, the voltage phase angle, at bus j.
n	The number of states.
m	The number of potential measurements.
a_i	Whether measurement i is required to be altered for the attack.
c_j	Whether state j is infected/affected due to false data injection.
h_j	Whether any measurement residing at bus j is required to be changed.
t_i	Whether potential measurement i is taken (*i.e.*, reported by a meter).
r_i	Whether measurement i is accessible to the attacker.
s_i	Whether the measurement is secured or not.
u_i	Whether line i exists in the true (real) topology.
v_i	Whether line i is fixed in the topology.
w_i	Whether the status information regarding line i is secured.
p_i	Whether line i is excluded from the topology by an exclusion attack.
q_i	Whether line i is included in the topology by an inclusion attack.
k_i	Whether line i is considered (though it may not exist) in the topology.

4.5 Attack Vector Verification

In this section, we present our model of verifying the potentiality of UFDI attacks. In order to model UFDI attacks, we need a number of parameters to denote different system properties and attack attributes. These parameters are shown in Table 21.

4.5.1 Preliminaries

According to the DC power flow model, the admittance of a line or branch is computed from its reactance. The direction of the line is taken based on the current flow direction, *i.e.*, from one end-bus to another end-bus. The two end-buses of line i are denoted using f_i (*from-bus*) and e_i (*to-bus*), where $1 \leq i \leq l$, $1 \leq f_i, e_i \leq b$, and b is the number of buses. The admittance of the line is denoted by d_i.

Each row of $\mathbf{H}$ corresponds to a power equation. The first l rows correspond to the forward line power flow measurements. The next l rows are the backward line power flow measurements, which are the same as the first l except the directions of the power flows are opposite. We use P_i^L to denote the power flow through line i, while P_j^B to denote the power consumption at bus j, and θ_j to denote the state value (*i.e.*, the voltage phase angle at bus j). Then, we have the following relation between the power flow of line i (P_i^L) and the states of the connected buses (f_i and e_i):

$$\forall_{1\leq i\leq l} \;\; P_i^L = d_i(\theta_{f_i} - \theta_{e_i}) \tag{19}$$

Equation (19) specifies that power flow P_i^L depends on the difference of the connected buses' phase angles and the line admittance. The last b rows of $\mathbf{H}$ correspond to the bus power consumptions. The power consumption at bus j is simply the summation of the power flows of the lines connected to this bus. Let $\mathbb{L}_{j,in}$ and $\mathbb{L}_{j,out}$ be the sets of incoming and outgoing lines of bus j, respectively. Then, the following equation represents the power consumption at bus j:

$$\forall_{1\leq j\leq b} \;\; P_j^B = \sum_{i\in\mathbb{L}_{j,in}} P_i^L \; - \sum_{i\in\mathbb{L}_{j,out}} P_i^L \tag{20}$$

The power consumption at a bus is also equal to the load power at this bus minus the power injected to it by the connected generators. If P_j^D and P_j^G denote the load power and generated power of bus j, respectively, the following equation holds:

$$\forall_{1\leq j\leq b} \;\; P_j^B = P_j^D \; - P_j^G \tag{21}$$

If bus j is not connected with any generator, then $P_j^G = 0$. Similarly, if bus j does not have any load, then $P_j^D = 0$.

Basically, state estimation in DC model is the solution to the linear equations for all of the measurements (P_i^Ls and P_j^Bs) given the line admittances (d_is).

4.5.2 Parameters for Modeling UFDI Attack

We use c_j to denote whether state x_j ($1 \leq j \leq n$) is affected (*i.e.*, changed to an incorrect value) due to false data injection. In the DC model, each state corresponds to a bus. Thus, n is equal to b. Parameter a_i denotes whether measurement z_i ($1 \leq i \leq m$) is required to be altered (by injecting false data) for the attack. If any measurement at bus j is required to be changed, h_j becomes true.

Here, we model incomplete information with respect to line admittance only and use the variable g_i to denote whether the attacker knows the admittance of line i. Note that if the end-buses of a line are unknown, the corresponding row in $\mathbf{A}$ is fully unknown to the attacker. In this case, there is no way for an adversary to launch UFDI attacks on the system. In the DC model, two measurements, the forward and backward power flows, can be taken for each line. We use the term "taken" to specify that the measurement is recorded by a meter/censor at a targeted point (here, one end of the transmission line), and it is reported to the control center. For each bus, a measurement can be taken for the power consumption at the bus. Therefore, for a power system with l number of lines and b number of buses, there are $2l + b$ number of potential measurements (z_is). Though a significantly smaller number of measurements are sufficient for state estimation, redundancy is provided to identify and filter bad data. We use t_i to denote whether potential measurement z_i is taken. Note that though m is often used to represent the taken measurements, in this model, m represents the maximum number of potential measurements (*i.e.*,

$2l+b$). The attacker may not be able to alter a measurement due to inaccessibility or existing security measures. We use r_i to denote whether measurement z_i is accessible to the attacker. We also use s_i to denote whether the measurement is secured.

4.5.3 Parameters for Modeling Topology Poisoning

The topology of a power grid represents the connectivity among the grid buses. An attacker can inject false data in the topology information sent by various circuit breakers and switches in order to change the topology. Changes in the topology that we assume in this work include: (i) exclusion of a closed line from the topology (*exclusion attack*), and (ii) inclusion of an open line in the topology (*inclusion attack*). Here, we also assume that the adversary can coordinate a topology error with other measurements to render the attack undetected. Therefore, a UFDI attack can be performed by leveraging the modified topology.

We assume that some of the lines in the topology are fixed (*i.e.*, they are never opened), which form the core part of the topology. We also allow the declaration of secure line statuses, *i.e.*, their topology is always faithfully represented in state estimation. In order to model all these properties plus the topology change, we use a list of notations as shown in Table 21. We use u_i to denote whether line i is the true or real topology, while v_i and w_i denote whether the line is fixed and the line status is secure, respectively. In order to denote exclusion and inclusion attacks on line i, we use p_i and q_i, respectively. Finally, k_i represents whether line i is considered/mapped in the topology.

4.5.4 Formalization of Change in State Estimation

The attack on state x_j specifies that the phase angle at bus j is changed. This condition is formalized as follows:

$$\forall_{1 \leq j \leq n} \ \ c_j \rightarrow (\Delta\theta_j \neq 0) \tag{22}$$

From Equation (19), it is obvious that a change of P_i^L is required based on the changes in state x_{f_i} (θ_{f_i}) and/or state x_{e_i} (θ_{e_i}). In the case of false data injection, P_i^L, θ_{f_i}, and θ_{e_i} are changed to P'^L_i, θ'_{f_i}, and θ'_{e_i}, respectively, and Equation (19) turns into the following form:

$$P'^L_i = d_i(\theta'_{f_i} - \theta'_{e_i})$$

The subtraction of Equation (19) from the above equation represents whether there are changes in the measurements and the states. The following is the resultant equation:

$$\Delta P_i^L = d_i(\Delta\theta_{f_i} - \Delta\theta_{e_i})$$

In this equation, $\Delta P_i^L = P'^L_i - P_i^L$, $\Delta\theta_{f_i} = \theta'_{f_i} - \theta_{f_i}$, and $\Delta\theta_{e_i} = \theta'_{e_i} - \theta_{e_i}$. If $\Delta\theta_{f_i} \neq 0$ (or $\Delta\theta_{e_i} \neq 0$), then it is obvious that state x_{f_i} (or x_{e_i}) is changed (*i.e.*, attacked). The above relation for line i holds only if the line is taken in the topology. We formalize this constraint as follows:

$$\forall_{1 \leq i \leq l} \ \ k_i \rightarrow (\Delta P_i^L = d_i(\Delta\theta_{f_i} - \Delta\theta_{e_i})) \tag{23}$$

If a line is not considered in the topology, then there should be no requirement of false data injection to corresponding measurements for launching UFDI attacks:

$$\forall_{1 \leq i \leq l} \;\; \neg k_i \rightarrow (\Delta P_i^L = 0) \tag{24}$$

4.5.5 Formalization of Topology Change

In the case of an inclusion attack, a line is considered in the topology though the line is open in reality. Conversely, a closed line in service is omitted in an exclusion attack. These are formalized as follows:

$$\forall_{1 \leq i \leq l} \;\; k_i \rightarrow (u_i \wedge \neg p_i) \vee (\neg u_i \wedge q_i) \tag{25}$$

A line can be excluded from the topology if and only if the line exists in the real or true topology and it is not a securely fixed line. This is formalized as follows:

$$\forall_{1 \leq i \leq l} \;\; p_i \rightarrow u_i \wedge \neg fl_i \wedge \neg w_i \tag{26}$$

Similarly, a line can be included in the topology if the following condition holds:

$$\forall_{1 \leq i \leq l} \;\; q_i \rightarrow \neg u_i \wedge \neg w_i \tag{27}$$

Note that for a topology error to remain undetected, it is necessary to alter certain measurements in necessary amounts. If a closed line is excluded from the topology, the corresponding line power flow measurement must be zero. As the states remain the same after the topology change, the corresponding connected buses' power consumption measurements are adjusted accordingly. On the other hand, when an open line is included in the topology, there should be a non-zero line power flow according to the phase difference between the connected buses. Let $\Delta \bar{P}_i^L$ be the change amount in the power flow measurement of line i in the case of a topology change. Then, the following constraints hold:

$$\forall_{1 \le i \le l} \;\; p_i \rightarrow (\Delta \bar{P}_i^L = -P_i^L) \tag{28}$$

$$\forall_{1 \le i \le l} \;\; q_i \rightarrow (\Delta \bar{P}_i^L = P_i^L) \tag{29}$$

If no exclusion or inclusion attack is done on line i, then $\Delta \bar{P}_i^L = 0$. Now, if line power flow measurement i (or $l+i$) needs to change, according to Equations (28) and (29), we need to know P_i^L. In the case of an exclusion attack, P_i^L already exists (*i.e.*, the actual measurement) and the attacker must have access to it. In the case of an inclusion attack, P_i^L needs to be estimated based on the difference between the states of the connecting buses.

4.5.6 Formalization of False Data Injection to Measurements

Here, we compute and formalize required changes to be applied to the measurements for coordinating the attack. The change for a power flow measurement is the summation of individual changes that are required for topology poisoning and state corruption. If $\Delta P_{i,total}^L$ is the total change required on the line i's power flow, then:

$$\forall_{1 \le i \le l} \;\; \Delta P_{i,total}^L = \Delta P_i^L + \Delta \bar{P}_i^L \tag{30}$$

According to Equation (20), the change in the measurement of the power consumption ($\Delta P_{j,total}^B$) at a bus depends on the total changes done in the power flow measurements of the lines incident to this bus. Therefore,

$$\forall_{1 \le j \le b} \; \Delta P_{j,total}^B = \sum_{i \in \mathbb{L}_{j,in}} \Delta P_{i,total}^L - \sum_{i \in \mathbb{L}_{j,out}} \Delta P_{i,total}^L \tag{31}$$

When $\Delta P^L_{i,total} \neq 0$, the measurements corresponding to line i (*i.e.*, t_i and t_{l+i}) are required to be altered if they are taken. Similarly, when $\Delta P^B_{j,total} \neq 0$, the power consumption measurement at bus j needs to be changed if this measurement is taken. Therefore:

$$\begin{aligned} &\forall_{1 \leq i \leq l} (\Delta P^L_{i,total} \neq 0) \rightarrow (t_i \rightarrow a_i) \wedge (t_{l+i} \rightarrow a_{l+i}) \\ &\forall_{1 \leq j \leq b}\ (\Delta P^B_{j,total} \neq 0) \rightarrow (t_{2l+j} \rightarrow a_{2l+j}) \end{aligned} \tag{32}$$

Conversely, measurement z_i is altered only if it is taken and the corresponding power measurement is changed:

$$\begin{aligned} &\forall_{1 \leq i \leq l}\ \ a_i \rightarrow t_i \wedge (\Delta P^L_{i,total} \neq 0) \\ &\forall_{1 \leq i \leq l}\ \ a_{l+i} \rightarrow t_{l+i} \wedge (\Delta P^L_{i,total} \neq 0) \\ &\forall_{1 \leq j \leq b}\ \ a_{2l+j} \rightarrow t_{2l+j} \wedge (\Delta P^B_{j,total} \neq 0) \end{aligned} \tag{33}$$

4.5.7 Formalization of Attack Attributes

4.5.7.1 Attacker's Knowledge

If the admittance of a line is unknown, then an adversary cannot determine the necessary changes that need to be applied to the measurements associated with the line. We formalize this condition as follows:

$$\forall_{1 \leq i \leq l}\ \ (\Delta P^L_i \neq 0) \rightarrow ((t_i \vee t_{l+i} \vee t_{f_i} \vee t_{e_i}) \rightarrow g_i) \tag{34}$$

The following equation shows an example of specifying the attacker's knowledge about the admittances of the lines:

$$g_1 \wedge g_2 \wedge g_3 \wedge \neg g_4 \wedge \cdots \wedge g_l \tag{35}$$

4.5.7.2 Attacker's Accessibility

The attacker usually does not have the necessary physical or remote access to inject false data into all the measurements. If a measurement is secured, then, although the attacker may have the ability to perform false data injection to the measurement, the false data injection will not be successful. Hence, the attacker will only be able to change measurement z_i if the following condition holds:

$$\forall_{1 \leq i \leq m} \ \ a_i \rightarrow r_i \wedge \neg s_i \tag{36}$$

It is necessary to specify whether a measurement is secured or not, as well as whether or not a measurement is accessible to the attacker. The following equations are examples of such specifications:

$$\neg s_1 \wedge s_2 \wedge \neg s_3 \wedge \neg s_4 \wedge \cdots \wedge s_m \tag{37}$$

$$r_1 \wedge \neg r_2 \wedge r_3 \wedge \neg r_4 \wedge \cdots \wedge r_m \tag{38}$$

4.5.7.3 Attacker's Capability for Simultaneous Attacks

The resource limitation specifies that, at a particular time, the attacker can inject false data into T_A number of measurements, at the maximum:

$$\sum_{1 \leq i \leq l} a_i \ \leq T_A \tag{39}$$

Due to limited resources, an attacker can only access or compromise a limited number of buses at a particular time. A bus is required to be accessed or compromised if a measurement residing at this bus is required to be altered. Therefore:

$$\begin{aligned} &\forall_{1\leq i\leq l} \;\; a_i \rightarrow h_{f_i} \\ &\forall_{1\leq i\leq l} \;\; a_{l+i} \rightarrow h_{e_i} \\ &\forall_{1\leq j\leq b} \;\; a_{2l+j} \rightarrow h_j \end{aligned} \tag{40}$$

Let T_H be the maximum number of substations that the attacker can compromise. Then:

$$\sum_{1\leq j\leq b} h_j \leq T_H \tag{41}$$

4.5.7.4 Attacker's Target

The attacker most often has a selected set of states for launching an attack. However, the attacker usually has no specification on the rest of the states. Thus, an unspecified state might be attacked or not. For example, if the attacker targets states 1, 4, and 6, then:

$$c_1 \wedge c_4 \wedge c_6 \tag{42}$$

It is possible to launch a UFDI attack on a number of measurements if the attacker can form a cut that divides the grid into two disjoint islands [80]. The attacker can attack all of the buses of one side of the cut with respect to the other side by altering the power flow and consumption measurements of the lines and the buses on the cut. However, in this case, all of the attacked buses have the same change of their states (*i.e.*, phase angles). If the state change of a bus is the same as that of the neighboring buses, then there is no state change relative to each other. In this case, the impact due to the attack might not be significant. Therefore, we also consider the constraints

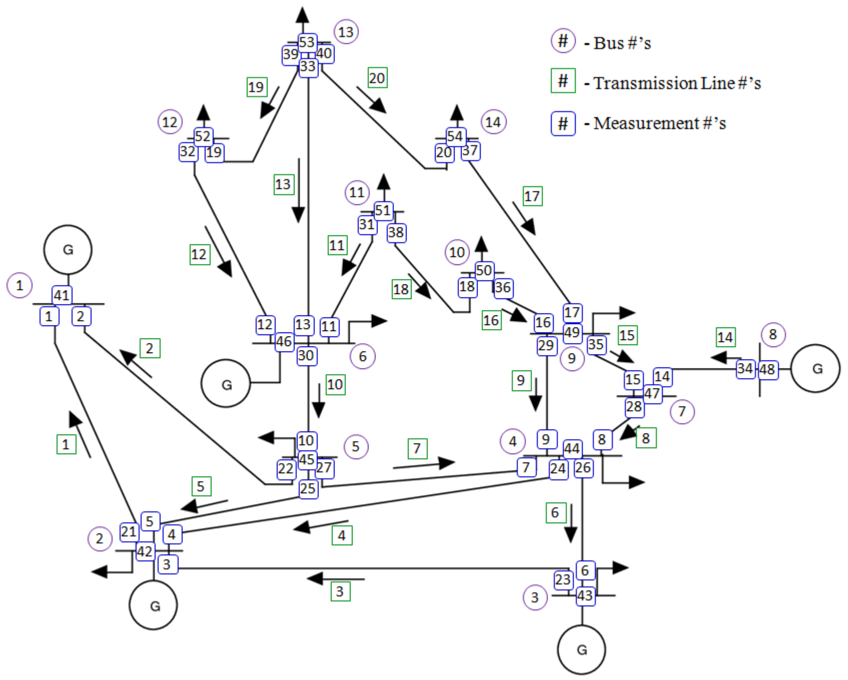


Figure 29: The diagram of the IEEE 14-bus test system. Red circles are used for bus numbers, green squares are for transmission line numbers, and round cornered blue squares are for measurement numbers.

specifying whether state changes are required to be different. For example, if the attacker requires that state 1 and state 4 must have a different amount of change, then:

$$(\theta_1 \neq \theta_4) \wedge \cdots \tag{43}$$

4.5.8 An Example Case Study

In this section, we briefly discuss the process of implementing our formal model. Later, we present a synthetic case study.

Table 22: Line information of the example in Section 4.5.8

Line #	From Bus	To Bus	Line Admittance	Knowledge?	True?	Core?	Secured?	Can Alter?
1	1	2	16.90	1	1	1	0	0
2	1	5	4.48	1	1	1	0	0
3	2	3	5.05	0 [a]	1	1	0	0
4	2	4	5.67	1	1	1	0	0
5	2	5	5.75	1	1	0 [b]	0	0
6	3	4	5.85	1	1	1	0	0
7	4	5	23.75	0	1	1	0	0
8	4	7	4.78	1	1	1	0	0
9	4	9	1.80	1	1	1	0	0
10	5	6	3.97	1	1	1	0	0
11	6	11	5.03	1	1	1	0	0
12	6	12	3.91	1	1	1	0	0
13	6	13	7.68	1	1	0	0	0
14	7	8	5.68	1	1	1	0	0
15	7	9	9.09	1	1	1	0	0
16	9	10	11.83	1	1	1	0	0
17	9	14	3.70	0	1	1	0	0
18	10	11	5.21	1	1	1	0	0
19	12	13	5.00	1	1	1	0	0
20	13	14	2.87	1	1	1	0	0

[a]The attacker does not know the impedance of this line.

[b]This line is not fixed in the topology (*i.e.*, it is not a part of the core topology).

4.5.8.1 SMT Encoding

We encode the system configuration and the constraints into SMT [23]. We write a program leveraging the Z3 Dot Net API [17] for encoding our formal model. We encode our formalizations mainly using Boolean (*i.e.*, for logical constraints) and real (*e.g.*, for the relation between power flows or consumptions with states) terms. The system configurations and the constraints are given in an input file. By executing the model (in Z3), we obtain the verification result as either satisfiable (*sat*) or unsatisfiable (*unsat*). If the result is *unsat*, it means that there is no attack vector that satisfies the constraints. In the case of *sat*, we get the attack vector from the assignments of the variables, a_is (and h_is), which represent the measurements that must be altered for the attack.

Table 23: Measurement info of the example in Section 4.5.8

Measurement #	Is Recorded?	Secured	Can Alter?
1	1 [a]	1 [b]	0
2	1	1	0
3	1	0	1 [c]
4	1	0	1
5	0	0	0
...	...	...	...
11	1	0	1
12	1	0	1
13	1	0	1
14	0	0	0
15	1	1	1
...	...	...	...
21	1	0	1
22	0	0	0
23	1	0	1
24	1	0	1
25	1	1	1
...	...	...	...
41	1	1	0
42	1	0	1
43	1	0	1
44	1	0	1
45	1	1	0
...	...	...	...

[a]The measurement is taken or recorded for state estimation.
[b]The measurement is secured, especially in terms of integrity.
[c]The attacker has the accessibility to alter the measurement.

4.5.8.2 Example

We present our results on the IEEE 14-bus test system (see Figure 29) [90]. The input about the line information is shown (partially) in Table 22. The line information includes a set of data for each line: line number, end buses of the line, a value indicating the line admittance, the knowledge status (*i.e.*, whether the line admittance is known to the attacker), and three types of data about this line regarding the grid topology (*i.e.*, whether this line is included in the actual topology, whether its existence is fixed in the topology, and whether associated topology information is secured). In this example, the admittances of lines 3, 7, and 17 are unknown. All of the 20 lines (as shown in Figure 29) are included in the true topology, though lines 5 and 13 are not a part of the core topology. The topology statuses regarding these two lines are crucial to estimate whether they are in open state or closed state.

The input about the measurements is partially shown in Table 23. Since this system has 14 buses and 20 lines, the maximum number of potential measurements is 54. Each row of Table 23 includes (i) whether the measurement is taken for state estimation (all the potential measurements are taken except measurements 5, 10, 14, 19, 22, 27, 30, 35, 43, and 52), (ii) whether the measurement is secured (measurements 1, 2, 6, 15, 25, 32, and 41 are secured) and (iii) whether the attacker has the accessibility to alter the measurement (*e.g.*, among the taken measurements 1, 2, 3, and 4, measurements 3 and 4 are accessible, while measurements 1 and 2 are inaccessible). Let us now consider two different objectives of the attacker.

Attack objective 1: Let the attacker's objective be to attack states 9 and 10 but in different amounts. Due to resource limitations, the attacker cannot alter more than 16 measurements at a time, and these measurements cannot be distributed in more than 7 substations (*i.e.*, buses). The execution of the model corresponding to this example returns *sat* along with the assignments to different variables of the model. From the assignments, we find that the measurements selected for attacking states 9 and 10 are 8, 9, 16, 18, 20, 28, 29, 36, 38, 40, 44, 47, 50, 51, 53, and 54. These measurements are distributed in buses 4, 7, 9, 10, 11, 13, and 14. If the attacker's resources are more limited (*e.g.*, 15 measurements and/or 6 buses only), then *unsat* is returned. However, if the attacks on states 9 and 10 can be the same, then there is a solution. In this case, the measurements for false data injection are 8, 9, 11, 13, 28, 29, 31, 33, 39, 44, 46, 47, 49, 51, and 53, while the corresponding buses are 4, 6, 7, 9, 11, and 13. In both of these cases, along with 9 and 10, some other states are also required to be corrupted; only states 9 and 10 cannot be attacked alone.

Attack objective 2: Here the attacker's objective is to attack state 12 only (*i.e.*, no other states will be affected). The execution of the corresponding model shows that measurements 12, 32, 39, 46, and 53 must be altered in this case. If measurement 46 is considered as secured, then no attack vector is possible. Let us now consider that the attacker has the ability to alter the topology information. In this scenario, we have a solution, where line 13 is excluded from the topology by injecting false data into the topology information. In this case, the measurements for false data injection are 12, 13, 32, 33, 39, and 53, which include necessary changes required for the state change along with the topology change.

4.6 Attack Impact Verification

In this section, we first discuss briefly the framework of verifying the impact of stealthy attacks on OPF. Then, we discuss the associated formal models. We provide explanatory examples to demonstrate the formal framework.

4.6.1 Framework

We follow the framework as shown in Figure 30 for verifying the impact of stealthy attacks on OPF. The framework includes two models: (i) the stealthy attack model that finds attack vectors corresponding to stealthy topology attacks, and (ii) the OPF model that verifies whether there is an OPF solution within a threshold cost. Since the objective is to launch a stealthy attack such that the cost of power generation (according to the OPF solution) increases by a specific amount or more, the idea of

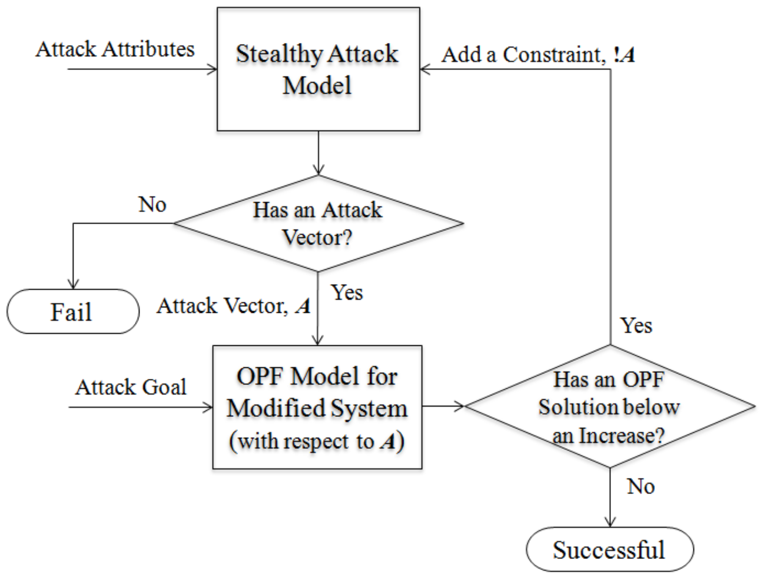


Figure 30: The framework for finding the impact of UFDI attacks on OPF.

impact analysis is as follows. First, we look for an attack vector according to the attack model (*i.e.*, attack attributes). If the attack model gives an attack vector, we update the system with respect to that vector (*i.e.*, according to the changed loads and the modified topology). Then, we verify whether there is an increase in the generation cost by executing the OPF model. In order to verify this increase, we set the threshold cost by adding the expected raise with the original (*i.e.*, in the no attack scenario) OPF solution and check whether there is still an OPF solution within this threshold value. If the result is no, then we are successful in finding an attack vector that causes a minimum amount of increase in the generation cost. Otherwise, the same process will be executed for a new attack vector until either we find a vector satisfying the objective or there are no more vectors. It is worth mentioning that the objective is to increase the generation cost while ensuring convergence of OPF, considering the power generation limit of each generator and the capacity of each transmission line. The framework combines the stealthy attack model and the OPF model into a single model, although they can be executed separately as shown in Figure 30.

4.6.2 Formalization of Impact of UFDI Attacks on OPF

In order to model the impact of stealthy attacks on OPF, we first model the OPF process as a problem of verifying whether there is a generation dispatch plan satisfying the demand within a threshold generation cost. Then, we model the feasibility of a stealthy attack inducing a particular increase in the generation cost.

4.6.2.1 Optimal Power Flow

The objective of the OPF is to optimally control the generation according to the load requirement. Let $\hat{P}_j^G$ be the changed power produced by the generator connected at bus j after considering the state estimation result. The main constraint for OPF is that the total generation must be equal to the total expected load. Therefore:

$$\sum_{1 \le j \le b} \hat{P}_j^G = \sum_{1 \le j \le b} \hat{P}_j^D \tag{44}$$

Each generator has lower and upper bounds on power production. If $\hat{P}_{j,max}^G$ and $\hat{P}_{j,min}^G$ denote the maximum and minimum generation limits of the generator at bus j, then this constraint is formalized as follows:

$$\forall_{1 \le j \le b} \; \hat{P}_{j,min}^G \le \hat{P}_j^G \le \hat{P}_{j,max}^G \tag{45}$$

The OPF process considers the entire set of power flow equations as constraints, as illustrated in Equation (16) (Section 4.1). In the case of OPF, let $\hat{\theta}$, $\hat{P}_i^L$, and

$\hat{P}^B_j$ be the state of bus j, the power flow on line i, and the power consumption at bus j, respectively. Then, in the case of a power flow measurement, the following equation, similar to Equation (19) must hold, if and only if the line is considered in the topology:

$$\forall_{1 \leq i \leq l} \;\; k_i \rightarrow (\hat{P}^L_i = d_i(\hat{\theta}_{f_i} - \hat{\theta}_{e_i})) \tag{46}$$

Consequently, the following equations, similar to Equations (20) and (21), must hold:

$$\begin{aligned} &\forall_{1 \leq j \leq b} \;\; \hat{P}^B_j = \sum_{i \in \mathbb{L}_{j,in}} \hat{P}^L_i \; - \sum_{i \in \mathbb{L}_{j,out}} \hat{P}^L_i \\ &\forall_{1 \leq j \leq b} \;\; \hat{P}^B_j = \hat{P}^D_j \; - \hat{P}^G_j \end{aligned} \tag{47}$$

Each line has a capacity for the power flow (*i.e.*, the maximum power that can flow through that line). Let $P^L_{i,max}$ be the upper bound for the line capacity. Therefore:

$$\forall_{1 \leq i \leq l} \;\; \hat{P}^L_i \leq P^L_{i,max} \tag{48}$$

Let $\mathcal{C}_j(.)$ denote the cost function for the generator connected at bus j, which takes the total generated power as the parameter and returns the total cost to generate that power. Usually, $\mathcal{C}_j(.)$ is a strictly increasing convex function. Many electric utilities prefer to represent their generator cost functions as piecewise linear equations (*i.e.*, single or multiple segment linear cost functions [1]). Considering the viability of modeling the cost function, we consider the latter form for cost functions, given by the following equation:

$$\mathcal{C}_j(\hat{P}^G_j) = \alpha + \beta \hat{P}^G_j \tag{49}$$

Here α and β represent the cost-coefficients for that particular generator. In OPF, the objective is to minimize the total generation cost based on expected or estimated loads at different buses. With the loss of generality, we model this objective as a constraint which specifies that the cost must be less than a limit, T_{OPF}. This constraint is sufficient to understand the minimum impact of a UFDI attack. The constraint is formalized as follows:

$$\sum_{1 \leq j \leq b} \mathcal{C}_j(\hat{P}^G_j) \ \leq T_{OPF} \tag{50}$$

We use notation OPF to denote the conjunction of the OPF constraints that we have described above.

4.6.2.2 Change in Loads Due to Stealthy Attacks

According to Equation (21), $\Delta P^B_j \neq 0$ specifies that there is a load and/or generation power change at the bus. In this work, we assume that a change in the measurement of a bus power consumption specifies a change exclusively in the load, which leads to $\Delta P^G_j = 0$. The reason behind this assumption is as follows: The measurement of the power produced by a generator (*i.e.*, the power injected to the bus by a generator) is well-defined, which is changed only if AGC suggests that. Typically, after the estimation of states, if any load change is found, the optimal power flow process (along with contingency analysis) is run, the result of which shows whether (and which) change in the generation is required for optimal efficiency. Therefore, according to Equation (21), the change in the power consumption of a bus specifies the change in the load at that bus. The following equation denotes this:

$$\forall_{1 \leq j \leq b} \ \ \Delta P^D_j = \Delta P^B_{j,total}$$

Let $\hat{P}_j^D$ be the estimated load (according to the result of state estimation) at bus j, which is also the input to the OPF model. Therefore:

$$\forall_{1\leq j\leq b} \ \ \hat{P}_j^D = P_j^D + \Delta P_j^D$$

At a particular bus j, there is usually an expected bound for the load. If $\hat{P}_{j,max}^D$ and $\hat{P}_{j,min}^D$ are the maximum and minimum loads at bus j, the following constraint holds:

$$\forall_{1\leq j\leq b} \ \hat{P}_{j,min}^D \leq \hat{P}_j^D \leq \hat{P}_{j,max}^D \tag{51}$$

4.6.2.3 Impact on OPF

In order to define the increase in the generation cost (*i.e.*, the increase of T_{OPF} in the OPF model), let $\mathcal{T}_{OPF}$ be the optimal cost of generation in the normal (*i.e.*, attack-free) situation. Now, if the attacker's objective is to increase the cost by $I\%$ of the optimal cost, then $T_{OPF} = \mathcal{T}_{OPF} I/100$. Therefore, the constraint to impose the desired impact by launching a UFDI attack is formalized as follows:

$$(T_{OPF} = \mathcal{T}_{OPF} I/100) \rightarrow \neg \ (\exists_{\hat{P}_1^G, \hat{P}_2^G, \cdots, \hat{P}_b^G} OPF) \tag{52}$$

The above constraint states that there is no possible allocation of generation that can cost less than T_{OPF}.

In addition, since the attacker's goal is not to make the OPF solution fail to converge (possible when the line capacity constraints fail), it must be ensured that there are OPF solutions for larger values:

$$(T_{OPF} >> \mathcal{T}_{OPF} I/100) \rightarrow \ OPF \tag{53}$$

Table 24: Input of the example in Section 4.6.3

```
# Topology (Line) Information
# (line no, from bus, to bus, admittance, line capacity, knowledge?, in true topology?,
# in core topology?, secured?, can alter?)
1 1 2 16.90 0.30 1 1 1 1 0
2 1 5  4.48 0.30 1 1 1 1 0
3 2 3  5.05 0.20 1 1 1 1 0
4 2 4  5.67 0.30 1 1 0 0 1
5 2 5  5.75 0.30 1 1 1 1 1
6 3 4  5.85 0.30 1 1 1 1 1
7 4 5 23.75 0.30 1 1 1 1 1
. . . . . . . . . . . .

# Measurement Information
# (measurement no, measurement taken?, secured?, can attacker alter?)
1 1 1 0
2 1 1 0
3 1 0 1
4 1 0 1
5 0 0 0
6 1 0 1
7 1 0 1
8 1 0 1
9 1 0 1
10 0 0 0
. . . . . . . . . . . .

# Attacker's Resource Limitation (measurements, buses)
28 7

# Generator Information (bus no, max generation, min generation, cost coefficient)
5
1 1.80 0.20 15 200
2 1.20 0.10 20 220
3 1.60 0.10 25 120
6 1.60 0.20 20 200
8 1.60 0.20 15 140

# Load Information (bus no, existing load, max load, min load)
11
2 0.20 0.30 0.10
3 0.40 0.40 0.10
4 0.15 0.40 0.05
5 0.15 0.40 0.05
6 0.25 0.40 0.05
9 0.15 0.30 0.05
10 0.10 0.30 0.05
11 0.20 0.30 0.10
12 0.15 0.30 0.05
13 0.15 0.40 0.00
14 0.10 0.30 0.05

# Cost Constraint, Minimum Cost Increase by Attack (in percentage)
412 5
```

4.6.3 An Example Case Study

Here, we present an example case study demonstrating the impact of the stealthy attacks on the topology and the states. In these examples, we consider the same

14-bus system as shown in Figure 29.

The complete input regarding the example is shown in Table 24. The line information includes a set of data for each line: line number, end buses (from-bus and to-bus) of the line, a value indicating the line admittance, the line capacity (*i.e.*, the maximum possible power flow through this line), the knowledge status, and the line status properties: (i) whether this line is included in the true topology, (ii) whether its existence is fixed in the topology, (iii) whether the topology information regarding this line is secured, and (iv) whether the attacker has the ability to alter the data. According to the input, all of the 20 lines are included in the true topology, while lines 4, 14, and 17 are not included in the core topology. The topology mapping information regarding lines 4 and 14 is not secured, while the attacker has the capability to change the topology information regarding all of the lines, except 1, 2, and 3. According to the measurement information, all of the potential measurements are taken except measurements 5, 10, 14, 19, 22, 27, 30, 35, 43, and 52. Measurements taken at bus 1 are secured. The attacker has access to all measurements that are taken.

The information about the buses in terms of load and generation is also shown in Table 24. The capability of the generators (*i.e.*, the maximum and minimum generations) corresponding to the buses are given. We assume that a generation bus only has a single generator connected. The generation cost of power is followed from the simple linear function as shown in Equation (49). The values of coefficient α and β for each generator are given in the input. Note that these coefficients are taken arbitrarily and do not correspond to the real costs. The total load of the system is 2.0 per unit, *i.e.*, 200 MW (considering a 100 MVA base). The cost constraint in the attack-free condition is \$4,120 (*i.e.*, there is a satisfied OPF solution in this cost).

In this example, the attacker's objective is to launch a stealthy topology attack, such that he or she can create at least a 5% increase in the generation cost. In this example, the attacker's resource constraints limit alteration to a maximum of 28 measurements at a time. These measurements can be distributed at no more than 7 buses. The execution of the model corresponding to this example returns *sat* along with the assignments to different variables of the model. From the assignments, we find that:

- An exclusion attack on the topology is launched such that lines 4 and 14 are unmapped in the topology.
- States 3, 4, 6, 7, 8, 9, 10, 11, 12, 13, and 14 have been attacked.
- In order to keep this attack undetected, measurements 3, 4, 6, 7, 11, 13, 18, 23, 24, 26, 31, 33, 38, 39, 42, 44, 46, 50, 51, and 53 need to be altered. These measurements are distributed in buses 2, 3, 4, 6, 10, 11, and 13.

The increased generation cost is almost \$4,550, which is approximately 10% more than the optimal value received in an actual (*i.e.*, without attack) scenario.

4.7 Security Hardening Against Stealthy Attacks

In the last section, we have described the model for determining potential UFDI attacks under given constraints. The formal verification model allows a grid operator to understand potential threats on state estimation with respect to an expected scale of attack (expressed in terms of different attack attributes) and to take necessary security measures accordingly. However, we need an automated mechanism to generate a security architecture. In this section, we present such a mechanism.

4.7.1 Preface

Although the authors in [74, 75] show that UFDI attacks can be defended if a strategically chosen set of measurements are secured, they only consider a specific attack model, where adversaries have perfect knowledge and unlimited capability. Based on this worst case attack model, the set of measurements to be secured can exceed the grid operator's resource (*e.g.*, budget). Therefore, a security design is required that can provide security within the resources of the grid operator, while keeping the power system state estimation secure with respect to an attack model (*i.e.*, security requirements).

Our solution utilizes the verification model to determine a security architecture, which typically includes a list of measurements that must be secured. With respect to the false data injection, we consider a measurement as secured if it is data integrity protected. Since securing a number of measurements distributed in many substations is very costly compared to a set of measurements distributed in a small number of substations, we mainly focus on the substation/bus specific security architecture. Moreover, securing a bus usually means securing all of the measurements taken in that bus. A bus can be secured by deploying a PMU (can be multiple for a large bus) at the bus with necessary security measures [91, 92]. By the security measures, we mainly consider the data integrity protection of the measurements. Since the PMU can provide the voltage phasor of the bus and current phasors of all the branches incident

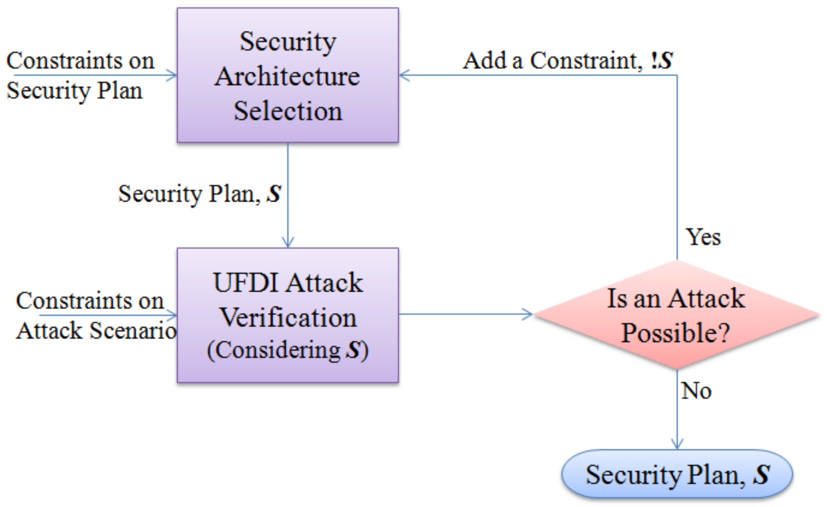

Constraints on Security Plan
Security Architecture Selection
Add a Constraint, !S
Security Plan, S
Yes
Constraints on Attack Scenario
UFDI Attack Verification (Considering S)
Is an Attack Possible?
No
Security Plan, S

Figure 31: The flow diagram of the security architecture synthesis mechanism for protecting state estimation attack.

to the bus, if the PMU is secured then all of these measurements become secured. At the unit level, security is being provisioned by existing PMU vendors [93]. Here, although we propose a mechanism to find the security architecture as a set of buses to be secured, a similar mechanism can be used for synthesizing security architecture with respect to measurements only.

4.7.2 Synthesis Design

Figure 31 shows the flow diagram of the security architecture synthesis mechanism for resisting state estimation attacks. It is an iterative approach with the combination of two formal models. One of these models is the candidate security architecture selection model. It selects the set of buses as a candidate of the security architecture considering some invariant and user-driven constraints on the security architecture. We discuss this candidate security architecture selection model in the following subsection.

The second model is our UFDI attack verification model, which verifies whether the selected candidate architecture can protect state estimation from UFDI attacks with respect to the security requirements (*i.e.*, an expected attack model). Security requirements are ensured when the verification model returns *unsat* (*i.e.*, no attack vector can be found). If a candidate architecture fails to ensure the required security, a constraint is added to the candidate security architecture selection model so that this architecture is removed from the potential candidate set. The updated model is solved for another candidate architecture and the verification model is used to ensure the security requirements. This process continues until a security architecture is found (*i.e.*, as long as the verification model returns *unsat*). However, when the candidate architecture selection model fails to return a candidate set, then no security architecture is possible according to the given security requirements.

4.7.3 Formalization of Candidate Architecture Selection

The main constraint for selecting the buses in the architecture is the resource limitations of the grid operator. The number of selected buses cannot exceed a limit (T_H). If h_j denotes whether bus j is secured, then:

$$\sum_{1 \leq j \leq b} h_j \leq T_H \tag{54}$$

Securing a bus implies that all of the measurements that are recorded at this bus are secured. If L_j denotes the lines connected to bus j, we formalize this as follows:

$$\begin{aligned} &\forall_{1 \leq j \leq b} \;\; h_j \rightarrow (t_{2l+j} \rightarrow s_{2l+j}) \\ &\forall_{1 \leq j \leq b} \;\; h_j \rightarrow \bigwedge_{i \in L_j} (t_i \rightarrow s_i) \wedge (t_{l+i} \rightarrow s_{l+i}) \end{aligned} \tag{55}$$

The grid operator may have the limitation that he or she is not capable of securing a particular set of buses. Those buses should be excluded from the candidate set, as shown in the following arbitrary example:

$$\neg h_2 \wedge \neg h_6 \wedge \cdots \tag{56}$$

Different analytical constraints can be used to limit the search space in the security architecture selection model. From Equation (23), we know that if no change is possible in the line power flow, the phase difference between the two buses connected by the line cannot be changed. Hence, if a bus is secured (*i.e.*, all the measurements at the bus are secured), the state of a connected bus cannot be changed with respect to the state of the secured bus. UFDI attacks on the states of these two buses are possible through a third bus which is not connected to the secured one but rather to the other. Therefore, securing the connected bus is not required to protect state estimation of the grid. Equation (57) formalizes this constraint.

$$\begin{aligned} \forall_{1 \leq j \leq b} \quad h_j \rightarrow \bigwedge_{i \in L_j} ((f_i = j) \wedge t_i) \rightarrow \neg h_{e_i}) \wedge \\ ((e_i = j) \wedge t_{l+i}) \rightarrow \neg h_{f_i}) \end{aligned} \tag{57}$$

4.7.4 An Example Case Study

In this subsection, we briefly describe the implementation of the synthesis technique followed by a case study illustrating our security architecture synthesis mechanism.

Algorithm 2 Security Architecture Synthesis

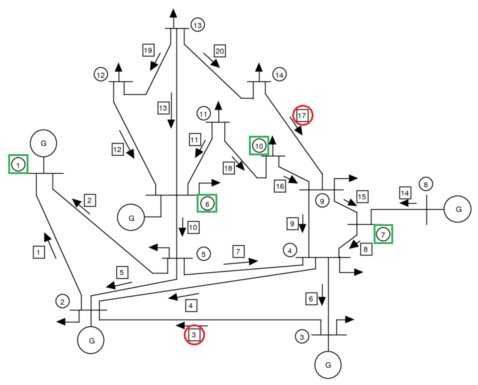
```
1: $F_{Attack}$ formalizes the UFDI attack verification model.
2: $F_{Secure}$ formalizes the security architecture selection model.
3: loop
4:    Save ($Push$) current $F_{Attack}$ into $\bar{F}_{Attack}$.
5:    if Solver returns a model $M$ ($i.e.$, $sat$) for $F_{Secure}$ then
6:       Get the security architecture $S$ from $M$.
7:    else
8:       Exit program.
9:    end if
10:   Add security constraints to $F_{Attack}$ based on $S$.
11:   if Solver returns $unsat$ for $F_{Attack}$ then
12:      Return $S$.
13:   else
14:      Add the constraint $!S$ to $F_{Secure}$.
15:   end if
16:   Retrieve ($Pop$) the saved formalization $\bar{F}_{Attack}$ into $F_{Attack}$.
17: end loop
```

4.7.4.1 Implementation

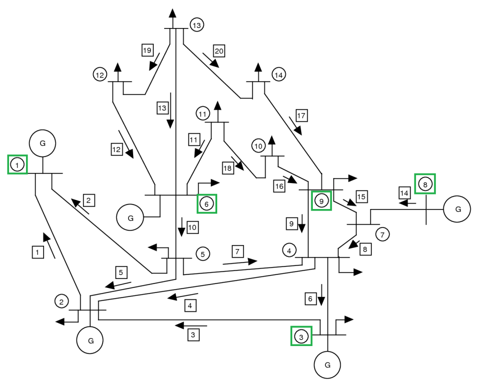
Similar to our verification model, we encode the candidate security architecture selection model using SMT [23]. Then, we implement the synthesis mechanism by combining the verification model and candidate selection model as shown in Algorithm 2. The algorithm is an iterative process, which stops when a security architecture is found (line 12) or there is no more candidate set available for verification (line 8).

4.7.4.2 Example

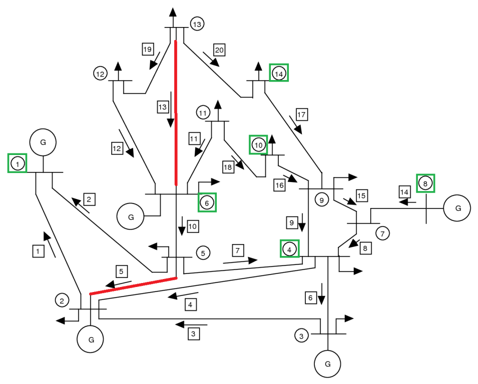
Here we present a case study based on the IEEE 14-bus test system illustrating how our security architecture synthesis mechanism produces different security architectures in various scenarios, as shown below:



(a)



(b)



(c)

Figure 32: The security architectures (the green squared buses needs to be secured) in different scenarios: (a) incomplete information (the red circled line's admittance is unknown), (b) complete knowledge, and (c) incorporating with topology poisoning attack (the read lines are potential to inclusion or exclusion topology attacks). In all scenarios, bus 1 is the reference bus.

Scenario 1: The attack model of the first scenario is similar to the first part of the example (attacker's objective 1) as shown in Section 4.5.8. In this scenario, the attacker has limited information (*i.e.*, admittances of lines 3 and 17 are unknown). The grid operator can consider such a constraint on the attacker's knowledge, if he or she is certain that the admittance information regarding this set of lines is neither disclosed nor predictable. The attacker also has limited resources, such that he or she cannot attack more than 12 measurements simultaneously. The grid operator, due to resource constraints, can secure 4 buses maximally. Bus 1 is considered as the reference bus. In this scenario, the security architecture produced by our mechanism suggests that buses 1, 6, 7, and 10 are must be secured, as shown in Figure 32(a) (*i.e.*, all the measurements in these buses are data integrity protected). However, there can be different sets of buses, which also can secure the system. Our synthesis mechanism can synthesize all of these sets.

Scenario 2: In the second scenario, the attacker knows the complete information (*i.e.*, all line admittances) for launching UFDI attacks and he or she has the ability to inject false data into any number of measurements. In this case, there is no solution with 4 buses that can secure state estimation of the grid against UFDI attacks. If the grid operator can secure 5 buses, there is a solution. In this solution, we need to secure buses 1, 3, 6, 8, and 9 (see Figure 32(b)).

Scenario 3: This scenario is the worst case situation compared to the last two scenarios. Here, the attacker has complete knowledge of the grid and he or she has the ability to inject false data into any number of measurements. In addition, the attacker can change the topology by injecting false data to the topology information. In this scenario, only lines 5 and 13 are considered vulnerable to line exclusion or inclusion attacks. However, in this case, no solution is possible by securing 5 buses only. If it is possible to secure 6 buses, then we have a satisfiable security architecture (*i.e.*, buses 1, 4, 6, 8, 10, and 14), which is shown in Figure 32(c).

4.8 Proactive Defense Against Persistent Attacks

4.8.1 Preface

The idea of moving target defense (MTD) has been studied for a decade, especially in the field of cybersecurity [94]. Typical information technology systems operate in a static environment. Configuration parameters, such as IP addresses, DNS names, network topology, routing entries, security policies, software stacks, etc. remain mostly static over relatively long periods of time. When a system is static, attackers get enough time to know the configuration and behavior of the system, to understand the vulnerabilities and corresponding attack vectors, and consequently to launch attacks on the system. The same is true for cyber-physical systems like power grids, where the physical and cyber systems are highly static, the operations are fixed, and the protocols are known.

Moving target defense is the concept of controlled change across multiple system dimensions in order to (i) increase uncertainty and apparent complexity for attackers, (ii) reduce their opportunity space, and (iii) increase the costs of their probing and attack efforts [95]. Usually, MTD is not meant to provide perfect security. The aim of MTD is to enable the operations to be executed safely in a compromised environment, where the system is defensible rather than perfectly secure, particularly against persistent attacks. We have already discussed the attack attributes for UFDI. The potential of moving target defense mechanisms lies in being able to randomize or perturb one or more of these attributes, where possible. In this work, we devise a moving target defense mechanism considering the knowledge attribute, where we add uncertainty to the information by changing the set of measurements and the topology properties (*i.e.*, line admittances). Although a persistent attacker may still be successful in launching UFDI attacks, due to the uncertainty introduced by the MTD strategy, the attack space reduces.

4.8.2 Moving Target Defense Strategy

In order to increase the uncertainty of the attacker's knowledge about the power grid system with respect to state estimation, our MTD mechanism takes two properties of the system: (i) the set of measurements that are considered in state estimation, and (ii) the admittances of a group of lines in the topology.

4.8.2.1 Randomization of the Set of Measurements

In regular practice, a fixed number of measurements is used in the state estimation process. According to the bad data detection algorithms, some of the measurements can be ignored in the process, if they are noisy enough (*i.e.*, bad) relative to the rest of the measurements. An adversary needs to know the set of measurements used in state estimation and alter a group of measurements from the set in order to launch a specific UFDI attack. If the attacker does not know the measurement set correctly, he or she may be unable to identify this group of required measurements perfectly (*i.e.*, one or more measurements can be missing in the group or included without necessity). Therefore, if we can randomize the measurement set used in state estimation by including a number of measurements from the unused (but possible) measurements, attackers' knowledge about the measurement set becomes uncertain.

For example, let us consider the IEEE 14-bus test system [90], which has 14 buses and 20 lines. With respect to the DC power model, it is possible to have 54 measurements (considering forward and backward power flows through transmission lines and power consumption at buses). Among these possible measurements, let us assume that a fixed set of 30 measurements is taken (*i.e.*, recorded and reported using sensors/meters) for state estimation, while the remaining 24 potential measurements are not. According to our MTD mechanism, we can take a set of 7 measurements from the unused measurements by deploying sensors there if necessary. Then, from the total of 37 measurements, we can select 30 at random to be used in state estimation. However, the selected set must be capable of observing the system. Later in this section, we present a formal model for selecting a measurement set according to the observability requirement.

4.8.2.2 Perturbation of Line Admittances

There are distributed flexible AC transmission system (D-FACTS) devices, which can be deployed on transmission lines and are capable of performing active impedance (*i.e.*, reactance) injections [96]. Leveraging this capability of D-FACTS devices, we consider the randomization of line admittances in our MTD mechanism. We assume that the admittance of a line can only be randomized if a D-FACTS device is deployed there. However, there are some limitations of using D-FACTS devices. Changes in impedance have impacts on the power flows, which can easily affect the power system operations (*e.g.*, the optimal power flow of the system [69]).

In order to obtain the effect on the power flows due to the deliberate changes in impedance of power lines with the help of D-FACTS devices, a sensitivity analysis related to D-FACTS devices is thoroughly explained in [97]. In our MTD mechanism, we consider a feasibility constraint in changing line admittances, which ensures that the secured optimal power flow solution [69] remains the same in spite of the changes in the admittances, although some of the power flows must change. We also need to ensure that the changes cannot be trivial. Further, all the lines with D-FACTS devices will not always be randomized. A set of lines among them will be chosen during each state estimation, and only admittances of these chosen lines will be perturbed. We assume that an adversary may know the actual admittance (*i.e.*, base admittance) of each of these lines, although he or she does not know the change amount. Therefore, the changed admittance is assumed to be unknown to the adversary. We also assume that when a set of line admittances is changed, the previously changed admittances are returned back to the base admittances. As a result, at a particular time, admittances of only the selected set of lines are unknown to the adversary.

Arguably, power system operations personnel may not be willing to perturb line impedances for the exclusive purpose of detecting attacks. However, D-FACTS-based perturbation of line parameters has been considered for minimization of power system losses and voltage control applications [97]. In practice, such line parameter changes could be leveraged for detecting attacks. In the rest of the paper, we illustrate the MTD through perturbation of line parameters as exclusively done for attack detection, while keeping in mind that perturbation done for other optimization applications could be leveraged instead.

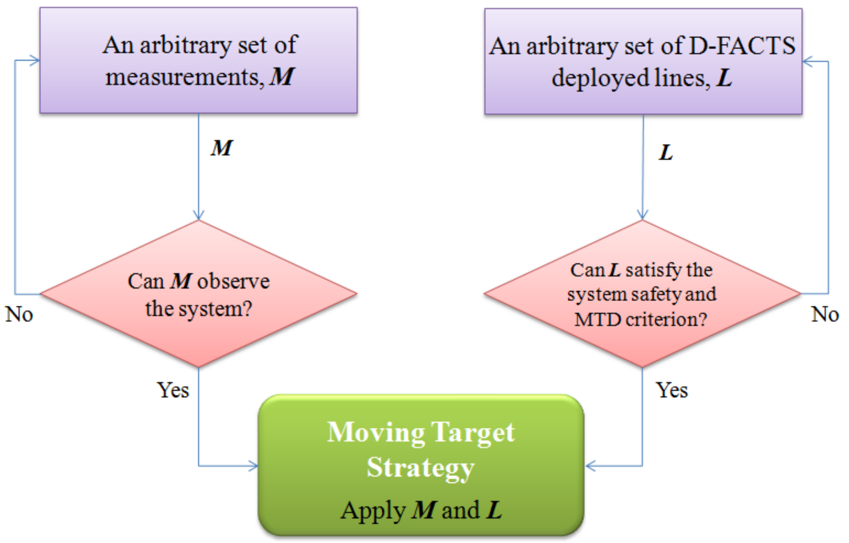

An arbitrary set of measurements, *M*
M
Can *M* observe the system?
No
Yes
An arbitrary set of D-FACTS deployed lines, *L*
L
Can *L* satisfy the system safety and MTD criterion?
No
Yes
Moving Target Strategy
Apply *M* and *L*

Figure 33: The moving target defense (MTD) mechanism for hardening the security of state estimation.

4.8.3 Formal Model for Strategy Selection

In Figure 33, we show the architecture of our MTD mechanism. It is a combination of two modules, as shown in the figure: one for the selection of an arbitrary set of measurements for state estimation, and another for the selection of an arbitrary set of lines and corresponding admittance perturbations. In this section, we present the formal designs of these two modules.

4.8.3.1 Selection of a Measurement Set

The power system is observable when the measurements, each of which represents a power equation, must solve the unknown states. Therefore, we consider Equations (19) and (20) as constraints. Now, if a measurement is taken, its power flow or consumption measurement value is assumed to be zero. That is:

$$\forall_{1 \leq i \leq l} \;\; (t_i \vee t_{l+i}) \rightarrow (P_i^L = 0)$$

$$\forall_{1 \leq j \leq b} \;\; t_{2l+j} \rightarrow (P_j^B = 0)$$

If the set of taken measurements can observe the system, when we consider each of them as zero, all of the states must be the same, *i.e.*, the difference between the states of each connecting pair of buses should be zero. Therefore, if the system is not observable with this set, then there exists at least a pair of buses which have different states with respect to each other (*i.e.*, nonzero difference). We find whether a set is observable using this contradiction. Therefore, we take the following constraint that all of the states cannot be the same:

$$\exists_{1 \leq j_1, j_2 \leq b, j_1 \neq j_2} \;\; \theta_{j_1} \neq \theta_{j_2}$$

Now, if there is no satisfiable solution to this model, we can conclude that the set of measurements can observe the system.

4.8.3.2 Selection of Lines and Admittance Perturbations

In the selection of the lines and corresponding changes in admittances, the main constraint is that the changes need to be done such that the OPF cost does not increase. Specifically, our aim is to keep the generation dispatch as it is according to the existing OPF, so that there is a minimum impact on the system operation due to the topology change.

The main constraint for OPF is that the total generation must be equal to the total expected load. Since we are not changing the demands at different buses, the required total generation remains the same. Now, the existing OPF solution can remain optimal after the admittance changes, if and only if the changed power flows still remain within associated transmission limits. Since all the power flow and consumption equations must hold, we consider them (*i.e.*, Equations (19) and (20)) as constraints:

$$\forall_{1 \le i \le l} \;\; P_i^L = \hat{d}_i(\theta_{f_i} - \theta_{e_i})$$

$$\forall_{1 \le j \le b} \;\; P_j^B = \sum_{i \in \mathbb{L}_{j,in}} P_i^L - \sum_{i \in \mathbb{L}_{j,out}} P_i^L$$

Here, $\hat{d}_i$ is the changed admittance of line i, such that $\hat{d}_i = d_i + \Delta d_i$, where Δd_i is the change made on line i. The admittance of a line can be changed only if D-FACTS devices are deployed. Therefore, considering that a line will be chosen for admittance change when the necessary D-FACTS facility is installed there, we define h_i for denoting whether the line is chosen for admittance change. Then, the following constraint holds on Δd_i:

$$\forall_{1 \le i \le l} \;\; \neg h_i \rightarrow (\Delta d_i = 0)$$

If there is a change in the line admittance, the change cannot be so small that it does not have an impact. If R is the ratio of the minimum change over the line admittance, then we can express this constraint as follows:

$$\forall_{1 \le i \le l} \;\; h_i \rightarrow (\Delta d_i \ge R \times d_i) \vee (\Delta d_i \le -R \times d_i)$$

Each line has a capacity for the power flow (*i.e.*, the maximum power that can flow through that line). Let $P^L_{i,max}$ be the line capacity. Therefore:

$$\forall_{1 \le i \le l} \;\; P^L_i \le P^L_{i,max}$$

The change of a line's admittance is useful to hinder adversaries from launching an attack, if one or more measurements associated with this line are taken. It is worth mentioning that there are four measurements associated with a line: two (forward and backward) line flow measurements and two bus consumption measurements at the end buses. Usually, it is beneficial to take a larger number of measurements associated with a line so that we can have greater impact if the admittance of the line is perturbed. However, in this model, we consider the minimum case as a constraint such that at least one of the measurements associated with the line needs to be taken:

$$\forall_{1 \le i \le l} \;\; h_i \rightarrow m_i \vee m_{l+i} \vee m_{f_i} \vee m_{e_i}$$

The solution to this model verifies whether a given choice of admittance changes on a selected set of lines satisfies the constraints. This model can even synthesize all (or a number of) potential sets of lines for admittance randomization with changed admittance values.

4.8.3.3 Impact of MTD on Attack Attributes

In order to launch a UFDI attack, power flows through various lines and power consumptions at different buses are impacted (*i.e.*, changed by ΔP^L_i and ΔP^B_j amounts, as shown in Section 4.5.4). The attacker needs to inject necessary false data to the measurements (*i.e.*, meter readings associated with those power flows and consumptions). However, the attacker only needs to inject necessary false data to measurement

i when it is taken. That is:

$$\forall_{1 \le i \le l}\ (\Delta P_i^L \neq 0) \rightarrow (t_i \rightarrow a_i) \wedge (t_{l+i} \rightarrow a_{l+i})$$

$$\forall_{1 \le j \le b}\ \ (\Delta P_j^B \neq 0) \rightarrow (t_{2l+j} \rightarrow a_{2l+j})$$

The randomization of the set of measurements, considered in state estimation, make t_i uncertain for the adversary.

If the admittance of a line is unknown to the attacker, he or she cannot determine the necessary changes that need to be applied on the power flow measurements of the line. The condition is formalized as:

$$\forall_{1 \le i \le l}\ \ (\Delta P_i^L \neq 0) \rightarrow ((t_i \vee t_{l+i}) \rightarrow g_i)$$

Moreover, when the admittance of a line is perturbed (*i.e.*, randomized), we also consider that the admittance is now unknown to the adversary, although the actual/base admittance of the line may be known to the adversary. Therefore, we take the following constraint:

$$\forall_{1 \le i \le l}\ \ h_i \rightarrow \neg g_i$$

4.8.4 An Example Case Study

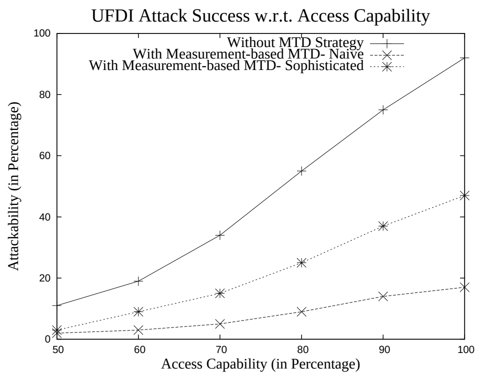
In this section, we present a case study demonstrating the performance of our MTD mechanism with respect to successful UFDI attacks on the IEEE 14-bus test system [90]. We use *attackability*, defined as the number of states which can be attacked (*i.e.*, infected by UFDI attacks) over the total number of states, as the performance metric.

4.8.4.1 Implementation

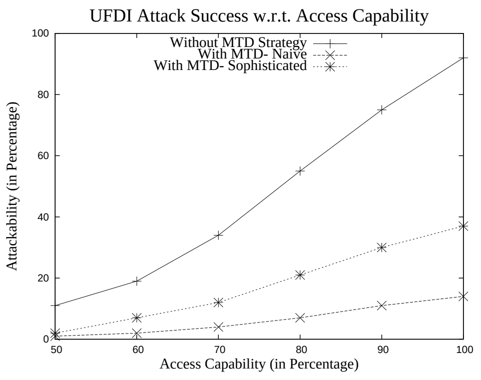
In order to implement a prototype of the MTD mechanism, we again use SMT. We encode the formal model of verifying whether a measurement set is observable. By solving this model using Z3, we generate a number of measurement sets to be used in state estimation. In our MTD mechanism, we randomly choose one measurement set among them following the uniform distribution. We also encode the formal model for the line admittance randomization. We first use the uniform distribution to select a subset of lines among those where the D-FACTS devices are deployed. Then, by executing this model in Z3, we determine whether the admittances of these lines can be changed while satisfying all the necessary constraints.

4.8.4.2 Case Analysis Results

We analyze the performance of our MTD mechanism by analyzing attackability under different scenarios considering access capabilities, knowledge limitations, and security measures. In this case study, we mainly consider two kinds of adversaries: (i) naive and (ii) sophisticated. The first type of adversary, as the name indicates, is unaware of the MTD scheme. He or she believes that a fixed set of measurements is used in state estimation. The second type of adversary knows that the MTD mechanism is running at the grid operator's side. As a result, in order to maximize the chances of a successful attack, he or she picks an attack vector that can cover as many potential sets of measurements as possible within resource and access limits. For both kinds of adversaries, we consider the same resource constraints. An adversary can attack 13-15 measurements at a time, while these measurements cannot be distributed



(a)



(b)

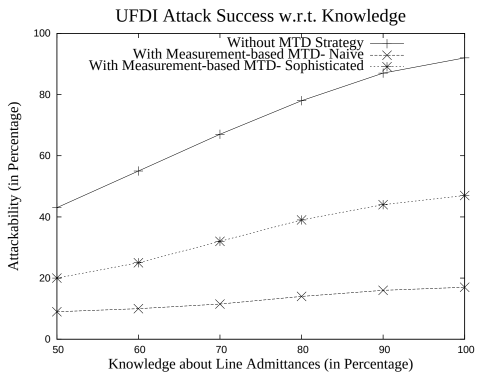
Figure 34: The probability of attack success in the cases of different access capabilities: (a) measurement-based MTD strategy and (b) measurement and line admittance-based MTD strategy.

over more than 7-8 buses of the system. We execute each experiment at least 30 times and take the arithmetic average of them.

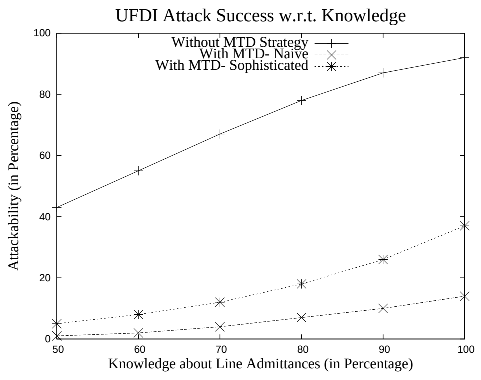
Figure 34(a) shows the attackability (*i.e.*, the number of states that can be attacked out of the total) in three different cases with respect to the application of our MTD mechanism and the adversary type. In the first case no MTD strategy is applied, while in the latter two cases the MTD is used but the type of adversary is different. In the second case the adversary is naive, while in the third case he or she is sophisticated. In this set of experiments, only the MTD strategy of randomizing the set of measurements used for state estimation is applied. Here, we consider the 14-bus test system. We take 100 sets of 30 measurements arbitrarily chosen from 37 taken measurements. We vary the accessibility of the adversary in the experiments from 50% to 100%. We observe that the attack success probability is always high when there is no MTD. In both of the cases of naive and sophisticated adversaries, the attackability reduces significantly. In the case of a sophisticated adversary, the attackability reduction is smaller compared to a naive adversary. This is because the sophisticated adversary uses all available resources to cover as many potential sets of measurements as possible, while the naive adversary only knows one particular set of measurements to be used in the state estimation process. The graphs in Figure 34(a) also show the impact of the adversary's access capability on attackability. The results are obvious: the lower the attacker's access capability, the better the performance of MTD strategy. In this case, the MTD mechanism is able to reduce attackability down to 5% when the access capability is no more than 60%.

Figure 34(b) shows the attackability under different attack capabilities of the adversary, as well. However, in this set of experiments, the MTD strategy of perturbing line admittances is applied along with the randomization of the set of measurements used for state estimation. We assume that D-FACTS devices are deployed on an arbitrary set of 5 lines, while only 2 lines are chosen among them for admittance perturbation at each time. According to the graphs in Figure 34(b), we can see that the MTD mechanism shows improved performance when we apply both of the MTD strategies. This performance improvement is more than 10% with respect to the measurement set randomization-based MTD alone.

We analyze the impact of the adversary's knowledge limitation on the performance of the MTD. Again, we consider the same three cases: without MTD, MTD with naive adversary, and MTD with sophisticated adversary. Figure 35(a) shows the impact of knowledge limitation when only measurement based-MTD strategy is applied. We observe that when the adversary has limited knowledge, MTD strategies perform better. However, the impact of knowledge limitation is significant in the case of the



(a)



(b)

Figure 35: The attackability in the cases of different levels of knowledge about line admittances: (a) measurement-based MTD strategy and (b) measurement and line admittance-based MTD strategy.

sophisticated adversary. Since a sophisticated adversary leverages knowledge about the system and the MTD strategy in order to increase attack success, when the knowledge is limited to less than 80%, the success drops significantly.

In the case of the MTD mechanism with both randomization of the measurement set and perturbation of line admittances, we see similar behavior (see Figure 35(b)). The only difference is that the impact of limited knowledge is higher in this case. That is, the performance of the MTD increases with the decrease of the adversary's knowledge, and this increase is more significant when both MTD strategies are applied.

4.9 Evaluation

In this section, we present the evaluation results showing the scalability of the our formal verification model, as well as that of the security architecture synthesis mechanism.

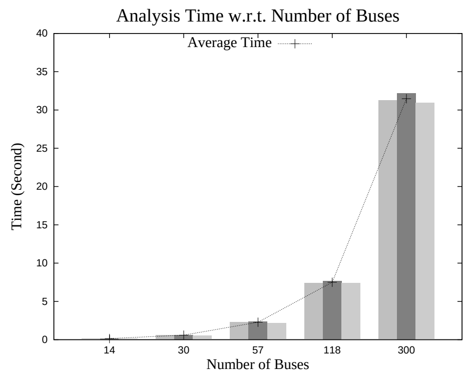
4.9.1 Methodology

We evaluate the scalability of our verification model by analyzing the time and memory requirements for executing the model in different problem sizes. Problem size depends mainly on the number of buses. We evaluate the scalability of our model based on different sizes of IEEE test systems: 14-bus, 30-bus, 57-bus, 118-bus, and 300-bus [90]. We also evaluate the impact of constraints on the scalability. Similarly, we evaluate the scalability of our security architecture synthesis mechanism. We run our experiments on an Intel Core i5 Processor with 8 GB memory. In this evaluation, we do not compare the time complexity of our formal model with that of the related work, especially with respect to [74] and [75], as neither of them provide results showing the complexity of their respective mechanisms.

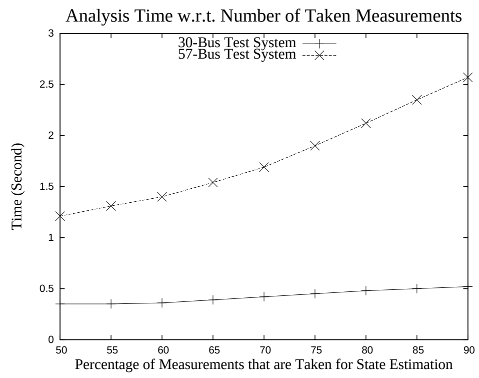
4.9.2 Time Complexity of Verification Model

4.9.2.1 Impact of the Problem Size

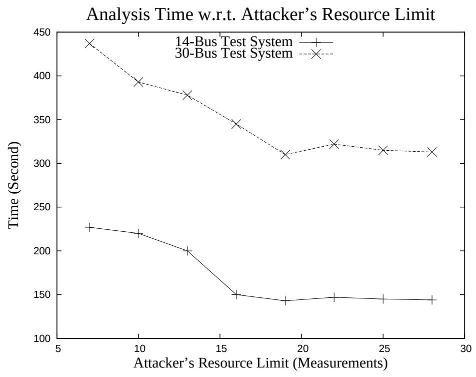
Figure 36(a) shows the execution time of our UFDI attack verification model with respect to the problem size. We vary the problem size by considering different IEEE bus test systems. We execute three experiments taking different states to be attacked for each test case. The execution time of each case is shown in Figure 36(a) using a bar chart. A graph is also drawn using the average execution time for each bus system. We observe that with respect to the bus size the increase in the execution time lies between linear and quadratic orders. For a specific bus size, we also observe that the execution time differs with a different choice of states to be attacked. It is worth mentioning that, although the general problem seems to have a quadratic growth considering the



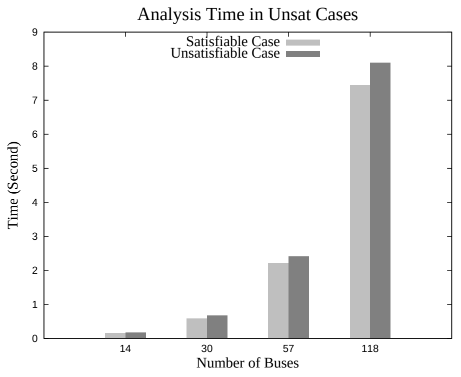
(a)



(b)



(c)



(d)

Figure 36: The verification model execution time in different experiments: (a) the execution time with respect to the number of buses, (b) the execution time with respect to the number of recorded measurements, (c) the execution time with respect to the attacker's resource limit, and (d) the execution time in unsatisfiable cases with respect to the number of buses.

number of buses and the connectivity between them, we observe smaller execution time. This is because the complexity depends not only on the number of buses, but also on the number of lines, measurements, and attack attributes. An important feature of power grid networks is that the average degree of a node (or bus) is roughly 3, regardless of the number of buses in the system [98]. This feature can explain why the complexity is not strictly quadratic.

We also analyze the impact of the number of taken measurements (represented as the percentage of the total potential measurements) on the model execution time. Figure 36(b) presents the evaluation results for the 30 and 57-bus test systems. The results show that the execution time increases linearly with the increase in the number of taken measurements. We also observe similar results for the other test systems. When the number of recorded measurements increases, the number of measurements to be considered for false data injection also increases, which results in a longer verification time.

4.9.2.2 Impact of the Constraints

The verification of potential UFDI attacks depends on the given constraints, especially the attacker's access capability and resource limit. We evaluate the impact of the attacker's resource limit on the analysis time. We consider IEEE 14- and 30-bus systems. The analysis result is shown in Figure 36(c). We observe that the analysis time decreases with the increase in the attacker's resources (*i.e.*, the resource constraint is relaxed) because the potential of UFDI attacks increases with the increase of the attacker's resources. However, this increase does not help in UFDI attacks after some point (*e.g.*, when the attacker's resource limitation is almost 20 measurements, as shown in Figure 36(c)). This is because the attacker already has resources which are sufficiently large to launch a UFDI attack to one or more states.

4.9.2.3 Performance in Unsatisfiable Cases

When constraints are tight (*e.g.*, when the attacker can attack a very limited number of measurements), there can be no satisfiable solution. In such cases, the SMT solver often takes a longer amount of time to give the unsatisfiable (*unsat*) results compared to the execution time in satisfiable cases. In unsatisfiable cases, the SMT solver needs to explore the entire solution space to conclude that there is no solution based on the given constraints. Figure 36(d) shows a comparison between the execution times for satisfiable and unsatisfiable cases, with respect to different bus systems. Since we consider different constraints and specific attack goals (corresponding to the attack attributes) for an attacker, the potentiality of an attack vector is already limited. Therefore, in our experiments we observe smaller execution time differences between satisfiable and unsatisfiable cases.

4.9.3 Time Complexity of Impact Analysis

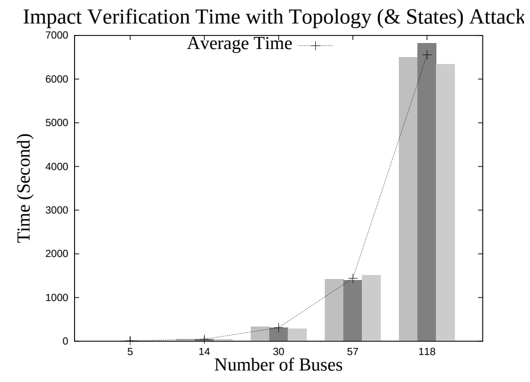
As we are considering real values, there is usually an extremely large number of stealthy attack vectors possible in an attack scenario. We observe that finding the impact on OPF, considering such a large number of attack vectors, becomes exceedingly time consuming when the number of buses becomes large. In order to keep the computation cost tractable, we enhance our formal framework with the following ideas:

- Although there can be a larger number of attack vectors, many vectors are close to each other and the difference between them is insignificant with respect to changes in loads. Therefore, it is enough to consider one of these similar attack vectors to see the impact for each of them. According to this idea, the number of attack vectors considered for finding the impact becomes limited, which leads to a reduced execution time. In our experiments, we take the precision of 2 digits to consider two attack vectors as the same one.

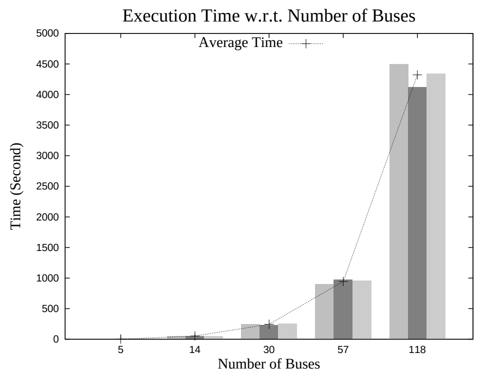
- The typical OPF model, as we present in Section 4.6.2, takes a very long time for 57, 118, or larger bus systems, which makes the impact verification often infeasible. In order to reduce the OPF model execution time, we adopt the idea of using generation-to-load distribution factors for calculating the line power flows in the OPF model [69, 99]. The use of shift factors alone cannot replace the voltage phase angle based line power flow calculation as in Equation (46), because it is conditioned with the existence of the line in the topology. Therefore, we use the line outage or line closure distribution factors (LODF/LCDF) to work with any line exclusion or inclusion attack [100]. However, since these LODF/LCDF are usually calculated for single line breakage or closure, in our evaluations, we only consider single line inclusion or exclusion-based topology attacks.

4.9.3.1 Impact of the Problem Size

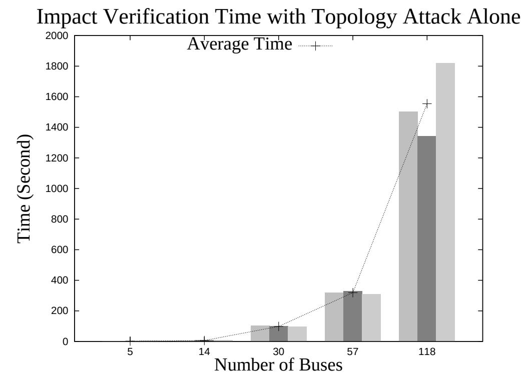
Figures 37(a), 37(b), and 37(c) show the execution time of our impact analysis formal model in different scenarios, considering different kinds of stealthy attacks. The graphs show the impact of the problem size on the execution time. We vary the problem size by considering different IEEE bus test systems. At each problem size, we perform three experiments taking different random scenarios, especially in terms of the attacker's resource limitation. We consider a 1-2% increase in the generation cost. The execution time of each of these experiments is shown using a bar chart. A graph is also drawn using the average execution time for each bus system. We see that, with respect to the bus size, the increase of the execution time follows exponential order.



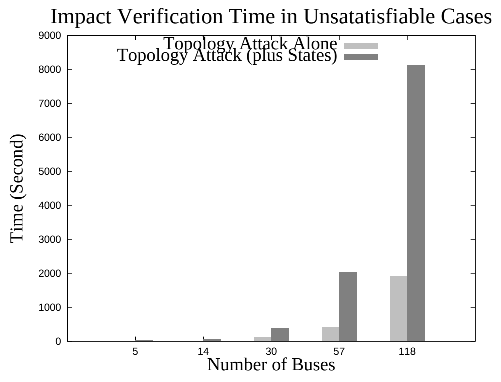
(a)



(b)



(c)



(d)

Figure 37: The execution time of Impact verification on OPF with respect to the number of buses in (a) UFDI attacks including topology attacks, (b) UFDI attacks without topology attacks, (c) topology attacks without infecting states, and (d) unsatisfiable cases.

The execution time of an SMT model depends on the number of variables and the complexity of the theories applied in the model. The number of variables increases with the problem size, particularly in this model due to the number of generators and lines. However, we observe that the execution time is much higher in the scenario when infection to the topology and states (Figure 37(a)) is considered together than the cases when either one of them performed (Figures 37(b), and 37(c)). This is because it is possible to launch multiple attacks on one or more states with respect to a single line inclusion or exclusion attack, which increases the attack space (*i.e.*, search space), significantly. It is worth mentioning that, due to this larger attack space, the second scenario can make larger (and various) impact on OPF compared to the first.

4.9.3.2 Performance in Unsatisfiable Cases

Figure 37(d) shows the execution time in the unsatisfiable cases. If we compare the graphs in this figure with those in Figure 37(c) and Figure 37(a), we can see that the execution time in unsatisfiable cases is higher than the time in the satisfiable cases. This is because the SMT solver requires verification of all the potential attack vectors in order to conclude that there is no attack that can create the desired impact.

4.9.4 Time Complexity of Synthesis Mechanism

4.9.4.1 Impact of the Problem Size

The execution time of our security architecture synthesis mechanism with respect to different test bus systems is shown in Figure 38(a). We consider two scenarios in our experiments: (i) 90% of the measurements are recorded for state estimation, and (ii) all of the measurements are recorded for state estimation. We can see in the figure that the increase in the execution time is quadratic in order. However, this execution time is significantly longer than that of the UFDI attack verification model that we see in Figure 36(a). This is because, in order to synthesize the security architecture, the verification model may need to be executed many times till a security architecture is found.

We again analyze the impact of the number of taken measurements, specified as the percentage of the total potential measurements, on the time of security architecture

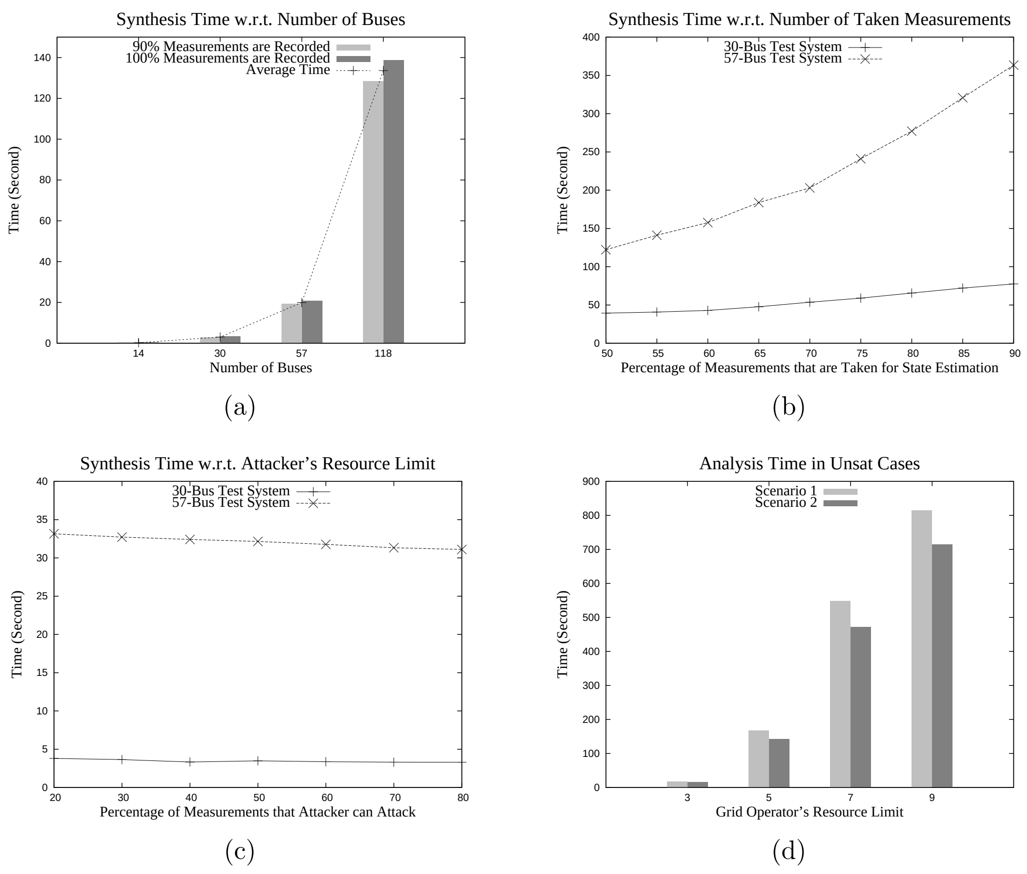


Figure 38: The execution time of the security architecture synthesis mechanism in different experiments: (a) the execution time with respect to the number of buses, (b) the execution time with respect to the number of taken measurements, (c) the execution time with respect to the attacker's resource limit, and (d) the execution time in unsatisfiable cases with respect to the number of buses.

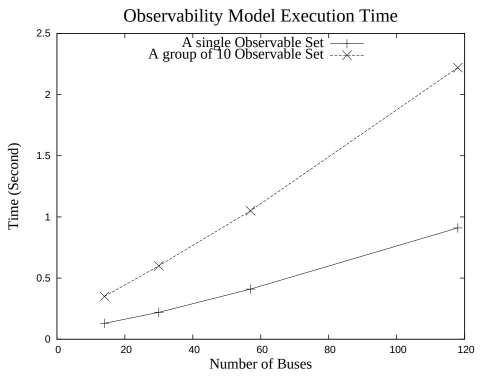
synthesis. Figure 38(b) shows the evaluation results corresponding to the 30 and 57-bus test systems. We observe that, with the increase in the number of taken measurements, the execution time increases linearly. Since the selection of security architecture is based on the buses, any increase in taken measurements does not increase the selection time. However, we know that verification time increases with the increase in taken measurements (as shown in Figure 36(b)). As a result, the time for the security architecture synthesis increases.

4.9.4.2 Impact of the Constraints

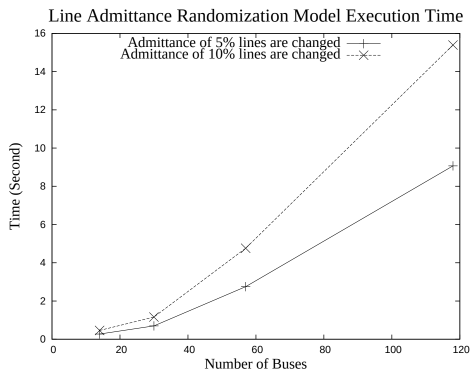
A security architecture depends on the given constraints (*e.g.*, the attacker's resource limit). We analyze the impact of this resource limit, represented as the percentage of the total measurements, on the security architecture synthesis time. The analysis result is shown in Figure 38(c). We observe that the synthesis time decreases slowly with the increase in the attacker's resource limit value. This is because the increase of the attacker's resources decreases the time to find that a candidate security architecture is unsuccessful, which is actually the satisfiability of the UFDI attack model. As a result, the synthesis time decreases.

4.9.4.3 Performance in Unsatisfied Cases

When the grid operator's resources can be so limited that there is no security solution. The execution time in such an unsatisfiable case is usually high because the synthesis mechanism requires verification of all the potential security architectures to conclude that there is no security solution based on the given constraints. Figure 38(d) shows the execution times of the synthesis mechanism in unsatisfiable cases. In this analysis, we take the IEEE 30-bus test system and vary the resource limit values in two different scenarios. In the first scenario a security plan needs a minimum number of 10 buses, while in the second the number is 12. No security plan is possible with less than this many buses. In the figure, we see that the closer the resource limit is to the minimum number of necessary buses, the higher the execution time is to discover that there is no solution. When the limit is too close to the minimum requirement, the unsatisfiability comes at the very end of the search, and thereby the early rejection



(a)



(b)

Figure 39: These two graphs shows the model execution time with respect to the number of buses: (a) the execution time of the observability model and (b) the execution time of the line admittance randomization model.

of a potential search mostly does not take place.

4.9.5 Time Complexity of MTD Strategy Selection Models

We run experiments to evaluate the efficiency of our developed models for MTD strategy selections and we find that both of these models are time-efficient. Figure 39(a) shows the execution time of our observability verification model with respect to the problem size. We observe that the model execution time lies within a few seconds in the case of 100 buses. The execution time of the line admittance perturbation model is presented in Figure 39(b), which shows that the model takes approximately 15 seconds for 100 buses.

4.9.6 Memory Complexity

The memory required by the SMT solver [17] for executing our verification model (refer to Section 4.5) and candidate security architecture selection model (refer to Section 4.7) is evaluated in different IEEE bus test systems. The memory requirement

Table 25: The required memory space (in MB)

# of Buses	Verification Model	Candidate Selection Model
14	1.32	0.05
30	2.60	0.10
57	4.56	0.16
118	9.69	0.31

for an execution of the SMT model depends mainly on the number of variables defined in the model and the number of intermediate variables generated by the solver to implement the satisfiability modulo theories used in the model. The memory analysis results are presented in Table 25, which shows that memory usage of our models increases almost linearly with the number of buses.

We also evaluate the memory or space required by the SMT solver [17] for executing our MTD strategy selection models as presented in Section 4.8. Here, we consider the 14-bus test system. The memory requirement for an execution of the SMT model depends mainly on the number of variables defined in the model and the number of intermediate variables generated by the solver to implement the satisfiability modulo theories used in the model. Our analysis results show that the observability model takes less than 4 MB of memory, while the line admittance randomization model needs less than 22 MB of memory.

4.10 Conclusion

Securing state estimation against cyber-attacks is of paramount importance to maintain the integrity of the power grid. We propose an SMT-based formal framework to systematically investigate potential security threats, particularly the feasibility of stealthy cyber-attacks, on state estimation. We also extend the framework to show the impact of stealthy attacks on the economic operation of a power system by considering the OPF module. Thus, our formal framework is capable of modeling stealthy attacks considering an adversary's constraints, analyzing the attack's feasibility, and quantifying consequences in terms of increases in overall generation costs. The framework also allows an operator to capture interdependency among attack attributes to synthesize a security architecture, which secures a set of buses for immunity against UFDI attacks. In addition, we propose an agility-based security solution by introducing uncertainty in the system following the MTD idea. In this mechanism, we apply randomization on the power grid system properties, particularly the set of measurements that is used in state estimation and the admittances of a set of lines. We present formal models to find the observable sets of measurements and the lines for randomizing admittances. The scalability of the models is evaluated with experiments and case-studies on different IEEE bus test systems. Our results show that our frameworks can solve problems up to a few hundreds of buses, although with different levels of efficiency. Our stealthy attack verification model executes in a few seconds for a grid system with 100 buses, while the time for the attack impact verification model takes several hours for a similarly sized system. The formal models provide a basis for the development of cyber-security tools for modern power grids. In the future, we would like to investigate stealthy attacks and their impact on the energy market's security.

CHAPTER 5: SUMMARY AND FUTURE WORK

Throughout this dissertation, we discuss the security of different major components of a smart grid and develop formal models for its security analysis. The research has two primary goals: (i) to develop analytics to proactively identify potential security threats, and (ii) to devise mechanisms for hardening the security of the grid against those threats. We achieve these two primary goals in three research thrusts that we have already discussed in the previous chapters in detail. In this chapter, we summarize those works. Next, we present an outline of potential research problems for future work, some of which are potential extensions of this dissertation, while the rest are new problems with a potential for applying the technical approaches similar to this dissertation.

5.1 Overview of Contributions, Technical Approaches, and Evaluation Results

In this section, we summarize our contributions and corresponding technical approaches. We also summarize the evaluation results.

5.1.1 Security Verification for AMI and SCADA

In the first chapter, we present a security verification framework for smart grids. Due to the heterogeneity in physical and network device configurations and emerging cyber-physical security threats, an automated analysis of smart grid configuration is an important as well as challenging problem. Considering this challenge, we first build a formal model for verifying threats on AMI. We analyze and identify different invariant and user-driven constraints, violations of which can cause various threats. Then, based on these constraints, we develop SmartAnalyzer, an SMT solver based tool. Under any constraint violation, the tool generates a threat report that includes the reason for the violation and the possible remediation plan.

Second, we develop an identical formal model for verifying the security threats for SCADA systems, considering the aspect of control operations. Next, we evaluate the scalability of SmartAnalyzer in different test configurations. As we apply the property level abstractions to model the configurations, we achieve high scalability and we observe that the constraint verification time lies within 10 seconds for a network of 1 million collectors.

5.1.2 Automated Synthesis of Resiliency Architecture

According to our second thrust, we first present an automated framework for synthesizing correct and cost-effective network isolation-based resiliency configurations for cyber systems in smart grids. The framework formalizes the resiliency architecture synthesis problem as the conjunction of all the security requirements (in terms of isolation) and business constraints (in terms of usability and deployment cost). This formal model is encoded using SMT and its solution provides a resiliency architecture that includes necessary network isolation measures along with placements of security devices. Although this framework produces a solution that satisfies the resiliency requirements, it is interesting and useful to get an optimal solution, considering the given requirements as the base necessity. However, the formal model takes a substantially long time to execute when the isolation requirement is close to the optimal. Thus, we also develop a hypothesis testing-based mechanism that follows an iterative and feedback process to find an optimal/improved resilient architecture.

Second, we present a formal framework that automatically provides a redundancy-based resilient architecture for AMI which includes safe, secure, and robust configurations. We model various constraints that are crucial for safe and robust data delivery in AMI systems. Although we model the robustness considering a single node or link failure, this design is extendable for further robustness requirements.

We also evaluate both of these frameworks in different synthetic test networks. We find that isolation-based resiliency architecture synthesis generates a satisfiable security design in $10-20$ minutes for a problem with several hundreds of hosts. In the case of redundancy-based resiliency architecture synthesis, the model takes few minutes for a problem with thousands of smart meters.

5.1.3 Threat Verification and Security Hardening for Power System State Estimation

We present security analytics for the power system state estimation. Correct state estimation is critical for the stability and reliability of a power grid. An adversary can attack state estimation by intelligently injecting false measurements. Thus, it is important for the grid operators to understand the potential threats based on security requirements and attack attributes. Therefore, we propose a formal framework for the power system state estimation security verification, which is comprehensive with respect to different constraints and requirements. The solution to this formal model gives potential threat vectors showing the attack feasibility with respect to the attacker's capabilities. We extend this formal model to show the impact of stealthy attacks on the economic operation of a power system by considering the OPF module.

We present a mechanism utilizing this framework to capture the relation between attack attributes and security properties to synthesize a pan which secures a set of buses to protect the grid against stealthy attacks. We also propose an interesting security solution by introducing agility in the system, in which we randomize the set of measurements considered in state estimation and the admittances of a set of lines. We develop formal models to select safe randomizing measures. We use SMT to formalize these models, and demonstrate our solutions with respect to different case studies. We evaluate our models in terms of scalability by performing a number of experiments using different IEEE test systems. Our evaluation results show that our model can efficiently solve problems with hundreds of buses. In the case of the IEEE 118-bus test system, our threat verification model executes in 20 seconds.

5.2 Future Research

From our deep study and contributions in developing automated formal analytics for smart grid security and resiliency, we have found various potentially challenging problems for future research, including the following:

- Automated synthesis of resilient architectures for cyber and cyber-physical systems: We know that resiliency is the ability of the system to keep the potential damage limited while the attack intensity increases. In this dissertation, we address resiliency architecture synthesis based on some pre-attack and during-attack resiliency mechanisms. In our future work, we would like to extend our work in order to include other forms of pre-attack and during-attack resiliency techniques, such as diversity and moving targets. We would also like to explore post-attack resiliency (*i.e.*, by using recoverablility techniques), and develop a comprehensive formal model that will consider all kinds of resiliency patterns and will automatically provide an optimal resiliency architecture within the business constraints.

- Analytics for threat identification and mitigation in smart grid energy markets: The restructuring of the power industry, led by Federal Energy Regulatory Commission (FERC)-mandated deregulations, has established a number of energy markets. The participants in these markets engaged in short-term and long-term contracts according to different trading information, *e.g.*, supply and demand, capacities, and offered bids. Adversaries can launch attacks on energy markets directly by manipulating the trading information or indirectly by attacking measurements or topology statuses to disrupt the energy market, leading to an unfair benefit to dishonest participants as well as uneconomical operations of the grid. Therefore, we would like to develop formal analytics to identify potential threats on energy markets, analyzing their impacts, and synthesize mitigation plans.

- Agility for the sustainability of smart grids: In this particular work, we would like to extend our idea of applying moving target defense mechanisms to increase the agility of smart grids, including AMI, SCADA, and Vehicle-to-Grid

(V2G) systems in order to increase the unpredictability of the system properties, thereby reducing the attack window for adversaries.

- Security analytics considering PMU-based SCADA measurements: In our threat analytics framework in this dissertation, we consider SCADA measurements (meter measurements as well as circuit breaker and switch statuses) only. In this particular research, we would like to explore other types of measurements, such as PMU and AMI data, and study the characteristics of the stealthy attacks and the corresponding idea of security hardening.

- Electric power supply chain management: There is an electric power supply chain between different generating stations, transmission and distribution substations, and power consumers. The different participants/components within this chain are interconnected with one another. The security of each participant is crucial for the security of the supply chain, while the chain should be resilient in contingencies. Although significant research has been done on the power grid's robustness in contingencies, it is still important to formally model the supply chain by considering each participant's characteristics, interactions between the participants, and contingencies in a single framework in order to understand the interdependency between the components and explore cost-effective resiliency plans.

REFERENCES

[1] Allen J. Wood and Bruce F. Wollenberg. *Power Generation, Operation, and Control, 2nd Edition*. Wiley, 1996.

[2] What is critical infrastructure. U.S. Department of Homeland Security. http://www.dhs.gov/what-critical-infrastructure.

[3] Bernie Lawrence, Martin Hancock, and Ginni Stieva. How unreliable power affects the business value of a hospital. Schneider Electric, August 2010. http://www.schneider-electric.com/documents/support/white-papers/wp_healthcare-how-unreliable-power.pdf.

[4] National institute of standards and technology. U.S. Department of Commerce. http://www.nist.gov/, http://www.nist.gov/publication-portal.

[5] North american electric reliability corporation. http://www.nerc.com, http://www.nerc.com/pa/Stand/Pages/default.aspx.

[6] Federal energy regulatory commission. http://www.ferc.gov/, http://www.ferc.gov/industries/electric/indus-act/reliability/standards.asp.

[7] The smart grid: An introduction. U.S. Department of Energy, 2008. http://energy.gov/oe/downloads/smart-grid-introduction-0.

[8] Silver Spring. How the smart grid makes restoration faster and easier for utilities. http://www.silverspringnet.com/outage/pdfs/SilverSpring-Whitepaper-Outage.pdf.

[9] J.S. Vardakas, N. Zorba, and C.V. Verikoukis. A survey on demand response programs in smart grids: Pricing methods and optimization algorithms. *IEEE Communications Surveys Tutorials*, pages 1–1, 2014.

[10] R. Alimi, Y. Wang, and Y. R. Yang. Shadow configuration as a network management primitive. In *ACM SIGCOMM Conference on Data communication*, pages 111–122, 2008.

[11] Energy Research Council. Best practices: Demand response. http://energyresearchcouncil.com/best-practices-demand-response.html.

[12] Nistir 7628: Guidelines for smart grid cyber security. Smart Grid Interoperability Panel- Cyber Security Working Group, August 2010. http://www.nist.gov/smartgrid/upload/nistir-7628_total.pdf.

[13] Guide to industrial control systems (ics) security. NIST Special Publication 800-82 (Revision 1), May 2013. http://dx.doi.org/10.6028/NIST.SP.800-82r1.

[14] Recommended security controls for federal information systems and organizations. NIST Special Publication 800-53 (Revision 4), April 2013. http://dx.doi.org/10.6028/NIST.SP.800-53r4.

[15] North-American Electric Reliability Corporation. Critical infrastructure protection (cip) standards. http://www.nerc.com/pa/Stand/Pages/CIPStandards.aspx.

[16] North American Electric Reliability Corporation. Comments of the north american electric reliability corporation in response to nist smart grid cyber security strategy and requirements (draft nistir 7628), 2009.

[17] Z3: Theorem prover. Microsoft Research, 2013. http://research.microsoft.com/en-us/um/redmond/projects/z3/.

[18] Ami smart grid testbed at unc charlotte. http://www.cyberdna.uncc.edu/events.php.

[19] Matthew W. Moskewicz et al. Chaff: Engineering an efficient sat solver. In *Annual ACM IEEE Design Automation Conference*, 2001.

[20] R. Nieuwenhuis and A. Oliveras. On sat modulo theories and optimization problems. In *Theory and Applications of Satisfiability Testing (SAT)*, volume 4121 of *Lecture Notes in Computer Science*, pages 156–169. Springer, 2006.

[21] E. Al-Shaer, W. Marrero, A. El-Atawy, and K. Elbadawi. Network configuration in a box: Towards end-to-end verification of network reachability and security. In *IEEE International Conference on Network Protocols (ICNP)*, pages 107–116, NY, USA, 2009.

[22] P. Bera, S. Ghosh, and P. Dasgupta. Policy based security analysis in enterprise networks: A formal approach. *IEEE Transactions on Network and Service Management*, 7(4):231–243, 2010.

[23] Leonardo de Moura and Nikolaj Bjørner. Satisfiability modulo theories: An appetizer. In *Brazilian Symposium on Formal Methods*, 2009.

[24] N. Bjørner and L. Moura. Z3^{10}: Applications, enablers, challenges and directions. In *Int. Workshop on Constraints in Formal Verification*, 2009.

[25] A. Greenberg. Congress alarmed at cyber-vulnerability of power grid, 2008. http://www.forbes.com/2008/05/22/cyberwarbreach-government-tech-security_cx_ag_0521cyber.html.

[26] P. McDaniel and S. McLaughlin. Security and privacy challenges in the smart grid. *IEEE Security Privacy*, 7(3):75–77, 2009.

[27] S. McLaughlin, D. Podkuiko, S. Miadzvezhanka, A. Delozier, and P. McDaniel. Multi-vendor penetration testing in the advanced metering infrastructure. In *26th ACSAC*, pages 107–116, 2010.

[28] Punya Prakash. Data concentrators: The core of energy and data management. http://www.ti.com/lit/wp/spry248/spry248.pdf?DCMP=induvid1&HQS=ep-pro-sit-induvid1-toolsinsider-20140605-mc-en, 2013. Texas Instruments.

[29] The EEI and AEIC Meter Committees. Smart meters and smart meter systems: A metering industry perspective. http://www.eei.org/issuesandpolicy/grid-enhancements/Documents/smartmeters.pdf.

[30] Lontalk protocol specification. http://www.enerlon.com/JobAids/Lontalk\%20Protocol\%20Spec.pdf.

[31] E. Al-Shaer and H. Hamed. Discovery of policy anomalies in distributed firewalls. In *23rd IEEE International Conference on Computer Communications (INFOCOM)*, pages 2605–2616, 2004.

[32] X. Ou, S. Govindavajhala, and A. Appel. Mulval: A logic-based network security analyzer. In *14th USENIX Security Symposium*, pages 113–128, 2005.

[33] Steven Noel and Sushil Jajodia. Attack graphs for sensor placement, alert prioritization, and attack response. In *Cyberspace Research Workshop of Air Force Cyberspace Symposium*, Shreveport, Louisiana, 2007.

[34] R. Dewri, N. Poolsappsi, I. Ray, and D. Whitley. Optimal security hardening using multi-objective optimization on attack tree models of networks. In *14th ACM conference on Computer and Communications Security (CCS)*, pages 204–213, 2007.

[35] I. Kotenko and M. Stepashkin. Attack graph based evaluation of network security. In *10th IFIP International Conference on Communications and Multimedia Security*, pages 216–227, 2006.

[36] J. Homer and X. Ou. Sat-solving approaches to context-aware enterprise network security management. 27(3):315–322, 2009.

[37] ABB Group. Security in the smart grid, 2009. http://www02.abb.com/db/db0003/db002698.nsf/0/832c29e54746dd0fc12576400024ef16/\$file/paper_Security+in+the+Smart+Grid+(Sept+09)_docnum.pdf.

[38] Ami system security requirements. AMI-SEC Task Force, 2008. http://osgug.ucaiug.org/utilisec/amisec/.

[39] Honeywell. The communications requirements of electric utilities. http://energy.gov/gc/downloads/nbp-rfi\-communications\-requirements-honeywell\-responses-request-information-rfi.

[40] S. McLaughlin, D. Podkuiko, and P. McDaniel. Energy theft in the advanced metering infrastructure. In *4th International Workshop on Critical Information Infrastructure Security*, pages 176–187, 2009.

[41] Y. Wang, D. Ruan, J. Xu, M. Wen, and L. Deng. Computational intelligence algorithms analysis for smart grid cyber security. *Advances in Swarm Intelligence*, 6146:77–84, 2010.

[42] Z. Anwar, R. Shankesi, and R. H. Campbell. Automatic security assessment of critical cyber-infrastructures. In *IEEE/IFIP DSN*, pages 366–375, 2008.

[43] Z. Anwar and R. H. Campbell. Automated assessment of critical infrastructures for compliance to cip best practices. In *2nd IFIP WG 11.10 International Conference on Critical Infrastructure Protection*, Arlington, Virginia, 2008.

[44] Yao Liu, Peng Ning, and Michael K. Reiter. False data injection attacks against state estimation in electric power grids. In *ACM Conference on Computer and Communications Security (CCS)*, pages 21–32, Nov 2009.

[45] L. de Moura and N. Bjrner. Z3: An efficient smt solver. In *Conf. on Tools and Algo. for the Construction and Analysis of Systems*, 2008.

[46] IBM Software. Managing big data for smart grids and smart meters. `http://www.smartgridnews.com/artman/publish/Business_Strategy/Managing-big-data-for-smart-grids-and-smart-meters-5248.html`.

[47] HP Technical. Using the hp vertica analytics platform to manage massive volumes of smart meter data. `http://www.vertica.com/wp-content/uploads/2014/05/SmartMetering_WP.pdf`.

[48] The yices smt solver. CSL, SRI International, 2013. `http://yices.csl.sri.com/`.

[49] A. Abur and A. G. Exposito. *Power System State Estimation : Theory and Implementation*. CRC Press, New York, NY, 2004.

[50] H. Krawczyk, M. Bellare, and R. Canetti. Hmac: Keyed-hashing for message authentication, 1997.

[51] J. Jonsson and B. Kaliski. Public-key cryptography standards (pkcs) #1: Rsa cryptography specifications version 2.1, 2003.

[52] William Stallings. The advanced encryption standard. *Cryptologia*, 26(3):165–188, July 2002.

[53] H. Gharavi and Bin Hu. Multigate communication network for smart grid. *Proceedings of the IEEE*, 99(6):1028 –1045, Jun 2011.

[54] H. Hamed, E. Al-Shaer, and W. Marrero. Modeling and verification of ipsec and vpn security policies. In *IEEE ICNP*, 2005.

[55] L. Yuan et al. Fireman: A frameworkkit for firewall modeling and analysis. In *IEEE Symposium on Security and Privacy*, 2006.

[56] Charles C. Zhang, Marianne Winslett, and Carl A. Gunter. On the safety and efficiency of firewall policy deployment. In *IEEE Symposium on Security and Privacy*, 2007.

[57] O. Sheyner et al. Automated generation and analysis of attack graphs. In *IEEE Symposium on Security and Privacy*, 2002.

[58] S. Noel, S. Jajodia, B. O'Berry, and M. Jacobs. Efficient minimum-cost network hardening via exploit dependency graphs. In *19th Annual Computer Security Applications Conference*, pages 86–95, Dec 2003.

[59] J. Homer and X. Ou. Sat-solving approaches to context-aware enterprise network security management. In *IEEE JSAC Special Issue on Network Infrastructure Configuration*, 2011.

[60] M.A. Rahman and E. Al-Shaer. A formal approach for network security management based on qualitative risk analysis. In *IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 244–251, May 2013.

[61] Anoop Singhal and Xinming Ou. Techniques for enterprise network security metrics. In *5th Annual Workshop on Cyber Security and Information Intelligence Research: Cyber Security and Information Intelligence Challenges and Strategies*, pages 25:1–25:4, 2009.

[62] M.S. Ahmed et al. Objective risk evaluation for automated security management. In *JNSM*, volume 19, 2011.

[63] M. Amezziane, E. Al-Shaer, and M. Ali. On stochastic risk ordering of network services for proactive security management. In *IEEE NOMS*, 2012.

[64] S. Narain, G Levin, S Malik, and V. Kaul. Declarative infrastructure configuration synthesis and debugging. *Journal of Network and Systems Management*, 16(3):235–258, 2008.

[65] B. Zhang et al. Specifications of a high-level conflict-free firewall policy language for multi-domain networks. In *ACM 12th ACM symposium on Access control models and technologies (SACMAT)*, 2007.

[66] B. Zhang and E. Al-Shaer. Synthesizing distributed firewall configurations considering risk, usability and cost constraints. In *International Conference on Network and Service Management (CNSM)*, 2011.

[67] A. Ipakchi and F. Albuyeh. Grid of the future. In *IEEE Power and Energy Magazine*, pages 52–62, March 2009.

[68] D. Kundur, X. Feng, S. Liu, T. Zourntos, and K. Butler-Purry. Towards a framework for cyber attack impact analysis of the electric smart grid. In *IEEE International Conference on Smart Grid Communications*, pages 244 – 249, Oct 2010.

[69] A. Monticelli. *State estimation in electric power systems: a generalized approach.* Kluwer Academic Publishers, Norwell, MA, 1999.

[70] A. Teixeira, S. Amin, H. Sandberg, K. Johansson, and S. Sastry. Cyber security analysis of state estimators in electric power systems. In *IEEE Conference on Decision and Control*, pages 5991–5998, Dec 2010.

[71] Kin Cheong Sou, H. Sandberg, and K.H. Johansson. Electric power network security analysis via minimum cut relaxation. In *50th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC)*, pages 4054–4059, 2011.

[72] O. Vukovic, Kin Cheong Sou, G. Dan, and H. Sandberg. Network-layer protection schemes against stealth attacks on state estimators in power systems. In *IEEE International Conference on Smart Grid Communications*, Oct 2011.

[73] Yao Liu, Peng Ning, and Michael K. Reiter. False data injection attacks against state estimation in electric power grids. *ACM Transactions on Information and System Security*, 14(1):13:1–13:33, Jun 2011.

[74] R. Bobba, K. Rogers, Q. Wang, H. Khurana, K. Nahrstedt, and T. Overbye. Detecting false data injection attacks on dc state estimation. In *IEEE Workshop on Secure Control Systems, CPS Week*, Apr 2010.

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

[76] O. Kosut, L. Jia, R. J. Thomas, and L. Tong. Limiting false data attacks on power system state estimation. In *IEEE Conference on Information Sciences and Systems (CISS)*, Princeton, NJ, Mar.

[77] O. Kosut, L. Jia, R. J. Thomas, and L. Tong. On malicious data attacks on power system state estimation. In *International Universities Power Engineering Conference (UPEC)*, Aug 2010.

[78] Yi Huang, Husheng Li, K. A. Campbell, and Zhu Han. Defending false data injection attack on smart grid network using adaptive cusum test. In *Annual Conference on Information Sciences and Systems (CISS)*, pages 1 – 6, Mar 2011.

[79] Kin Cheong Sou, H. Sandberg, and K. H. Johansson. On the exact solution to a smart grid cyber-security analysis problem. *IEEE Transactions on Smart Grid*, 4(2):856–865, 2013.

[80] M. Ashfaqur Rahman and H. Mohsenian-Rad. False data injection attacks with incomplete information against smart power grids. In *IEEE Conference on Global Communications (GLOBECOM)*, Dec 2012.

[81] Jinsub Kim and Lang Tong. On topology attack of a smart grid: Undetectable attacks and countermeasures. *IEEE Journal on Selected Areas in Communications*, 31(7):1294–1305, Jul 2013.

[82] M. Esmalifalak, Ge Shi, Zhu Han, and Lingyang Song. Bad data injection attack and defense in electricity market using game theory study. *IEEE Transactions on Smart Grid*, 4(1):160–169, Mar 2013.

[83] S. Antonatos, P. Akritidis, E. Markatos, and K. Anagnostakis. Defending against hitlist worms using network address space randomization. *International Journal of Computer and Telecommunications Networking*, 51(12):3471–3490, August 2007.

[84] J. H. Jafarian, E. Al-Shaer, and Q. Duan. Openflow random host mutation: Transparent moving target defense using software defined networking. In *Workshop on Hot topics in Software Defined Networks (HotSDN)*, August 2012.

[85] Qi Duan, Ehab Al-Shaer, and Jafar Haadi Jafarian. Efficient random route mutation considering flow and network constraints. In *IEEE Conference on Communications and Network Security*, October 2013.

[86] Yilin Mo and Bruno Sinopoli. Secure control against replay attacks. In *47th Annual Allerton Conf. on Communication Control and Computing*, pages 911–918, September.

[87] K. Morrow, E. Heine, K. Rogers, R. Bobba, and T. Overbye. Topology perturbation for detecting malicious data injection. In *45th Hawaii International Conference on System Science (HICSS)*, pages 2104–2113, January 2012.

[88] K. Davis, K. Morrow, R. Bobba, and E. Heine. Power flow cyber attacks and perturbation-based defense. In *IEEE Third International Conference on Smart Grid Communications*, pages 342–347, November 2012.

[89] W. Niemira, R. Bobba, P. Sauer, and W. Sanders. Malicious data detection in state estimation leveraging system losses & estimation of perturbed parameters. In *IEEE International Conference on Smart Grid Communications*, pages 402–407, October 2013.

[90] Power systems test case archive. `http://www.ee.washington.edu/research/pstca/`.

[91] J. Chen and A. Abur. Placement of pmus to enable bad data detection in state estimation. *IEEE Transactions on Power System*, 21:1608–1615, Nov 2006.

[92] J. De La Ree, V. Centeno, J. S. Thorp, and A. G. Phadke. Synchronized phasor measurement applications in power systems. *IEEE Transactions on Smart Grid*, 1:20–27, Jun 2010.

[93] J. Stewart, T. Maufer, R. Smith, C. Anderson, and E. Ersonmez. Synchrophasor security practices, 2011. `https://www.selinc.com/WorkArea/DownloadAsset.aspx?id=8502`.

[94] S. Jajodia, A. K. Ghosh, V. Swarup, C. Wang, and X. S. Wang, editors. *Moving target defense- creating asymmetric uncertainty for cyber threats*, volume 54 of *Advances in Information Security*. Springer, 2011.

[95] Moving target defense (mtd). Cyber Security, R & D Center, U.S. Homeland Security, 2011. `http://www.cyber.st.dhs.gov/moving-target-defense/`.

[96] D. Divan and H. Johal. Distributed facts - a new concept for realizing grid power flow control. In *Power Electronics Specialists Conference, 2005. PESC '05. IEEE 36th*, pages 8–14, June 2005.

[97] K.M. Rogers and T.J. Overbye. Some applications of distributed flexible ac transmission system (d-facts) devices in power systems. In *40th North American Power Symposium (NAPS'08)*, pages 1–8, 2008.

[98] Dennis J. Brueni and Lenwood S. Heath. The pmu placement problem. *SIAM Journal on Discrete Mathematics*, 19(3):744–761, 2005.

[99] R. Treinen. Shift factors: Methodology and example. `http://www.caiso.com/docs/2004/02/13/200402131609438684.pdf`, 2005. CRR Educational Class, CAISO Market Operations.

[100] P.W. Sauer, K.E. Reinhard, and T.J. Overbye. Extended factors for linear contingency analysis. In *Annual Hawaii International Conference on System Sciences*, pages 697–703, Jan 2001.