

2015-06-26

# Formal Analysis and Verification of Cyber-Physical Systems for the Smart Grid

Henry Senyondo

*University of Miami*, [henrykironde@gmail.com](mailto:henrykironde@gmail.com)

Follow this and additional works at: [https://scholarlyrepository.miami.edu/oa\\_theses](https://scholarlyrepository.miami.edu/oa_theses)

---

## Recommended Citation

Senyondo, Henry, "Formal Analysis and Verification of Cyber-Physical Systems for the Smart Grid" (2015). *Open Access Theses*. 571.  
[https://scholarlyrepository.miami.edu/oa\\_theses/571](https://scholarlyrepository.miami.edu/oa_theses/571)

This Embargoed is brought to you for free and open access by the Electronic Theses and Dissertations at Scholarly Repository. It has been accepted for inclusion in Open Access Theses by an authorized administrator of Scholarly Repository. For more information, please contact [repository.library@miami.edu](mailto:repository.library@miami.edu).

UNIVERSITY OF MIAMI

FORMAL ANALYSIS AND VERIFICATION OF CYBER-PHYSICAL SYSTEMS  
FOR THE SMART GRID

By

Henry Senyondo

A THESIS

Submitted to the Faculty  
of the University of Miami  
in partial fulfillment of the requirements for  
the degree of Master of Science

Coral Gables, Florida

August 2015

©2015  
Henry Senyondo  
All Rights Reserved

UNIVERSITY OF MIAMI

A dissertation submitted in partial fulfillment of  
the requirements for the degree of  
Master of Science

FORMAL ANALYSIS AND VERIFICATION OF CYBER-PHYSICAL SYSTEMS  
FOR THE SMART GRID

Henry Senyondo

Approved:

---

Saman Zonouz, Ph.D.  
Assistant Professor of Electrical  
and Computer Engineering  
Rutgers University

---

Mohamed Abdel-Mottaleb, Ph.D.  
Professor of Electrical and Computer  
Engineering

---

Kamal Premaratne, Ph.D.  
Professor of Electrical and Computer  
Engineering

---

Dean of the Graduate School

SENYONDO, HENRY

(M.S., Electrical and Computer Engineering)

Formal Analysis and Verification of  
Cyber-Physical Systems for the Smart Grid

(August 2015)

Abstract of a thesis at the University of Miami.

Thesis supervised by Professor Saman Zonouz.

Co-supervised by Professor Mohamed Abdel-Mottaleb.

No. of pages in text. (53)

The current development in cyber-physical systems technology, from a static to a more dynamically distributed environment, has contributed towards the need for the development of the future cyber-physical security support systems. These power systems have evolved from a unidirectional to a bidirectional infrastructure with millions of nodes from the source to the destined power user. The existing security tools can not provide the required level of trusted platform for these system. The monitoring of this dynamic network involves ensuring that the network is in a stable state under all circumstances. The circumstances could include natural disasters, attacks from terrorist activities, undetected malfunctions and poor configurations. The existing security schemes in power control systems only consider securing the the power grid at single point of the infrastructure level especially using firewalls. In this thesis, we present a series of threat models that could be used against the evolving cyber-physical system and we model tools that prevent these attacks. We utilize the SMT verification solver engine to perform the formal analysis of the system components.

*Dedicated to my Parents*

## Acknowledgments

I want to thank my advisor Prof. Saman Zonouz for giving me this opportunity to work under his guidance. This has only been possible because of his brilliant mentorship and great ideas during my time at 4N6 Research Group.

I would also like to thank Prof. Mohamed Abdel-Mottaleb for being my co-supervisor, as well as providing all the additional support needed for this work. In addition, I would like to thank him for the parental guidance he provided.

I would also like to thank all the people who contributed to the project, Dr. Stephen McLaughlin, Clement Garrigues, Luis Garcia Antonio, Alessio Antonini, Ying Chen. This work could never be successful without your contributions.

I also thank the members of my committee, Prof. Mohamed Abdel-Mottaleb, Prof. Saman Zonouz, Prof. Kamal Premaratne for providing valuable feedback and insights for this work as well as the time they spent reviewing this thesis.

Many thanks to the 4N6 Research and Fortinet Cybersecurity Laboratory members for the support and creating a very conducive research environment. I also thank our staff.

Lastly, I would like to thank my Parents, relatives and friends for being so supportive. They are always there for me, and given me a listening ear.

HENRY SENYONDO

*University of Miami*

*August 2015*

# Table of Contents

<b>LIST OF FIGURES</b>	<b>viii</b>
<b>LIST OF TABLES</b>	<b>x</b>
<b>1 INTRODUCTION</b>	<b>1</b>
1.1 Summary of Contributions . . . . .	4
1.2 Related Work . . . . .	4
1.3 Thesis Organization . . . . .	5
<b>2 CYBER-PHYSICAL SYSTEMS</b>	<b>7</b>
2.1 Control System Security . . . . .	8
2.2 Automation Control Architecture . . . . .	9
<b>3 THREAT MODEL</b>	<b>13</b>
3.1 Control System Threats . . . . .	14
3.2 Current Countermeasures . . . . .	17
<b>4 EXPERIMENTAL SETUP</b>	<b>19</b>



4.1	Reverse Engineering . . . . .	19
4.1.1	Disassembler . . . . .	19
4.1.2	Compiler . . . . .	21
4.2	Data Exchange . . . . .	22
4.3	Performance . . . . .	24
4.3.1	Compiler Evaluation . . . . .	24
4.3.2	Disassembler Evaluation . . . . .	24
4.3.3	Latency Evaluation . . . . .	24
<b>5</b>	<b>MODEL VERIFICATION</b>	<b>30</b>
5.1	Parallel Model Generation . . . . .	30
5.2	Symbolic Execution of PLC Code . . . . .	31
5.2.1	Model Checking . . . . .	34
5.2.2	Model Refinement . . . . .	35
<b>6</b>	<b>COVERT CHANNEL COMMUNICATION</b>	<b>38</b>
6.1	PHYCO Introduction . . . . .	38
6.2	PHYCO Threat Model . . . . .	39
6.3	Message Transmission . . . . .	40
6.3.1	Message Transfer . . . . .	42
6.3.2	Message Reception . . . . .	43
6.4	Data Transfer Reliability . . . . .	43

<b>7 CONCLUSION</b>	<b>48</b>
7.1 Ongoing and Future Work . . . . .	48
<b>BIBLIOGRAPHY</b>	<b>50</b>

# List of Figures

2.1	programmable logic controller . . . . .	9
2.2	Statement List Sample . . . . .	10
2.3	Function Block Diagram Sample Sample . . . . .	11
2.4	Ladder Logic . . . . .	11
2.5	PLC Architecture Overview . . . . .	12
3.1	Industrial Control Systems Attacked: ICS-CERT 2014 . . . . .	14
3.2	Threats Attack Methods: ICS-CERT 2014 . . . . .	15
4.1	MC7 figure . . . . .	26
4.2	Instruction bit representation. . . . .	27
4.3	STL source. . . . .	27
4.4	Compiler Evaluation . . . . .	28
4.5	Disassembler Evaluation . . . . .	28
4.6	PLC Latency Evaluation . . . . .	29
5.1	RCR Execution Overview . . . . .	31
5.2	Instruction List Intermediate Language grammar . . . . .	33
5.3	Sample IL To ILIL conversion . . . . .	36

5.4	Symbolic Execution Example . . . . .	37
6.1	PHYCO Thread Model . . . . .	41
6.2	Power System Topology. . . . .	44
6.3	Received signal affection for decoding. . . . .	45
6.4	PHYCO decoding time requirements. . . . .	46

# List of Tables

6.1	Received signal affection for decoding. . . . .	45
-----	---	----

# CHAPTER 1

## Introduction

A shift in the power system infrastructure from a static to a more dynamically distributed environment has contributed towards the need for the enhancements of the future cyber-physical support systems. The power systems have evolved from a unidirectional to a bidirectional infrastructure with millions of nodes from the source to the destined power users.

These systems have also moved from physically connected networks to wireless power transmission systems, where electromagnetic energy is distributed across the grid without interconnecting wires [1]. The end users could be storage facilities or applications. These storage facilities could later be used as power sources. The monitoring of this dynamically changing network involves ensuring that the entire network is in a stable state under all circumstances that could include natural disasters, attacks from terrorist activities, undetected malfunctions and poor configurations. This ubiquitous nature of the power grid is referred to as the smart grid. A smart grid, refers to a power grid environment where power distribution and information follows a bidirectional model in an automated fashion to improve the efficiency, reliability, economics, and sustainability of the production and distribution of electricity. The

smart grid infrastructure should have highly effective fault detection and self-healing mechanisms.

In order to have good control of these systems, industrial control systems (ICS) and supervisory command and data acquisition (SCADA) systems have been used. These systems are connected to different components of the smart grid including networks of the critical cyber-physical infrastructure and the internet. The systems monitor large amounts of data over the smart grid that is collected at the different points. The complexity of the systems is associated with a high cost of security since cyber-attacks can easily be tailored at different parts. Any intrusion on the cyber-physical infrastructure could cause catastrophic damage including a big economic loss.

The existing security schemes in power control systems only consider securing the power grid at single points of the cyber-physical infrastructure [2, 3]. In this work, we categorize the power grid into two layers. The top layer, which we refer to as the network/distribution, is comprised of the distribution and the end users of the smart grid environment. The core/lower layer deals with the automation and control processing. All the energy distribution in the grid takes place at this layer. The layer constitutes the control systems which govern the power flow and information flow. A programmable logic controller (PLC) is a general-purpose programmable digital computer that is mainly used for automation [4]. PLCs can be compared to normal desktop computers, however they are more reliable. PLCs are designed with longevity in mind can withstand extreme temperatures, humidity and vibration. At the smart grid automation level, PLCs are used for processing the distribution of power, data handling, data storage, and communication.

Monitoring and control of the smart grid infrastructure is done by the SCADA systems [5–8]. The SCADA systems can be distributed over the entire smart grid since the environment is large. Due to the evolution of the power grid, SCADA systems have evolved and they have included the internet to their architecture cite.

A SCADA system usually consists of the following subsystems:

- Remote terminal units (RTUs) are special units that connect to sensors. They are used to convert sensor signals to digital data. These RTUs have telemetry hardware capable of sending and receiving the converted digital data to the supervisory system.
- Programmable logic controller (PLCs) which perform the automation by using sensors. The sensor data is converted to digital data and provided to the PLCs as inputs. The PLC provide digital outputs after processing the instructions using the input data.
- A data server is a software service which stores all the data over the smart grid. This data could be from wired components like the PLCs or wireless components like the Satellites. The data center is connected to most parts of the Smart grid. Data in the smart grid is distributed and managed by a distributed database management system. The main data elements of the SCADA are called tags or points [9–11]. These include the input values, output values and operation data. The input and output can be obtained from the various sensors. This type of data is known as soft data. The operation data is obtained from the control systems and is known as hard data.



## 1.1 Summary of Contributions

Stuxnet which was discovered in June 2010 is described as a computer worm [12] with zero-day attack [13]. It was designed to attack industrial programmable logic controllers. PLCs are used to automate electro-mechanical processes like those used to control machinery on factory assembly lines, or centrifuges for separating nuclear material. Stuxnet worm compromised Iranian PLCs, collecting information on industrial systems and causing the fast-spinning centrifuges to tear themselves apart. Stuxnets design and architecture are not domain-specific and it could be tailored as a platform for attacking modern SCADA and PLC systems. Stuxnet is said to have ruined almost one-fifth of Iran's nuclear centrifuges [12].

The current goal of this work is to develop tools that could support these critical cyber-physical infrastructures from such attacks and can ensure a reinstatement in case of such disasters. This would require both securing the infrastructure at the hardware level and at the application level.

## 1.2 Related Work

The current evolution of the smart grid environment has led to development of new strategies that support and enhance the security of this cyber-physical infrastructures. Several protocols and policies have been put in place to enhance the security of the cyber-physical infrastructure. The National Institute of Standards and Technology (NIST) created guidelines to allow the secure integration of devices on to the smart grid [14]. However, there has been a high increase in the number of attacks on the cyber-physical systems. A report from ICS-CERT, shows over 70% increase in the

number of threats and attacks on these systems. Data mining and analysis concepts have been utilized in network communications as a means to intrusion detection. These tools and approaches can help in detecting intrusion based on the expected behavior of the various data points in the network [15–18]. The smart grid has communication platforms with several nodes and data is transmitted between these nodes.

The communication platform of the smart grid can be attacked using several approaches. Some researchers have shown techniques to attack these systems. Some of the most complicated attacks have shown stealth capabilities. Xiao, et al. [19] worked on a covert channel using copy-on-write-based memory deduplication implementations. In the work, they showed how these communications could be exploited by the attackers in virtualized environments specifically. Additionally, Castiglione, et al. [20] proposed stealth communication using spam emails. Khan, et al. [21] designed a cluster-based covert channel so as to evade disk investigation and forensics attempts. The covert channel avoids detection by not using the storage on the filesystem of the compromised system hence lesser chances to be detected by the detection tools. PHYCO avoids detection by the cyber security sensors to zero since it does not use the cyber assets, computer networks nor PLC links of the cyber-physical infrastructure

### 1.3 Thesis Organization

This Thesis illustrates the concepts and approaches that could be used to enhance the security of cyber-physical infrastructures. The main underlying feature of this work is the use of formal methods to model and detect intrusions in the cyber-

physical systems. We define threat models and provide well tailored approaches for the threats. Chapter one provides an overview of the current state of the cyber-physical systems/smart grid. Chapter two explains the architecture of the control systems in the cyber-physical environment. In chapter three, we give a description of the threat models and some current counter measures that have been proposed. Chapter four show the reverse engineering of the core components of the cyber-physical infrastructure. Additionally, we describe the architecture of our experiment in this chapter. In chapter five, we give details of the formal methods and how they are mapped into our system to perform model verification. Chapter six demonstrates our threat model giving a fully descriptive and well tailored stealth intrusion example. In chapter seven we provide a conclusion and future work.

## CHAPTER 2

# Cyber-Physical Systems

The current trend and shift in the power system infrastructure from a static to a more dynamically distributed environment has contributed towards the need for the enhancements of the future cyber-physical support systems. The power systems have evolved from a unidirectional to a bidirectional infrastructure with a millions of nodes from the source to the destined power user. The users could be storage facilities or applications/end-users. Energy could be transmitted from one source and stored; later the storage becomes the source providing energy to the end user. For example, the green cars evolution has utilized energy storage facilities to enable recharging of electric cars. The monitoring of this dynamic network involves ensuring that the network is in a stable state under all circumstances that could include natural disasters, attacks from terrorist activities, undetected malfunctions and poor configurations. The existing security schemes in power control systems only consider securing the the power grid at single point of the infrastructure level. The current goal of this work is to develop tools that could support this infrastructure and could ensure a reinstatement in case of such disasters. This would require both securing the infrastructure at the hardware level and at the application level.

## 2.1 Control System Security

The power grid has transformed from a static to a more dynamically distributed environment [22]. This evolution has brought about new improved technology like smart meters to be added to the original power grid. This development has contributed to a smarter power grid referred to as the smart grid. However, the evolution requires deep understanding of these added technologies in order to keep the grid secure. This new hardware is developed by different manufacturers. The hardware have different protocols that are used when communicating with the critical cyber-physical infrastructure. A fault in the production system could create an entry point for intrusion. Detecting attacks over this dynamic grid becomes a very difficult process and needs very diverse approaches.

According to Camacho, et al [23], smart grid is an incorporation of intelligence, communication and the power grid. To provide efficient cyber-security for these cyberphysical components, first we need to understand the core components that support this infrastructure. In our work we classify the cyberphysical environment into layers. At the top of the stack is the network/distribution layer. This layer includes all the elements that enable power and communication to be transferred to different locations. In the case of communication, the network of sensors and physical lines are utilized for this purpose. Data in the sensors could either be analog or digital. The sensors are also known as data points in the cyberphysical environment since each sensor is a source of data.

As per the Industrial Control System Cyber Emergency Team (ICS-CERT), there has been a high increase in the number of attacks that have been tailored toward industrial control systems.

## 2.2 Automation Control Architecture

The core component of the cyberphysical infrastructure is the automation or control center. This comprises special purpose computers (PLCs) to monitor and control the flow/distribution of the power. PLCs are micro-controllers which are specifically

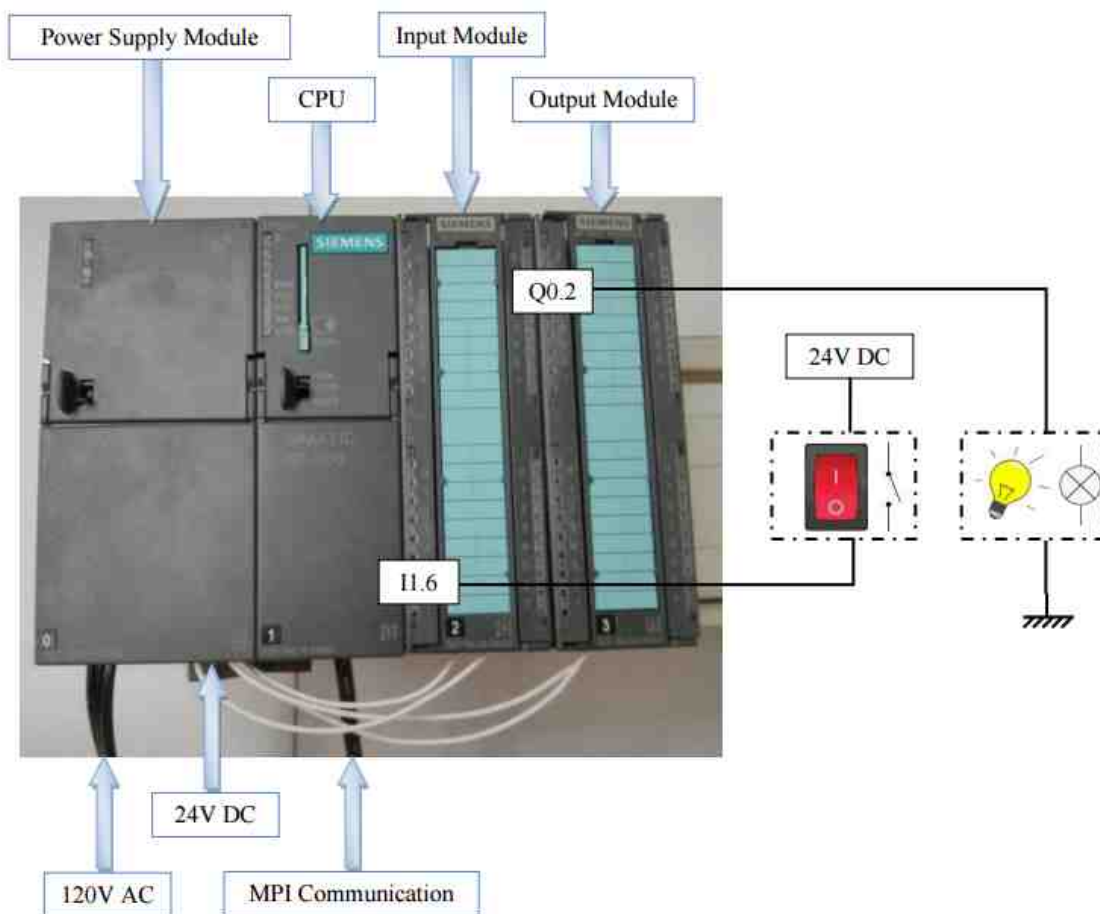
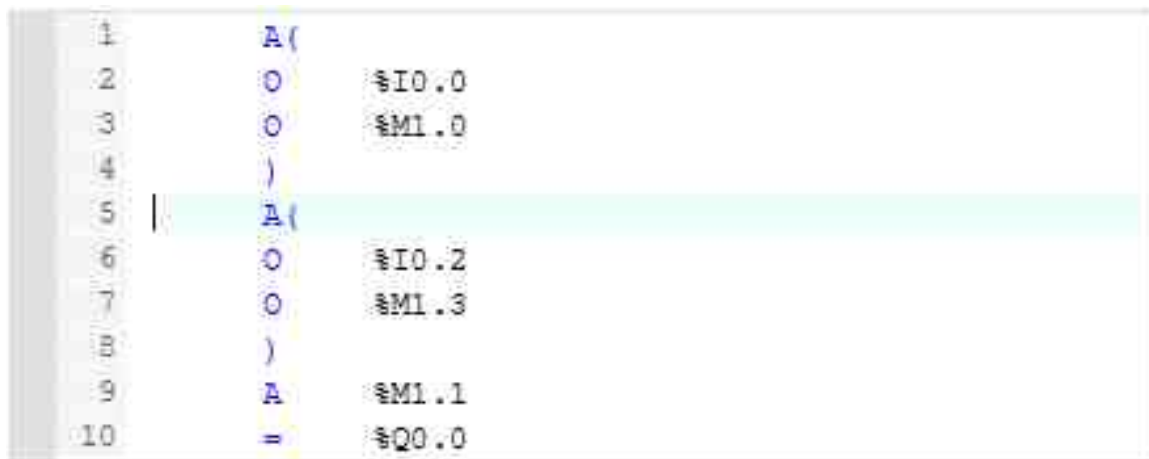


Figure 2.1: programmable logic controller

built for industrial automation. Unlike desktop computers, PLCs are designed to

withstand the harsh environment of industrial processing. As input from sensors are taken in, the PLCs runs a set of instructions based on these inputs and they create the corresponding outputs. Each single execution of these machines is called a scan cycle. Figure 2.1 shows a diagram of a PLC.

PLCs are programmed in mainly three languages: Statement List (STL), Ladder logic (LAD) and Function Block Diagram (FBD). Both Statement List and Ladder Logic provide a graphical circuit like diagram interface. On the other hand, the Statement List PLC programming language is low-level machine-oriented language. Instructions in the Statement List language are line-oriented; Each instruction takes up one line. The instruction line begins with an operation or command and one or many operands. In some cases like loops/jumps, the instructions may be preceded by a label followed by a colon. Figure 2.2 shows a sample of STL programming language.



```
1      A(
2      O      %I0.0
3      O      %M1.0
4      )
5      | A(
6      O      %I0.2
7      O      %M1.3
8      )
9      A      %M1.1
10     =      %Q0.0
```

Figure 2.2: Statement List Sample

The Function Block Diagram is a graphical programming language that connects block-boxes to form a program. These blocks may be arithmetic, Boolean, or other

functional elements and function blocks. Figure 2.3 shows a sample of the FBD language.

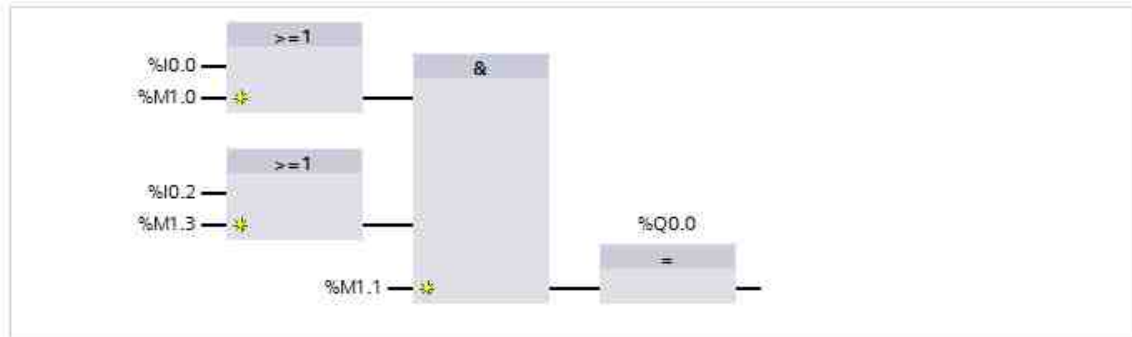


Figure 2.3: Function Block Diagram Sample Sample

Ladder Logic is a graphical programming language comparable to relay controls. They utilize the Boolean variables and connect elements using vertical or horizontal lines. Figure 2.4 shows a sample of LD language.

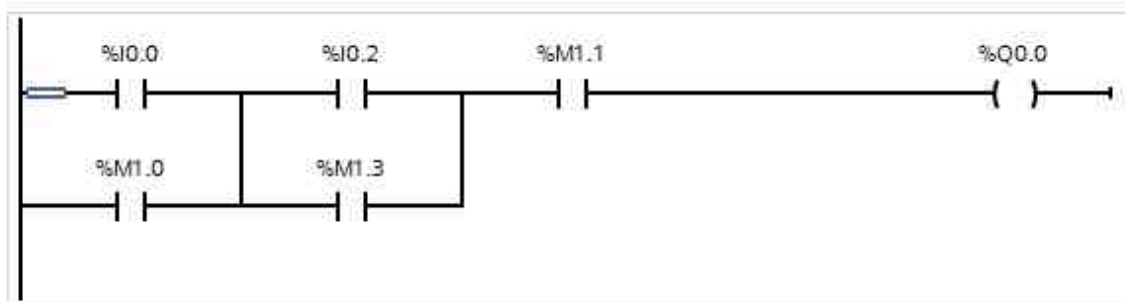


Figure 2.4: Ladder Logic

The PLC firmware is comprised of the several blocks; organization blocks (OB), Function Block (FB), Data Block (DB) and Function (FC). PLC execution of the instruction set starts at the organization block. An industrial process can have more than one OB block. The Function Block stores the static variables of the PLC. The Function does not contain static variables but can be assigned parameter for



processing. At the start of the program execution, the operating system identifies the OB block and the control is passed to this block. The flow of control during execution is based on the instructions in that process implemented. In the case where the program has other OB blocks, control is passed on to these blocks sequentially based on the block number.

In addition to the main blocks (OB, FB, DB and FC), the PLC has three system blocks SFB, SFC and SDB. These blocks are not available to the users but can be accessed from the main building blocks (OB, FB and FC). The figure 2.5 below gives an overview of the PLC architecture.

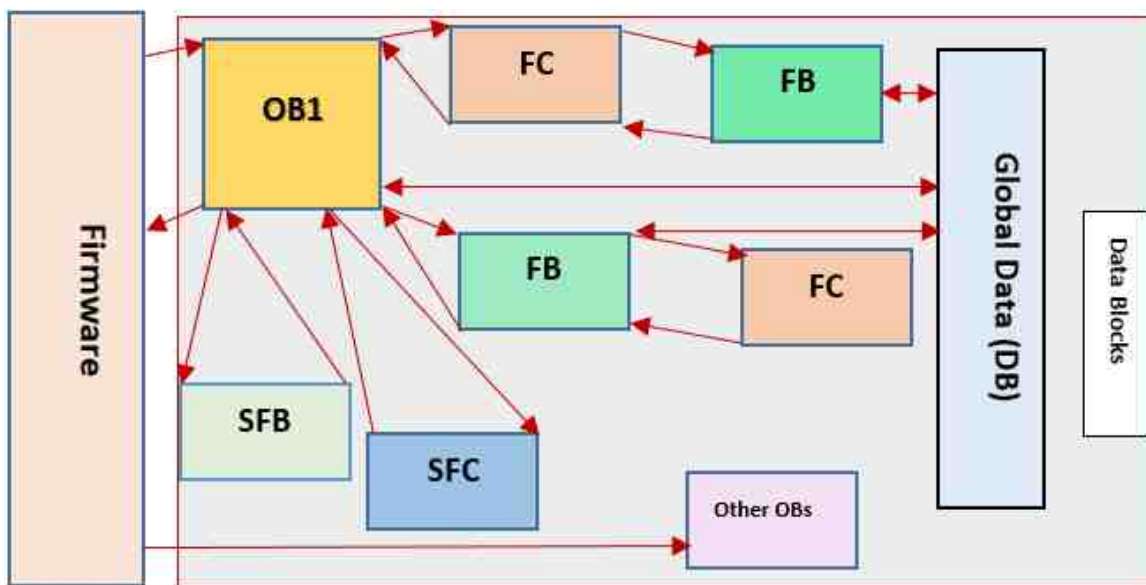


Figure 2.5: PLC Architecture Overview

## CHAPTER 3

# Threat Model

Stuxnet, discovered in 2010 [12], is one of the most powerful malware that is known to have attacked cyber-physical systems. The malware was developed to target the control systems of the Iranian nuclear power plant. This malware may have been introduced to the system using a USB, however it had the capabilities of reproducing and infecting other PLC control systems on the network. The malware was designed to read data while sending it to a different control unit undetected. At the Iranian nuclear plant, Stuxnet also provided false data readings to the human machine interface (HMI) as it operated the centrifuges with its own sets of data and commands. This eventually managed to cause the fast-spinning centrifuges to tear due to the high speed caused by the malware. There are many approaches and protocols that have been proposed to protect cyber-physical systems [24,25] However, cyber-physical systems still lack sufficient approaches and tools for protection. In this chapter, we describe the various attack vectors that can be used against the cyber-physical systems and last we give an overview of the current counter measures.

### 3.1 Control System Threats

CyberPhysical system attacks can be tailored using various vectors. Different attack vectors have been demonstrated and several of these attacks have been reported and documented by the ICS-CERT. In 2014, ICS-CERT reported 245 industrial control systems attacks that were responded to. Figure 3.2 shows the 2014 ICS-CERT fiscal year report. According to the report, 32% targeted the Energy Sector, 27% targeted Critical Manufacturing systems, 6% targeted Health care, Water supply systems and Communications each with 6%, and Government Facilities at approximately 5%.

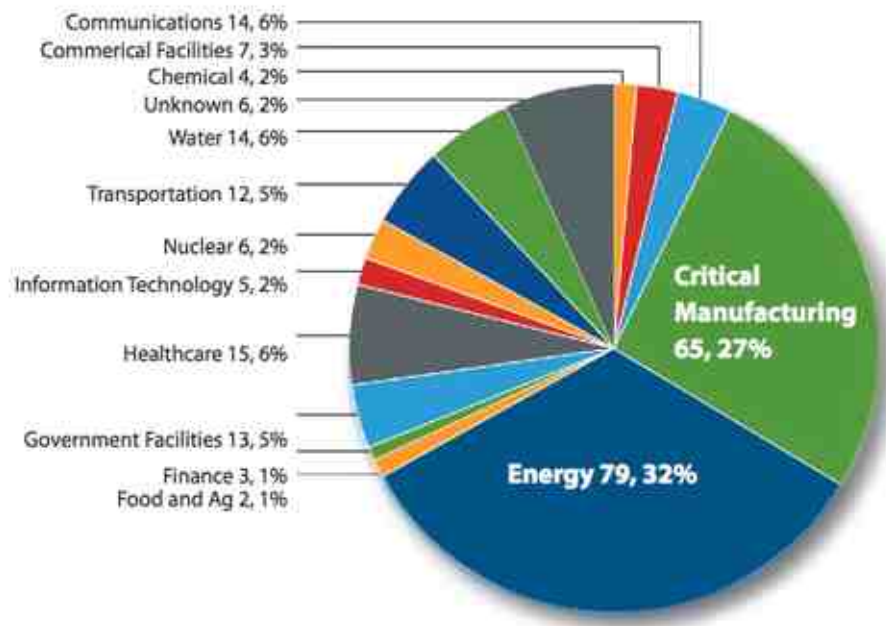


Figure 3.1: Industrial Control Systems Attacked: ICS-CERT 2014

According to the report, more than half of the incidents reported, the method of attack remained unclear, while 17% were spear phishing ops, 22% were network scanning, and 5% targeting weak authentication. Figure 3.2 shows the various methods used in these attacks. The effectiveness of these attacks depends on how well the mal-

ware is developed. In addition, the extent of penetration depends on the protection protocols at various layers in the cyber-physical infrastructure. To understand the possibility of the attack vectors, we discuss the various entry points and the possible extent of penetration. The extent of penetration, after an entry point has been found, depends on the internal security of the system components in that network.

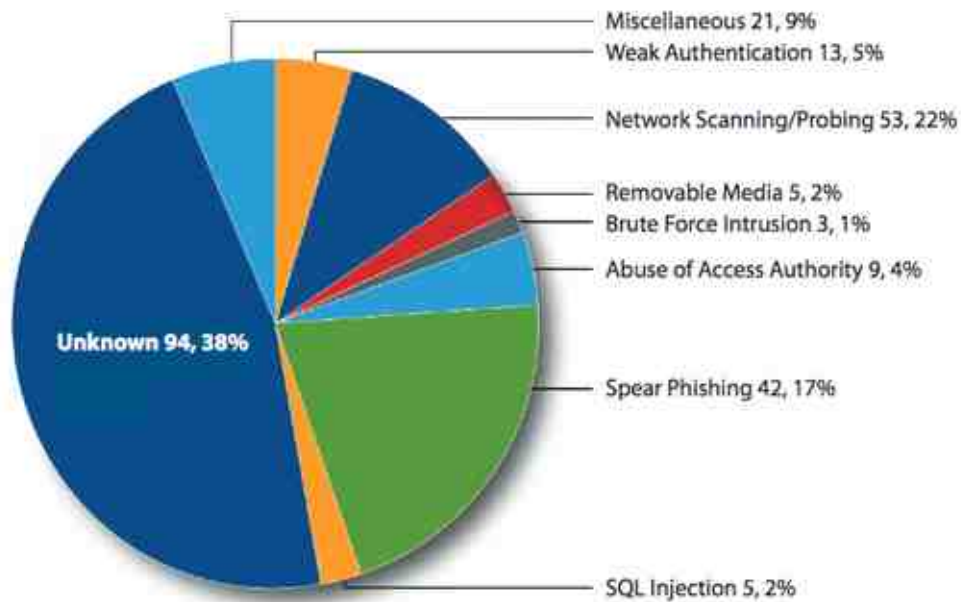


Figure 3.2: Threats Attack Methods: ICS-CERT 2014

The HMI is an important component of the smart grid or cyber-physical systems. This HMI is comprised of computers that are used for the monitoring of the various states of the systems. An attack can be launched by compromising these systems internally. Since these machines are operated by humans, an insider could introduce the malware to the system using various memory devices such as USBs. In another case, a legitimate employee could intentionally give wrong instructions to the control system with a goal of destroying the system.

CyberPhysical systems make up very large distributed networks. The networks spread from the end user applications to the power sources. As the network expands rapidly with the integration of new devices on the grid, security measures have been developed to protect the network. The main method used is the use of firewalls. These firewalls could easily be exploited if some of the devices are not well configured [26,27]. Once a network is compromised, several attacks could be launched that would cause catastrophic damages to the cyber-physical critical infrastructure. An intelligent malicious attack [28] can also use the network topology to calculate the appropriate attack.

Network attacks can be performed both at the power control layer and at the distribution/application layer. In the former case, a single PLC attack over a distributed network could lead to the compromise of other PLCs in the same network [29–32]. An attacker could use the topology of this compromised network to perform a more sophisticated attack on the entire system. For example, PHYCO [28] utilizes the topology of these systems to perform a targeted and well calculated attack.

At the distribution or application level, several threat models have been proposed targeting end users [33]. These security threats include smart meter data collection for consumer profiling. In this case, smart meters are attacked and the data collected is used to spy on the consumers.

Another attack vector could be compromising the grids components at the manufacturing centers. Since the hardware that supports these cyber-physical infrastructure uses software, a slight twist in the requirements could cause these machines to have entry points that may not be detectable at manufacturing and testing time. However, some measures have been proposed to reduce the risks of such incidences [34].

## 3.2 Current Countermeasures

programmable logic controllers are the core components of power-distribution control in the cyber-physical infrastructure. These programmable logic controllers are embedded with software that control and executes the instructions on the machine. This software is referred to as firmware and can be updated. In order to program the PLCs, source code written in either STL, LAD or FBD is compiled and is installed on the machines. The use and the update of software exposes these systems to vulnerabilities. Many researchers have shown attacks targeted at firmwares.

Rouf, et al. [35], demonstrated an attack in which they perform an eavesdropping at 40ms on moving car sensors. In the work, they reverse engineered the underlying protocols of the firmware. They revealed the static 32 bit identifiers used in the protocols and showed how messages could be triggered remotely. Similarly, Koscher, et al. [36], showed how, if the Electronic Control Unit (ECU) was infiltrated, an attacker can leverage the ability to completely circumvent a broad array of safety-critical systems.

In this thesis, we provide a similar approach of infiltration. We demonstrate the possible vulnerabilities in the embedded systems used in the cyber-physical systems. Specifically we perform our case study on the programmable logic controllers. In the study, we demonstrate how the firmware could be compromised. Our work is based on a tailored remote attack. In addition, we show a stealth channel of attack vector on the cyber-physical systems.

Last, we provide countermeasures for such attacks that could be targeting the cyber-physical systems. In our research, we propose several methods that could be

used to detect any intrusion on these critical infrastructures. The models are based on formal analysis approaches in which we demonstrate the Trusted Safety Verifier. In order to provide an efficient trusted base for the cyber-physical systems, we lower the detection of intrusion and malware to the firmware level.

## CHAPTER 4

# Experimental Setup

## 4.1 Reverse Engineering

### 4.1.1 Disassembler

The disassembler converts the machine readable mc7 executable binary files running on the PLC to Statement List language source. Each mc7 file is processed linearly, reading each bit and there after transforming a group of bytes to the corresponding source instruction statements. The number of bits representing each instruction for this architecture vary in size depending on the type of instruction and the operation performed by the instruction. The bit pattern design is dependent on the category of instruction which we classified under the following categories; data transfer operations, arithmetic operations, branching operations, logical operations, and machine-control operations. The number of bits for the instructions that operate on data depend on the number of bytes for that data. For example, the opcode Load input byte (L IB) takes 16 bits to represent the data that ranges between 0 to +127 and it uses 32 bits to represent the data between +128 to +65535.

To understand the organization of these bits in the binary, we analyzed permutation sets of instruction in the different memory blocks of the PLC. The organization



block (OB), was used as the initial start point for the analysis since it is the beginning point of program execution. We started with an analysis of the mc7 file created when the project's main block (OB) is empty. The instructions in the OB block were incremented starting with an empty network, followed with sets of instruction and later calls to different function blocks were included.

Figure 4.2 shows an example of the bit organization in an instruction. The first two instruction opcodes A and ON need 2 bytes, X requires 4 bytes. In this case, the m bits are used for either input-output or memory area, the b bits are for the address line and the v bits are used to represent either input or output or memory data.

The mc7 executable binary files are obtained after compiling source code from either STL, LAD or FBD source programs. These files contain both the meta data of the program and the instruction hexadecimal representation. Instruction are grouped in networks and the size of each network is equal to the total number of bytes for these instructions. Depending on the language used, a hexadecimal value at the address 0x04 is used to match the corresponding language that was used. For each mc7 file, memory is allocated at the address 0x05 to identify the block types (OB, FB, FC). The next two memory addresses are reserved for the block number. Two bytes before the start of the code section (0x22 and 0x23) are allocated to store data related to the size of code ( $SC$ ). The size of code is calculated using the equation  $SC = \varphi[0x23] + 100 * \varphi[0x22] - 0x02$ . To understand these semantics used for the PLC instructions, we compared executable files compiled and run on the PLC. The source code are compiled and the binaries are directly loaded into the PLC, no copy of the binaries are stored in the compiling software. To obtain these compiled executable files we download the copies directly from the PLC. This requires the data to be read

from certain memory location of the PLC after a connection has been established. Using a PLC communication framework {libnodave reference}, we obtained copies of the executable files that were loaded in to the PLC. The copies of these mc7 executable files obtained from the PLC are progressively analyzed to determine the structures of the bits and the mnemonic representation of each corresponding STL as shown in figure 4.2.

PLC programs are made of a combination of one or more networks. A network defines the simple task for automation, a group of these networks would coordinate a given task for a particular automation. Since the source code is structured in networks, the total size of individual networks ( $S_n$ ) equals to the size of the code i.e  $SC = \sum_{n=0}^N S_n$ . The address of the total number of networks in a given block is dependent on the block type and block number. To obtain this address, each block type has a block number that is proportionally associated with a constant ( $\delta C$ ) which is used when calculating the address of the total number of networks ( $N$ ). The address of the final instruction is obtained by adding the size of code to the address marking the beginning of the code  $N = \varphi [\delta C + \mathbf{SC}]$ .

$$N = \varphi \left[ \delta C + \sum_{n=0}^N S_n \right] \quad (4.1)$$

### 4.1.2 Compiler

Our compiler converts the statement list source code to the mc7 executable binary files based on the PLC architecture. The processing of the STL source code is performed linearly just like the case of the disassembler. Initially the source code is parsed, since the syntax of the language permits only one instruction statement per

line, single lines are processed one at a time. The amount of memory to be used from for the instruction is calculated after determining the bytes to be used by the instruction. Each instruction line is pre-processed to identify the opcode or both the opcode and operand. The number of bits used for only the opcode is predetermined since it does not vary. Using a look up table with a predefined bit structure for that particular opcode, the opcode is transformed to the corresponding bit representation. Each structure of the opcode defines a the format for the data associated with the instruction giving a range of values that are acceptable for the instruction. The operand is then transformed based on this range of values. To obtain the bit representation for the instruction, we perform an AND operation on the opcode and the operand. The result is converted to the respective hexadecimal equivalent. Figure 4.2 shows a sample opcode and the their respective bit structures.

Mc7 files constitute meta data, this data varies from one file to another. In order to compile the source code, we identify the memory locations for some of meta data. The locations for the different meta data depends on the attributes within the source files to be compiled. For example, the block number, block type, and the source language used for the source program.

## 4.2 Data Exchange

In order to verify the set of instructions running on the PLC, we need to have a copy of the source code, PLC concrete values and the set of specifications for the given automation task. During the program compilation of the source code, the binaries are uploaded to the PLC and no copy of this binary is externally stored. Obtaining these binary files (mc7) that are running on the PLC requires to establish a communication

with the PLC. In the experiment, libnodave is used to communicate and download these executable file from the PLC. Programs on the PLC are executed in cycles called scan cycles. Each scan cycle reads the memory image on an average of 150ms and based on the input a given set of outputs are produced. To perform the verification for real time systems, we needed to keep a count each scan cycle number. In the experiment, a static variable is included in the source program and this is updated for every scan cycle as the scan cycle number.

Each scan cycle reads data from the input memory locations of the memory image. This input data comes from various input sensors. RCR monitors the scan cycles and obtains the concrete values corresponding to that scan cycle. The concrete values are then converted to ILIL format. Since we have the system running on two platforms, the connection from the PLC using libnodave acts as the server. Data from the server is then sent to the client RCR program. The server continuously reads the concrete values in real time. Since each TEG state corresponds to a given scan cycle, we AND the concrete values to the respective TEG state for that scan cycle.

Symbolic execution is known to have an explosion problem due to the number of states generated. In RCR, for each path condition created after the concrete values have been added, we perform a satisfiability check for that path. If the path condition is not satisfied, the graph generation for the particular path is expelled.

## 4.3 Performance

### 4.3.1 Compiler Evaluation

In our experiment we study the compilation performance in relation to time. To do so, we consider the following experiment. We initially designed a small project with 3 networks and 15 lines of statement list instructions. The code size of this project was increased by a scale of three more networks each containing 15 instructions. In order to have a fair comparison we performed 1,000 executions for each code size, hence obtaining the average time for the code compilation. The graph in figure 4.4 shows how the size of code performs with an increased size of code. From the graph, we observed that the as the size of the code increases the times for compilation tends towards a constant time.

### 4.3.2 Disassembler Evaluation

In order to evaluate the performance of the disassembler, we followed the same technique used for the compiler evaluation. We obtained already compiled mc7 files with known sizes of both networks and instruction list code lines. We disassembled these files 1000 times for each code size and an average time was obtained. The graph in figure 4.5 shows the Correlation between the size of code and the time for disassembling.

### 4.3.3 Latency Evaluation

Our measurement technique utilizes the PLC scan cycle time and the size of bytes read from the memory image of the PLC. In a normal no interactive environment, no bytes are read from the memory image. RTV reads the memory image in real time

obtaining different concrete cycle values. In order to determine the overhead caused by the running RTV, we evaluated the execution time for a sample piece of code on the PLC, taking note of the PLC time. The analysis reveals that the PLC execution is not dependent on the size of bytes or the number of times bytes from the image memory are read. However since the size of data read is increasing and the number of iterations, from the result, we observed that there is an increase in the transfer times of various sizes of data.

Offset (h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00000000	70	70	01	01	01	0B	00	01	00	00	01	B2	00	00	00	00
00000010	03	41	0F	C9	2A	4A	03	A1	63	83	21	A7	00	1C	00	0E
00000020	00	1A	01	40	C0	01	C0	01	10	09	41	60	00	18	FB	7C
00000030	FB	79	00	04	FE	6F	00	14	C7	00	41	50	00	00	FB	79
00000040	00	04	7E	53	00	12	FB	79	00	04	7E	57	00	02	FE	0B
00000050	84	00	00	00	FB	76	00	04	FE	6B	00	14	FB	7C	10	0A
00000060	C0	01	C0	01	10	01	41	60	00	14	3D	01	70	0B	00	02
00000070	10	02	C0	01	10	01	41	60	00	14	3D	0A	70	0B	00	02
00000080	10	02	C0	01	10	01	41	60	00	14	FB	70	01	04	70	0B
00000090	00	02	10	02	C0	01	10	03	41	60	00	18	FB	7C	FB	79
000000A0	00	01	FE	6F	00	14	FE	0B	84	00	00	00	75	01	FE	6B
000000B0	00	14	FB	7C	10	04	C0	01	10	03	41	60	00	18	FB	7C
000000C0	FB	79	00	0C	FE	6F	00	14	FE	0B	84	00	00	00	75	0A
000000D0	FE	6B	00	14	FB	7C	10	04	C0	01	10	03	41	60	00	18
000000E0	FB	7C	FB	79	01	04	FE	6F	00	14	FE	0B	84	00	00	00
000000F0	FB	72	01	04	FE	6B	00	14	FB	7C	10	04	C0	01	10	07
00000100	41	60	00	14	FB	74	00	2B	70	0B	00	02	10	08	C0	01
00000110	1D	01	70	0B	00	02	C0	01	1D	0A	70	0B	00	02	C0	01
00000120	FB	71	01	04	70	0B	00	02	C0	01	55	01	C0	01	55	0A
00000130	C0	01	FB	73	01	04	C0	01	3D	01	70	0B	00	02	C0	01
00000140	3D	0A	70	0B	00	02	C0	01	FB	70	01	04	70	0B	00	02
00000150	C0	01	C0	01	75	01	C0	01	75	0A	C0	01	FB	72	01	04
00000160	C0	01	*	00	01	00	00	14	00	00	00	02	05	02	05	02
00000170	05	02	05	02	05	02	05	05	05	05	05	05	05	0E	05	20
00000180	05	00	40	00	9A	00	12	00	29	00	2A	00	00	00	00	00
00000190	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000001A0	00	00	00	00	00	00	01	00	55	2F	00	00	00	00	00	00
000001B0	00	00														

- Type of Block (OB,FC,FB,SFC,SFB) (always @ 0x05)
- Block Number (always @ 0x06 and 0x07)
- (Size of Code + 0x02) (always @ 0x22 and 0x23)
- Code
- Number of Network
- Size of Network n°1
- Size of Network n°2
- Size of Network n°3
- Size of Network n°4
- Size of Network n°5

Size of Code = Size of Network n°1 + Size of Network n°2 + Size of Network n°3 + Size of Network n°4 + Size of Network n°5  
 Ex: 0x13E = 0x40 + 0x9A + 0x12 + 0x28 + 0x2A

★ End of Code = Size of Code + @ Begin of Code  
 Ex: 0x160 = 0x13E + 0x24

Figure 4.1: MC7 figure

```

A      ?? ??      - 1m00 0bbb ixxx xxxx And
ON     ?? ??      - 1m10 1bbb ixxx xxxx Or Not
L IB   4A ??      - 0100 1010 0ggg gggg
L IB   7E 11 ?? ?? - 0111 1110 0001 0001 gggg gggg gggg gggg
X      05 ?? 00 ?? - 0000 0101 00vv 0bbb 0000 0000 0xxx xxxx
                    Exclusive Or.

```

Figure 4.2: Instruction bit representation.

```

1 NETWORK
2 A I 0.0 //C000
3 A I 0.1 //C0C1
4 = Q 0.0 //D880
5 NETWORK
6 O I 0.2 //CA00
7 O I 0.3 //CB00
8 = Q 0.1 //D980
9 NETWORK
10 X I 0.4 //05140000
11 X I 0.5 //05150000
12 = Q 0.2 //DA80
13 NETWORK
14 L C#140 //Counter preset value.
15 A I 0.1 //Preset counter after detection of rising edge of I 0.1.
16 S C1 //Load counter 1 preset if enabled.
17 A I 0.0 //One count down per rising edge of I 0.0
18 CD C1 //Decrement counter C1 by 1 when RLO transitions.
19 AN C1 //Zero detection using the C1 bit.
20 = Q 0.0 //Q 0.0 = 1 if counter 1 value is zero.
21 NETWORK
22 AN I 2.6 //E602
23 = Q 0.7 //DF80

```

Figure 4.3: STL source.



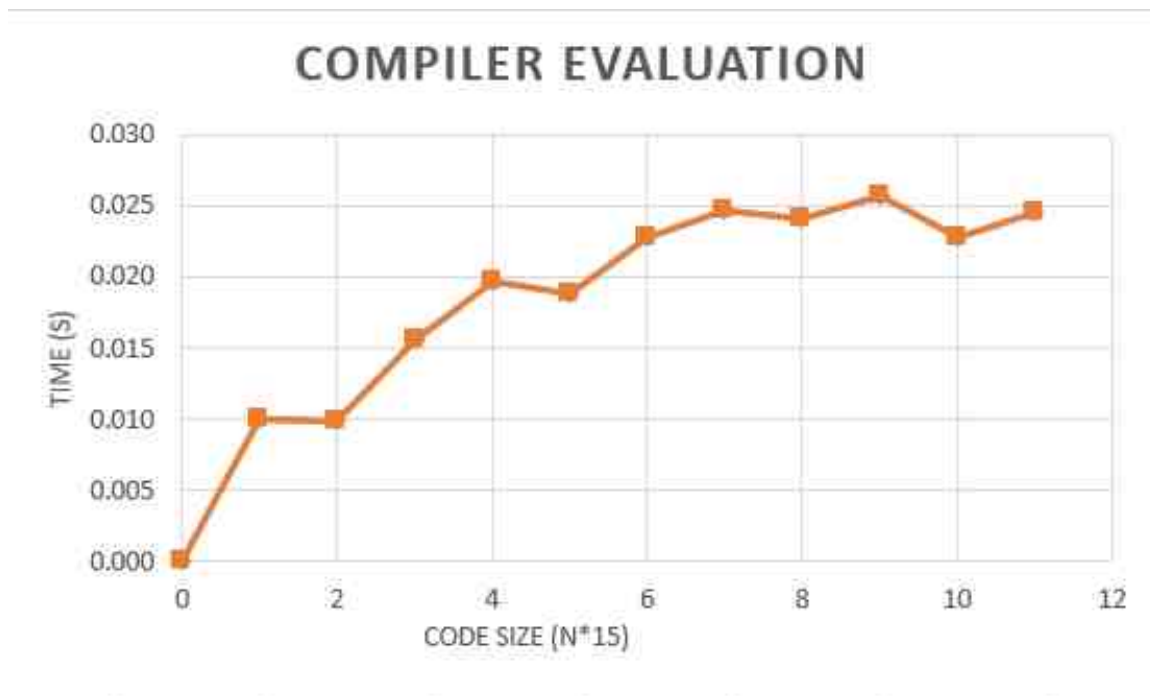


Figure 4.4: Compiler Evaluation

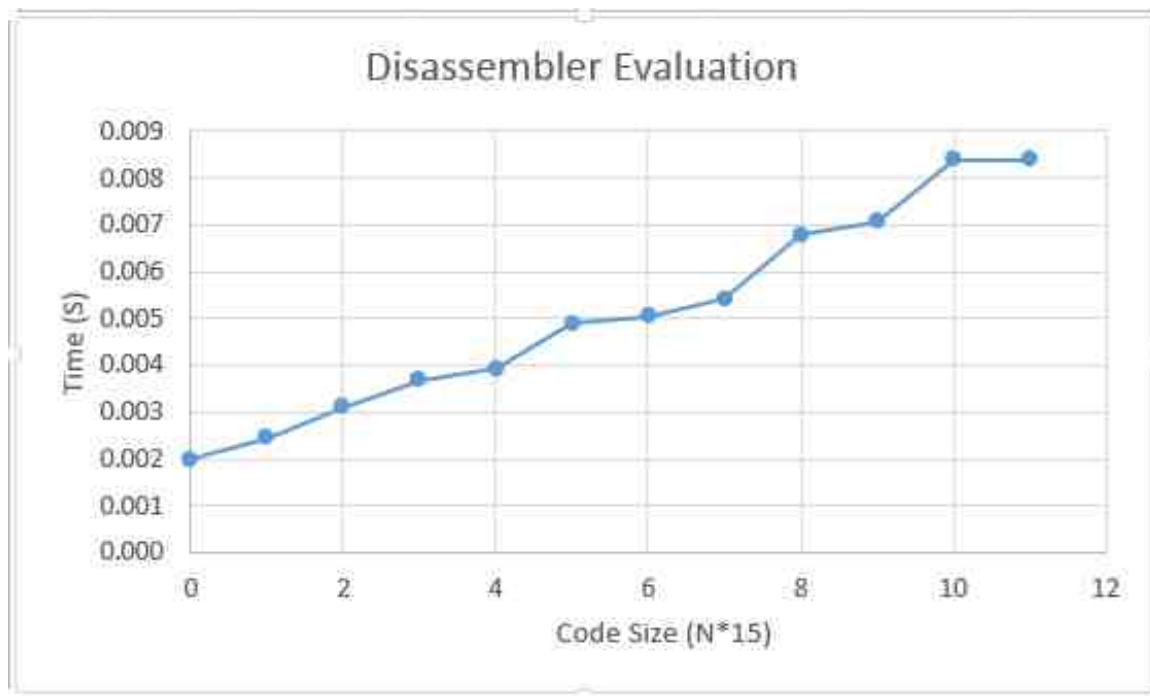


Figure 4.5: Disassembler Evaluation

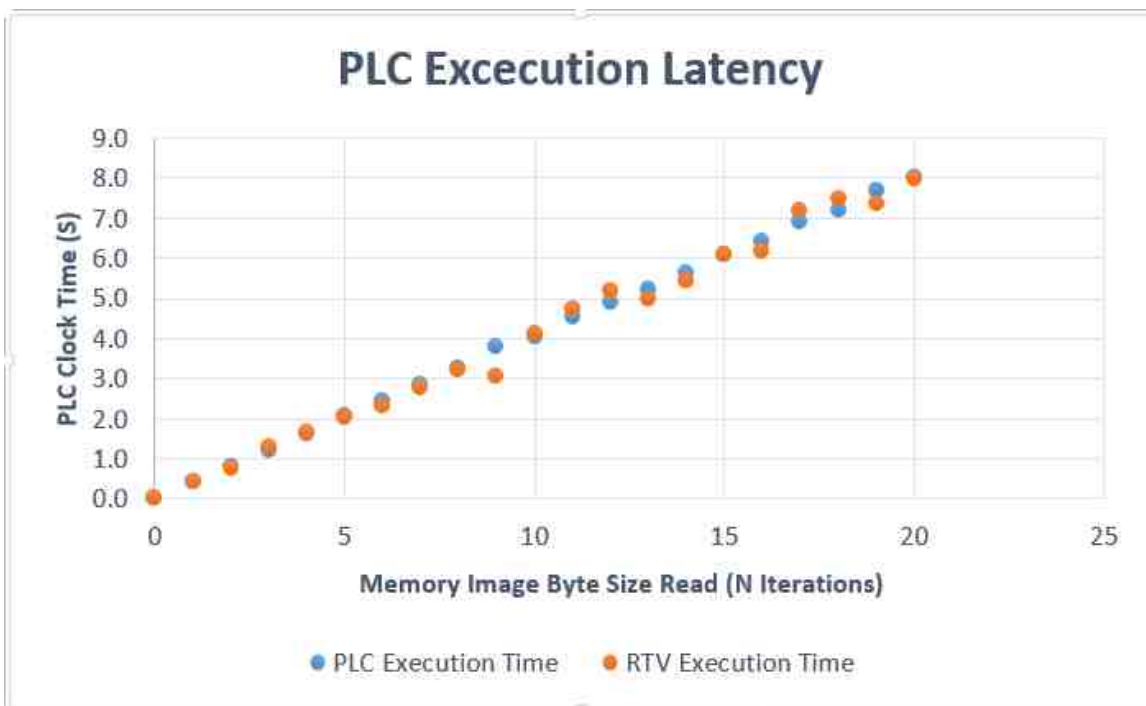


Figure 4.6: PLC Latency Evaluation

## CHAPTER 5

# Model Verification

### 5.1 Parallel Model Generation

To enable real-time system monitoring, RCR needs to generate the controller program models and perform the corresponding model checking procedures at a faster, or at least equal, pace than/to the actual evolution of the underlying physical system. Otherwise, if RCR gets behind the actual execution, RCR's outputs would be useless, because the operators would notice the actual malicious consequences of the attacker program before RCR could analyze them. To facilitate real-time model generation and formal verification, RCR makes use of several design and programming techniques to accelerate the whole process. First, given a PLC controller program, RCR implements the model generation procedures to run in parallel threads. To that end, given the recursive DFS-like generation process of the temporal execution graph, RCR assigns every recursive exploration call to an available thread in a global thread pool. Initially, the thread pool size is set to the number of the available cores in the system, and every time a recursive function returns its corresponding thread it is added back to the pool. Figure 5.1 shows the overview of the RCR model.

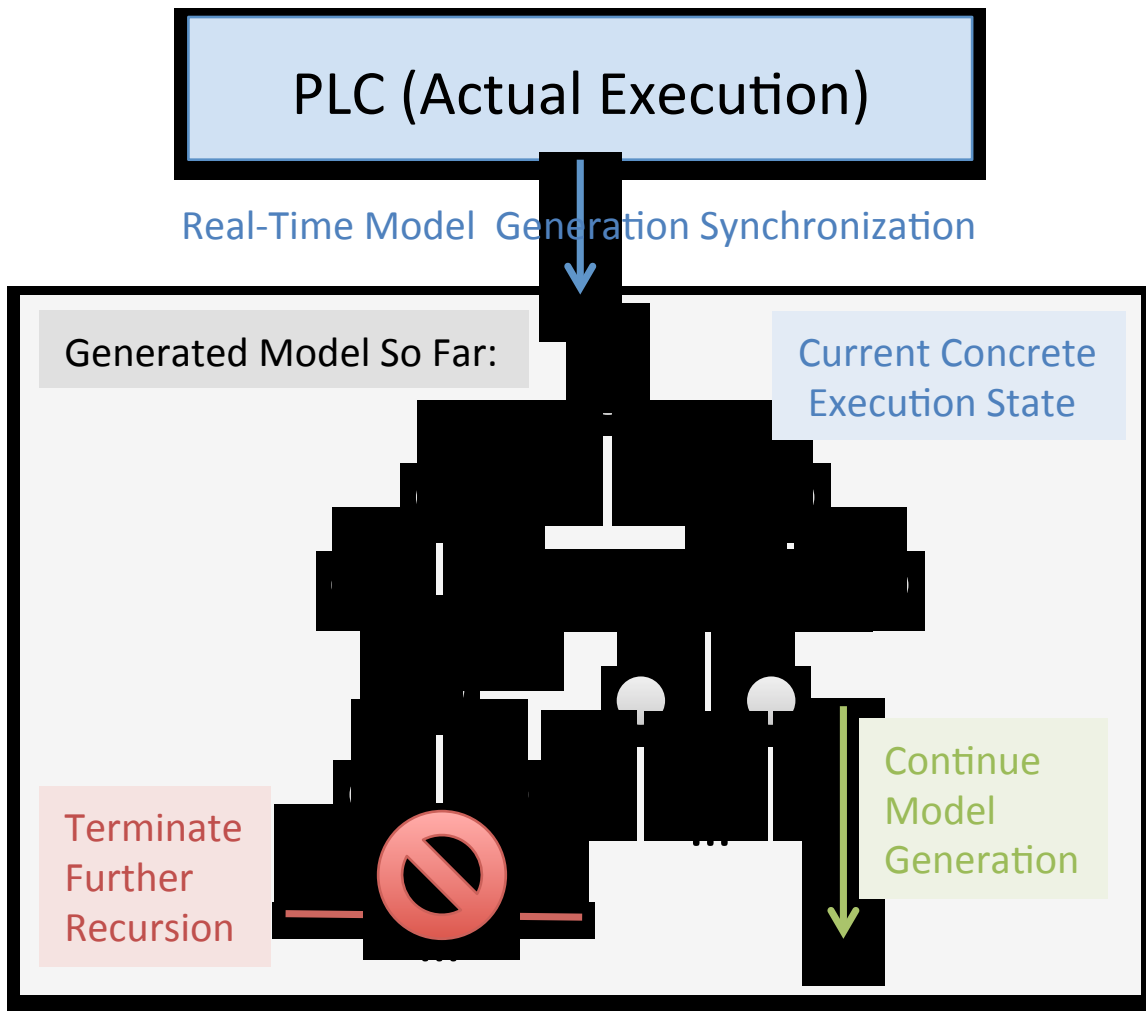


Figure 5.1: RCR Execution Overview

## 5.2 Symbolic Execution of PLC Code

The execution of the binary files on a PLC is done in cycles. These cycles are called scan cycles. Each scan cycle reads the the memory image of the PLC before each execution. With the set if sensor inputs, the PLC executes the control logic and creates a new set of output that correspond to the inputs. In our experiment, logic and input are analyzed using symbolic execution hence calling the cycle a symbolic scan cycle. PLC logic can be programmed in three different languages; Ladder

logic, Function Block Diagram and Statement List. Ladder logic and Function Block Diagram languages provide circuit-like diagrams and are more graphical. However, the underlying fundamental part is an assembly-level language. On the other hand, Statement List is purely an assembly-level language and enables easy binary analysis. Since the source code is compile to an executable binary file, our analysis is centered on these binary codes. This enables us to focus on the lower level analysis and hence make it possible to analyze binary files compiled using other languages. In order to perform binary analysis on the PLCs, we use and intermediate language, the Instruction List Intermediate Language. The main reason for an intermediate language is because of the PLC-specific features that make it difficult to leverage existing binary tools in our analysis.

ILIL is based on the Vine IL, used in the BitBlaze binary analysis platform we developed the Instruction List Intermediate Language (ILIL). An ILIL program is a list of top-level instructions followed by a list of function definitions. Each function corresponds to a single function block in the PLC program. The top-level code consists of the organization block, which is where each scan cycle begins execution. The organization block then calls function blocks in turn.

ILIL uses the two basic Vine types: registers and memories. A single register variable is used to represent each CPU register in a particular PLC architecture. These are implemented as bit vectors of 1, 8, 16, 32, and 64 bits. Memories are implemented differently in ILIL than in Vine. ILIL memories are mappings from hierarchical addresses to integers. Memory loads return the integer for a given address, and memory stores return a new copy of the memory with the specified location modified.

In addition to registers and memories, ILIL adds a third type: addresses. In Vine, memories are mappings between integers. This is reasonable as most architectures use 32- or 64-bit address registers. This isn't sufficient for PLCs, which use hierarchical addresses. A hierarchical address has several namespace qualifiers before the actual byte or bit address. For example, in Siemens PLCs, addresses have a single namespace qualifier called a memory area. In Allen Bradley, there are three namespace qualifiers: rack, group, and slot. ILIL addresses are essentially integer lists where the leftmost  $n$  entries represent the  $n$  namespace qualifiers. We also extend the memory type to include  $n$ . Thus, the ILIL statement  $mem := : mem_t$  (1 initializes an empty memory with a single namespace qualifier, where  $mem_t$  is the memory type defined in the ILIL grammar. Figure 5.2 shows the ILIL grammar. Figure 5.3 shows a sample transformation of Statement List into ILIL.

```

prog ::= inst*fun*
fun ::= ident(var){inst*}
inst ::= cjmp e,e,e | jmp e | label ident | ident := e
       | call ident(var=e) | ret | assert e
e ::= load(ident,addr) | store(ident,addr,int) | e binop e
   | unop e | var | val | (e)
binop ::= +, -, *, /, mod, &, &&, <<, ... (And signed versions.)
unop ::= - (Negate), ~ (Bitwise), ! (Boolean)
var ::= ident (: τ)
val ::= mem | addr | int (: τ)
mem ::= {addr ↦ int, addr ↦ int, ...}
addr ::= [int :: int :: ...]
τ ::= reg1_t...reg64_t | mem_t(int) | addr_t

```

Figure 5.2: Instruction List Intermediate Language grammar

Since we do not use the PLCs clock time, we used a data memory address to keep track of the times. During symbolic execution when a timer is checked, values at this

address will generate a fresh symbol. In the model-checking step, this symbol will be non-deterministic, meaning it will cause both paths to be explored if used in a branch condition.

Symbolic execution follows all possible paths of a single scan cycle of the program being executed. A Satisfiability Modulo Theory (SMT) solver enables us to follow only the feasible paths during the execution. We also enforced the execution of loops at a minimal number to reduce the load. The result of a symbolic execution is the symbolic scan cycle, which represents all possible executions of a particular scan cycle. In order to scale up our tool, we combine scan cycles to form a temporal graph that represents all the possible system executions. Figure 5.4 demonstrates the symbolic execution of

### 5.2.1 Model Checking

In order to check for violation of the PLC during the execution, we convert IL code to ILIL code. The ILIL is then symbolically executed, creating the output mappings between the path conditions and the symbolic output variable values. Using this mapping, we verify if the PLC IL program violates the specifications of the PLC that it is being executed on. These outputs are used to form the temporal execution graph (TEG). The TEG graph is then used to formally verify that the PLCs execution meets the power network security requirements. These requirements are defined as the linear temporal logic specifications which should not be violated.

### 5.2.2 Model Refinement

As the TEG is created, to void the path explosion challenge, we perform a few heuristics to refine the TEG graph [37]. This makes the formal analysis feasible. The atomic propositions representing the specifications are converted into linear temporal logic, these are added to the TEG as we recursively traverse it. For each iteration, we use the SMT solver to check if for each atomic proposition can take on true or false values or otherwise we assign concrete values. In case the state can take both values, the states are duplicated and we add the negated atomic proposition predicate to the original duplicate states existing predicates using a logical AND operator.

At this state, we obtain an abstract graph in which each state is represented by the atomic proposition values only. The concrete proposition value assignment helps in the model-checking state to determine whether the formula is satisfiable. If it doesn't, the model-checking engine comes up with a counter example that demonstrates how the security specification can be violated through a listed execution path. The HMI operator can look at these paths so as to resolve the problem or detect an attack if the code is injected maliciously by an adversary.



```

A I  0.5  ;; And input bit 5
= Q  0.1  ;; Store at output bit 1

0. // Initialize PLC state.
1. mem := {} : mem_t(1);           // Main memory.
2. I   := 0;                       // Input memory qualifier.
3. Q   := 1;                       // Output memory qualifier.
4. RLO := 1 : reg1_t;             // Boolean accumulator.
5. FC  := 0 : reg1_t;             // System status registers.
6. STA := 0 : reg1_t;
7. ...
8.
9. // A I 0.5
10. STA := load(mem, [I::0::0::0::5]);
11. cjmp FC == 0 : reg1_t,L1,L2;
12. label L1;
13. RLO := STA;
14. label L2;
15. RLO := RLO && STA;
16. FC  := 1 : reg1_t;           // Side effects.
17. ...
18.
19. // = Q 0.1
20. STA := RLO;
21. mem := store(mem, [Q::0::0::0::1], RLO);
22. FC  := 0 : reg1_t;           // Side effects.
23. ...

```

Figure 5.3: Sample IL To ILIL conversion

<b>Code Segment</b>	<b>Concrete execution</b>
input i;	Input? $\leftarrow$ 16
output o;	Output: 8
read("Input?", i);	
if (i<10)	
o = i+9;	
else	<b>Symbolic execution</b>
o = i/2;	Input? $\leftarrow$ 'a'
print("Output:", o);	Output:
	PC [a<10 ] $\rightarrow$ a+9
	PC [a>=10] $\rightarrow$ a/2

Figure 5.4: Symbolic Execution Example

## CHAPTER 6

# Covert Channel Communication

### 6.1 PHYCO Introduction

Apart from the security level developed at the core levels of the smart grid, there is a need to secure the communication infrastructure used in electric power systems [38]. As reported by the Industrial Control System Cyber Emergency Team (ICS-CERT), there is an increase in the number of cyber-physical attacks. The current security models in power control systems only consider explicit communications. In our research, we developed a novel covert channel that leverages physical substrates within a power system, to transmit information between compromised device controllers. In this chapter, we discuss the capabilities of PHYCO. Basically, using PHYCO we can enable compromised controllers that are far apart to coordinate an attack if they manipulate relays to modify the power network's topology. The performance of PHYCO has been evaluated using PLCs on a realistic simulated power grid. We have also shown how PHYCO can bypass existing intrusion detection sensors and physical inspections.

Based on the reported increase of cyber threats against critical cyber-physical infrastructures [13], a new trend that we predict is to implement synchronization

protocols to deploy coordinated intrusions in order to be effective against large-scale power grid infrastructures. Current security protocols have shown good performance with single point attacks. However, these current protocols and tools do not have the capabilities to handle coordinated attacks.

Efforts have been made to make a stealthy form of communication channels. Xiao, et al. [19] worked on a covert channel using copy-on-write-based memory deduplication implementations. These could be exploited by the attackers in virtualized environments specifically.

Castiglione, et al. [20] proposed stealth communication using spam emails. Khan, et al. [21] designed a cluster-based covert channel so as to evade disk investigation and forensics attempts. The covert channel avoids detection by not using the storage on the filesystem of the compromised system hence lesser chances to be detected by the detection tools. PHYCO avoids detection by the cyber security sensors to zero since it does not use the cyber assets, computer networks nor PLC links of the cyber physical infrastructure.

## 6.2 PHYCO Threat Model

Our model assumes that single point of entry have been compromised using several attack vectors like the example of Stuxnet [29]. Assuming that the adversaries have already compromised two computing host systems  $\mathbf{h}_s$  and  $\mathbf{h}_r$ . The two systems make efforts to communicate to each other in order to coordinate a large-scale distributed attack against the physical grid. Assuming the compromised machines are PLCs.

In this model,  $\mathbf{h}_s$  which we call the sending host, can send a message to the receiving host  $\mathbf{h}_r$ . Since the physical network communication is not part of the PHYCO

model, these compromised hosts need to be equipped with the necessary sensing and actuation capabilities. The sending  $\mathbf{h}_s$ , needs to be in charge of a PLC that has the capabilities to manipulate the power grids topology. This can be done through changing some of the parameters of the PLC data image. As the sending host sends the changes, the receiving port reads the data corresponding to the changes in real time.

In particular,  $\mathbf{h}_s$  needs to be in charge of a local power grid component control so that it could manipulate the power grid's topology and/or parameters whenever needed. On the other hand,  $\mathbf{h}_r$  has to receive real-time sensor measurements regarding the power grid's current status. These actuating and sensing capabilities are mutually dependent. That is, the required sensing capabilities on the receiving end  $\mathbf{h}_r$ , depend heavily on what actuation capabilities the sending party  $\mathbf{h}_s$  owns. As a case in point, if is sending message bits through the opening and closing of a power line switch,  $\mathbf{h}_r$  needs to sense the affected power system parameter, such as the current on a transmission line.

### 6.3 Message Transmission

In order to coordinate a large scale, PHYCO, compromises two PLCs. PLCs are the core components of the cyber-physical infrastructure. They control the power flow in the grid. Since the two compromised components (PLCs) cannot communicate to each other due to the firewall between them as shown in the figure 6.1, they take advantage of the power control capabilities. In order to transfer data, the sending PLC makes a small change in the power generation  $P\Delta$ . The change can be a decrease

or increase in the power generated along the line by a small amount that is used to represent a 1 or 0 bit.

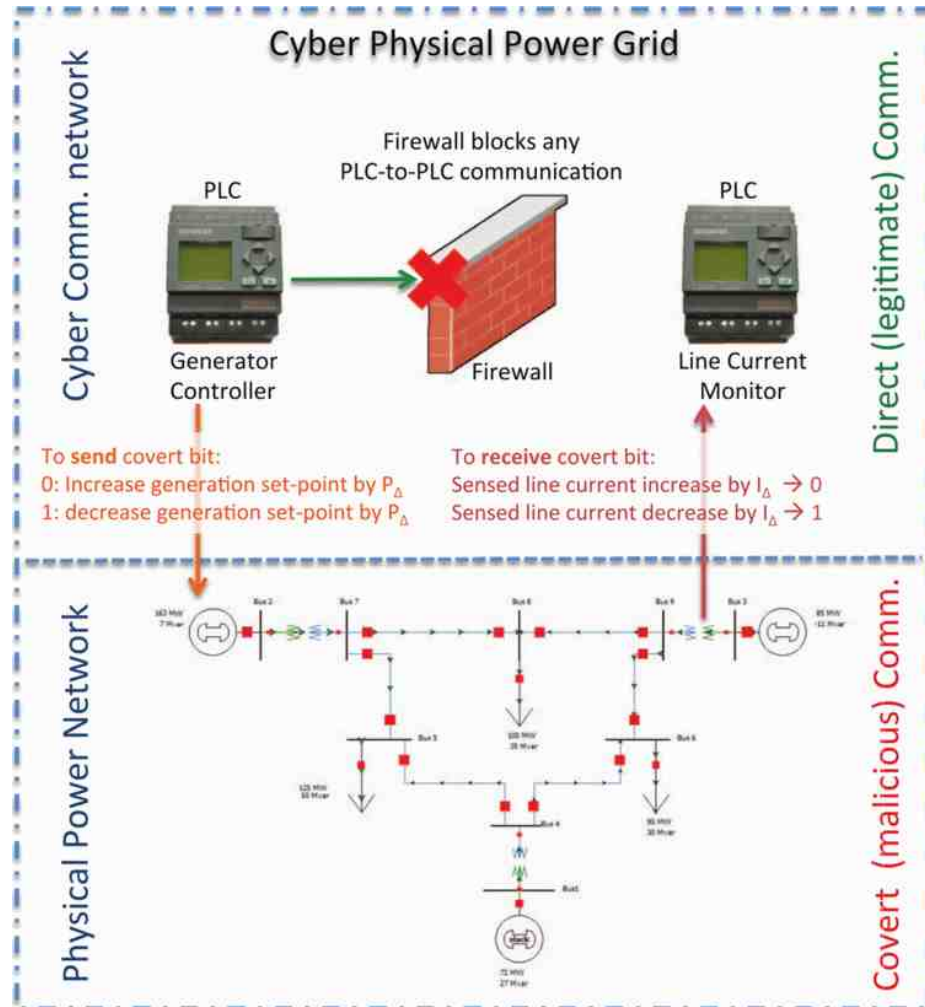


Figure 6.1: PHYCO Thread Model

As the power changes travel along the power distribution lines, the receiving compromised PLC listens and captures these power changes. The power changes are transferred to bits 1 and 0. In order to coordinate an attack, PHYCO uses the power flow equations to calculate potential effects. Since steady state power systems obey the law of conservation of power in any given network, we use these equation with

the set of collected bits data to determine the set of voltages and angles that satisfy power balance. We write the system state as  $f(x, u) = 0$  which is a vector representing the injection at each node in the system that could be performed to cause a terrible breakdown of the entire system using the programmable logic controller.  $x = [V, \theta]$  where  $\mathbf{V}$ , corresponds to a vector of voltage angles. We take as a vector of real power loads and the vector of reactive power loads is  $\mathbf{Q}_i$  The generator control specifications which define the limit of operation are also collected as the vector of controls,  $\mathbf{U}$ . The power flow problem is then transformed to The function  $f(x, u)$  can then be represented as

$$f_i^p = -P_i^g + P_i^l + \sum_{k \in C} |V_i||V_k|(G_{ik} \cos \theta_{ik} + B_{ik} \sin \theta_{ik}) \quad (1)$$

$$f_i^q = -Q_i^g + Q_i^l + \sum_{k \in C} |V_i||V_k|(G_{ik} \sin \theta_{ik} - B_{ik} \cos \theta_{ik}) \quad (2)$$

The equations represent the nonlinear problem which is referred to as power flow equations.

### 6.3.1 Message Transfer

Messages transmitted by the sender depend on the available actuators. We have two categories of the events that could result into creating a message. A sender  $\mathbf{h}_s$ , could cause changes in the power topology updates such as opening and closing a relay or a transformer tapping ratio modification. These actions modify the inter-connectivity of the power system components, and thereby modify the power flow equations. A second alternative is the sender  $\mathbf{h}_s$ , could change the values of the power parameters and control variables making it possible for the receiver to keep track of the modifications. These actions however do not change the power system

but do bring a change in some parameters. To satisfy the power flow equations, other parameters of the system will change as well and the message delivery will be accomplished if the receiving party  $\mathbf{h}_r$  happens to be measuring one of the indirectly updated system parameters.

### 6.3.2 Message Reception

The host  $\mathbf{h}_r$  on the receiving side of PHYCO would be in charge of reception and interpretation of the delivered signal. Considering the set of actions available for the sender, any action taken by  $\mathbf{h}_s$ , the received signal by the receiver depends on what point of the underlying power system resides as well as the parameter that it measures.

The receiver  $\mathbf{h}_r$ , initially needs to know the local effect of each action taken by the sender  $\mathbf{h}_s$ . During an offline phase before the message transfer,  $\mathbf{h}_r$  emulates each action in the current power system state and solves the power flow equations (Equations (1) and (2)) to find out how each possible action would impact the power parameter measured by  $\mathbf{h}_r$ . The whole procedure resembles the power system contingency analysis [39] that the control centers often perform every few minutes. However, the emulation involves the action set instead of the contingency list.

## 6.4 Data Transfer Reliability

As the sender sends data to the receiver  $\mathbf{h}_r$ , PHYCO ensures that the data is not corrupted by any noises. To achieve this, we utilize a periodic slice in which the sender sends data. The receiver will only be required to detect the interval slices that contain the data. For this approach to yield good performance, we need to set





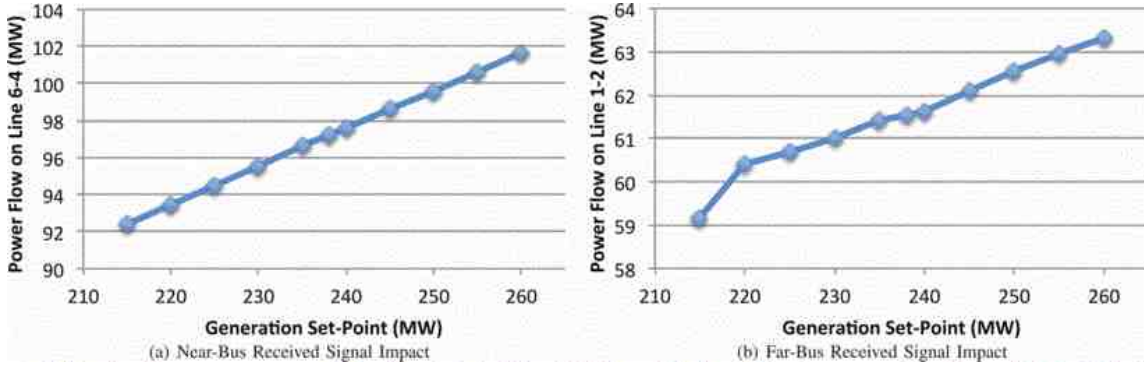


Figure 6.3: Received signal affection for decoding.

Action Set size	Message length	Block size	Convert Encoding
8	3	7	1.18 (sec)
16	11	15	3.00 (sec)
32	27	31	6.65 (sec)
64	59	63	14.03 (sec)
128	123	127	29.6 (sec)

Table 6.1: Received signal affection for decoding.

is one sending party and two receivers. In particular, the sending party  $\mathbf{h}_s$  resides in Bus 4 and controls and regulates the generation set-point of the corresponding generator. The receiver  $\mathbf{h}_{r,1}$  sits on a near Bus 6, where it measures the power flow on the transmission line that connects Bus 6 to Bus 4 of the power system. The receiver  $\mathbf{h}_{r,2}$  sits on a far Bus 1, where it measures the power flow on the transmission line that connects Bus 1 to Bus 2 of the power system

Figure 6.3 illustrates the results of how the actions taken by sending party  $\mathbf{h}_s$  results in power system variations sensed by the receiver  $\mathbf{h}_r$ . Figure 6.3(a)) shows the sensed measurement updates on the near-bus receiver. The receiver  $\mathbf{h}_{r,1}$  receives the signals more clearly as the measurement modifications are higher due to the geographical vicinity to the sender  $\mathbf{h}_s$ . In particular, for every 5 MW change due to the PHYCO message transfer results in approximately 1 MW of change in the receiver-side power

flow measurements. On the other hand, the far-bus receiver  $\mathbf{h}_{r,2}$ 's signal reception is not very clear due to the impact level on the receiver-end signal, i.e., the power flow on Bus 1, is lower because of the distance between the sender  $\mathbf{h}_s$  and the receiver  $\mathbf{h}_r$ . The received signal changes on average 0.4 MW for every subsequent action pair by the sender  $\mathbf{h}_s$ . The changes on the receiver side are not linear (see Figure 6.3(b)); therefore, the receiver  $\mathbf{h}_r$  has to pre-calculate the received signal changes before communicating through the covert channel.

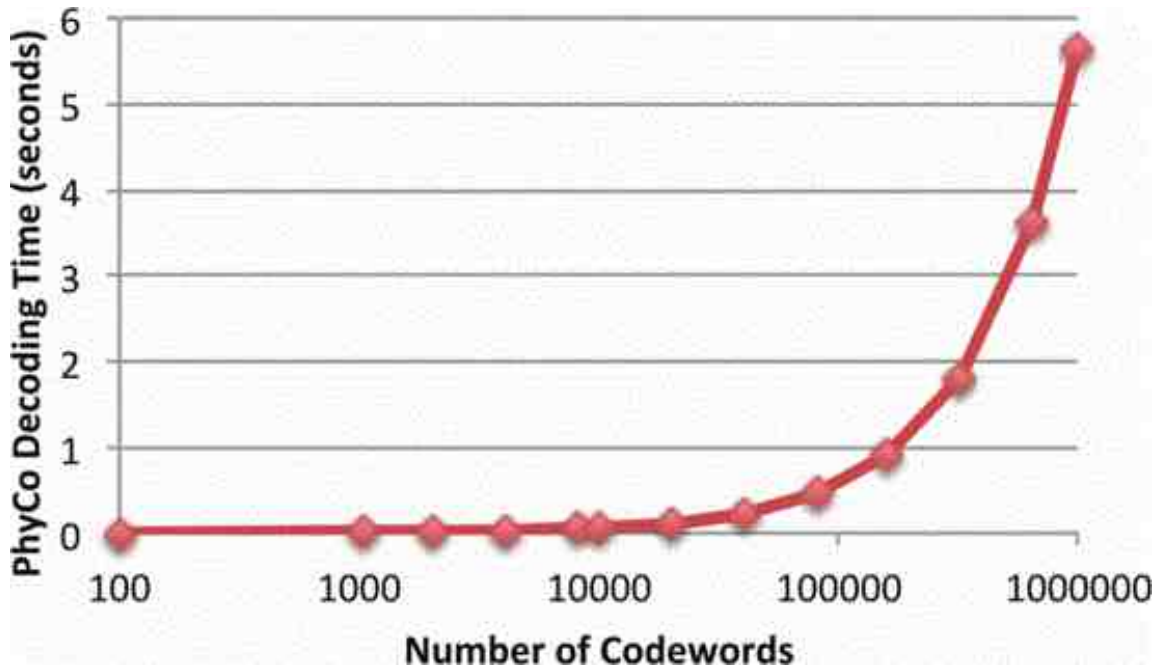


Figure 6.4: PHYCO decoding time requirements.

The relatively low signal changes on the far-bus receiver encourages using PHYCO's error correction module. We implemented a prototype of the message encoding and decoding scheme. Our implemented encoding scheme could tolerate any single symbol transfer error. The toleration level could additionally be configured based on the channels noise level. Table 6.1 shows the final results for encoding various action

set cardinalities, covert message lengths and PHYCO block lengths. According to our results, the sender side  $\mathbf{h}_s$ , for a reasonable covert channel action set size of 32, PHYCO takes approximately 7 seconds to complete encoding of a given message before its transfer. When the signals are received, they are decoded by the receivers. Figure 6.4 shows the results of various number of codeword transfers. Decoding 1M codewords takes the receiver in PHYCO less than 6 seconds that is very suitable for practical uses of the proposed channel.

## CHAPTER 7

# Conclusion

In this thesis, we have presented a formal analysis and approach for modeling and the verification of cyber-physical systems. The thesis has given a detailed review on the current state of the cyber-physical systems. This review, involves a detailed discussion of the cyber-physical system threats and the counter measures that have been developed. In the work, we have specifically shown an example of a well tailored stealth attack on the critical cyber-physical infrastructure. Using formal methods, we have additionally shown how the core components of the cyber-physical infrastructure can be presented as formal expressions. The main goal of the work is to provide tools that could protect the cyber-physical infrastructure effectively.

### 7.1 Ongoing and Future Work

Due to the dynamic and smarter evolution of cyber-physical systems, there is need for more and better security tools. The current growth of applications that are integrated into the power grid make it more difficult to detect any malware intrusions into the critical cyber-physical infrastructure. Our goal in this research is to enhance

the security of cyber-physical systems in such a way that the cost of tailoring an attack becomes exponentially more expensive than the effects from the attack.

# Bibliography

- [1] S. Alexzander and I. Anbumalar, “Recent trends in power systems (wireless power transmission system) and supercapacitor application,” in *Sustainable Energy and Intelligent Systems (SEISCON 2011), International Conference on*, July 2011, pp. 416–420.
- [2] A. Teixeira, S. Amin, H. Sandberg, K. Johansson, and S. Sastry, “Cyber security analysis of state estimators in electric power systems,” in *Decision and Control (CDC), 2010 49th IEEE Conference on*, Dec 2010, pp. 5991–5998.
- [3] A. Monticelli, “Electric power system state estimation,” *Proceedings of the IEEE*, vol. 88, no. 2, pp. 262–282, Feb 2000.
- [4] E. Hryniewicz, A. Milik, and D. Polok, “Programmable logic controller based on reconfigurable logic,” in *Design and Technology of Electronics Packages, (SIITME) 2009 15th International Symposium for*. IEEE, 2009, pp. 227–231.
- [5] M. Mollah and S. Islam, “Towards iee 802.22 based scada system for future distributed system,” in *Informatics, Electronics Vision (ICIEV), 2012 International Conference on*, May 2012, pp. 1075–1080.
- [6] D. Gaushell and H. Darlington, “Supervisory control and data acquisition,” *Proceedings of the IEEE*, vol. 75, no. 12, pp. 1645–1658, Dec 1987.
- [7] F. F. Wu, “Power system state estimation: a survey,” *International Journal of Electrical Power & Energy Systems*, vol. 12, no. 2, pp. 80–87, 1990.
- [8] C. A. Bejan, M. Iacob, and G. Andreescu, “Scada automation system laboratory, elements and applications,” in *Intelligent Systems and Informatics, 2009. SISY’09. 7th International Symposium on*. IEEE, 2009, pp. 181–186.
- [9] C. Patil, H. Sonawane, and K. Patil, “Overview of scada application in thermal power plant,” *International Journal of Advanced Electronics and Communication Systems*, 2014.
- [10] R. K. Chauhan, M. Dewal, and K. Chauhan, “Intelligent scada system.”

- [11] Y. Wang, “sscada: securing scada infrastructure communications,” *International Journal of Communication Networks and Distributed Systems*, vol. 6, no. 1, pp. 59–78, 2011.
- [12] N. Falliere, L. O. Murchu, and E. Chien, “W32.Stuxnet Dossier,” Symantic Security Response, Tech. Rep., Oct. 2010.
- [13] T. M. Chen, “Stuxnet, the real start of cyber warfare?[editor’s note],” *Network, IEEE*, vol. 24, no. 6, pp. 2–3, 2010.
- [14] U. De, “Smart grid cyber security: Vol. 2, privacy and the smart grid,” 2010.
- [15] J. Gómez, C. Gil, N. Padilla, R. Baños, and C. Jiménez, “Design of a snort-based hybrid intrusion detection system,” in *Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted Living*. Springer, 2009, pp. 515–522.
- [16] J. Safarik, P. Partila, F. Rezac, L. Macura, and M. Voznak, “Automatic classification of attacks on ip telephony,” *Advances in Electrical and Electronic Engineering*, vol. 11, no. 6, pp. 481–486, 2013.
- [17] J. E. Diaz-Verdejo, P. Garcia-Teodoro, P. Muñoz, G. Maciá-Fernández, and F. De Toro, “A snort-based approach for the development and deployment of hybrid ids,” *Latin America Transactions, IEEE (Revista IEEE America Latina)*, vol. 5, no. 6, pp. 386–392, 2007.
- [18] G. Kurundkar, N. Naik, and S. Khamitkar, “Network intrusion detection using snort,” *International Journal of Engineering Research and Applications*, vol. 2, no. 2, pp. 1288–1296, 2012.
- [19] J. Xiao, Z. Xu, H. Huang, and H. Wang, “A covert channel construction in a virtualized environment,” in *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, ser. CCS ’12. New York, NY, USA: ACM, 2012, pp. 1040–1042. [Online]. Available: <http://doi.acm.org/10.1145/2382196.2382318>
- [20] A. Castiglione, A. D. Santis, U. Fiore, and F. Palmieri, “An asynchronous covert channel using spam,” *Computers & Mathematics with Applications*, vol. 63, no. 2, pp. 437–447, 2012. [Online]. Available: <http://dx.doi.org/10.1016/j.camwa.2011.07.068>
- [21] H. Khan, M. Javed, S. A. Khayam, and F. Mirza, “Designing a cluster-based covert channel to evade disk investigation and forensics,” *Computers & Security*, vol. 30, no. 1, pp. 35–49, 2011. [Online]. Available: <http://dx.doi.org/10.1016/j.cose.2010.10.005>



- [22] X. Miao, X. Chen, X.-M. Ma, G. Liu, H. Feng, and X. Song, “Comparing smart grid technology standards roadmap of the iec, nist and sgcc,” in *Electricity Distribution (CICED), 2012 China International Conference on*, Sept 2012, pp. 1–4.
- [23] E. F. Camacho, T. Samad, M. Garcia-Sanz, and I. Hiskens, “Control for renewable energy and smart grids.”
- [24] Y. Mo, T.-H. Kim, K. Brancik, D. Dickinson, H. Lee, A. Perrig, and B. Sinopoli, “Cyber-physical security of a smart grid infrastructure,” *Proceedings of the IEEE*, vol. 100, no. 1, pp. 195–209, Jan 2012.
- [25] T. Basso and R. DeBlasio, “Advancing smart grid interoperability and implementing nist’s interoperability roadmap,” in *Proc. NREL/CP-550-47000, Grid-Interop Conf*, 2009.
- [26] W. Enck, P. Traynor, P. McDaniel, and T. La Porta, “Exploiting open functionality in sms-capable cellular networks,” in *Proceedings of the 12th ACM conference on Computer and communications security*. ACM, 2005, pp. 393–404.
- [27] Y. Zhang, C. Guo, R. Chu, G. Lu, Y. Xiong, and H. Wu, “Ramcube: exploiting network proximity for ram-based key-value store,” in *Proceedings of the 4th USENIX conference on Hot Topics in Cloud Computing*. USENIX Association, 2012, pp. 5–5.
- [28] L. Garcia, H. Senyondo, S. E. McLaughlin, and S. A. Zonouz, “Covert channel communication through physical interdependencies in cyber-physical infrastructures,” in *2014 IEEE International Conference on Smart Grid Communications, SmartGridComm 2014, Venice, Italy, November 3-6, 2014*, 2014, pp. 952–957. [Online]. Available: <http://dx.doi.org/10.1109/SmartGridComm.2014.7007771>
- [29] D. Beresford, “Exploiting siemens simatic s7 plcs,” 2011.
- [30] A. Clark, L. Bushnell, and R. Poovendran, “A passivity-based framework for composing attacks on networked control systems,” in *Communication, Control, and Computing (Allerton), 2012 50th Annual Allerton Conference on*. IEEE, 2012, pp. 1814–1821.
- [31] G. Sandaruwan, P. Ranaweera, and V. Oleshchuk, “Plc security and critical infrastructure protection,” in *Industrial and Information Systems (ICIIS), 2013 8th IEEE International Conference on*, Dec 2013, pp. 81–85.
- [32] S. McLaughlin and P. McDaniel, “Sabot: specification-based payload generation for programmable logic controllers,” in *Proceedings of the 2012 ACM conference on Computer and Communications Security*, 2012, pp. 439–449.

- [33] P. Lee and L. Lai, “A practical approach of smart metering in remote monitoring of renewable energy applications,” in *Power Energy Society General Meeting, 2009. PES '09. IEEE*, July 2009, pp. 1–4.
- [34] A. Lunkeit, T. Voss, and H. Pohl, “Threat modeling smart metering gateways,” in *Smart Objects, Systems and Technologies (SmartSysTech), Proceedings of 2013 European Conference on*, June 2013, pp. 1–5.
- [35] I. Rouf, R. Miller, H. Mustafa, T. Taylor, S. Oh, W. Xu, M. Gruteser, W. Trappe, and I. Seskar, “Security and privacy vulnerabilities of in-car wireless networks: A tire pressure monitoring system case study,” in *Proceedings of the 19th USENIX Conference on Security*, ser. USENIX Security’10. Berkeley, CA, USA: USENIX Association, 2010, pp. 21–21. [Online]. Available: <http://dl.acm.org/citation.cfm?id=1929820.1929848>
- [36] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, and S. Savage, “Experimental security analysis of a modern automobile,” in *Proceedings of the 2010 IEEE Symposium on Security and Privacy*, ser. SP '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 447–462. [Online]. Available: <http://dx.doi.org/10.1109/SP.2010.34>
- [37] K.-K. Ma, K. Y. Phang, J. S. Foster, and M. Hicks, “Directed symbolic execution,” in *Proceedings of the 18th International Conference on Static Analysis*, ser. SAS’11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 95–111. [Online]. Available: <http://dl.acm.org/citation.cfm?id=2041552.2041563>
- [38] V. Gungor, D. Sahin, T. Kocak, S. Ergut, C. Buccella, C. Cecati, and G. Hancke, “Smart grid technologies: Communication technologies and standards,” *Industrial Informatics, IEEE Transactions on*, vol. 7, no. 4, pp. 529–539, Nov 2011.
- [39] S. Varshney, L. Srivastava, M. Pandit, and M. Sharma, “Voltage stability based contingency ranking using distributed computing environment,” in *Power, Energy and Control (ICPEC), 2013 International Conference on*. IEEE, 2013, pp. 208–212.
- [40] S. B. Wicker and V. K. Bhargava, *Reed-Solomon codes and their applications*. John Wiley & Sons, 1999.