PHYSICS FOR THE SAKE OF SECURITY, SECURITY FOR THE SAKE OF PHYSICS

By

LUIS GARCIA

A dissertation submitted to the

School of Graduate Studies

Rutgers, The State University of New Jersey

In partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

Graduate Program in Electrical and Computer Engineering

Written under the direction of

Professor Saman Zonouz

And approved by

_____

_____

_____

_____

New Brunswick, New Jersey

OCTOBER, 2018

ABSTRACT OF THE DISSERTATION

Physics for the Sake of Security, Security for the Sake of Physics

by Luis Garcia

Dissertation Director:
Professor Saman Zonouz

In the current cyberwarfare climate, industrial control systems (ICS) are increasingly becoming focal points of security research as they interconnect, monitor, and control safety-critical processes. ICS comprise a class of cyber-physical systems (CPS) across a wide range of domains, including but not limited to the electric power grid, factory automation, biomedical applications, as well as nuclear reactors. As the interconnectivity and accessibility of ICS system components expands, the attack surface for such systems expands as well. Because these ICS control safety-critical physical processes, there is a need for security solutions that have the physical dynamics integrated into the design process in order to ensure safe operation.

This dissertation investigates the security and verification of ICS at different levels of abstraction. The goal is to bridge the gap between practical security analyses and sound theoretical approaches to verifying cyber-physical systems. In particular, we propose to not only leverage the physical properties of an ICS for security purposes, but to also provide fine-grained hybrid systems modelling of embedded cyber-physical systems such as a programmable logic controllers (PLCs).

First, this dissertation introduces novel and practical security and verification solutions for ICS that leverage the cyber-physical interdependences between the *cyber* components and the underlying *physical* system. This dissertation then explores the feasibility of utilizing formal methods in the context of complex ICS control processes. Finally, this dissertation introduces a balanced approach to cyber-physical intrusion detection that enforces control behavioral integrity of a distributed ICS by integrating physical state-estimation into control-flow monitoring of the associated software.

Acknowledgements

This work would not have been possible without the financial support of the Graduate Assistance in Areas of National Need (GAANN) fellowship program from the Department of Education as well as my initial support from the Energy Department's Advanced Research Projects Agency-Energy (ARPA-E). I am especially grateful to Dr. Athina Petropulu, Dr. Raheem Beyah, Dr. Narayan Mandayam, Dr. Kate Davis, Dr. Matt Davis, Dr. Andre Platzer, Dr. Stefan Mitsch, Dr. Andrew Sogokon, Dr. Khalil Ghorbal, Dr. Rakesh Bobba, Dr. Mehdi Javanmard, as well as Dr. Ahmad Sadeghi, all of whom have provided me with invaluable guidance and support as I embark on this journey to a career in academia.

I would like to express an endless amount of gratitude and appreciation to my committee chair and Ph.D. advisor, Dr. Saman Zonouz, whose guidance and encouragement elevated my work to a level beyond anyone's expectations. He provided me the mental support when I needed it the most, and helped me stay the course whenever it seemed all hope was lost. He has set an incredible example of a successful life-work balance for my labmates and I, and he has always prioritized our mental health. I appreciate every extra mile that you went for all of us, and I hope to do the same for my students in the future.

Finally, I would like to thank my family, friends, and girlfriend for their incredible support throughout this entire chapter in my career. Several events of adversity during this time have proven to be some of the more trying and stressful times of my life, and I would not have been able to endure these instances without your help. Furthermore, I would like to also thank you for your patience and understanding. There are times

where my research had engulfed me and not allowed me to be there to support you when you needed it the most. This experience has taught me, among other things, the value of time, and I hope that I have graduated to be a better partner, a better friend, and a better son.

Dedication

This dissertation is dedicated to the memory of my brother, Rafael Garcia, whose life is carried on in the memories of those who loved him as well as through his son, Mason.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Industrial control systems (ICS) interconnect, control, and monitor safety-critical industrial environments, e.g., water treatment plants, factory automation, and the electric power grid. They represent a subset of cyber-physical systems (CPS) whose attack surfaces are increasing due to the trend of networking components to the internet European Network and Information Security Agency (ENISA) (2011). Although the evolution of such systems are necessary for the advancement and integration of the associated technologies in today's society, it often germinates convenience at the expense of safety and security.

ICS are composed of multiple layers of abstraction, where at the highest level there exists a supervisory control and data acquisition (SCADA) architecture and at the lowest level you have the sensors and actuators interfacing with the physical components of the system. At the core of several ICS control system architectures lie embedded CPS, so-called programmable logic controllers (PLC), that act as the cyber-physical interface between the supervisory control and the physical components of the ICS. PLCs are digital computing devices used for automating industrial electromechanical processes. They control the state of the output ports based on signals received from the input ports and stored programs, and operate typically under hard environmental conditions, such as excessive vibration and high noise Bolton (2015); Erickson (1996). PLCs control standalone equipment and discrete manufacturing processes. Their logical behavior with regard to inputs and outputs can be programmed by the operator.

Because of their criticality in ICS, PLCs are commonly the target of nation-state ICS malware. The Stuxnet worm Falliere et al. (2010) against Iranian nuclear uranium enrichment facilities and BlackEnergy crimeware F-Secure Labs (2016) against

the Ukrainian train railway and electricity power industries were exemplars of ICS-targeted malware. Furthermore, these malware were designed with the ability of evading traditional cyber security detection mechanisms and caused catastrophic failures that impacted the physical processes controlled by the ICS.

ICS-targeted malware such as Stuxnet and BlackEnergy motivated several security solutions. In particular, the Trusted Safety Verifier, TSV McLaughlin et al. (2014), presented a bump-in-the-wire defense against such attacks by verifying any code that was loaded onto PLCs against some safety specifications. This solution was implemented with legacy PLCs due to the vast amount of such devices deployed in the industry. Replacing so many devices with secure devices would be impractical and too expensive. Having a cost-effective bump-in-the-wire solution is a much more flexible approach. Although the intention of such a solution is to provide a practical approach to verifying safety properties of the application-layer code, it relies on the correct safety properties being specified by the system designer. Furthermore, the solution is currently limited to properties specified in linear temporal logic, LTL, which has severe limitations for verifying real-time systems and is more suited for falsification than verification of safety-properties. In order to generate the correct and invariant safety properties of an ICS, one needs to model the CPS as a hybrid system.

Hybrid systems are systems whose continuous evolution–e.g., the physical equations of a system–is determined by discrete-state transitions. In the context of ICS, the "cyber" actions represent the discrete-state transitions that affect the underlying physical evolution of the system. For instance, a PLC that opens a water valve will change the flow rate from zero to some constant value. The change in flow rate will then change the physical equations associated with the water tank level sensors. A sound and relatively-complete logic that has been used to model and formally verify cyber-physical systems as hybrid systems is differential Dynamic Logic (d$\mathcal{L}$) Platzer (2010). The d$\mathcal{L}$ approach comes with a formal programming language to model hybrid systems. The formalized models are called hybrid programs. The language extends the imperative constructs of Dynamic Logic Harel et al. (2000) with additional constructs to encode constrained continuous evolutions. The model reads as a list of actions that the system has to

perform to achieve its goals. The models can then be proven using an automatic theorem prover, KeymaeraX Fulton et al. (2015). Although this modelling approach has proven to be a sound and relatively complete approach to modelling complex CPS, the models are typically simplified models that typically abstract the control mechanisms of the hybrid system. For instance, the model for a train control system used to prevent collisions with trains on the same track will abstract the braking force to some constant value without considering the underlying variables of the actually breaking system Platzer and Quesel (2008).

The aforementioned solutions for both security and verification of ICS provide either heuristic methods to defend against attacks or sound, verified models of a coarse-grained abstraction of the hybrid dynamics. Given these limitations, this dissertation aims to answer the following questions:

- *How can the physical dynamics of a system be exploited to provide a proper vulnerability assessment of the ICS?*
- *If controllers cannot be trusted in the context of ICS, can the physical interdependence between components and their side-channels provide means of security and verification?*
- *How can the gap between practical security practices and theoretical approaches to safety verification be bridged in the context of ICS?*

This dissertation investigates the security and verification of ICS at all levels of abstraction. First, novel approaches to cyber-physical vulnerability assessment of ICS are explored to motivate physical awareness in cyber security solutions. Second, analyses of side-channels in industrial processes are instrumented for the implementation of novel cyber-physical security and verification approaches. Third, the dissertation explores the feasibility of utilizing d$\mathcal{L}$ in the context of complex ICS control processes by providing high-level abstractions of PLC actuation. The dissertation then presents a solution that will provide a more fine-grained hybrid systems analysis of the actual code that is loaded onto PLCs in the context of ICS. Finally, this dissertation presents

a practical implementation for monitoring a distributed ICS as a hybrid system for the purpose of intrusion detection. The preliminary results are as follows.

Harvey: attacking PLCs with a physical model aware rootkit. Trustworthy operation of industrial control systems (ICS) requires secure code execution on the embedded programmable logic controllers (PLCs). The controllers monitor and control the underlying physical plants such as electric power grids and continuously report back the system status to human operators. chapter 2 presents Harvey[1], a PLC rootkit that implements a physics-aware stealthy man-in-the-middle attack against cyber-physical power grid control systems. Harvey sits within the PLC's firmware below the control logic and modifies control commands before they are sent out by the PLC's output modules to the physical plant's actuators. Harvey replaces legitimate control commands with malicious, adversary-optimal commands to maximize the damage to the physical power equipment and cause potentially large-scale failures. To ensure system safety, the operators often fetch system parameter values from PLC devices and observe the status of the power system. To conceal the maliciously caused anomalous power system behavior from operators, Harvey also intercepts the sensor measurement inputs to the PLC device. Harvey simulates the power system with the legitimate control commands (which were intercepted/replaced with the malicious ones), and calculates/injects the sensor measurements that operators would expect to see. We reverse engineered the firmware binary of a widely spread PLC, an Allen Bradley PLC and evaluated Harvey on a real-world electric power grid test-bed. The results are very promising and empirically prove Harvey's deployment feasibility in practice nowadays.

Malicious fill pattern detection in additive manufacturing. Additive Manufacturing is an increasingly integral part of industrial manufacturing. Safety-critical products, such as medical prostheses and parts for aerospace and automotive industries are being printed by additive manufacturing methods with no standard means of verification. chapter 3 presents a scheme of verification and intrusion detection that is

[1]Harvey Dent (Two-Face) is a fictional super-villain adversary of the superhero Batman. The right half of his face looks normal/benign, unlike the hideously scary left side of his face.

independent of the printer firmware and controller PC. The scheme incorporates analyses of the acoustic signature of a manufacturing process, real-time tracking of machine components, and post production materials analysis. Not only will these methods allow the end user to verify the accuracy of printed models, but they will also save material costs by verifying the prints in real time and stopping the process in the event of a discrepancy. We evaluate our methods using three different types of 3D printers and one CNC machine and find them to be 100% accurate when detecting erroneous prints in real time. We also present a use case in which an erroneous print of a tibial knee prosthesis is identified. This work shows how the physical properties of an ICS can be leveraged for defense purposes in a practical, domain-specific fashion.

Formal verification of hybrid controller logic for transient stability in hybrid systems. chapter 4 presents a method for proving properties of systems governed by non-linear ordinary differential equations with mixed polynomial and trigonometric functions under semi-algebraic evolution constraints whose proofs require semi-algebraic invariants. Proofs of such safety properties as well as the associated invariant generation typically require an axiomitisation of such transcendental functions by introducing fresh variables and eliminating non-polynomial non-linearities. Although such techniques allow for the automatic generation of semi-algebraic invariants and proofs of associated safety properites, certain systems may specify safety properties that require access to the previously axiomatised state variables. In order to prove the safety properties of such systems while maintaining the soundness of previous approaches, we present an automatic approach that exploits properties of the taylor series approximations for such transcendental functions. We further present a case study in which we apply our method to prove the safety of a hybrid systems representation of an electric power grid system with respect to its transient stability. This work provides a promising preliminary results that will lead to future directions of exploration for more providing safety constraints for more complex CPS as well as for developing compositional models that will provide safety guarantees across different levels of hybrid states as well as different levels of abstraction. In the subsequent chapter, we will discuss the verificaiton of the

controllers that are abstracted for the simplified electric power grid system modeled in this work.

Hybrid PLC program translation for verification. As has been discussed, programmable Logic Controllers (*PLCs*) provide a prominent choice of implementation platform for safety-critical industrial control systems applications. Formal verification provides ways of establishing correctness guarantees, which can be quite important for such safety-critical applications. But since PLC code does not include a model of the system plant, their verification is limited to "shallow" discrete properties.

In chapter 5, we, thus, start the other way around with hybrid programs that include continuous plant models in addition to discrete control algorithms. Even "deep" correctness properties of hybrid programs can be formally verified in the theorem prover KeYmaeraX that implements differential dynamic logic d$\mathcal{L}$for hybrid programs. After verifying the hybrid program, we now present an approach for translating hybrid programs into PLC code. The new tool HyPLC implements this translation of discrete control code of verified hybrid program models to real PLC controller code and vice versa. We define the translation of hybrid programs that are written in differential dynamic logic and whose models are validated through model compliance methods to programmable logic controller (PLC) code. HyPLC can also translate existing PLC code into the discrete control actions for a hybrid program given an additional input of the continuous dynamics of the system to be verified. This approach allows for the generation of real controller code while preserving, by compilation, the correctness of a valid and verified hybrid program. PLCs are known to be the cyber-physical interface for safety-critical industrial control applications, and HyPLC serves as a pragmatic tool for bridging formal verification of complex cyber-physical systems at the algorithmic level of hybrid programs with the execution layer of concrete PLC code implementations.

Control behavior integrity for distributed cyber-physical systems. Cyber-physical control systems, such as industrial control systems (ICS), are increasingly

targeted by cyberattacks. Such attacks can potentially cause tremendous damage, affect critical infrastructure or even jeopardize human life when the system does not behave as intended. Cyberattacks, however, are not new and decades of security research have developed plenty of solutions to thwart them. Unfortunately, many of these solutions cannot be easily applied to safety-critical cyber-physical systems. Further, the attack surface of ICS is quite different from what can be commonly assumed in classical IT systems.

chapter 6 presents Scadman, a system with the goal to preserve the *Control Behavior Integrity* (CBI) of distributed cyber-physical systems. By observing the system-wide behavior, the correctness of individual controllers in the system can be verified. This allows Scadman to detect a wide range of attacks against controllers, like programmable logic controller (PLCs), including malware attacks, code-reuse and data-only attacks. We implemented and evaluated Scadman based on Secure Water Treatment (SWaT)– a real-world water treatment testbed for research and training on ICS security. Our results show that we can detect a wide range of attacks–including attacks that have previously been undetectable by typical state estimation techniques–while causing no false-positive warning for nominal threshold values.

chapter 7 discusses possible future research directions along with the conclusion of this dissertation.

U

Chapter 2

Attacking PLCs with a Physical Model Aware Rootkit

2.1 Introduction

Industrial control systems (ICS) interconnect, control and monitor industrial environments such as electrical power generation, transmission and distribution, chemical production, oil and gas refining and transport, and water treatment and distribution. In recent years, ICS have received considerable attention due to security concerns originated by the trend to connect ICS to the Internet European Network and Information Security Agency (ENISA) (2011). In particular, critical infrastructures connected to and controlled by ICS substantiate these security concerns. Nevertheless, the ICS market is expected to grow to $10.33 billion by 2018 TechNavio (2014).

Nation-state ICS malware such as the Stuxnet worm Falliere et al. (2010) against Iranian nuclear uranium enrichment facilities and BlackEnergy crimeware F-Secure Labs (2016) against the Ukranian train railway and electricity power industries show that targeted attacks on critical infrastructures can evade traditional cybersecurity detection and cause catastrophic failures with substantive impact. The discovery of Duqu Chien et al. (2011) and Havex Rrushi et al. show that such attacks are not isolated cases as they infected ICS in more than eight countries.

ICS security has been traditionally handled using network security and information technology (IT) practices Zhu and Sastry (2010). ICS security goals, however, differ from traditional IT security goals due to additional requirements and conditions of operation. The interconnection of the physical world and virtual world, bridged by cyberphysical systems (CPS), is a unique feature of ICS compared to traditional IT infrastructures. And unlike most traditional IT systems, high availability is critical for ICS. A process failure can have fatal consequences threatening human lives and

resulting in immense financial loss.[1]

ICS are monitored and operated in a centralized fashion: embedded CPS, known as programmable logic controllers (PLC), are connected to a central control terminal (human-machine interface HMI) through which a human operator can supervise the system. PLCs are digital computing devices used for automating industrial electromechanical processes. They control the state of the output ports based on signals received from the input ports and stored programs, and operate typically under hard environmental conditions, such as excessive vibration and high noise Bolton (2015); Erickson (1996). PLCs control standalone equipment and discrete manufacturing processes. Their logical behavior with regard to inputs and outputs can be programmed by the operator.

The main goal of sophisticated attackers is to remain stealthy from ICS operators. In particular, the HMI's view of the system should not indicate any effect caused by attacks. For this, the adversary can either compromise and manipulate the HMI itself, or launch a more sophisticated attack on PLCs. A prominent example of HMI related attacks is the infamous Stuxnet Falliere et al. (2010). However, HMIs are often based on commodity computer systems for which a wide variety of security solutions exist, including anti-virus software, security enhanced operating systems, run-time attack protections, and many more. This makes the HMI an unattractive attack target. On the other hand, although many PLC related attacks have been published in recent years Beresford; Schuett (2014); Klick et al. (2015); Brüggemann and Spenneberg (2015), they have limitations with regard to stealthiness and result in obvious effects, such as disrupting the operation of PLCs Schuett (2014). Other attacks operate on the PLC's application level (called *control logic*), which allows the operator to detect their presence through the PLC's remote management capabilities Klick et al. (2015); Brüggemann and Spenneberg (2015).

[1]ICS are also not what is usually considered Internet of Things (IoT) as there are substantial differences (cf. subsection 2.8.1).

Goals and Contributions. In this paper we present a novel class of *stealthy* PLC attacks that we refer to as Man-in-the-PLC. Our exploit intercepts the PLC's input and output values, provides an arbitrary view of the system to the control logic (i.e., the program running on the PLC), and simulates a semantically correct system state towards the central control unit while changing the actual system state. We provide the following main contributions:

- We present a novel attack class on industrial control systems: a cyber-physical attack which is completely invisible to the control center of an ICS.
- We reverse engineered the central control loop mechanism of a widely deployed Allen Bradley 1769-L18ER-BB1B CompactLogix 5370 L1 Rev. B PLC.
- We developed a prototype implementation of HARVEY, and tested and evaluated it on an Allen Bradley PLC. Allen Bradley is one of the most used ICS suppliers in the United States.
- We evaluate our attack in a real power grid test environment.

We would like to stress that our main contribution and novelty of our rootkit lies in its physics-awareness. This makes our solution more general than all solutions published before, including real world attacks like Stuxnet Falliere et al. (2010). To be able to implement and evaluate our prototype, we had to reverse engineer the PLC to gain the required insight into its inner working, in particular, the PLC's control of input and output lines, and the connection between the firmware and control logic programs.

Following the standard responsible disclosure policies, we have taken necessary steps and have contacted the vendor, Allen Bradley, informing them about the possibility of such malicious exploits against their controllers. Allen Bradley gave us clearance to publish our findings.

The remainder of the paper is structured as follows. First, we provide the reader with a background on industrial control systems in general and programmable logic controllers in particular, and present our system model and adversary model in section 2.2. In section 2.3, we explain the general concept of HARVEY before providing details on the physics-aware data manipulations of our attack in section 2.4. In section 2.5, we describe how we reverse engineered the firmware of an Allen Bradley PLC

and implemented our rootkit. In section 2.6, we provide an extensive evaluation for our physics-aware PLC rootkit against a real power grid test-bed. section 2.7 provides a review of related work in the area of ICS security. We discuss our findings and possible mitigation strategies in section 2.8, and conclude in section 2.9.

2.2 Background and System Model

In this section, we provide basic knowledge for the rest of the paper. We provide detailed information about industrial control systems (ICS) and programmable logic controllers (PLC), and we define the system model and adversary model we will consider throughout the paper.

2.2.1 Background

An ICS is a distributed system which is composed of physical components, like sensors and actuators, which interact with the physical system (e.g., power grid) and cyber components (e.g., cyber-networks and servers). Although ICS are largely self-contained, interfaces exist through which external components can interact with the systems. For instance, a human operator can monitor the systems state and influence it through a human-machine interface (HMI). Most PLCs are connected to the ICS via an Ethernet network, and hence, often indirectly connected to the Internet. It is also quite common that PLCs are directly connected to the Internet Klick et al. (2015).

Centralized operation and maintenance is an essential feature of ICS. An operator can program and monitor PLCs and the applications running on them remotely, i.e., retrieve the status of a PLC and re-program it over the network. The information which can be retrieved from the PLC contains, among others, the control logic applications on the PLC. All applications, including their source code and further meta information, can be loaded from a PLC.[2]

[2]This enables the operator to detect malicious modification of the PLC on the control logic layer.

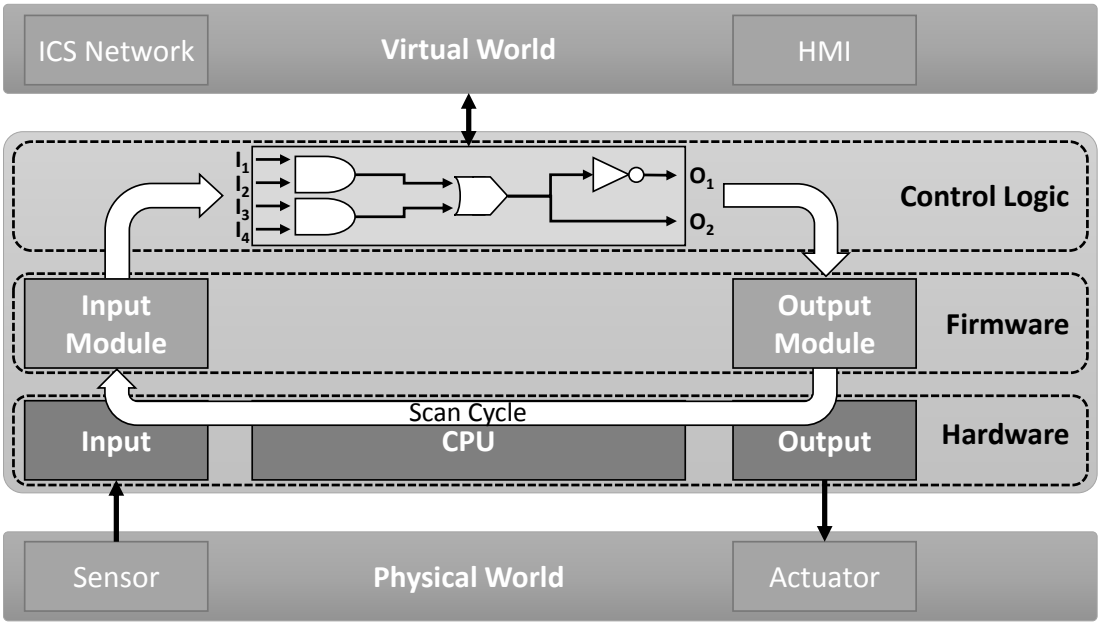


Figure 2.1: PLC Architecture

Programmable Logic Controllers (PLC). Programmable logic controllers (PLC) are cyber-physical systems that are used to control industrial appliances. PLCs have input and output modules to interact with the physical world. They can translate physical inputs, mostly current on a wire, into digital values and vice versa. Connected to physical appliances such as sensors and actuators, PLCs can convert sensor readings into digital values, process the readings with the built-in computing unit, and forward the outputs to the physical world. The logical behavior of PLCs, i.e., the processing of the input data, is programmable.

Such control loops can be either local, i.e., the inputs and outputs are handled by a single PLC, or distributed, i.e., the inputs are read by one PLC and forwarded over the network to another PLC.

The two main software components of a PLC, control logic and firmware, are shown in Figure 2.1. The firmware is acting as a kind of operating system (OS). The firmware contains, among other functionality, modules to read and write the inputs and outputs of the PLC from/to the physical world. These modules can be seen as drivers for the I/O hardware. Control logic programs can be considered the PLC's application layer. The firmware provides services to read from input lines and write to output lines of

the PLC. They are used by the control logic to program the behavior of the PLC. The control logic programs are executed repeatedly in fixed intervals, called scan cycles. The control logic program reads input values from memory and stores the output values to memory. The underlying firmware is responsible for the interchange of these updated values to and from the PLC's general purpose input/output (GPIO) ports, i.e., the interface to the physical system, as well as the reporting mechanisms of the PLC, i.e., the LED display on the device and the HMI. The scan cycle is illustrated by the white arrows in Figure 2.1.[3]

For instance, a pump can increase the pressure in a pipe. In order to have a constant pressure in the pipe the pump must be active until the predefined pressure is reached. Whether the desired level has been reached is determined based on sensor readings. However, the sensor and the pump are not directly connected. The sensor measurements are read by the PLC through its input lines. The value is then processed by the control logic, and the result is translated back into the physical system to steer the pump.

This approach allows for high flexibility, e.g., the sensor and actuator (pump) might not operate in close proximity and the sensor measurements have to be sent to the PLC controlling the actuator. This can be done cost-efficiently through existing computer networks, even over the internet. Furthermore, processing data in the cyber network allows for more complex relations between sensor measurements and actions.

2.2.2 System Model

In this paper, we mainly consider large distributed ICSs that are operated in a centralized manner. Figure 2.2 shows an abstract view of our system model in which a physical system is operated from a central control terminal. The control terminal provides a HMI that allows an operator to monitor the system and interact with the system (by sending control commands). The connection between the control terminal in the cyber world of the ICS and the physical world is provided by PLCs. PLCs capture the physical system's state by reading measurements from sensors. Additionally, PLCs

[3]PLCs can be programmed with multiple independent control logic applications which are executed sequentially within in each scan cycle.

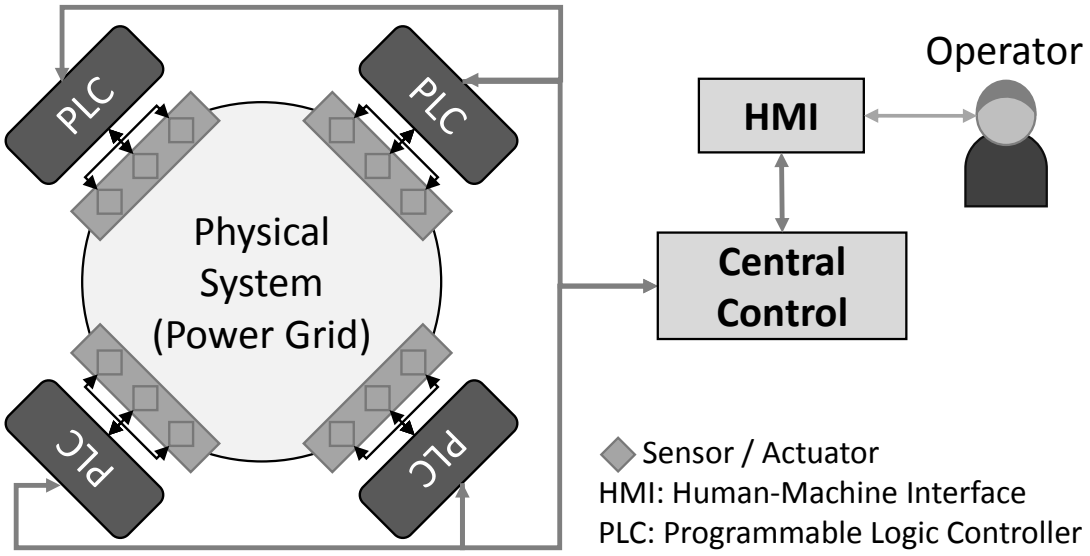


Figure 2.2: System Model

control the system's actuators, based on both the control actions generated by their local control logic and the control commands sent from the operator.

2.2.3 Adversary Model and Assumptions

Stealthiness. The main goal of the adversary is to launch a stealthy attack on an industrial control system (ICS).[4] Stealthy means that the attack does not cause unintentionally observable effects. For instance, sensor readings analyzed by a system operator or automated tools should align with what they are expected to be. Real world examples like Stuxnet Falliere et al. (2010) have shown that stealthy attacks have a more enduring impact on a system than a short attack which will rapidly break down a system.

The attack exploits the circumstance that in real world systems the operator's view of the entire system is limited to the information provided by the HMI, i.e., he cannot directly observe the physical system and detect attack effects through an out-of-band channel like visual contact. This limitation can be due to different reasons, e.g., in large and distributed systems the operator is physically not capable of observing the entire system, or the system operates in a hazardous environment and for safety reasons the

[4]Obviously the attacker could also use HARVEY's capabilities to cause very visible attacks if he chooses to.

operator only has remote access to the system.

PLC-only Attack. We assume that the adversary compromises only PLCs and no other components of the ICS, hence the name Man-in-the-PLC. In particular, we do not assume that the adversary has manipulated the human-machine interface (HMI), e.g., to hide suspicious activities from the operator.[5] Besides an operator observing the HMI, the ICS might include security mechanisms like SCADA-specific[6] intrusion detection systems (IDS) that monitor the system Zhu and Sastry (2010). We assume the adversary cannot compromise (all) monitoring entities (i.e., IDS systems) in an ICS in order to hide an attack.

Furthermore, ICS components like HMI's are usually based on commodity hardware and software, e.g., a workstation PC running Windows operating system. Security solutions for those systems already exist, e.g., anti-virus software and automated software update solutions, increasing the probability for detection of an attack.

Physical Model Extraction. We assume the adversary has knowledge about the inner workings of his target and uses this information to build a model of the correct behavior of the target to be able to hide suspicious effects of the attack. The adversary can get the required information, for instance, through an insider, or through preceding information gathering attacks Klick et al. (2015). Although physical model extraction is outside of the scope of this paper, it is worth noting that monitoring and management systems of the ICS can be leveraged to extract information about the physical model. For instance, in power systems, energy management systems (EMS) are used to control the power grid infrastructure. An EMS is a collection of computer-aided tools used by operators of electric utility grids to monitor, control and optimize the performance of generation and transmission systems. A typical suite of EMS applications includes several components that feed sensor measurements into state estimation systems, contingency analysis software, optimal power flow control analysis software, as well as an HMI. Hence, the power system topology information can be extracted through insider

[5]Stuxnet for instance utilized a compromised HMI to hide itself from the operator Falliere et al. (2010).

[6]SCADA: supervisory control and data acquisition.

intruders (e.g., Stuxnet Falliere et al. (2010)) or EMS compromises. Additionally, unlike in purely-cyber settings, the physical power system and its topology is often exposed to outside world; therefore, physical system reconnaissance is relatively simpler.

2.3 Harvey: Model-aware Rootkit

The central property of our rootkit HARVEY is the fact that it takes into account the physical topology of the industrial control system (ICS) it infects. This gives HARVEY unique capabilities. Most importantly, it allows our rootkit to be stealthy. HARVEY is completely invisible to the ICS's virtual world. This means that the effects of attacks launched by HARVEY can neither be detected by human operators monitoring system measurements nor by security tools like intrusion detection systems (IDS) monitoring the ICS network. This makes HARVEY uniquely powerful and goes beyond attacks known today.

The idea of our model-aware rootkit is to infect the firmware of a programmable logic controller (PLC), allowing HARVEY to control all inputs and outputs of the PLC. The control logic program gets access to the PLC's input values through the firmware from the physical world, processes them, and then provides outputs that are forwarded to the physical world through the firmware. The control logic can also interact with other cyber components in the industrial control system (ICS) over network.

Because HARVEY lives in the firmware layer and intercepts the control and information flow of the firmware, it is completely transparent to the control logic. Each output which is passed from the control logic to the firmware is captured by HARVEY and can be changed, e.g., to maximize the effects of the attack. Similarly, HARVEY can change the input values seen by the control logic arbitrarily, e.g., to hide effects of its attack.

HARVEY only compromises the PLC's firmware, hence, it cannot be detected by the operator's PLC management tools. In contrast, malware that operates on the control logic level can be detected through the PLC's remote management capabilities. Control logic malwares need to rely on additional techniques or assumptions to hide themselves from the operator. Stuxnet compromised the operators workstation to hide

itself Falliere et al. (2010); Brüggemann and Spenneberg rely on the observation that they can cause the operator's remote management software to crash by manipulating meta-data stored on the PLC Brüggemann and Spenneberg (2015).

Although our Man-in-the-PLC rootkit cannot be detected directly by the operator, it could still be detected indirectly through the effects it causes. To prevent unintended, possibly suspicious effects in the ICS, HARVEY does not change input and output values arbitrarily. Instead, our rootkit acts according to a model of the target system which ensures that the operator's view of the system stays consistent with his expectations. This means that if the malware's goal is to increase the pressure in a pipe to damage it, it is not sufficient to activate the pump by setting the output of the PLC accordingly. A sensor would measure the increasing pressure and would alert the operator or trigger an automatic safety mechanism. Hence, for the attack to be successful, the malware must also ensure that sensor readings presented to the operator are not suspicious, e.g., hide the increasing pressure.

Since our rootkit uses a model to manipulate inputs *and* outputs of a PLC in a coordinated fashion, it can present a normal operation view towards the cyber world while manipulating the physical world.

Model. The model according to which our malware is operating can be created and/or obtained in different ways. HARVEY essentially makes use of the same models that are used to control the underlying physical platform legitimately. However, the malware optimizes the control commands for an adversarial objective function.

For attacks on simple systems the model would be simple, too, and can be created manually. For more complex systems the model can be created with automated tools.[7] The attacker's advantage is that he does not need to have a comprehensive model of the entire system. She only needs a model for parts of the system her attack will operate in. For instance, if the attacker aims to damage a specific pipe in a large plant, she only needs a model covering those components that her attack will effect, which might be as few as a single pump and a single sensor. Additionally, the model can also

[7]Related work focusing on modeling industrial control systems is described in section 2.7.

incorporate the deployed (if any) intrusion detection sensors to ensure the malicious control commands do not trigger the alerts.

In section 2.6, we will present our evaluation results of our model-aware malware on a real power grid. It shows that manipulations in the physical world can be hidden from the cyber world.

PLC Infection. Our rootkit works by compromising a PLC's firmware, which the attacker can achieve in different ways. The attack can use the built-in firmware update mechanism to replace the firmware on a PLC. Depending on the PLC model, firmware updates are not secure against manipulations.[8] Firmware updates can be deployed over the network for most PLCs. Hence, PLCs which are reachable by the attacker over the network can be compromised directly.

The attacker can also compromise PLCs locally. Many PLCs allow firmware updates from local media such as SD cards. Additionally, the attacker can use hardware interfaces like JTAG[9] to connect to the PLC and manipulate its firmware. An attack has recently been presented that shows the ease of JTAG implantation FitzPatrick and King (2015). In section 2.5 we describe in detail our attack on a PLC through its JTAG interface.[10]

Lastly, if the previous attack methods are not available, the adversary could facilitate run-time attacks, e.g., network exploits. To the best of our knowledge, there are no PLCs available that have run-time attack mitigation techniques deployed. This means that attacks like code injection are likely to succeed on PLCs. Remote code execution vulnerabilities on PLCs have been issued just this past year, e.g., CVE-2016-0868, CVE-2016-5645 and all vulnerabilities associated with the FrostyURL vulnerability, such as CVE-2015-6490, CVE-2015-6492, CVE-2015-6491, CVE-2015-6488, and CVE-2015-6486. However, even in the presence of protection mechanisms, the developments in the desktop and server world have shown that attackers will find new means of

[8] We will elaborate on our findings on firmware update security in section 2.5.

[9] Joint Test Action Group (JTAG) is an IEEE standard for testing and debugging integrated circuits.

[10] Note that the goal of our work is to show the effectiveness of physics-aware malware, providing novel infection vectors were not the scope of our efforts.

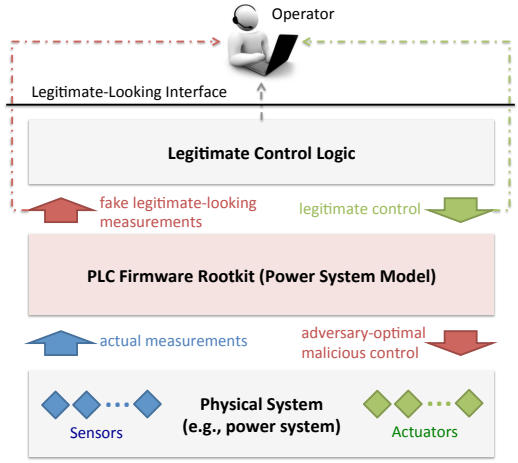


Figure 2.3: HARVEY Two-Way Data Manipulation Attack

circumventing these protection mechanisms. Code reuse attacks, like return oriented programming (ROP), have already been applied on embedded systems Francillon and Castelluccia (2008a).

2.4 Physics-Awareness

This section explains how HARVEY manipulates the control actuation actions and sensor measurements within a PLC.

Figure 2.3 shows how HARVEY manipulates data streams to damage the physical plant while ensuring the operators see what they would expect to see based on their inputs to the system. HARVEY performs the attack through manipulation of data in both directions: *i)* control commands sent by the operators and/or the control logic on the PLC to the actuators deployed on the physical plant; and *ii)* sensor measurements from the deployed sensors to the operators. The control commands are corrupted to damage the physical system and cause system failures, whereas the sensor measurements are corrupted to evade the operator detection of the caused failures.

2.4.1 Control: Malicious Plant Actuation

In real-world control systems, e.g., power systems, the system dynamics continuously change due to various factors, e.g., electricity consumption changes by the civilians. Such changes require updated control of the physical system to ensure it maintains its

core functionalities. For instance, in the power grid control systems, the generation set-points (control commands) are updated dynamically by the controllers to ensure the amount of power that is consumed equals the generated amount by the generators.

The control of the physical plant is often performed based on the models of the underlying physical dynamics, e.g., fluid dynamics for the water networks and Kirchhoff laws for the power systems. The models encode the correlations between the system parameters due to the physics laws caused by inter-component connections and dependencies. The most popular generic modeling follows linear dynamical state-based formalism $\dot{x} = f(x, u)$, where

$$f(x, u) = Ax + Bu + \omega \tag{2.1}$$

$$y = Cx + \epsilon. \tag{2.2}$$

The matrices A, B, C represent the dynamics of the physical system, x is the state vector of the system, y is the sensor measurements sent to the PLC, and u are the control outputs from the PLC to the actuators. ω and ϵ encode the noise in system dynamics and the sensor measurements, respectively. The PLC control logic receives the sensor measurements y and calculates the corresponding optimal control commands u^* to maximize a domain-specific objective function $u^* = \arg\max_u \text{obj}(x, f)$, e.g., the minimum total generation cost in power systems. The calculated control commands are sent to the PLC output modules through its firmware facilities.

HARVEY sits within the firmware and intercepts the output module write requests, and replaces them with malicious control output u_m^*. It calculates the malicious control outputs through similar logic as the legitimate controller with only one difference: it calculates the control outputs that minimize the corresponding objective function $u^* = \arg\min_u \text{obj}(x, f)$ to maximize the damage on the physical plant. HARVEY writes the calculated commands to the corresponding PLC output ports that are connected to the physical plant actuators. The corrupted control commands drive the plant towards unsafe states leading to potential physical failures, e.g., power system blackouts.

It is noteworthy that HARVEY's malicious control output calculation can also incorporate additional constraints that may be introduced to evade the detection of the attack. For instance, in a water plant, the constraints may avoid an extreme increase of the pump rotation speed because of its potentially noticeable noise. Additionally, without the loss of generality, the linear model mentioned above can be replaced with either simpler or more complicated models used by the PLC control logic depending on the specific control system domain (see section 2.6 for an empirical power system case study.)

2.4.2 Monitoring: Sensor Data Corruption

The control command manipulation attack, discussed above, causes unsafe physical plant states, but it does not contain the necessary amount of stealth for practical real-world deployment. Hence, it can be easily detected by the control system operators, who continuously monitor the real-time physical plant state on the human-machine interface (HMI) screens. The HMI screens are frequently updated automatically by fetching the PLC's memory for the most recent physical system sensor measurements that have been delivered to the PLC's input module.

To evade the detection, HARVEY intercepts the sensor measurements on the PLC and replaces them with fake values that would make the underlying physical system status look normal to the operators. Stuxnet implemented an innovative system status record-and-replay attack. The worm would record the plant dynamics for 13 days before it injected malicious control logic on the PLC. Once the attack started on the plant (malicious control logic execution on the PLC), Stuxnet would replay the recorded data stream back to the operator screens.

Such record-and-replay attacks would work in specific control system plants with static and stationary low-pace dynamics such as uranium enrichment, which was Stuxnet's target. However, the sole record-and-replay attack would not be practical and can be easily detected if used in typical control systems with high-pace dynamics such as a power grid. In power grids, the operators constantly change the system configuration/-topology and parameter values as a result of many external unpredictable factors, such

as real-time demands by the end-point electricity consumers and real-time climate for renewable energy sources such as solar and wind generation plants. Therefore, a replay of a previously recorded sensor measurement stream back to the operators is very unlikely to match exactly the expected status of the power system following the operator's most recent control commands.

HARVEY addresses the challenge above in highly dynamic control system environments through real-time and lightweight physical plant simulation within the PLC firmware. HARVEY takes the legitimate controller commands that either the operator or the legitimate controller logic on the PLC issues to be sent to the output modules and actuators. HARVEY then simulates the physical plant dynamics by solving the corresponding plant models (e.g., Equation 2.1). Through the simulation of the physical system, HARVEY essentially calculates how the power system would "look" if the legitimate control commands would really be sent to the deployed actuators. HARVEY replaces the actual sensor measurements with their corresponding simulated fake values before they are written to the PLC memory. The following PLC memory reads by the operators' HMI software would be receiving the fake measurements. Hence, the HMI screens would show a legitimate-looking physical system state to the operators.

The fabricated PLC memory values are used as sensor measurements by the legitimate control logic that is developed by the operators and runs on top of the malicious PLC firmware. Consequently, the legitimate control logic will calculate control commands that satisfy the operators' expectations on the HMI screens. It is noteworthy that HARVEY does not replace the legitimate control logic execution. Instead, it runs its malicious code in parallel to the legitimate control logic, and hence the outputs from both executions are calculated and used for different purposes (i.e., for faking the physical system appearance and damaging its actual components).

2.4.3 Distributed Monitoring and Control

In practice, a large-scale control system is often maintained by a set of distributed PLCs, each in charge of their assigned local "zone". As an example, in power systems, the electricity grid is typically partitioned into several sub-areas each maintained by a

separate controller Gómez-Expósito et al. (2016). In water plant facilities, the water treatment is often performed in a sequence of several serial phases such as chlorination, pH control, filtration, and disinfection. Individual phases are usually operated by separate control logic programs either all on the same PLC or each sitting on a separate PLC. For real-time monitoring and control, each PLC takes the monitoring and control responsibility of its associated zone independently such that its local sensor measurements suffice for its zonal control operations.

The distributed monitoring and control paradigm is traditionally employed to ensure real-time and reliable operation; however, it can be leveraged by the PLC firmware attacks such as HARVEY to ensure its stealthiness against large-scale control systems even when one or just a few of PLCs are infected. Put in other words, to perform an attack against a large-scale platform, the adversaries do not have to compromise all the controllers to maintain stealth.

2.5 Harvey Implementation

This section describes our rootkit implementation for an Allen Bradley 1769-L18ER-BB1B CompactLogix 5370 L1 Rev. B. It is noteworthy that the attack implementation is explained specifically for the PLC model above. The PLC software/hardware architectures are fundamentally similar across various vendors. Hence, the proposed techniques can be generalized to other widely-used industrial PLCs.

HARVEY deeply interferes with the core functionality of the PLC's firmware. This interference allows complex behavior manipulations of the PLC, which is required for stealthy control system attacks as described in the previous section. Since the firmware of PLC is not openly available, we had to reverse engineer it as the first step of our prototype implementation. Most techniques in this multi-step process are known but nonetheless technically challenging and needed.

Device Selection and Specification. Before we get into the analysis details of the Allen Bradley 1769-L18ER-BB1B CompactLogix 5370 L1 Rev. B, we shortly explain which criteria we used to select the target device for our implementation. On one hand,

groundwork on reverse engineering Allen Bradley PLCs had been done before by the Air Force Institute of Technology Basnight et al. (2013); Schuett (2014). On the other hand, unlike Siemens PLCs–which have received a lot attention in recent years–Allen Bradley PLCs are mostly uncharted. Nevertheless, Allen Bradley is one of the largest vendors for industrial control systems internationally. In particular, the CompactLogix L1 series is widely used in several safety-critical infrastructure applications such as power grids, water plants, oil&gas refineries, and medical devices (e.g., the Adept Viper S650 surgical assistant robot).

The 1769-L18ER-BB1B CompactLogix 5370 L1 Rev. B is based on a Texas Instruments Stellaris LM3S2793 Microcontroller,[11] which uses the ARM Cortex-M3 instruction set architecture (ISA).

Two sets of pins from the processor's pin configuration were of relevance to our work: (1) the processors pins associated with the JTAG interface, and (2) the input/output port pins of the processor.

Joint Test Action Group (JTAG) is commonly used to refer to IEEE Standard 1149.1 jta (2001). JTAG can be used as a hardware debugging interface in the processor.[12] We used JTAG to develop our HARVEY prototype, as the JTAG connection allowed us to read out the CPU's memory, including flash memory, read-only memory (ROM) and static random access memory (SRAM). Although HARVEY is not limited to JTAG as method to infect a PLC exploring further infection paths is out of scope in this work.

Figure 2.4 shows the PLC's CPU, i.e., a Texas Instruments LM3S2793. The CPU's pin allocations are marked as well as the connection of the JTAG pins to the solder pad on the right side of the board.

---

[11] The processor's data sheet can be found online and reveals important details about the processor Texas Instruments (2007-2014).

[12] Although the JTAG header was physically compatible to typical ARM Cortex-M3 10-pin JTAG connection, the pin configuration was different.

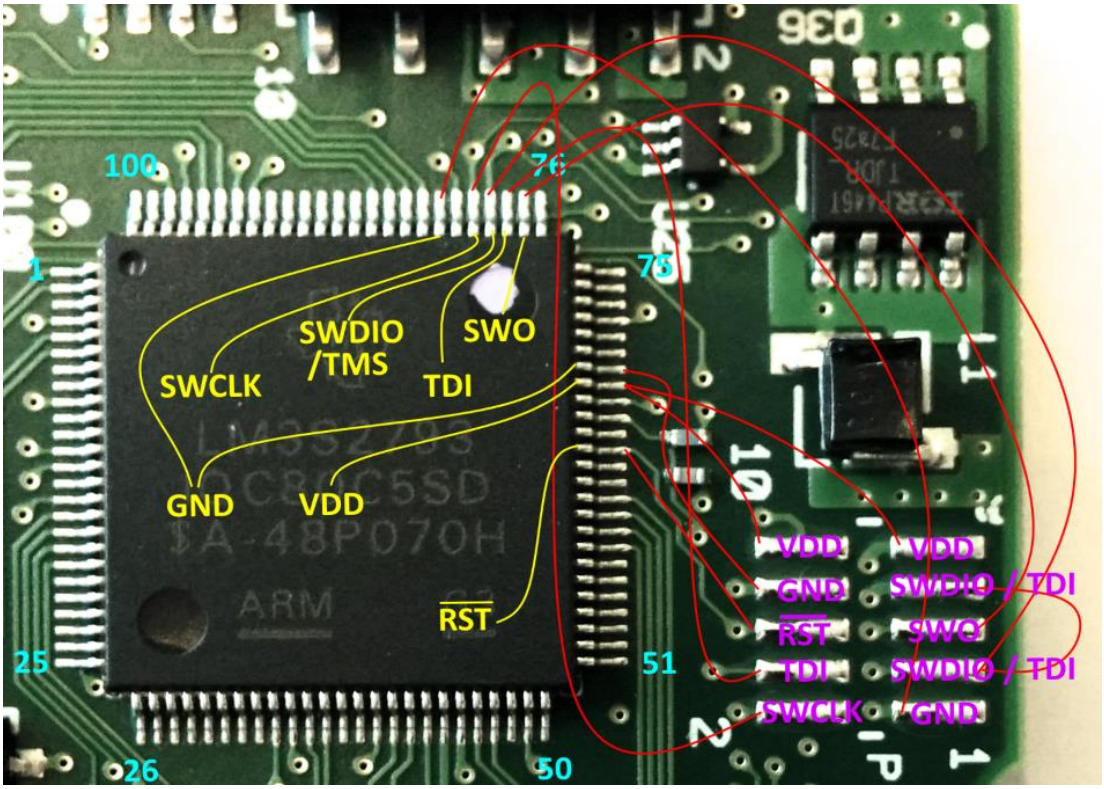


Figure 2.4: LM3S2793 JTAG Pins and Their Connections to the 10-pin ARM-JTAG Connector.

2.5.1 Preparation

The ultimate goal of our work was to modify the firmware of the PLC to manipulate input and output values of the PLC without changing the PLC's control logic. To accomplish that we had to find the firmware functions which are responsible for handling the PLC's inputs and outputs.

The first step in analyzing the firmware of our prototyping platform was to obtain images of the firmware. We used two approaches: (1) We downloaded firmware update packages from the vendor's website and extracted the firmware images from them.[13] (2) We extracted the firmware from the PLC's memory using the JTAG interface of the PLC's main processor.

[13]Vendors like Siemens encrypt their firmware images in update packages. However, the key to decrypt them have been published Beresford. The firmware images associated with our PLC are not encrypted.

Table 2.1: TI LM3S2793 Memory Map

Start	End	Description
0x00000000	0x0001FFFF	On-chip Flash
0x00020000	0x00FFFFFF	Reserved
0x01000000	0x1FFFFFFF	ROM
0x20000000	0x2000FFFF	On-chip SRAM
0x20010000	0x21FFFFFF	Reserved
0x22000000	0x221FFFFF	Bit-band alias of SRAM
...	...	
0x4005C000	0x4005CFFF	GPIO Port E (AHB)
0x4005D000	0x4005DFFF	GPIO Port F (AHB)
0x4005E000	0x4005EFFF	GPIO Port H (AHB)
0x4005F000	0x4005FFFF	GPIO Port G (AHB)
...	...	

Firmware Images. Allen Bradley PLCs (at least CompactLogix L1 and ControlLogix L61 models) have two firmwares. (1) A base firmware which is shipped with the PLC which provides a minimal function set of the PLC, and (2) a full flashed firmware which provides all functionality for operation. The latter will be referred to as *full firmware* for the rest of the paper, while the first we call *base firmware*.

The base firmware can only be extracted from the PLC's memory as it is not contained in the firmware update package. The base firmware is intended to have one central functionality: to recover the PLC in case an update of the full firmware did not succeed. Hence, the base firmware should not be updated.

The full firmware of the PLC can be updated, with several versions are available for download from the vendor's website Rockwell Automation (2015). It can be updated remotely over Ethernet or locally via USB or SD card.

Memory Layout and I/O mapping. The TI LM3S2793 maps all flash memory, ROM, SRAM as well as peripheral devices into one contiguous memory address space. Table 2.1 shows parts of the memory map of the CPU Texas Instruments (2007-2014).

2.5.2 I/O Interception

To recover the functionality of the firmware we use offline as well as online analysis of the firmware. Offline analysis was done with standard reverse engineering tools such as hex editors and dis-assemblers. The online analysis was possible due to the JTAG connection we could establish with the PLC.

Offline Firmware Analysis. We disassembled the downloaded and extracted firmware binary files from the firmware update file. However, only a small portion of the code was initially disassembled correctly, to improve the results we utilized techniques presented in Basnight et al. (2013). This code provided a greater insight of higher level functionality of the firmware, but the memory we extracted directly from the CPU through the JTAG interface proved to be more fruitful. Using the JTAG-extracted memory files, we first aimed to get a general understanding of the functionality of the firmware. Using the ARM Cortex-M3 documentation, we were able to identify the nested vector interrupt controller (NVIC) table. This table contained the address of the reset handler, i.e., where the device starts execution after a reset. Using this address, we were able to follow the boot sequence and disassemble the core functionality in the PLC's flash memory as well as all functions called in SRAM and ROM.

More importantly, we used the ROM data sheets for Stellaris LM3S devices to identify calls to the microcontroller's built-in functions Texas Instruments (2011-2013). This helped to identify when important calls to functions that interacted with system peripherals were executed. Identifying these functions gave us a basic understanding as to how the firmware was configuring the controller. Specifically, the functions to control the CPU's general purpose input/output (GPIO) ports, such as the ROM_-GPIOPinTypeGPIOInput, ROM_GPIOPinTypeGPIOOutput, ROM_GPIOPinWrite and ROM_GPIOPinRead functions, provided us with functions to look out for as we disassembled the firmware.

One other detail worth mentioning is that we were able to find where the NVIC table was being re-based after the initial boot sequence. Typically the NVIC is re-based to address `0x4000` in ARM Cortex-M3 processors. This was confirmed in our dis-assembly

as the vector table offset (located at address `0xE000.ED08`) was set to `0x4000` at the end of the boot sequence. By knowing where the NVIC is, we knew where the addresses of the interrupt service routines (ISR) specified in the LM3S2793 documentation were located. In the following section, we will see the significance of this detail.

Online Firmware Analysis. The JTAG connection of the PLC allowed us to analyze and debug firmware during its execution. We could set break points, step through functions, read and write memory and modify registers of the CPU while it is executing. We used this to follow the control flow of the firmware and discover reachable code sections.

Through the analysis we could identify the *main loop* of the firmware. We coupled our online analysis with our offline analysis to step through functions and follow along the disassembled paths. By knowing the boundaries of the main loop, we could investigate where the interaction between the LEDs/HMI and the GPIO Ports occurred. The PLC is equipped with LEDs, one per I/O pin. Similarly, the input values sent to the HMI allow the operator to observe the system state.

When the PLC is power-cycled, an LED sequence is generated where each LED associated to the embedded I/O is sequentially blinked, starting from the Input Ports to the Output Ports. This sequence is relatively slow and involves a sleep period. This allowed us to halt the processor in between two LEDs being blinked. After stepping through the LED sequence, we were able to determine the subroutine associated with sending the LED values over I^2C. Additionally, we identified the address where the I/O values are stored before they are sent to the LEDs. We confirmed this by modifying the associated registers (while the CPU was halted) and stepping through to force arbitrary values different from the typical LED sequence. Although this confirmed that we could control what values were being sent to the LEDs, we still needed to take control of the interchange between the LEDs/HMI and the GPIO Ports.

We found that the main loop has one reference to this LED update function. We noticed that before this update function, a few timer interrupts were being disabled. Using the information from the re-based NVIC table as well as the data sheet, we were

able to find the associated ISRs located in SRAM. In particular we found that the Timer 0A ISR was responsible for the interchange between the GPIO Pins and the LEDs/HMI.

As described in subsection 2.2.1 PLCs operate on the basis of so-called scan cycles, i.e., in fixed intervals inputs are read, the control logic is executed, and the results are written back. We are careful to identify this main loop as being directly correlated to the scan cycle. The Timer 0A ISR seems to be independent of the scan cycle as it is set to run every 0.25 µ s. It is only interrupted when updating the LED values. Because this process has not been fully reverse-engineered, we cannot make much stronger inferences than those already mentioned.

2.5.3 I/O Interception Code Modifications

As described in the previous section, we identified the exact two subroutines where the values from GPIO Ports E and F are being forwarded to the PLC memory associated with the input values, i.e., the input values sent to the control logic program, the LEDs and the HMI, as well as where the output values from the PLC memory are forwarded to the associated output memory for the LEDs, HMI, and GPIO Ports G and H.

For the output update routine, HARVEY uses the physical models to send legitimate-looking data to the LEDs/HMI, and sends malicious values to GPIO Ports G and H. For the input update routine, HARVEY uses the legitimate input data from GPIO Ports E and F to update the malicious model and possibly coordinate with the output modifications. HARVEY again uses the physical models to report legitimate-looking input values to the LEDs/HMI. The input values, which the control logic might report, e.g., to the system operator, can be crafted such that they correspond to the observation HARVEY makes on the control logic's output, as described in section 2.4.

I/O Interception Details. In order to implement our attacks, we modified two subroutines within the Timer 0A ISR that are responsible for the interchange of values to and from the GPIO Ports. Figure 2.5 shows the aforementioned first attack scenario where we reported false output values to the LEDs and HMI.

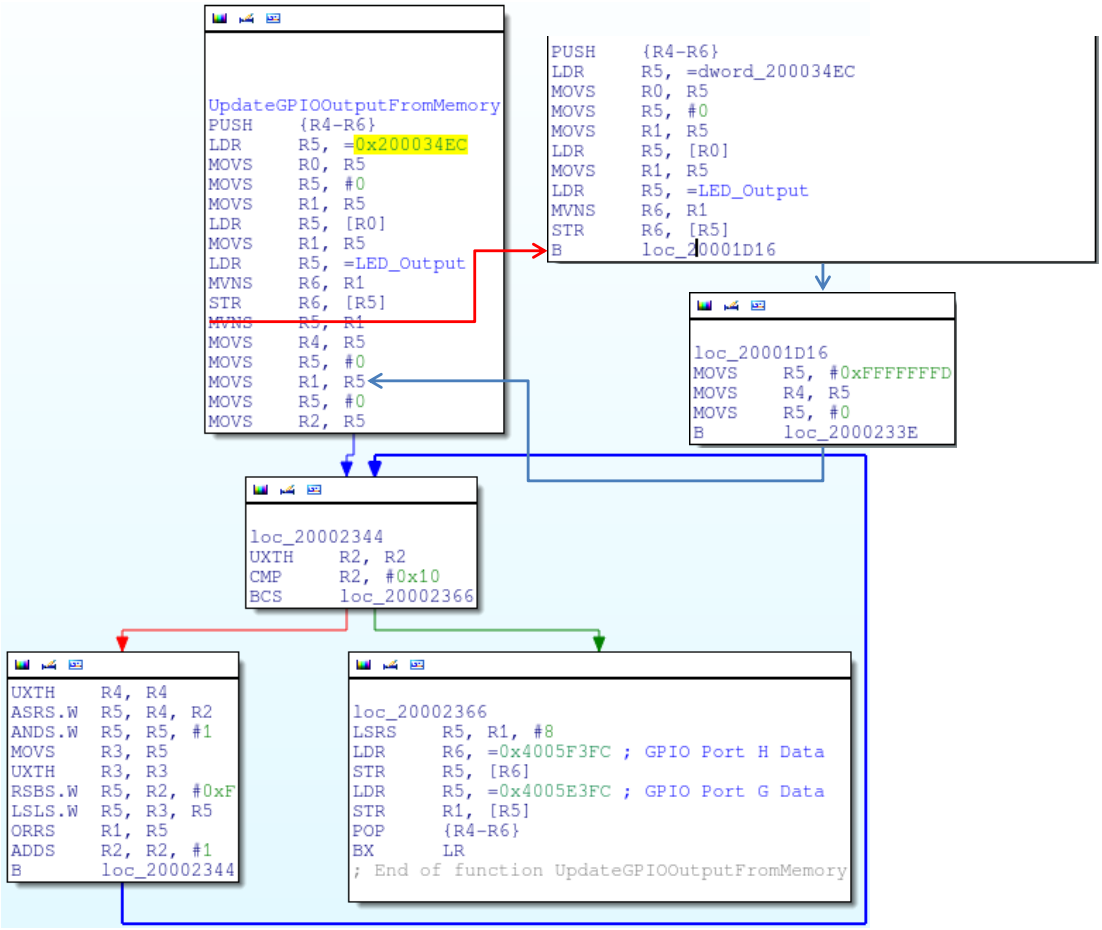


Figure 2.5: Original GPIO-output update ISR assembly code compared to modified subroutine with branch to malicious code.

The figure shows the original subroutine that was responsible for forwarding an updated output value from memory to the GPIO Ports G and H. The subroutine first updates the address corresponding to the LED and HMI output, `=LED_Output`, and then calculates the correct values to send to the GPIO Ports. For our attack, we first modified an arbitrary location of usable (or re-usable) memory in SRAM and injected our malicious assembly code. Once the malicious code was written, we modified the subroutine to branch to our malicious code. The malicious code would then branch back to the subsequent instructions once the appropriate values were modified. In our attack scenario, a safe system state with Output Port 0 high would have a "0" at the least significant bit, representing the 0 port, and the rest of the bits would be set to 1, i.e., a value of `0xFFFFFFFE`. In our attack, we want to set Output Port 0 to low and Output Port 1 to high, i.e., write a value of `0xFFFFFFFD` to the associated memory address. We

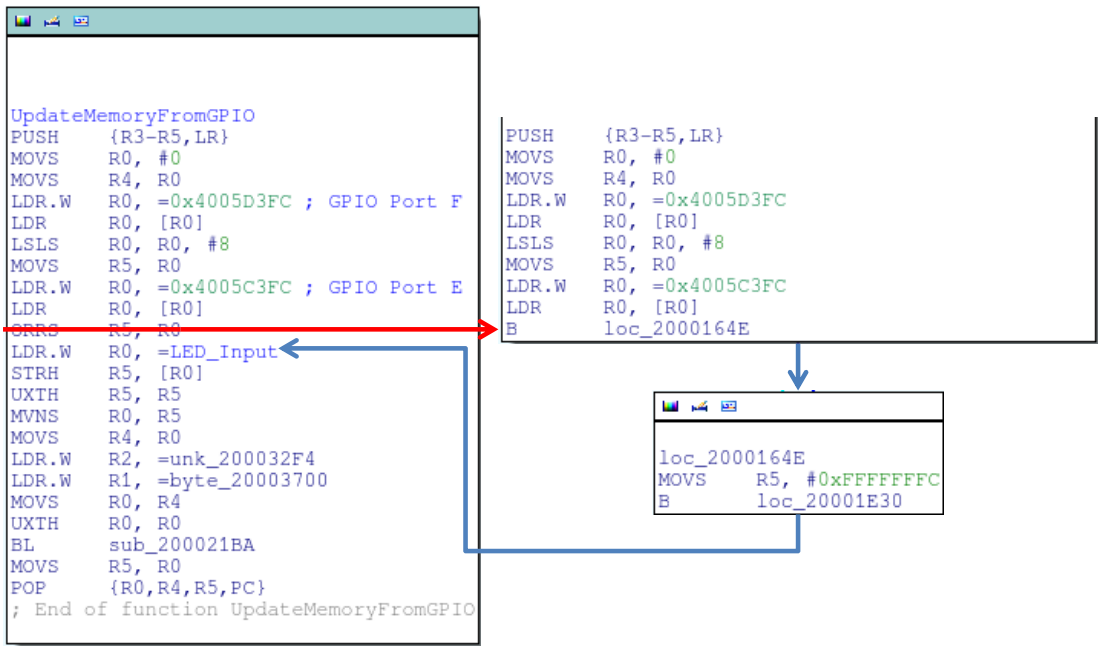


Figure 2.6: Original GPIO-input update ISR assembly code compared to modified subroutine with branch to malicious code.

chose to branch to an arbitrary memory location to prove that we can make use of the available memory to implement more complex attacks. Figure 2.6 shows the second attack implemented in a similar fashion.

In this case, the goal was to fake the input values being sent to the LED's/HMI as well as the actual ladder logic program running on the PLC. With no inputs, the expected value would be `0xFFFFFFFF`. In our attack, we disregard the values read from the GPIO Ports E and F and simply wrote a value of `0xFFFFFFFC` to the input LEDs, setting Input Port 0 and Input Port 1 to high.

2.5.4 Firmware Update

We believe that the built-in remote firmware update functionality of PLCs is the most likely method for an attacker to compromise a device. This is based on observations in related work that firmware updates are not protected against malicious modifications Basnight et al. (2013); Schuett (2014). For our PLC model, the situation is different. Firmware updates are protected by cryptographic means. Firmware updates are delivered together with certificates in the X.509 standard Cooper et al. (2008). The certificate contains a SHA-1 of Standards and Technology (2012) hash of the firmware

binary file and is signed with 1024 bit RSA Rivest et al. (1983). Although the certificate is self-signed, the PLC will abort the update process when provided with a self-signed certificate using a different key than the original one. This makes it practically impossible for an attacker to change the firmware and install it on the PLC. To succeed, the attacker has two options, (1) he could attempt to find a pre-image hash collision for the SHA-1 hash of the benign firmware binary, or (2) he could factorize the public key used to sign the certificate.

However, an attacker can always compromise a device through the JTAG interface, like we did.

2.6 Evaluations

We evaluated several aspects of HARVEY. On one hand, we evaluated the effects of HARVEY on the individual programmable logic controllers (PLC), its influence on execution times and its memory consumption. On the other hand, we evaluated HARVEY in a real-world power system to empirically prove that HARVEY can (1) maximize the effects on the physical system, and (2) hide its malicious effects from the operator.

2.6.1 PLC Evaluation

Our PLC model is equipped with an ARM Cortex-M3 processor with 64 KB RAM Texas Instruments (2007-2014). It has 512 KB memory for user programs (control logic), as well as 16 DC digital inputs and 16 DC digital outputs.

Experimental Setup. To evaluate the effects of HARVEY on a PLC we set it up with a typical control logic. We installed a control logic program for a PID (proportional-integralderivative) controller which is shipped by the vendor of our PLC as a standard control logic instruction used in many environments.

In order to model fake input and output values, we used a custom implementation of a PID controller. Figure 2.9 shows an extract of its assembler code. The code represents a sample PID update function that takes in the current system error and the difference in time since the last iteration and updates the control output based on

the summation of the scaling terms. These scaling terms are determined by the type of PID controller. In this case, we defined proportional, derivative and integration error terms to be summed for the control output. A windup guard is used to set a maximum value for the integration term. We compiled this code using a pre-built GNU toolchain for ARM Cortex-M3 processors as well as the StellarisWare libraries for our processor.

To validate the modifications of HARVEY we had to compare the physical outputs of the PLC with the information provided to the operator. To measure the physical output of the PLC we wired it to a voltmeter. To determine the operator's view of the system state, we used the built-in LEDs of the PLC as well as the online monitoring provided by the vendor's control logic development suite, which we will refer to as our HMI. The PLC has a dedicated LED per input and output pin which lights up according the logical state of the pin, i.e., when there is current on the pin the LED will turn on, otherwise, the LED will be off. On the HMI side, online updates of the downloaded control logic program are displayed in real-time.

We used HARVEY to break the relationship between the displayed system state and the actual inputs and outputs on the physical pins of the PLC.

Attack. We were able to change the LED and HMI states of the PLC arbitrarily and independently of the state of the input and output pins. Similarly, we were able to set the output values of pins regardless of the commands sent by the control logic.

Execution Time. To evaluate the performance of HARVEY, we measured its execution time and compared it to the execution time of the PID control logic. The measurements are provided in Figure 2.7, depending on the mode the PID control logic takes between 31.03 μs and 31.74 μs. The timer A0 interrupt handler, which controls the input/output interchange between the GPIO ports and the LEDs/HMI, takes only 0.25μs, which is two orders of magnitude faster. Our attack code, which implements a simple relay logic, takes only 0.18μs.

As described in subsection 2.2.1 PLCs work in scan cycles. This means inputs are read and outputs are written at a fixed rate, while the control logic gets executed

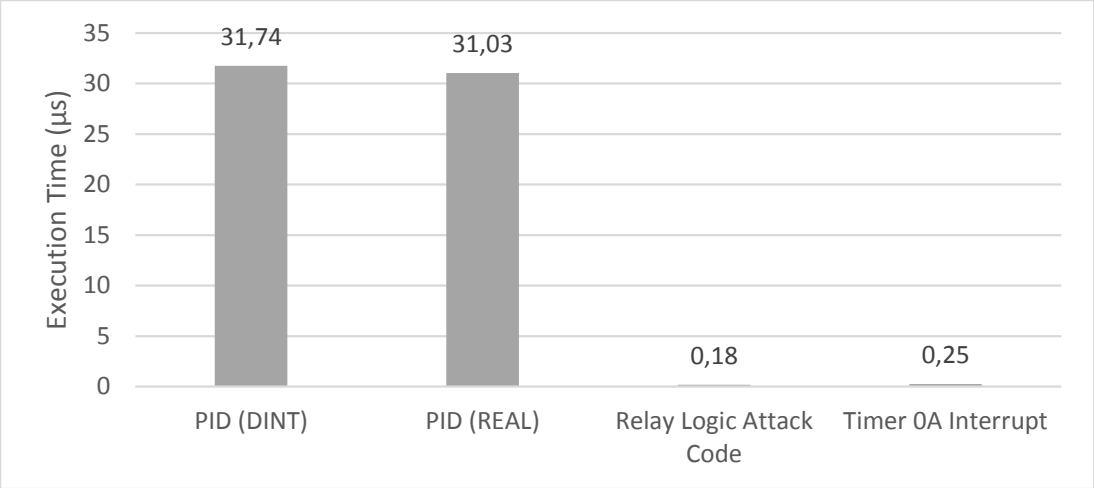


Figure 2.7: Feasibility Analysis: Performance Overhead

in between. The control logic execution time may not exceed the scan cycle length, otherwise it gets interrupted. Usually, the execution time of control logic is well below the length of a scan cycle. This means that HARVEY can utilize the time difference between control logic execution length and scan cycle length.

If the unused time of a scan cycle is not sufficient for HARVEY, there are several potential solutions that can be implemented to influence the length of a scan cycle. For instance, HARVEY's periodic execution depends on a timer interrupt configuration. Because HARVEY is executing within the firmware, the timer interrupt configuration can be modified to better suit the attacks needs. Similarly, the reporting mechanisms for the control logic/HMI also depend on the timer interrupt configurations that are implemented within the firmware. Therefore, there are several permutations of an attack vector that would allow HARVEY to execute in a timely fashion with a reasonable amount of independence from the scan cycle duration.

Memory Consumption. Our PLC has a finite amount of memory which must be shared between the benign firmware and HARVEY. Although the firmware initially occupies most parts of the memory, large parts are never used. These sections were determined during our online analysis. Furthermore, we examined subroutines that were no longer called after the initial boot sequence. Once the PLC reached the aforementioned main loop, we were able to identify the subset of subroutines that were called

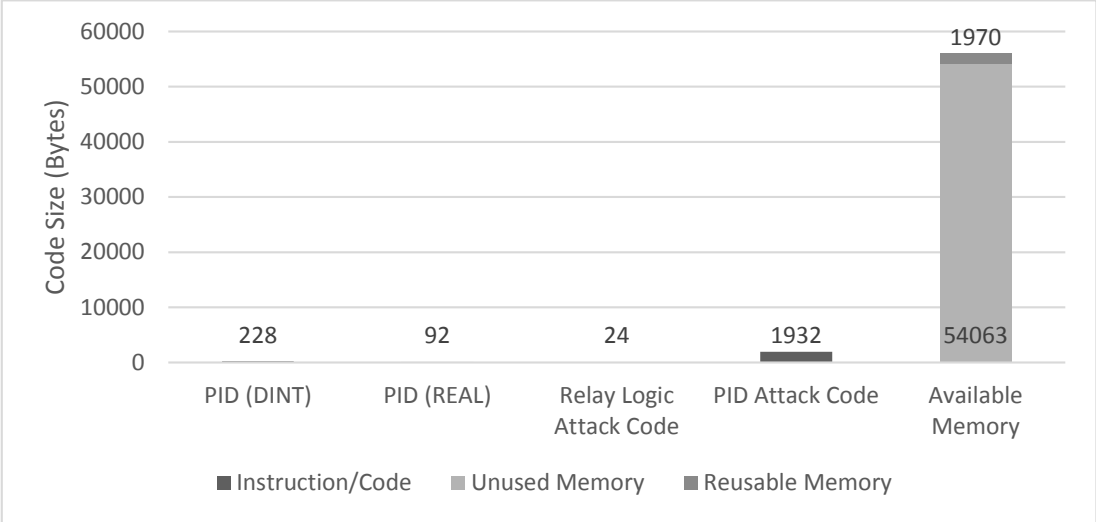


Figure 2.8: Available Memory vs. Malware Size

by the PLC during the boot sequence but were no longer referenced within the main loop. We refer to this memory as reusable memory. We verified that these functions were no longer used by setting breakpoints at the function addresses as well as any referenced locations within the subroutine. We consider unused memory as memory parts which contain regular patterns that indicate that the memory is not used, for instance, memory sections filled with all 0x00000000 or all 0xFFFFFFFF. Additionally, we found large chunks of memory that contain what seems to be garbage code that is never referenced throughout the firmware execution.

For the practical feasibility of HARVEY, Figure 2.8 lists the memory consumption of the PID control logic as well as reusable and unused memory in the firmware. It also shows the memory consumption of the custom PID update function we used as the adversary's system model in our setup.

HARVEY can utilize the unused memory as well as the reusable memory parts, which is significant portion of the PLC's memory (56.033 Bytes out of 65.536 Bytes).

2.6.2 Real-World Power System Case Study

We evaluated HARVEY on a real-world power system test-bed, where distributed PLCs with installed PID modules along with more complicated control algorithms (discussed below) maintain safe power system operation.

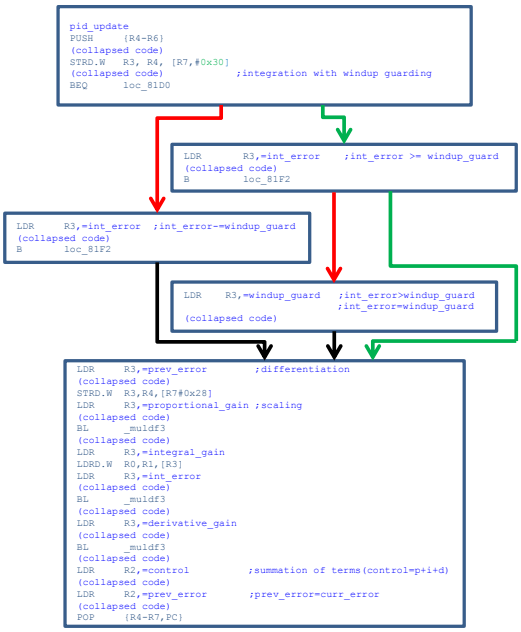


Figure 2.9: Injected Malicious PID Controller

The electricity grid is modeled using the mathematical power flow equations (physical Kirchhoff laws):

$$f_i^p = -P_i^g + P_i^l + \sum_{k \in C} |V_i||V_k|(G_{ik}\cos\theta_{ik} + B_{ik}\sin\theta_{ik}), \tag{2.3}$$

which mandate how the sensor measurements (e.g., real/reactive power values on i-th power node (bus) P_i/Q_i, power bus voltages V_i, inter-bus phase angles θ_{ij}, and the admittance (inverse resistance) parameters (G_{ij}, B_{ij}) on the transmission line between the buses i and j correlate due to well-known physics Kirchhoff laws. P_i^g represents the amount of power that is injected to the i-th power bus by a generator, and P_i^l is the amount that is consumed by the end-users at that bus.

Optimal power flow (OPF) is the most widely used control algorithm that is used in practice nowadays to calculate the optimal control commands continuously. In power systems, OPF finds an optimal power generation set-point that minimizes total cost c while meeting operational safety constraints FER (2010). The control commands typically include power output (set-points) of generators Dommel and Tinney (1968). The OPF's equality constraints are the power balance equations at each bus in the system. Its inequality constraints are the network operating safety limits such as line

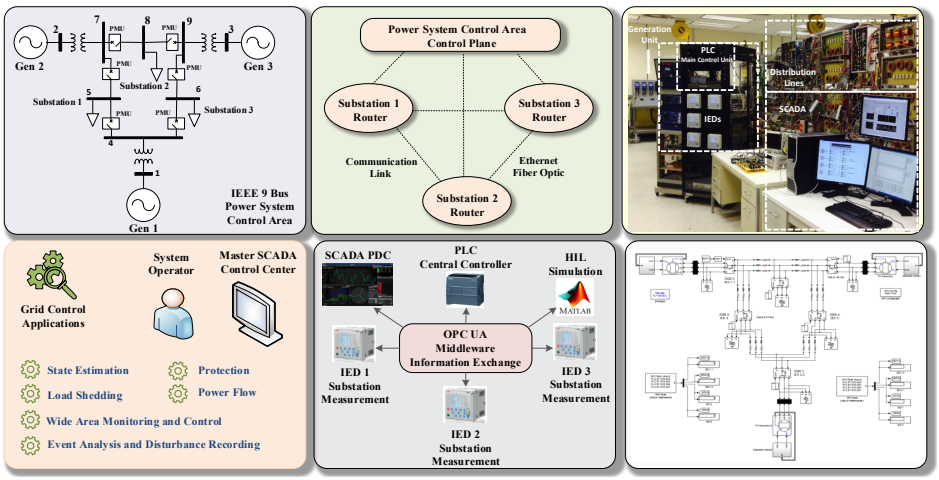


Figure 2.10: The Evaluation Smart Grid Test-Bed

flow capacities and generator power output limits:

$$
\begin{aligned}
\min_{u} \quad & c(x,u) \\
\text{s.t.} \quad & P_i^g - P_i^l = \sum_k |V_i||V_k|(G_{ik}\cos\theta_{ik} + B_{ik}\sin\theta_{ik}) \\
& Q_i^g - Q_i^l = \sum_{k\in C} |V_i||V_k|(G_{ik}\sin\theta_{ik} - B_{ik}\cos\theta_{ik}) \\
& P_l^g \leq P_l^{gmax} \\
& \forall i,j \in N,\ \forall l \in G,\ \forall k \in C
\end{aligned}
\tag{2.4}
$$

where u denotes the controls commands to be calculated; x represents dependent variables; V and θ denote the bus voltage magnitudes and angles, respectively.

The legitimate OPF's objective is to minimize the cost while ensuring the system operates safely. HARVEY implements a modified version of the algorithm, malicious optimal power flow (mOPF), to maximize the cost without the need for compliance

with safety constraints:

$$\begin{aligned} \max_{u} \quad & c(x,u) \\ \text{s.t.} \quad & P_i^g - P_i^l = \sum_k |V_i||V_k|(G_{ik}\cos\theta_{ik} + B_{ik}\sin\theta_{ik}) \\ & Q_i^g - Q_i^l = \sum_{k\in C} |V_i||V_k|(G_{ik}\sin\theta_{ik} - B_{ik}\cos\theta_{ik}) \\ & \forall i,j \in N,\ \forall l \in G,\ \forall k \in C \end{aligned} \tag{2.5}$$

where the calculated control commands would *maximize* the amount of possible damage to the power system. The calculated commands are used as set-points to be maintained by the inner-loop PID controllers. Please note that the specific objective function for different malicious goals can be simply used instead in the formulation above.

Our power system test-bed implements IEEE nine-node (bus) benchmark topology Christie (2000) including three synchronous power generators and controlled by nine distributed PLCs. Figure 2.10 shows the test-bed (top right), its cyber network topology (top middle), power system topology (top left), Lab-View control diagram (bottom right), supervisory control and data acquisition device interconnections (bottom middle), and monitoring and control operations (bottom left). The model has three substations and corresponding loads (which consume power). The power nodes are connected through power transmission lines. To follow real world implementations, we equipped each substation with protection functions such as over-current, voltage and frequency, i.e., the substation will open a transmission line if they carry current beyond its physical capacity or cause over-voltage or over-frequency situations. To monitor the power system, the voltage and current sensors (phasor measurement units PMUs) send their measurements to PLC controllers that act as monitoring/control agents and are responsible for all operational functions in the system.

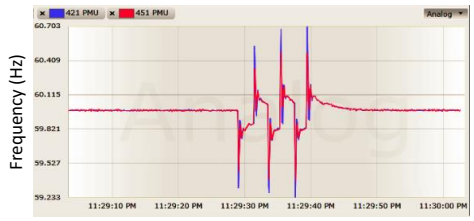
On the cyber end, the testbed includes a human-machine interface (HMI) server to provide the system status to the operators through its connections to the PLC. The data exchange between different field devices is established by open platform communications (OPC) Client I/O servers OPC Foundation (2015). Kepware OPC Server provides

embedded drivers to connect to the PLC. The testbed employs a ReLab device driver to connect to and obtain measurements (IEEE C37.118) from PMU sensors. Briefly, using Kepwares IEC 61850 MMS clients, the KEPServerEX OPC Server drivers create an interface for any of the OPC clients running in the network.

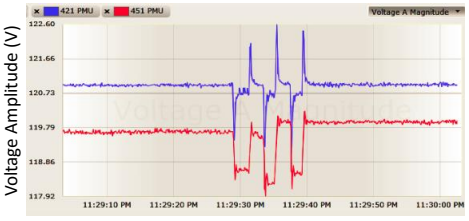
We evaluated HARVEY for two attack scenarios.

Steady-state system malicious attack: Repeated heavy load circuit breaker open/-close triggering without loss of power system stability. In this scenario, the malicious PLC firmware randomly (blindly) selects a circuit breaker to attack and triggers the opening/closing of the breaker several times, i.e., a transmission line opened and closed repeatedly. The power system was able to withstand this attack scenario without losing the stability since the target circuit breaker load was in the limits of power system generation reserve capacity. The SEL-451 PMU is located on generator 1 bus, and the 421-PMU is located at generator 2 bus. Figure 2.11 shows the power system status during the attack that starts at 11.29.30 PM. The circuit breaker was opened and closed three times sequentially within ten seconds. The heavy loading in the system deteriorated the system frequency (Figure 2.11a) and voltage (Figure 2.11b). The AC phase angle difference between generator 1 and generator 2 exceeded permissible limits (Figure 2.11c). The power flow magnitudes (Figure 2.11d) also violated safety thresholds temporarily. As shown, although the instant voltage and frequency of the system exceeded permissible limits, the power system was able to withstand this type of attack. During the attack, HARVEY was able to run the power system model on the PLC in parallel and generate fake legitimate-looking sensor measurements to be viewed by the operators. Figure 2.12 shows the results (before the noise was added for the presentation clarity). As the attack on the physical plant would result in noticeable side effects such as equipment operational noise, HARVEY's outputs show a minor system perturbation within safety limits that is normally observed on daily dynamic power system operations. From the operators' viewpoint, the system acts safely and no corrective action is needed.

Adversaryoptimal control attack: optimal malicious attack using real-world control algorithms. In this attack scenario, HARVEY implements a real-world power system

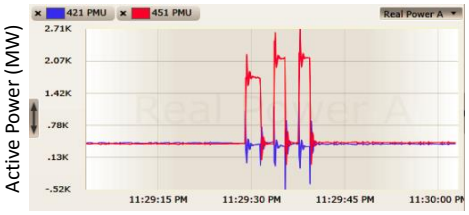


(a) Frequency



(b) Voltage Magnitude

(c) AC Voltage Phase Angle



(d) Power

Figure 2.11: Actual Power System Measurements

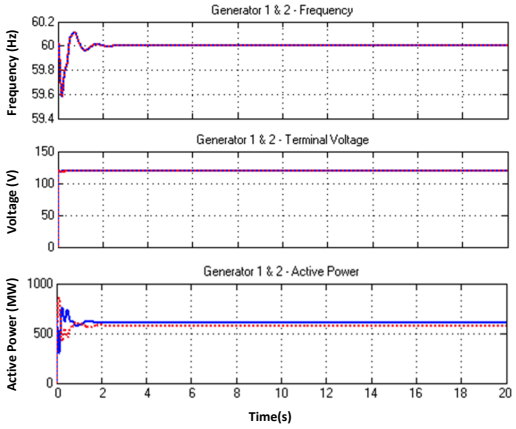
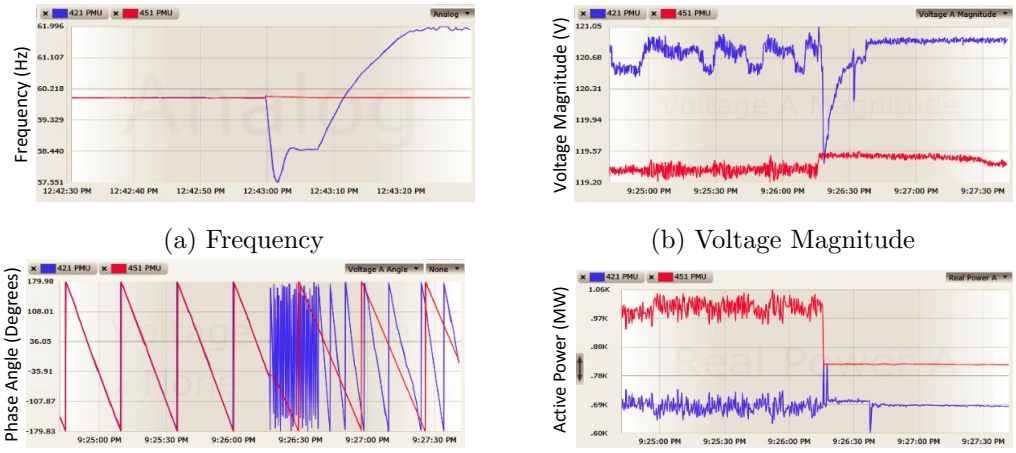


Figure 2.12: Fake Measurements to Mislead the Operator

controller algorithm, called optimal power flow (OPF) Bose et al. (2015), that is widely used in power system control centers internationally. OPF is implemented as a linear programming function: it typically finds the optimal power system control strategy that minimizes the overall cost while ensuring the system safety. The system safety is usually defined by a set of lower and upper bound thresholds for various system parameters such as power transmission line current capacities, and minimum/maximum allowed system frequency 59.5-61Hz (60Hz is the nominal power grid frequency in USA). The control strategy is essentially a set of control commands that the PLC sends to the actuators, e.g., generation set-points to the generators that mandate how much power each generator should generate. HARVEY implements the same control algorithm on the PLC after making three modifications to the algorithm (we call it malicious OPF - mOPF): *i)* it removes the condition that ensures the system is within safety margins; *ii)* it replaces the cost minimization function with maximization so that the adversarial impact becomes maximum; and *iii)* HARVEY adds predefined stealthy conditions to ensure its malicious control actions do not get noticed/detected by the local operators on site due to the noise the actions generate. Example conditions are "no power generator disconnect from the rest of the power grid" in large power plants, since such disconnects cause a noticeable sound noise to the potential local operators. In practice, there are typically few or no operators present on remote power system substations. This gives HARVEY more freedom in terms of what malicious actions it can carry out.

In this attack scenario, HARVEY's objective was to implement mOPF on the PLC to calculate adversary-optimal control strategy for the power plant. Using the power system's safety constraints, HARVEY intercepts the legitimate control action outputs and instead sends out its optimally-calculated malicious control commands to the power actuators at specific time points. HARVEY sets the nominal frequency reference to 62 Hz, and its malicious controller calculates and sends out control commands accordingly.

Figure 2.13 shows the actual power system measurements. HARVEY makes the power system frequency exceed its safety margins through its malicious commands (Figure 2.13a). The system's voltage magnitude (Figure 2.13b), AC voltage phase angle (Figure 2.13c), and electric power values (Figure 2.13d) experience serious instability as



(a) Frequency (b) Voltage Magnitude

(c) AC Voltage Phase Angle (d) Power

Figure 2.13: Actual Power System Measurements

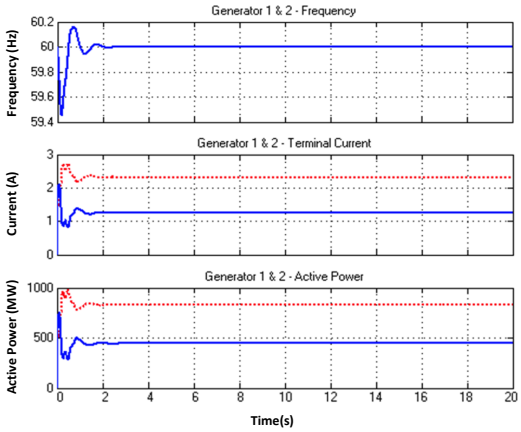


Figure 2.14: Fake Measurements to Mislead the Operator

well. However, in order to mislead the operator, HARVEY implements a legitimate OPF algorithm in the background to simulate the power system and calculate individual system parameters assuming that the legitimate OPF control commands were carried out on the power system. The fabricated fake sensor measurements (Figure 2.14) are sent back to the operators' HMI screens. Consequently, from the operators' viewpoint, the underlying power system follows their expectation, while in reality, the system goes through serious instability situations facing potential large-scale failures. An experienced operator might get suspicious of small disturbances visible in the graph. However, such disturbances can also occur in normal operation. Similarly, an automated tool monitoring the ICS must be tolerant to small disturbances to reduce the number of false positive alarms.

2.7 Related Work

We discuss related work on ICS security in terms of proposed defense mechanisms and possible attacks.

Defense mechanisms have been proposed on network and host/device levels. SOCCA Zonouz et al. (2014a) generates network-level attack graphs based on Markov decision processes considering the impact of the adversarial actions on the physical power system. CPMA Davis et al. (2015) uses the ICS attack graphs to perform security-oriented risk analysis, so-called contingency analysis, regarding potential threats against the power grid. Both solutions consider PLCs as the interface between cyber and physical assets of the infrastructures, and identify them as potential targets by the adversaries. SCPSE Zonouz et al. (2012) and CPAC Etigowni et al. (2016a) present a stateful detection mechanism to detect attacks against control systems based on the received sensor measurements by the operators. HARVEY evades such detectors completely through replacing them with legitimate-looking fake measurements.

Unlike traditional IT cyber networks, ICS networks often follow well-defined behavioral patterns. Therefore, online ICS intrusion detection solutions monitor the runtime operation for anomalous behaviors as opposed to the signature-based paradigm Yang et al. (2006); Cheung et al. (2007); Kleinman and Wool (2014). Formby et al. Formby et al. (2016) employ the behavioral profiles for device fingerprinting and access control. Security solutions in ICS has to be non-intrusive against safety-critical operations with real-time constraints that run mostly on resource-limited embedded devices/controllers Zhu and Sastry (2010). Such anomaly-based solution cannot identify HARVEY, since it uses the same power model to fake sensor measurements and make them look normal.

TSV McLaughlin et al. (2014) and Zonouz et al. (2014b) provide a bump-in-the-wire solution between the HMI and PLC device to intercept and analyze the control logic downloads on the PLC by the HMI server. TSV implements formal methods to verify the safety of the code regarding the physical plant safety requirements, and drops the control logic if a counterexample is found. TSV is unable to detect HARVEY as it

targets control logic updates and does not support firmware updates or network-level firmware exploits.

The WeaselboardMulder et al. (2013) is a PLC backplane analysis that captures backplane communications between modules with the intention of preventing zero-day exploits on PLCs. The inter-module traffic is forwarded to an external analysis system that detects changes to process control settings, sensor values, module configuration information, firmware updates, and process control program (logic) updates. Because the board monitors backplane communication, HARVEY would remain undetected as HARVEY would feed fake-legitimate-looking measurements to the PLCs backplane, i.e., HARVEY resides between the the I/O modules and the backplane. However, a Weaselboard implementation can prevent arbitrary firmware updates over the network. Therefore, an attacker would need to convince the operator to run a firmware update with a compromised firmware binary file or use a JTAG implantation to modify the firmware.

There have been several security solutions focused on the detection of firmware modifications that could prevent an attack like HARVEY, e.g., control-flow monitoring solutions Reeves (2011). However, our contribution is not in the evasion of firmware modification attacks, but rather the evasion of intrusion detection solutions that sit outside of the PLC. Attacks such as Ghost in the PLC Abbasi and Hashemi (2016) have shown that these firmware modification monitoring solutions can be circumvented.

Attacks on ICS and PLCs have grown significantly since their seminal example emergence, the Stuxnet worm Falliere et al. (2010) that targeted the Iranian Natanz nuclear enrichment plant. Stuxnet is categorized as a malicious control command injection attack. Through four Windows zero-days, Stuxnet compromised HMI and sent malicious control logic to the PLC. Stuxnet attacks would be identified using TSV McLaughlin et al. (2014) as violating the plant's safety requirements and blocked from the PLC execution. Similar to Stuxnet, PLC-Blaster worm Brüggemann and Spenneberg (2015) injects a malicious control logic on the vulnerable PLCs (Siemens S7-1200v3) after a network scan. For stealth, PLC-Blaster manipulates the meta-data of its control logic which will cause the HMI software (Siemens Step7) to crash when the operator tries

to retrieve information from an infected PLC. This would raise the operator's suspicion leading to potential detection. Ghost in the PLC Abbasi and Hashemi (2016) provided a PLC rootkit that exploited I/O pin control operations to provide a cyber framework for an undetectable rootkit. However, the rootkit does not provide a means of stealthiness with respect to the monitoring entity overseeing the physical evolution of the system.

On the sensing side, false data injection attacks McLaughlin and Zonouz (2014); Liu et al. (2011b); Xie et al. (2010) have been shown to be capable of misleading the operators. The attackers in control of a subset of sensors would send corrupted measurements to control centers to mislead the state estimation and controller servers. False data injection attacks do not consider the operator's control commands to the plant, and hence their fabricated system state may not satisfy the operators' expectation, and hence can be detected simply.

On the network side, Beresford Beresford discovered vulnerabilities in Siemens S7 series communication protocol for replay attacks leading to a remote shell access. The attack was specific to that PLC model number of Siemens only. The similar attacks on firmware vulnerabilities (e.g., insecure checksum validation during the update process Basnight et al. (2013), DDoS attacks against common industry protocol CIP package handler functions Schuett (2014)) are orthogonal to HARVEY since HARVEY's core contribution is to inject and run a power system model as a rootkit (after the firmware is compromised) to damage the physical plant while evading the operator detection. The above-mentioned PLC attacks did not leverage the domain-specific features to *i)* maximize their destructive physical impact using adversary-optimal control algorithms, and *ii)* simulate the physical plant model to fabricate legitimate-looking measurements to the operators.

Klick et al. Klick et al. (2015) show that internet-facing controllers can be compromised, act as a SNMP scanner or SOCKS proxy, and be misused by an adversary to attack devices that are not directly connected to the internet. This technique can be used to extend HARVEY's compromised devices. Additionally, there have been theoretical attack frameworks proposed against water SCADA Amin et al. (2010). The

attack drives the underlying physical plant towards unsafe states by solving the partial differential equations that model the water plant dynamics. The authors do not discuss details of how such attacks can be implemented in practical real-world settings and the involved challenges. Similarly, system-theoretic models Pasqualetti et al. (2011) Amini et al. (2015) have been proposed to identify stealthy cyber-physical attacks against the power grid either through compromised measurements or dynamic load-altering attacks. However, the models assume that attacker has already compromised the required assets. These models could be utilized by HARVEY to model the spoofed measurements being sent to the HMI.

2.8 Discussions and Mitigations

We discuss the generality of HARVEY, and describe potential mitigation mechanisms that could be deployed to protect critical infrastructures against similar attacks.

2.8.1 Unique Challenges in ICS

In this section we elaborate on some of the unique challenges in the ICS space by comparing our PLC rootkit against rootkits known from workstations and servers. We also elaborate on the relation of industrial control systems and the Internet of Things (IoT).

Rootkits. The concept of rootkits, or more generally, compromising the privileged software of a computing system with the goal of hiding has been known for decades Hay et al. (2008). However, most known rootkits target commodity operating systems like Windows or Linux which have vast amounts of resources that can be (mis-)used by the rootkit to hide itself and perform its malicious actions. PLCs, on the contrary, have a significantly different software design (cf. subsection 2.2.1) which requires new techniques for rootkits in the domain of ICS.

HARVEY, in particular, aims at manipulating the input and output of a PLC, i.e., interaction with the physical world. "Classical" rootkits operate only within the deterministic cyber world, which makes hiding and other actions simpler. For instance, a

typical hiding method of Windows rootkits is to remove themselves from a list structure maintained by the OS Symantic (2005). HARVEY, however, has to compensate for the changes to its physical world which will eventually feed back as input readings.

Internet of Things (IoT). In recent years, the term IoT is used over-extensively even though (or because) there is no widely accepted and precise definition of the term. While there are definitions that include ICS (e.g., every computing device with a network connection) we would argue that ICS are not part of the IoT. Many IoT devices are based on commodity hardware and software (e.g., ARM Cortex-A processors and Linux based OS) and thus not much different from systems like smartphones. Hence, attacks on IoT devices, such as the compromise of the Google nest thermostat Hernandez et al. (2014), are not applicable to ICS. Another significant difference between consumer IoT devices and ICS is the real-time operation of PLCs, reflected by the control loop design of the PLC's firmware. IoT devices rarely have strict real-time requirements. Further differences include software deployment schemes (IoT device's software is controlled by the device manufacturer, PLC control logic is installed by the programmer/operator), or device lifetime (ICS might be operated for decades). Lastly, the operation of IoT devices (and their effects on the environment) are often unsupervised, hence, an IoT rootkit does not need to provide a manipulated system state view to an operator.

2.8.2 Generality

HARVEY involved reverse engineering of a real-world commercial PLC device and binary software modules. Although we worked on a specific model, the techniques we used, such as JTAG debugging and binary analysis, can be generalized to PLC and controllers from other vendors, because they generally follow similar technical approaches such as scan-cycle-based execution paradigm followed by periodic I/O interrupts and memory updates. Additionally, the proposed two-way data manipulation attack can be implemented on other (not necessarily power grid) control system settings, where controller devices are used to monitor and control underlying physical plants.

HARVEY can be protected against using three major mitigation solutions: *i)* remote attestation allows a verifier to check the software integrity of a system. A trusted component provides an authenticated measurement of the memory of the device to be attested. Different approaches specifically for embedded systems have been developed and could be applied to PLCs Strackx et al. (2010); Defrawy et al. (2012); Brasser et al. (2015); *ii)* with secure boot, the integrity of a devices configuration is not verified by an external entity but by the device itself possibly using a trusted platform module Arbaugh et al. (1997). Secure boot ensures that only a known and trustworthy software can be loaded on a device. Secure boot could be used to ensure the integrity of PLC firmware; and *iii)* an external bump-in-the-wire device between the PLC controller and the physical plant could be monitoring the two-way sensor-to-PLC and PLC-to-actuator data streams (unlike TSV McLaughlin et al. (2014) that would sit between the HMI and PLC). The solution could possibly check whether the control commands issued by the PLC satisfy the plant's essential safety requirements that must be defined by the operators. Additionally, the solution could implement coarse-grained control consistency checks to validate whether sensor measurements and actuation commands are consistent in terms of how the plant should be controlled.

2.9 Conclusions

We presented HARVEY, a PLC rootkit that implements a physics-aware man-in-the-middle attack against cyber-physical control systems. HARVEY damages the underlying physical system, while providing the operators with the exact view of the system that they would expect to see following their issued control commands. Our experimental results with a commercial PLC controller on a real-world power system test-bed demonstrates the feasibility of HARVEY in practice.

Chapter 3

Malicious Fill Pattern Detection in Additive Manufacturing

3.1 Introduction

Additive Manufacturing (AM), also known as 3D printing, is an emerging field that shows promise in reducing waste, time, and infrastructure needed in a manufacturing process. Many major companies including Ford, GE, Airbus, SpaceX, Koenigsegg, and NASA are currently utilizing AM for both prototyping and production-quality manufacturing mRick Smith (2016); har (2017); arc (2016); Jeff; Davies et al.; Janaki Ram et al.. Additionally, AM has been employed as a useful tool for printing medical implants Berman, and cutting edge research is underway on producing food, drugs, and living tissue using AM techniques nat (2017); Hicks. Across industries, AM is expected to reach a market potential of 50% by 2038 Wohlers (2015).

Because of this potential for wide-spread use of AM in the coming decades, work has begun on understanding the security challenges that are unique compared to traditional manufacturing and cyber-physical security. Mark Yampolskiy, *et al.* Yampolskiy et al. outlined a taxonomy for the potential of the misuse of a 3D printer as a weapon (3D-PaaW). In their paper, they identify the elements which may compromise or manipulate an AM environment, the targets of attack (printed object, printers, or environment), and the parameters for understanding the potential effectiveness of a given attack.

In this paper, we focus on the use of a 3D-PaaW to manipulate the physical properties of a printed object through manipulation of the object specifications, manufacturing parameters, and/or source material. According to the taxonomy described by Yampolskiy, *et al.* each of these are classified as attacks which would be achievable by an adversary through the manipulation of printer firmware or the controller PC. It has

been shown that structural integrity can easily be compromised by introducing slight modifications in the model, e.g., a minuscule void injected into a manufactured dog bone can reduce the yield load by 14 percent Sturm et al. (2014). In order to combat these forms of attack, we propose three methods of verification of design parameters that utilize analysis of the acoustic signal, embedded materials, and spatial position of machine components. These are chosen because they provide information about the manufactured design *without* access to the STL file or the G-code instructions[1] read by the printer. We do not consider our techniques to be a panacea for all verification needs. They are meant to be complementary to domain-specific verification methods. In some cases, this may be means of saving costs, e.g., by detecting malicious prints in real-time and ending them at the onset of a detection. In other cases, this may be a means of ensuring safety, e.g., by detecting malicious materials or designs before the print is used. Throughout the course of this paper, we will consider the use case of printing the tibial portion of a knee prosthesis.

Our contributions are as follows:

- A multi-layered approach to the verification of design specifications, manufacturing parameters, and materials used in an AM.
- Proposed implementations of aforementioned approach for in-house and third-party AM producers.
- A case study of a scenario in which a malicious print of a medical prosthetic is identified.

The paper is organized as follows. We first provide a background in AM verification along with a system overview and threat model in section 3.2. We then provide details for the different types of verification methods that we proposed in section 3.3. In section 3.4, we evaluate the effectiveness of the combined verification scheme on a malicious print of a tibial knee implant. In section 3.5 we discuss the implementation

[1] An STL file is a STereoLithography file for CAD software used in 3D printing. G-code is the set of actual instructions for 3D printers that are generated for particular models given an STL file and the print configuration, e.g., print speed and infill density.

and limitations of the verification scheme. We conclude in section 3.6 and discuss future work.

3.2 Background and System Model

In this section we discuss the previous efforts related to side-channel analysis of AM and verification of the physical models. We then provide a system overview of our approach as well as the threat model that will be used for the rest of the paper.

3.2.1 Side-Channel Analysis

In this paper we provide a means of verification by utilizing the various side-channels of the printing process. We also use materials science based verification to verify that the intended physical model is printed. As such, we first review previous efforts that have been made for the analysis of the side-channels involved in the AM process. We then provide a brief review on materials-based verification techniques like Raman spectroscopy and computed tomography (CT).

Acoustic, Magnetic, and Motion Sensing. KCAD Chhetri et al. (2016) provided the first method of using the analog emissions of AM processes for the purpose of detecting so-called zero-day kinetic cyber-attacks. However, the work utilizes only one 3D printer and only investigates attacks in which simple variations in the exterior design. The paper also lacks any means of verifying the printed materials post-manufacturing. The focus of the majority of previous work on the analysis of side-channels from 3D printers used in AM has been its usefulness in obtaining intellectual property. Chen Song, *et al.* Song et al. and Avesta Hojjati, *et al.* Hojjati et al. each showed that the array of sensors available on a modern smart phone can be leveraged to re-create designs produced from 3D printers or CNC machines. The sensors used in each study to collect side-channel data included the microphone, magnetometer, and accelerometer. Each group was able to reconstruct simple printed designs using supervised machine learning and manual analysis of sensor signals respectively. However, each group was only able to reconstruct very simple shapes such as two-dimensional outlines of airplanes or keys

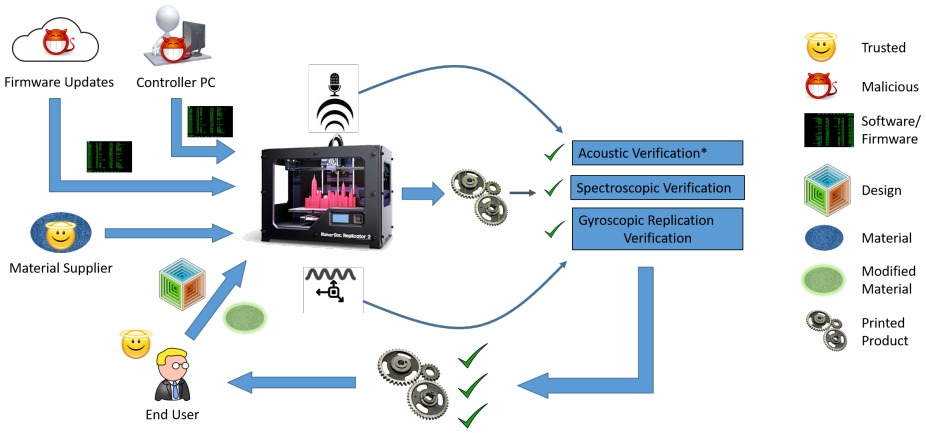


Figure 3.1: System Model.

with no fill structure.

Beyond 3D printing and manufacturing, acoustic signals have also been shown to be useful in a growing number of security applications. As an example, Guri Mordechai, *et al.* Guri et al. showed that information can be transmitted from a speakerless PC using information embedded in the sound of a cooling fan. Likewise, accelerometers have been used across industries as quality control sensors in CNC machines Lemaster et al..

3.2.2 Physical Model Verification

The physical model that is printed from the AM machines are typically verified in a manner specific to the domain, such as mechanical strength testing Sturm et al. (2014). Chien, *et al.* Chien et al. (2012) use several techniques such as surface morphology characterization to verify 3D-printed tissue scaffolds. Furthermore, several solutions have been presented as preventative measures to future physical failures, such as the solution presented by Stava, *et al.* Stava et al. (2012) for detecting and correcting models prior to being printed. However, these only correct the models that are being sent to the printer and do not verify the actual physical model in the event that the printer itself is compromised.

Imaging Analysis. We will now discuss the background for two modalities used for observing the composition of materials that will be explored in this paper for the

verification of 3D printed models. It is important to note that we do not consider these modalities to be the most effective imaging techniques nor the most cost-effective solutions. As we will discuss in section 3.4, we chose these two modalities as they were readily available and are generalizable. Both solutions will act as a template for imaging techniques that are used to identify embedded materials. The choices for both the imaging technique and the associated embedded materials will be specific to the context in which they are applied.

Raman Spectroscopy. Surface-enhanced Raman spectroscopy (SERS) has been shown to be sensitive to single-molecule detection Nie and Emory (1997); Kneipp et al. (1997); Michaels et al. (1999); Le Ru et al. (2006). Nie, *et al.* Nie and Emory (1997) have shown that silver colloidal nanoparticles can be used to amplify the spectroscopic signature of adsorbed Rhodamine 6G (R6G) and enable the single R6G molecule detection at room temperature. Furthermore, the sizes and shapes of the colloids enhance the spectral responses at different plasmon bands Nikoobakht and El-Sayed (2003); Orendorff et al. (2006). We find that this technique can be utilized for post-production verification of 3D printed objects. By embedding a series of detectable markers of contrast agents in SERS at specific location within the 3D printed object, the SERS process would be able to reconstruct the model and verify the integrity of the internal structure of an object.

Computed Tomography. CT is typically used in medical applications to enable doctors to view precise images of their patients' internal organs Kak and Slaney (2001). Additionally, CT scanning also has been used in a wide variety of applications for verifying structural integrity. Cnudde, *et al.* Cnudde and Boone (2013) discuss the application of CT scanning in the context of geomaterials. Akin, *et al.* Akin and Kovscek (2003) also discuss the use of CT as a non-destructive method for imaging multiphase flow in porous media in the context of petroleum engineering research. Similarly, Alymore Aylmore (1993) discusses how CT scanning was used as a non-destructive method for studying soil behavior and soil/plant/water relations in space and time. In this study, we utilize CT in a similar fashion to construct models and verify the integrity of completed objects.

3.2.3 System Model

Figure 3.1 provides an overview of the system model that includes all verification techniques presented in this paper. Our system assumes that there is an end user with a 3D model design. The design will be printed on a 3D printer that is controlled by a controller PC. The 3D printer may or may not be controlled by a third party entity. The end user will send her design to be printed. Throughout the printing process, the object will be verified using three verification layers. The first two layers are achieved through acoustic side-channel analysis and spatial sensing which analyze the sound and physical position of printing components respectively. The third layer is that of materials verification in which imaging techniques are used to verify that the print is made from the proper material and printed correctly.

The end user may supply her own modified set of materials to the printer so that physical model verification may be performed upon completion. The goal is to embed special materials into the filament that is used in 3D printing. The modified filament can be used for materials verification purposes.

For the remainder of the paper, acoustic side-channel verification, spatial side-channel verification and materials verification are referred to as the acoustic layer, spatial layer, and material layer respectively.

3.2.4 Threat Model

The threat model assumes that the attacker has full knowledge of both the printer and its control software. If a third party manufacturer or affiliate of the user is involved, they are trusted as an organization. Therefore, they are willing to provide information about the print for verification. However, malicious entities may include network intruders, disgruntled employees, or other insider threats. The attack is carried out such that the printer behaves maliciously despite being sent G-code [2] for a non-malicious print. Meanwhile, the controller PC indicates that the print is being carried out correctly.

[2] G-code is the set of instructions interpreted by a 3D-Printer, CNC, or other machine that includes information about motion direction, speed, and other operations.

This attack is feasable using a a cyber-physical rootkit such as Harvey described by Garcia, *et al.* Garcia et al. (2017a).

It is also assumed that training prints may be performed under supervised circumstances in which it may be reasonably assumed that no attack is taking place. This may be achieved by a direct connection between the controlling machine and the printer via USB. The materials supplier shown in Figure 3.1 is assumed to be trusted. Untrusted materials suppliers are beyond the scope of this paper. For the materials-based verification, the modified filaments with the embedded materials are to be supplied directly by the end user. Furthermore, all communication channels among trusted entities are assumed to be secure.

3.2.5 Use Case: Prosthetic Tibial Implant

For a specific use case example, the tibial implant portion of a prosthetic knee was chosen. Unlike the titanium alloy component of the prosthetic knee that attaches to the femur, the tibial portion of the implant is made from polyethylene and has been identified as a component that could easily be manufactured through AM Berman; kne (2017). Furthermore, the knee undergoes more mechanical stress than any joint Schmidler (2016). Thus much research has been conducted which describes the medical implications of its wear and tear Vandekerckhove et al.; Kilgus et al.. Therefore, an attack is considered in which alterations are made to the internal structure of tibial knee implant that would dramatically increase the rate of wear.

3.3 Verification Layers and Implementation

The main focus of this paper is to verify the unseen internal fill structure present in all 3D printed objects. When a print is converted from a design on a computer to G-code instructions for a 3D printer or CNC, an internal structure for the physical product must be generated. These can range from low density for prototyping or non-load bearing prints to high density for load bearing or industrial use. The fill itself may take on a honeycomb pattern, rectilinear pattern, or other various patterns as specified by

the user. Failure to produce the proper internal fill will render a final product that may externally look like the design intends, but fails to provide other required physical characteristics.

In order to develop a robust verification scheme, methods were needed that would allow for real-time identification and visualization of potentially malicious prints as well as visualization of a completed print to ensure its usability. Analysis of the acoustic side-channel was chosen as a non-intrusive method of identification. Instead of using traditional machine learning methods as have been used before, we use an audio classification scheme similar to popular apps used for identifying music. For real-time visualization, a method of tracking the moving components of a printer or CNC machine was determined to be a useful way of understanding the process without relying on control software. Finally, methods were borrowed from materials science by which the internal structure of an already completed print may be observed in a non-destructive way.

3.3.1 Side-Channel Verification

The side-channel analysis verification layers provide a means of verifying printed models in real-time. The goal is to infer as much information as possible from the given side-channels, but we do not expect each modality to be able to verify the entire print in itself. We will first discuss the experimental setup for each side-channel modality.

Acoustic Layer. As a physical byproduct of nearly any mechanical process, acoustic signals have been explored as a method of understanding information being processed by both traditional printers Backes et al. and 3D Printers used in AM Song et al.; Hojjati et al.; Chhetri et al. (2016). Because traditional printing methods now rely on lasers or ink jets, the information obtained from these is minimal. However, 3D printers will continue to rely on various actuators and fans for the foreseeable future which produce useful acoustic data. This is especially true for large-scale implementations of the technology.

In this verification layer, we assume that a particular design with a given infill structure will be printed multiple times. We use an open source audio classifier similar to

the Shazaam Avery Li-Chun Wang or SoundHound Applications. Using a training audio file, it locates noise-resistant peak frequencies and their temporal location within the file. It then locates frequency peaks in the test data that match the location, frequency, and spacing from other peaks. When a test file is identified, it is accompanied by a confidence score among other information. The confidence score indicates the number of peaks that the test has in common with the training data.

For AM verification, we use a single print as a training set by recording it with a microphone to obtain an audio file. Because even a simple print can take many minutes, the resulting file is separated into a number of segments of a given length (some number of seconds) and indexed in ascending order. Each indexed segment of the print is then trained as a different "song" and stored in a database. In many machine learning schema, common practice is to train multiple sets of data. However, because acoustic classification involves one-to-one comparison of audio files, a single-file training set is appropriate.

Test data is collected using the same method as training data and split into segments of the same length. Each indexed segment is then classified independently and a confidence score is returned. The confidence score represents the number of frequency peaks that a given file has in common with the training file. Verification that a repeated print is unaltered from the training set is determined in two ways:

1. The classification results are such that the index values appear in ascending order. If they are out of order, it is likely that a change has been made.

2. The confidence score of one or more indexed classification results falls below a given threshold value. The threshold value is referred to as the confidence threshold (CTh) for the remainder of the paper. Its value is optimized manually for each printer to maximize the true positive rate and minimize the false positive rate.

With this, a print will be considered verified if each indexed audio file is classified correctly, in the correct order, and with confidence values greater than the CTh. A non-verified print conversely will be classified but out of order or with one or more

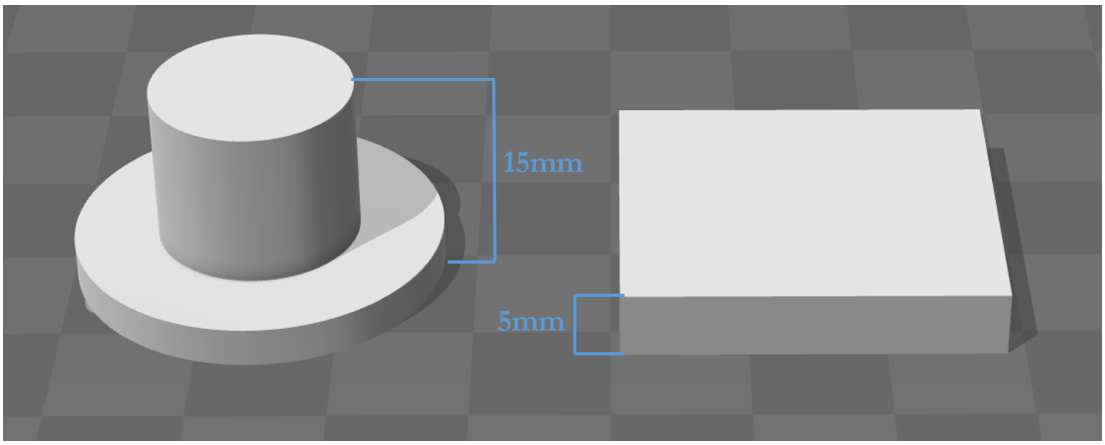


Figure 3.2: 3D Printed models described as (left) Top Hat and (right) Rectangular Prism.

confidence values less than CTh.

To test this method, two designs, shown in Figure 3.2 are used throughout this paper. They are described as a Rectangular Prism (right) and a Top Hat (left). Each was printed several times with "Honeycomb" and "Rectilinear" fill patterns of 20%, 40%, and 60% density. For each print style, a single set of audio data was split and stored in a unique database as described above.

In order to derive quantitative results to the test classifications, we assign a "score" to each segment of the audio data which are defined as follows:

- If a segment is in proper sequence and the confidence value is greater than CTh, its score is equal to that of the confidence value.
- If a segment is out of sequence, its score is equal to $-1 *$ confidence value.
- If a segment is in sequence, but the confidence value is less than CTh, its score is set equal to $-1 *$ confidence value.

If a negative score is calculated for any segment of the sliced audio file, a positive error classification may be determined. If no negative values are calculated, a negative error classification is determined.

Sample results are shown in Figure 3.3. The print is a Rectangular Prism with a 20% density Honeycomb fill pattern. The top chart shows the averaged results of three

known negative error classifications (true negatives). Each bar represents a 90 second slice of the printing data, and CTh is set to 35. Likewise, the bottom chart represents various positive error classifications (true positives) caused by incorrect fill densities or patterns. Each type of error is printed four times and the results are averaged. For errors involving the Honeycomb fill pattern with erroneous densities, a positive error classification is achieved within 270s or the first 60% of the print. For the erroneous Rectilinear fill pattern, positive error classification is achieved within 180s or 40% of the print. In each case, the first 90s slice is always receives high scores due to the fact that the design always starts with a 100% density fill of the first three layers. This is standard in 3D printing to ensure that the exterior is solid.

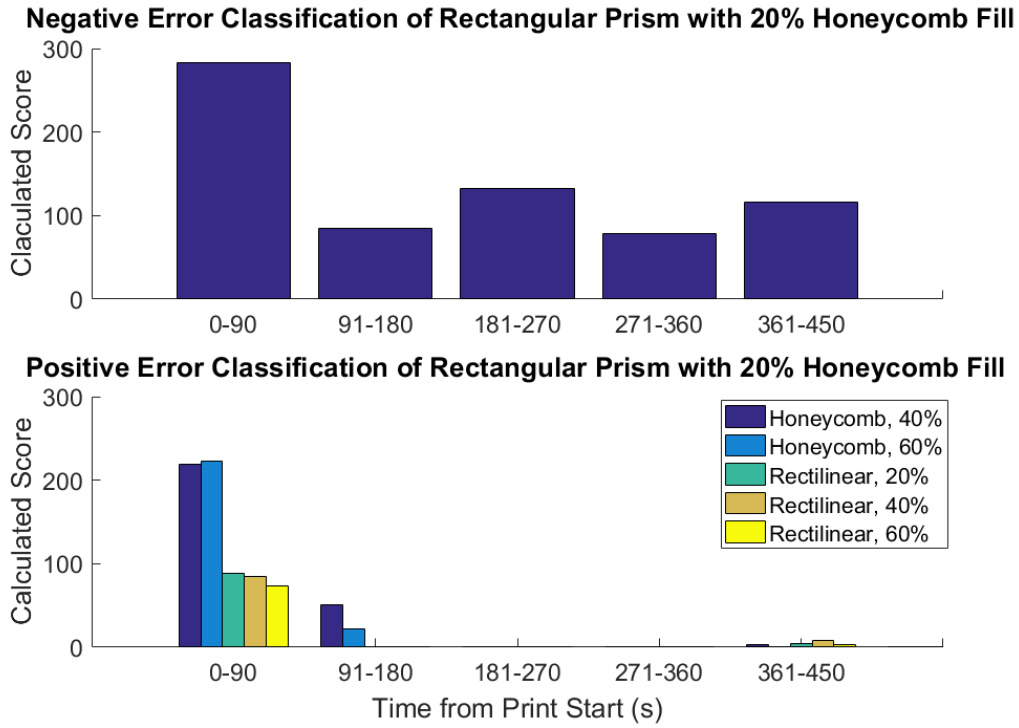


Figure 3.3: Classification example.

Spatial Sensing Layer. When performing 3D prints, it was found that the software used to monitor print progress simply displayed the progress of the G-code instructions being sent to the printer. This is regardless of the actual actions of the printer. The goal in setting up a spatial sensing verification scheme was to physically monitor the position of the printing nozzle with respect to the printing base, in order to observe

their actual positions throughout the printing process.

The first consideration was to use a ride-along accelerometer such as those described in section 3.2. However, due to the double integration from acceleration to position and the noisiness of the accelerometer data, visual representations of the printer's path became prohibitively difficult to obtain.

With this in mind, a scheme was developed in which the a gyroscopic sensor was paired with a linear potentiometer in order to construct a set of spherical coordinates to describe the printer's motion. This proved more effective because no integration was needed for the data, and only simple moving average filtering was necessary to reduce noise.

To obtain these measurements, the following devices were used: a Unimeasure linear potentiometer model number LA-PA-10-N1N-NPC, a SparkFun Triple Axis Accelerometer and Gyro Breakout MPU-6050, and a Teensy 3.2 board. The experiments were conducted in a setup as shown in Figure 3.4 with a Dobot Magician desktop CNC and 3D Printer. For experimental purposes, the actual 3D printing extruder was removed and "dummy" prints were performed. The test prints were a single layer of a circular disk printed with Honeycomb and Rectangular fills each with a 20% and 40% density. Data is collected at a rate of 100Hz. In Figure 3.5, each print is shown as the G-code representation next to the reconstructed path of the printer. The data shown is smoothed using a moving average filter with a window of five.

3.3.2 Materials Verification

The objective of our materials-based verification is to embed contrast agents that will act as signature markers for particular prints without compromising the structural integrity of the original model. The contrast agents are chosen based on the materials as well as the scanning modalities. This approach is similar to the approach presented by Le, *et al.* Le et al. (2016) for privacy-preserving techniques for secure point-of-care medical diagnostics in which they used synthetic beads with different dielectric properties for user identification. In our case, we embed a single type of nanoparticle at different points in the printed model to generate a pattern specific to the model.

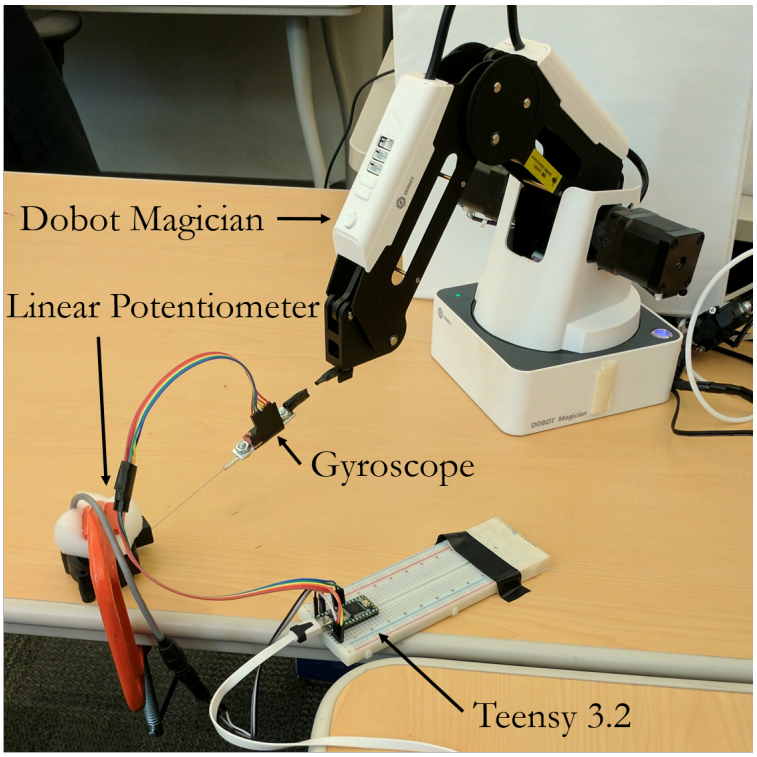


Figure 3.4: Spatial sensing setup with Unimeasure linear potentiometer model number LA-PA-10-N1N-NPC, SparkFun Triple Axis Accelerometer and Gyro Breakout, and Teensy 3.2 board.

This will allow us to ensure that the model was not modified by either an attacker who compromised the firmware and is duping the manufacturer, or a malicious insider who has physical access to the printing process. While it is arguable that embedded markers would change the integrity of the material itself, numerous studies have shown that the use of nanoparticles actually *improves* the materials' mechanical strength Wu et al. (2002); Crosby and Lee (2007); Fu et al. (2008); Liu and Webster (2010).

Here, we explore two types of scanning modalities: Raman spectroscopy and computed tomography (CT). Although both modalities are not necessarily cost-effective, our goal is to explore their effectiveness in our verification techniques. In both cases, we assume that the end user will provide the necessary materials to the manufacturer, who will be responsible for printing the model. The design sent to the manufacturer will not include any information about the embedded materials. We will now briefly discuss the different scanning modalities in detail.

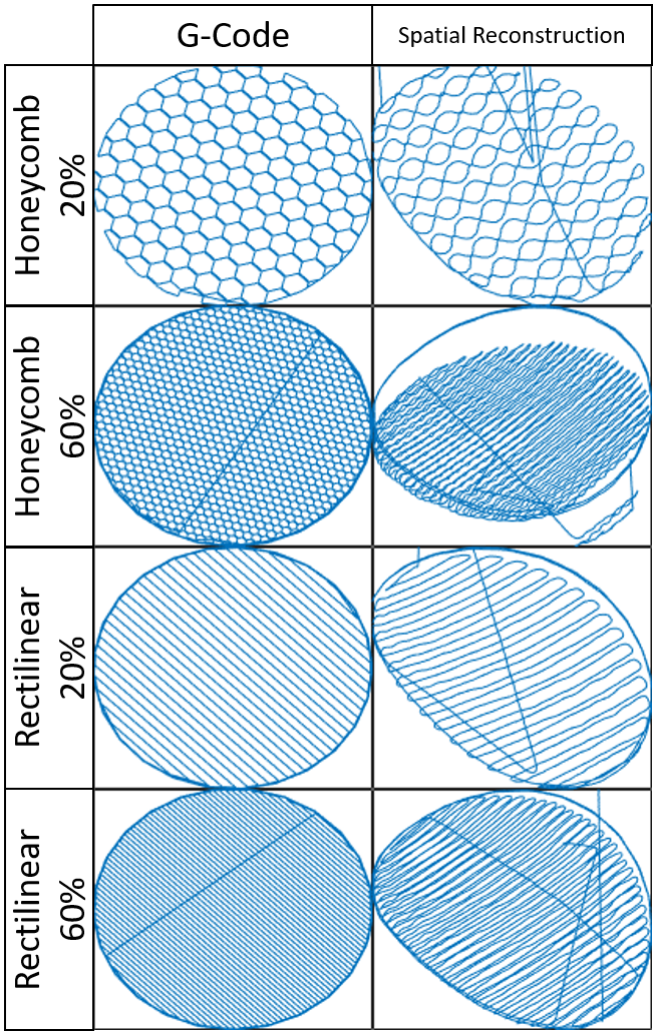


Figure 3.5: Comparison of G-code reconstruction to gyroscopic sensing reconstruction of single layers of various fill types and densities.

Raman Spectroscopy. The first of the aforementioned modalities is Raman spectroscopy, which has been shown to be applicable for specific target identification and quantification Nie and Emory (1997); Kneipp et al. (1997); Michaels et al. (1999); Le Ru et al. (2006); Qi and Berger (2005); Strachan et al. (2007); Zhu et al. (2007). The target sample is irradiated with a monochromatic light source such as laser. The majority of the scattering light has the same frequency of the incident light. This elastic scattering is called Rayleigh scattering. A small fraction of the scattering is inelastic. It has a small shift in photon frequency due to the energy transfer with the target molecules. When excited at a specific frequency, the target molecules can either increase or decrease in vibrational energy. Thus, the small fraction of the scattering light reduces

(Stokes shift) or gains (anti-Stokes shift) equally the energy of the molecule vibration.

Due to to the unique covalent bonds and atomic mass of the each molecule, different molecules require specific excitation energy to change the molecule vibration Lin-Vien et al. (1991). The combination of multiple energy shifts creates the unique spectrum for each target molecule. The distinct spectra can be use to identify the target molecule in Raman spectroscopy.

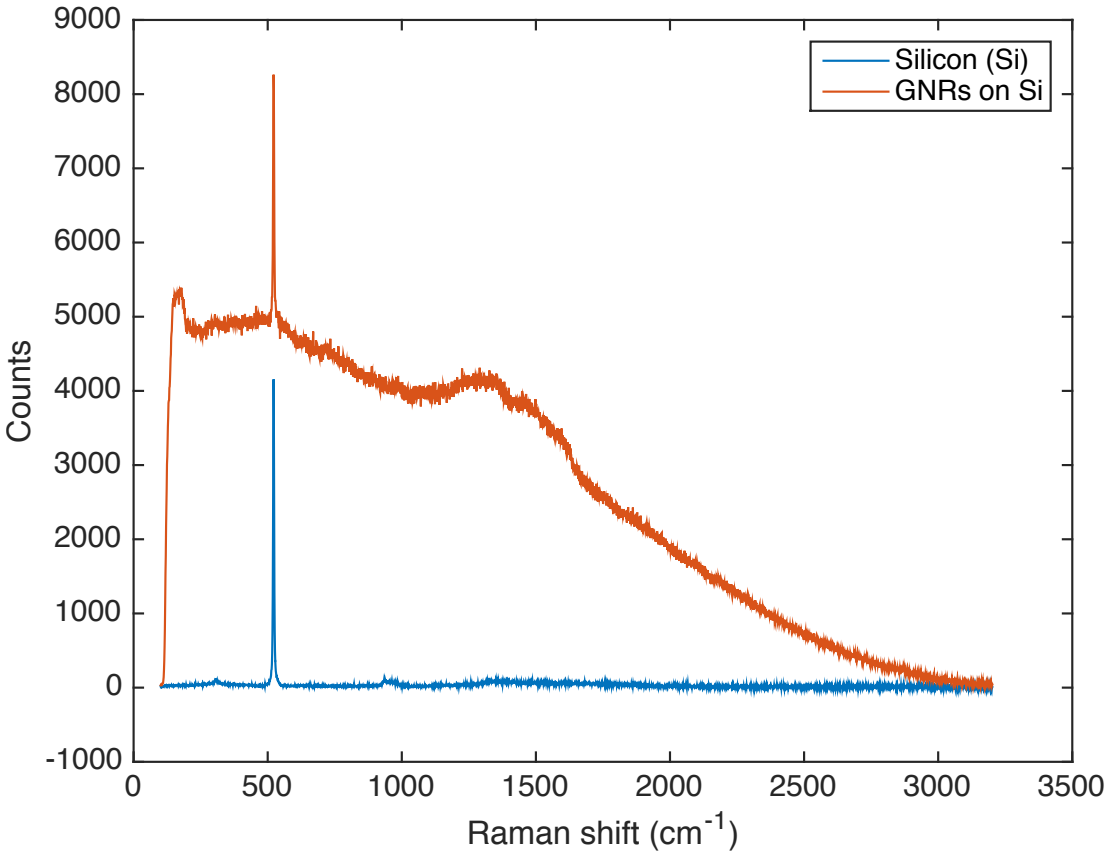


Figure 3.6: Raman scattering measurement of Silicon wafer with gold nanorods (GNRs) and 3,3'-Diethylthiatricarbocyanine iodide (DTTCI). The Raman spectrum of Si is amplified when using the enhancers.

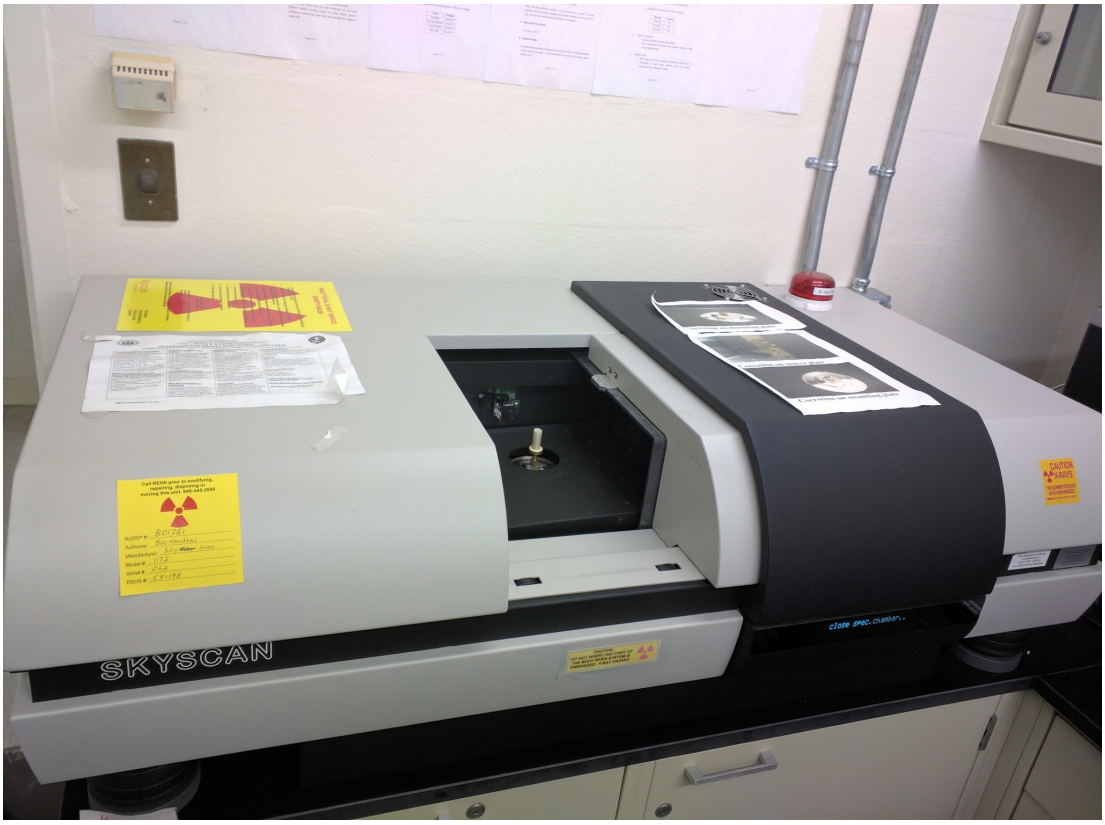
Contrast agents in Surface enhanced Raman spectroscopy (SERS) can be used to amplify the Raman spectra of the target samples. As the electromagnetic wave (laser) irradiates the contrast agent molecules, it excites the localized surface plasmons on the rough surface. This results in the enhancement of electromagnetic fields near the surface Fleischmann et al. (1974); Campion and Kambhampati (1998); Stiles et al. (2008). The increase in intensity of the electromagnetic fields would also increase the

intensity of Raman scattering. Thus, the Raman spectra is amplified. As a result, by coupling the contrast agents with the target molecules, SERS technique can be applied for identification of target molecules. Furthermore, SERS is also shown to be applicable for *in vivo* studying Qian et al. (2008); Huang et al. (2007). Qian, *et al.* has shown that pegylated gold nanoparticles can be used to target tumor cells in live animals in an *in vivo* study.

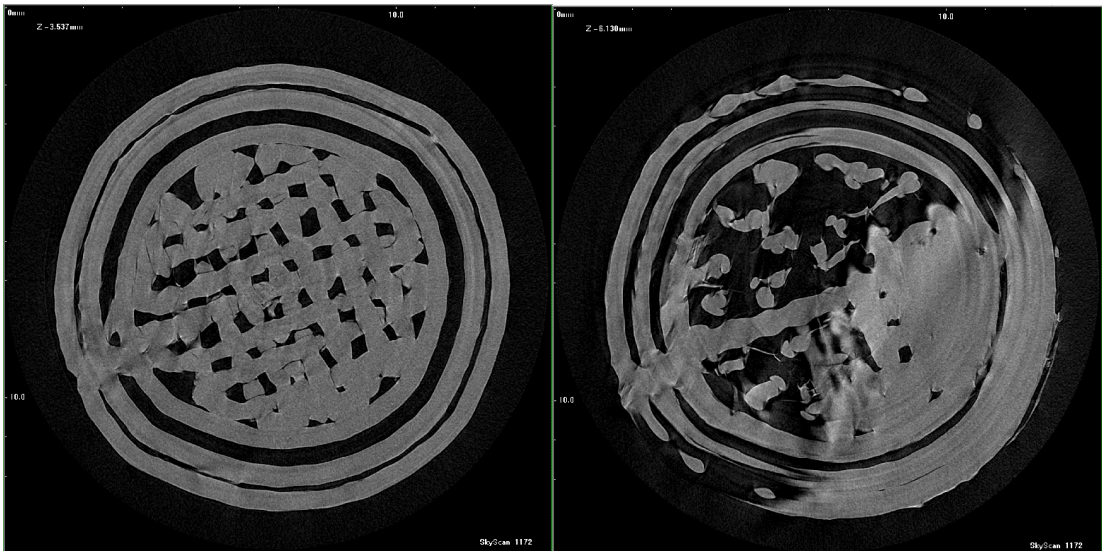
In this study, we utilize gold nanorods (GNRs - *Sigma Aldrich*) and 3,3'-Diethylthiatricarbocyanine iodide (DTTCI - *Sigma Aldrich*) as the two different contrast agents in SERS detections to verify the material of the 3D printed object. The contrast agent can be embedded in the filament at specific locations for material identification. The internal structure of the 3D printed object can be verified using the embedded materials. Figure 3.6 shows the result of the standard Raman scattering measurement of the Silicon (Si) wafer and the Raman scattering of GNRs and DTTCI drop coat on top of the wafer. The Si wafer is used to calibrate the Raman instrument prior to the experiments. The Si Raman spectra has been studied thoroughly Parker Jr et al. (1967); Temple and Hathaway (1973); Richter et al. (1981). In Figure 3.6, the GNRs and the DTTCI amplified the signal response of the Si Raman scattering intensity.

Computed Tomography. The second scanning modality is a computed tomography (CT) scan. Just as in the SERS experiment, we needed to find an effective contrast agent that would allow us to view the embedded materials within the 3D printed model. Because it has been shown that gold works as an excellent contrast agent due to its X-ray density Hainfeld et al. (2014) and because we already had the materials at our disposal, we decided to reuse the GNRs as our contrast agent. Furthermore, the GNRs' biocompatability will allow us to apply our verification procedures to the tibial prosthesis.

We initially experimented with the use of GNRs as a contrast agent for CT scanning by embedding them in a simple 3D printed model. We developed and printed a cylindrical 3D model using a standard acrylonitrile butadiene styrene (ABS) filament as the control material of the model. Multiple layers of ABS filament with embedded GNRs were deposited in between the bulk material.



(a) Skyscan 1172 MicroCT scanner.



(b) ABS control print. (c) GNR layer print.

Figure 3.7: CT scan of ABS cylindrical tube with embedded GNRs.

Figure 3.7 shows the initial results of the 3D printed model with a layer of injected GNR filament. We performed a CT scan using a Skyscan 1172 MicroCT scanner. As the figure shows, the GNRs did indeed contrast with the ABS filament. This was

sufficient to prove that GNRs could be used as a contrast agent for our printing use case. However, we will discuss in subsection 3.4.2 the limitations of the custom filament and as well as why we did not use the GNRs in our final evaluation.

3.4 Evaluation

In this section we evaluate the three-layered verification method. We describe the identification of a malicious print, the observation of the detected error, and the post-production materials verification. Then, we evaluate the effectiveness of the acoustic and spatial verification on the use case of a 3D printed tibial knee implant.

To quantify the accuracy of the results of the various tests, the data is fit into a logistic regression model with the binary dependent variable of "malicious print detected" or "no malicious print detected". From the model, we extract the probabilistic classification outcomes and create a receiver operating characteristic (ROC) curve. The area under the ROC curve (AUROC) is the metric used to predict classification accuracy.

Also, it is important to note that due to the fact that these machines are used to produce real 3D prints, large amounts of data were not practical to obtain. Furthermore, the imaging analysis techniques used for the materials verification were also time-consuming with limited availability. Therefore, sample sizes in this section will be significantly smaller than papers dealing with computer simulations.

3.4.1 Identification of Malicious Prints

In this section, we evaluate the usefulness of the proposed verification method in simply identifying an error in a potentially malicious print. This initial identification will be carried out primarily by the acoustic layer with redundancy in the spatial layer to reduce false classifications.

Classification Accuracy. In order to gain initial understanding of the parameters that affect the accuracy of the acoustic layer, several experiments were carried out with a small number of trials. The printers used in the tests were a Lulzbot Taz6, Lulzbot TazMini, and an Orion Delta. The AKG P170 condenser microphone was placed on a

stand as close to the moving extruder head without being knocked over by the moving components of the printer. The audio classifier is called dejavu Will (2017) and is an open-source project written in python.

In order to generate data useful for logistic regression, a vector of scores, **S**, is generated using the exact method as is described in subsection 3.3.1. For example, the components of **S** are what are shown in Figure 3.3. The vector **S** is of length n where $n = \lfloor \frac{\text{audio length}}{\text{audio slice length}} \rfloor$. We then calculate a print score, p, where

$$p = \sum_{n} S_n \ . \tag{3.1}$$

The value p associated with a given print now determines how likely the print is to be the same as the training print with higher values meaning more likely and lower values meaning less likely.

In Figure 3.8, the ROC curves are shown for the classification results of the Rectangular Prism design with Honeycomb and Rectilinear fills. The audio is segmented to 90 second and 120 second segments, each CTh = 35. The same original audio files are used whether the audio files are segmented to 90 seconds or 120 seconds. The Honeycomb and Rectilinear tests each consist of nine target prints and sixty malicious prints. The reason for the large number of known positive error classifications was that each print is considered an erroneous version of each other print.

The poorest performance was an AUROC of 0.7815 for the rectilinear fill with the audio segmented at 90 seconds. That was determined to be unacceptable especially considering the high likelihood of false positives. To find an explanation for the poor classification, the G-code was inspected. Upon investigation of the G-code which was generated by Slic3r, it was found 9 lines which specified x and y coordinates along with the extrusion rate were repeated 12 times each out of 15 layers needed to complete the print in both the Rectilinear and Honeycomb fill patterns. Also, upon investigating sequentially repeated blocks of code, it was found that blocks of G-code describing three entire layers were repeated twice during the course of the print. This symmetry was hypothesized to be the cause of the classification confusion.

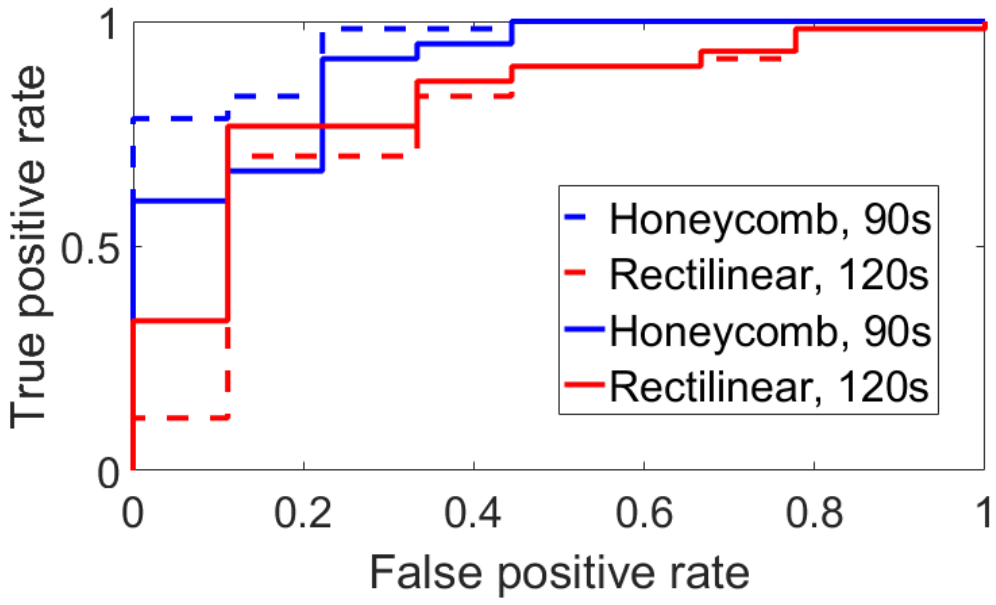


Figure 3.8: ROC Curve for Rectangular Prism, CTh = 35.

To test this hypothesis, a second set of tests were conducted with the Top Hat design, which is asymmetrical along the z axis. The same number of prints was performed with Honeycomb and Rectilinear fill being sliced to 90s and 120s each and CTh set to 35. The ROC curve of these experiments are shown in Figure 3.9. Each sample consists of nine target prints and sixty malicious prints, and the same data is used for the 90 second audio slice length as the 120 second slice.

Upon investigation of the G-code, the only repeated lines were those that define the nozzle speed at the beginning and do not include extrusion. Furthermore, there are no blocks of G-code or layers that are entirely repeated verbatim. This is suspected to contribute greatly to the increased performance seen in Figure 3.9. Here, least AUROC is 0.9852 which is suitable for verification purposes. Between the 120 second and 90 second slice lengths, we see little change in performance. Although audio classification is shown here to be effective in identifying malicious prints, it is still susceptible to both false positives. By introducing data from the spatial layer, these may be reduced. For instance, Figure 3.10 compares the data from the x, y, and z axes of the 40% Honeycomb and 40% Rectilinear fills from Figure 3.5. Here, we see a significant difference between

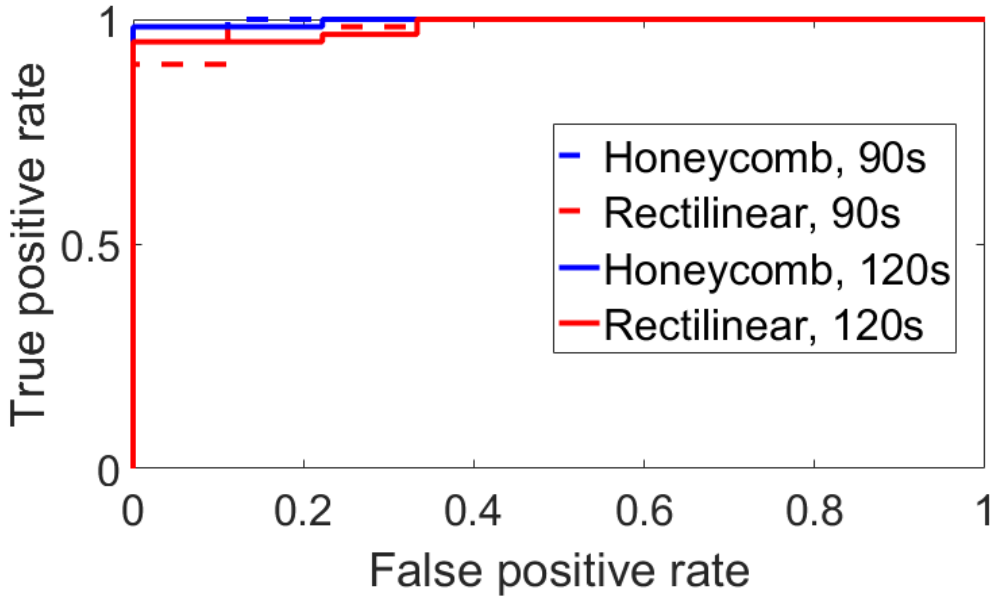


Figure 3.9: ROC Curves for Top Hat.

the two prints. Each frequency response has a similar shape, but the major features of the 40% Rectilinear fill are shifted to the right because the back-and-forth motion is not impeded by the creation of small Honeycomb structures.

For classification, the four most prominent peaks are used as features along with their locations. We conducted a test in which the target print was chosen to be the disk with 20% density Rectilinear fill shown above. All other prints were considered malicious. With this, we had 10 target prints and 12 malicious prints. Training using the linear regression model, an AUROC of 1.0 was achieved in differentiating between malicious and target prints.

While the spatial sensing layer is primarily for the purpose of print visualization, its role in conjunction with the acoustic layer allows for 100% accuracy in detecting malicious prints.

Varied Printer Models. In order to understand the effectiveness of audio classification for print verification on different printer models, several prints were performed on a Lulzbot TazMini and Orion Delta. Acoustic data recordings are obtained using the same microphone. In each print, a Top Hat design identical to the one described above was printed and the audio was sliced to 120s. The optimized CTh for the TazMini,

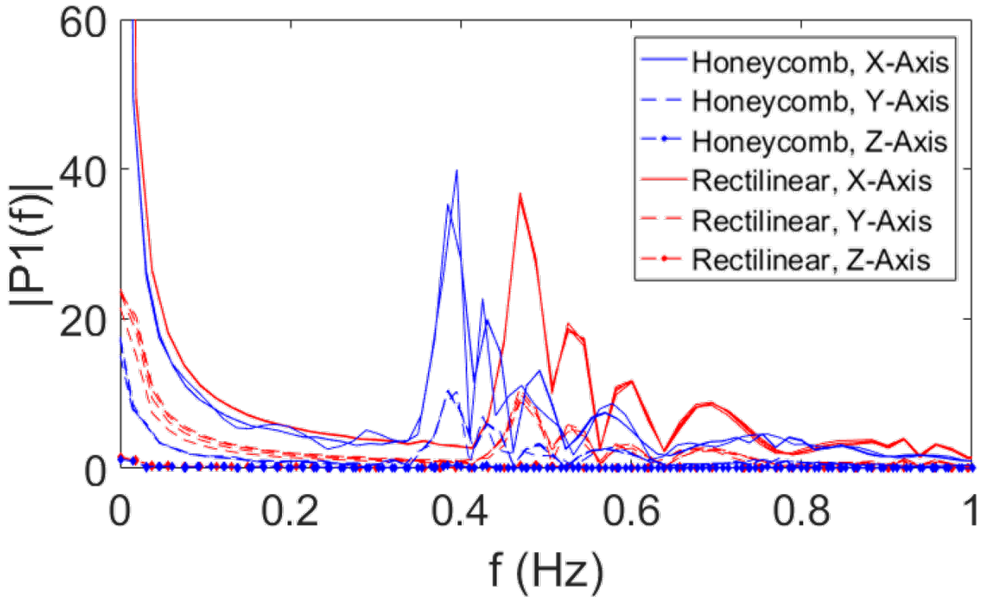


Figure 3.10: Comparison of the frequency response between a single layer of Honeycomb 40% fill and Rectilinear 40% fill. Four samples of each fill are compared.

Orion Delta, and Taz6 are 150, 20, and 35 respectively. The ROC curve results are shown in Figure 3.11. Because the Honeycomb and Rectilinear fill patterns are considered together, each data set consists of 18 target prints and 120 malicious prints. Consequently, the acoustic verification method is generalizable to printers of different sizes and configurations. The AUROC does not fall below 0.9542 in these tests.

Classification in Noisy Environments. Other experiments were conducted using an Afina H40 3D Printer with an eBoTrade Digital Voice Recorder wide-range microphone. This setup was in a noisy university makerspace with people talking near the printer. In this experiment, the classification accuracy suffered greatly (AUROC ≈ 0.5). Because it is shown that acoustic verification is useful on different types of printers above, we assume that the loss of classification accuracy is due to the noise in the environment. Also, because the microphone was wide range and not directional, the talking near the printer can be clearly heard. Therefore, in the implementation of this verification scheme it is important to use a directional microphone and noise isolation as much as possible.

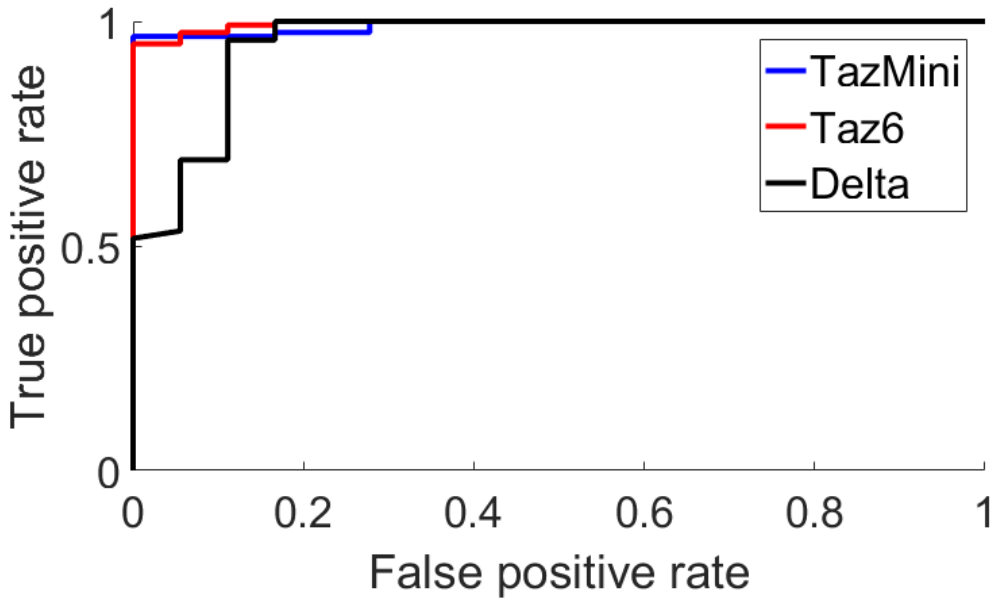


Figure 3.11: ROC curves for top hat design printed using a TazMini, Orion Delta, and Taz6 perint. Prints audio was sliced to 120 seconds and the confidence threshold is 150, 20, and 35 respectively.

3.4.2 Visualization of Malicious Prints

When a potentially malicious print is identified as described above, it is important to have the capability to visualize the potential threat. This visualization must be independent of the intended G-code which may be interpreted differently by malicious firmware. This is achieved in real time through use of the spatial sensing layer and in post-production by the materials inspection layer.

Real-Time Visualization. In the event that a potential malicious print is identified, a user has the capability of viewing the real-time print in progress through the spatial sensing as seen in Figure 3.5. By viewing the layer in progress, significant fill pattern changes such as those between the 20% Honeycomb and 20% Rectilinear fill are obvious. However, less obvious changes made to the print such as those between the 40% Honeycomb and Rectilinear fills are identifiable through FFT Analysis as in Figure 3.10. This is particularly true, as will be shown in subsection 3.4.3, if the user has access to the frequency response of a reference print.

While the spatial sensing layer is useful for identifying the type of fill pattern that

is being maliciously generated, it is less useful for identifying if the design itself has been altered due to the warping that occurs in the data. This, however, is an easy issue to solve through the use of a webcam which can easily identify the shape of the design. In this sense, it may seem that spatial sensing may be replaced altogether by a webcam, but it is important that the latter uses far more data and does not readily provide information about the frequency response.

Post Production Visualization. The aforementioned materials-based verification methods are meant to be generalized for any scanning method that can detect the embedded contrast material within a 3D model. In our case, we chose Raman spectroscopy and computed tomography because those modalities were readily available to us at the time of evaluation.

Given the results shown in Figure 3.6, we concluded that the GNRs and DTTCI can be combined for use as a contrast agent in Raman spectroscopy. The contrast agents amplify the photon count across the Silicon spectrum in Raman spectroscopy. To echo the results shown in Figure 3.6 for the 3D printed disk, we use 10 nm diameter GNRs 780 nm absorption, and DTTCI 765 nm absorption *(Sigma Aldrich)* diluted in ethanol as the two distinct contrast agents. Each contrast agent is drop coated on the surface of the 3D printed disk. The Raman spectra of the blank 3D printed disk is also taken as the control data.

To emulate the filament with the embedded contrast agent, we produced the filament from ABS pellets using the filament maker *(Filabot)*. For the GNRs embedded filament, the ABS pellets are submerged in a GNR solution and left to dry. In this test, a 4 mL GNR solution was mixed with 12 g of pellets. Based on the information from the manufacturer, we naively calculated the number of GNRs per mL of solution to be approximately 7.284e11. Per 12 g of pellets, we can produce approximately 2 m of filament with a 2.5 mm diameter. The 3D printed disk has 50 µm in layer thickness. Therefore, for the area of 1 μm^2 on each layer of the 3D printed disk, there are approximately 4 GNRs particles. This approximation only serves as the estimation of the GNRs within the measurement area. Due to the non-uniform mixing of the the GNRs in the pellets, the distribution of GNRs within the 3D printed disk varies considerably.

For the DTTCI embedded filament, while the quantity of DTTCI in the filament is not estimated, larger quantities of the DTTCI enhancer were available to produce the modified filament. The blank ABS filament is extruded using only ABS pellets.

Precise Embedding of Contrast Agents. In an ideal case, we would have the ability to embed the contrast agents or markers at precise Cartesian coordinates within the 3D printed models. However, for our proof of concept, we chose to simply create an ABS filament that was saturated in the GNRs or DTTCI throughout the entire spool of filament. The precise embedding of markers location is beyond the scope of current work. It can be explored in the near future. We then used a Lulzbot Taz dual extruder tool head to provide the capability of localize the embedded filament at precise locations.

In the following subsection, we evaluate the Raman spectra of the blank 3D printed disk, the 3D disk with GNRs or DTTCI drop coat on the surface, and the 3D printed disk with GNRs or DTTCI embedded filament. We wrote a simple C++ program that allowed the user to embed filament at desired locations by modifying the G-code where necessary, i.e., switching between the extruder nozzle containing the normal filament and the nozzle containing the GNR filament. The user can specify the beginning and end points of embedded material within the normal print path. This method was used for both the initial CT scan results as well as the final evaluation.

Imaging Analysis. In the evaluation using Raman spectroscopy, the 3D printed disk is excited with with 785 nm infrared light for 20 s per accumulation of data at 100 % power setting in Renishaw InVia micro-Raman system. Figure 3.12 shows the mean measurement results all data spectra of the 3D printed disks. Similar to the results from Figure 3.6, the spectrum of the 3D printed disk with DTTCI coated surface has significant improvement of photons counts across the spectrum comparing to the control data of the blank 3D printed disk. The spectra of the 3D printed disk from DTTCI embedded filament also shows the elevation of photons counts comparing to the control data. These spectra fall in between the spectra of the control data and the surface coated 3D printed disk. This conforms with the fact that the surface coated would accumulate more contrast agent at the measurement site comparing to the embedded

filament. While the Raman spectroscopy can be used to quantify the concentration of the target particles, the elevation of the photons count in Figure 3.12 does not reflect the approximate distribution of contrast agent embedded in the filament. The measurement site in Raman spectroscopy might be a cluster or spare of contrast agent or markers. As mentioned above, the markers might not be uniformly distributed in the filament. This is confirmed in Figure 3.7c as a result of the MicroCT scanner. The high reflection in the CT scan shows the large cluster of the GNRs in the embedded filament. Due to the low resolution of the MicroCT scanner, the scan would not highlight the areas where the GNRs are sparsely distributed. While the Raman spectroscopy results of the GNRs embedded filament are not shown, the similar response can be discerned.

In classification of 3D printed blank ABS, GNRs embedded, and DTTCI embedded disk, mean and standard deviation of the spectra are used to distinguish the cluster of data set. Figure 3.12 shows the mean of the typical response of Raman spectra of 3D printed disk with blank ABS, DTTCI coated disk, and DTTCI embedded ABS filament. By observation, the greatest change of Raman shift is in the range of $100cm^{-1}$ and $800cm^{-1}$. The details of the Raman scattering separation can be seen in Figure 1 in section .1. This is in the range of $791.21nm$ and $837.60nm$ scattering; whereas the sample is irradiated at $785nm$. Therefore, this is the reasonable range of interest for Raman scattering for all data selection. By training the logistic regression model, the classification using mean and standard deviation shows 100 % accuracy against the blank ABS (226 samples) filament for both GNRs (179 samples) and DTTCI (71 samples) embedded filaments.

In Raman spectroscopy, the maximum setting depth penetration for the Renishaw InVia micro-Raman system is approximately 300 µm, we cannot verify the 3D printed object where the GNRs or DTTCI embedded filament is implanted further inside the object. Therefor, the Raman spectroscopy would not be sufficient for the verification that require depth. In further analysis, we use the MicroCT scanner to evaluate the internal structure of 3D printed objects.

The initial results for the CT scan approach presented in Figure 3.7 showed that although the GNRs embedded filament contrasted well in the CT scan, we could not

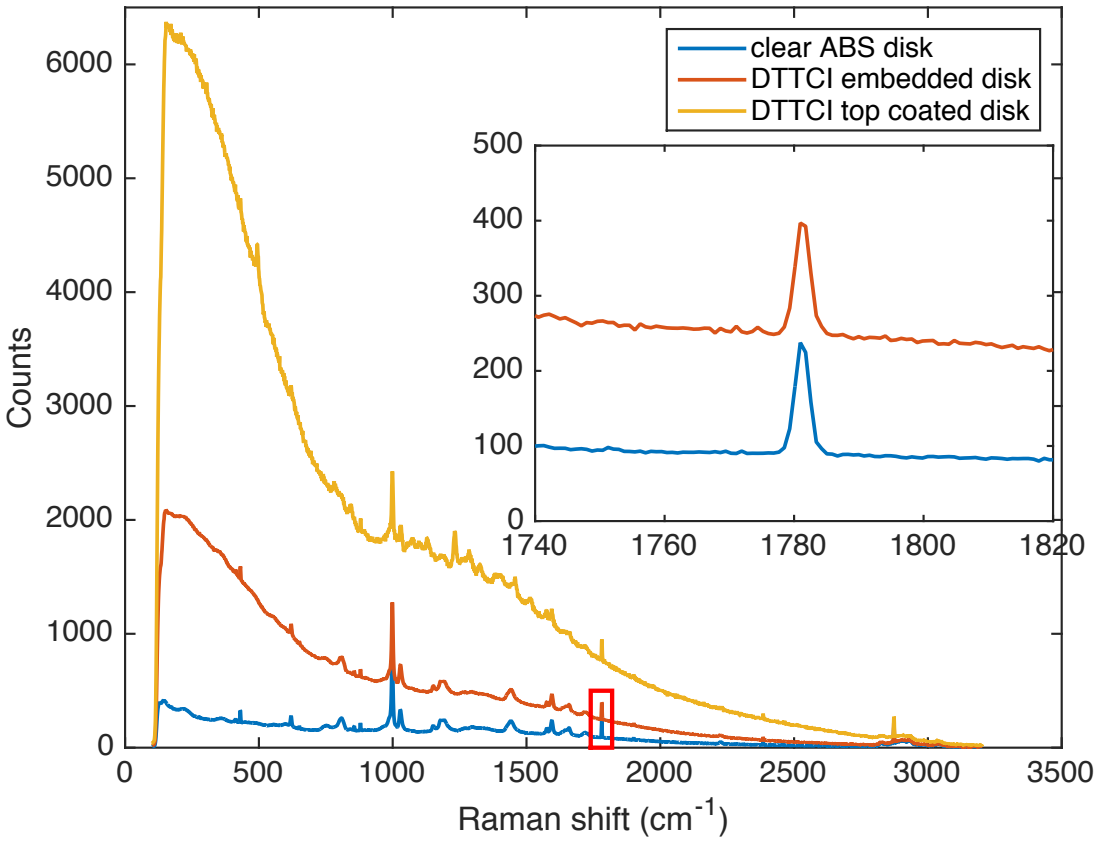


Figure 3.12: Mean measurement of Raman scattering of 3D printed disks using acrylonitrile butadiene styrene (ABS) filament and ABS with gold naonorods (GNRs) and 3,3'-Diethylthiatricarbocyanine iodide (DTTCI) embedded.

rely on the custom filament due to the sparse distribution of the GNRs. We did not have the equipment nor the expertise to manufacture a heavily saturated filament.

For a more precise proof of concept, we used commercially available stainless steel filaments where the filament is heavily saturated with stainless steel particles. Under the CT scanning, the steel particles would produce similar response to the GNRs due to high X-ray density. Although stainless steel is not biocompatible, it will serve as a substitute for the GNRs in order to provide precise visibility in the CT scan. Furthermore, we changed the control filament from ABS to polylactic acid (PLA) after comparing the densities in the CT scan. The X-ray properties of PLA versus ABS have been studied Veneziani et al. (2016), but we confirmed our assumption after simple trial and error. Figure 3.14 highlights the contrast in X-ray densities between the PLA filament and the stainless steel filament. We will discuss in the subsequent section how

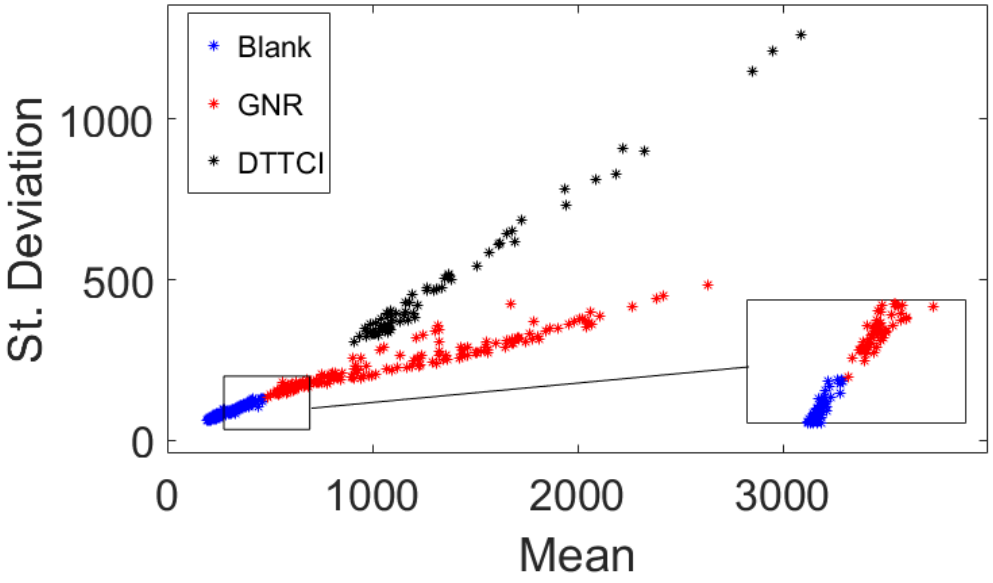
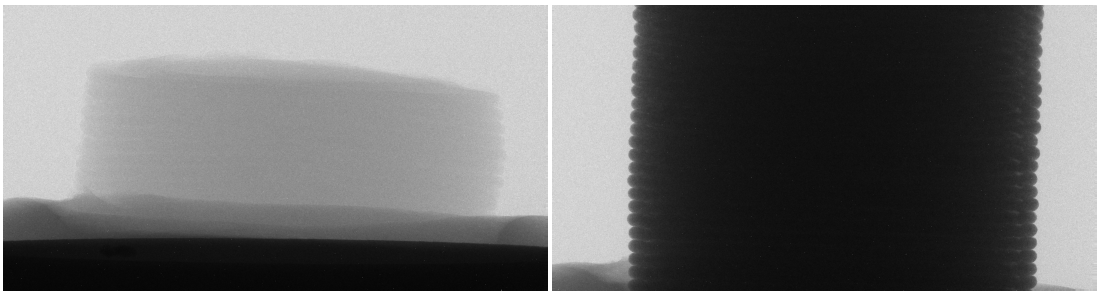


Figure 3.13: Classification of blank acrylonitrile butadiene styrene (ABS), gold nanorods (GNRs), and 3,3'-Diethylthiatricarbocyanine iodide (DTTCI) dye embedded filament in 3D printed disks.



(a) PLA filament. (b) Stainless steel filament.

Figure 3.14: Comparison of X-ray densities of PLA and stainless steel filaments.

we evaluated this approach on a tibial prosthesis.

3.4.3 Case Study: Prosthetic Knee

As described in subsection 3.2.5, a model of the tibial component of a prosthetic knee implant was used as a design for a use case test. Prosthetics differ slightly between patients, so we assume that malicious print identification is performed periodically with a known standard prosthetic design. Real-Time and post-production visualization are still performed on each print.

Error Identification. The acoustic verification results are shown in Figure 3.15 which

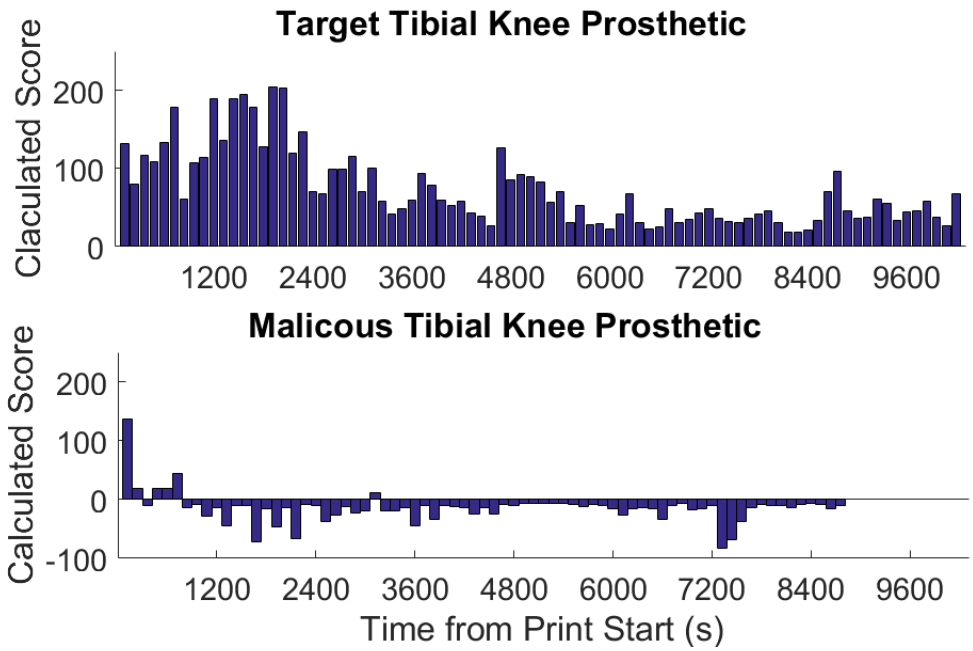


Figure 3.15: Comparison of target 60% Rectilinear Fill Tibial Prosthetic print acoustic classification (Top) vs. malicious 20% Honeycomb Fill (bottom). CTh = 0.

shows the confidence values of both the target print and the malicious print. These results are gathered using the same technique as those described in section 3.3 with audio slices of length 120s and CTh = 0. By setting CTh = 0, we see that a positive error classification can be made within the first 360s of the print or the first 4% of the total known print time by only observing out-of-sequence index classifications. The CTh may be set to anything less than 18 without causing a false positive. Overall, acoustic error detection itself saves over 2 hours of print time and prevents a potentially harmful print from being completed. A detailed table of the results shown here can be found in section .2.

In Figure 3.16, the FFT of a target print and a malicious print are compared to a training print. Similar to Figure 3.10, the malicious print shows a different frequency response near 0.2Hz as highlighted by the lower box. The upper box highlights the closeness of the peaks between the training and target prints and the difference between those and the malicious print. The full print of the object requires 111 layers, so it would take less 1% of the time of the total print to identify the erroneous pattern once

it begins.

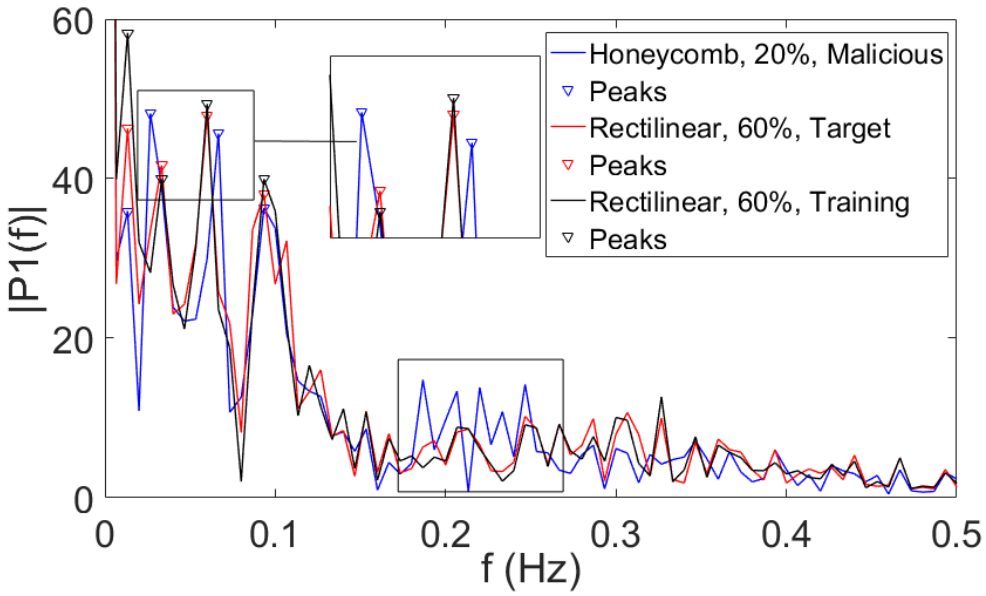


Figure 3.16: Comparison of x-axis frequency response for a layer of a layer of the tibial knee implant design.

Real-Time Visualization. In this test, the target print uses a 60% Rectilinear fill and the malicious print uses a 20% Honeycomb fill. In the attack, the visualization of the intended G-code remains unaltered for the user while the instructions sent to the printer are altered. The consequences of this attack would be to cause accelerated wear in the implant causing pain and financial loss for the victim who has the implant.

For the print identification and real-time visualization tests, a full sized prosthetic design is used. However, due to the size limitations of the MicroCT scanner, a significantly scaled down version of the same design is used.

The training, target, and attack prints were each recorded on the Lulzbot Taz6 printer. Due to the availability of the experimental setup, a single layer of each of these prints was performed by the Dobot Magician for the visualization tests. The exact same G-code was used for the Dobot prints as in the Taz6 with the exception of the extruder being disabled and the speeds decreased to suit the capabilities of Dobot. It should be noted that spatial verification testing is entirely plausible on the Taz6 which has a moving base because the measurements describe the relative position between the nozzle and the base. This is regardless of whether that base is a stationary table

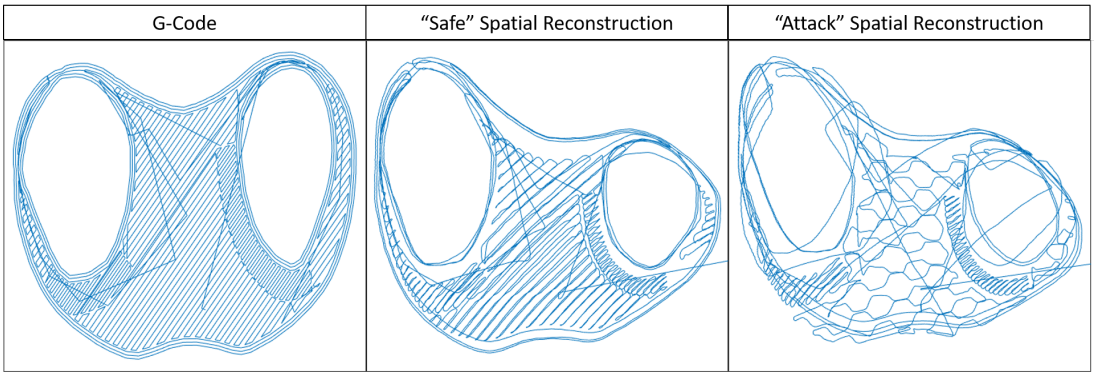


Figure 3.17: Comparison of target and malicious tibial knee implant prints. Left: G-code reconstruction of 60% Rectilinear fill, Middle: Spatial reconstruction of 60% Rectilinear fill, Right: Spatial reconstruction of malicious 20% Honeycomb fill.

or a moving part of the printer. It should also be noted that both acoustic and spatial verification would ideally be performed in tandem, but for testing purposes here, they are not.

Figure 3.17 shows the spatial verification visualization of, in order of left to right, a G-code visualization of the training print, a spatial reconstruction of the target print, and a spatial reconstruction of the malicious print. It is clear that the recreated target print uses a rectilinear fill at approximately the correct density while the malicious print differs significantly from the intended G-code. Due to the warping that occurs in the spatial reconstruction, a user would not be made aware if the shape of the print were altered by using this method alone.

Post Production Visualization. We only considered the CT scan approach for the post production visualization as the Raman spectroscopy would not be able to verify the internal structure of the tibial prosthesis due to its depth limitations. Figure 3.18 shows an X-ray scan of the front of a PLA tibial prosthesis with 2 infill layers of steel. Because we had to use a MicroCT scanner, the part of the tibial insert was scaled down to fit within a diameter of about 30 mm. The two large blotches of stainless steel are simple imperfections that mark points where the second extruder began printing.

Figure 3.19 compares the G-Code representation of the intended print of the top stainless steel layer–with the stainless steel path highlighted in red–versus the CT scan of that layer at a 15 μm/voxel resolution. The CT scan image is rotated about 45

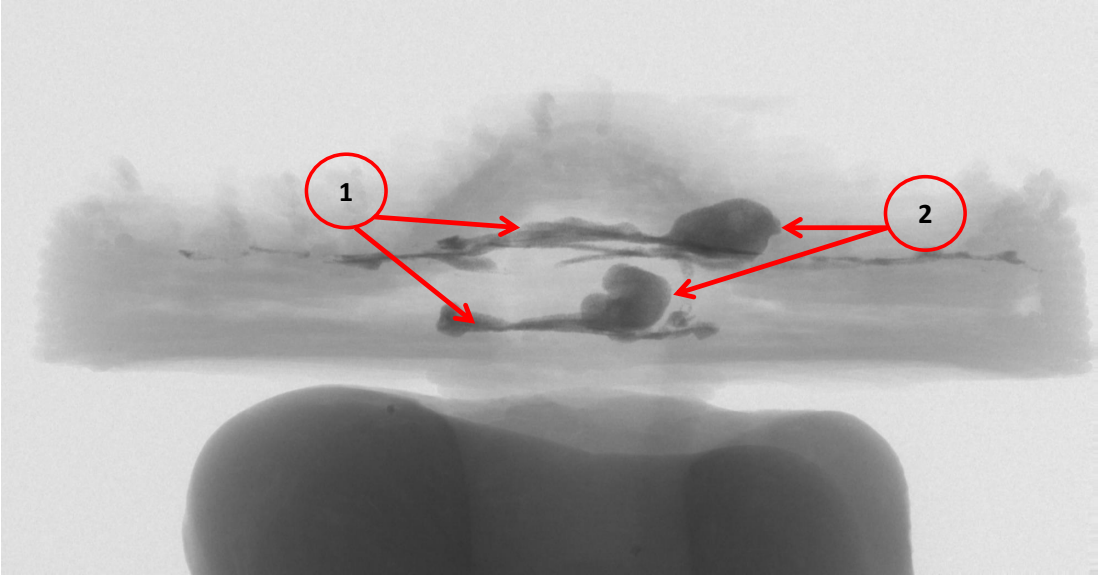


Figure 3.18: X-ray scan of front of PLA tibia with embedded stainless steel at a 15 μm/voxel size resolution. The first label shows the side view of the cross-sectional stainless steel infill, while the second label shows the two blotches where the stainless steel print began.

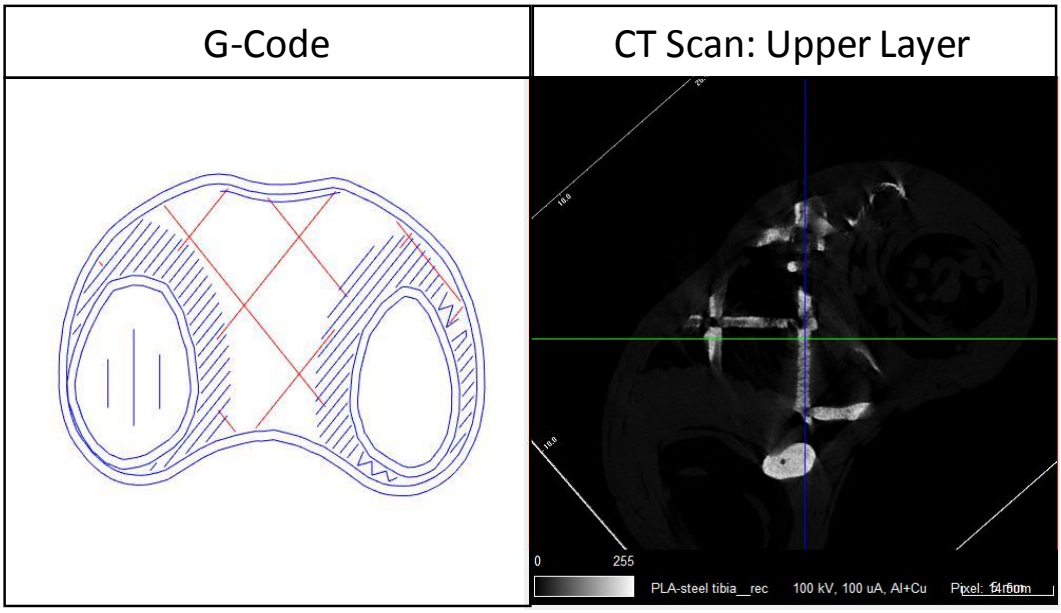


Figure 3.19: Comparison of G-code simulation of embedded steel (shown as red lines) versus CT scan of the printed model. The CT scan image is rotated about 45 degrees.

degrees in comparison to the intended print. Furthermore, the small model had to be mounted on a bed of silicone polymer to hold it in place, so it is not completely level. Despite the imperfections of the printed model and the scans, it can be seen that the steel was properly embedded within the walls of the model and is clearly detectable against the PLA filament.

3.5 Discussion

In this section, we discuss the various methods of implementing the proposed verification scheme. We then briefly discuss its limitations.

Implementation. The three layer verification and malicious print detection scheme described here is most readily suited for a mass production AM scenario. In this setting, many different standard designs may be produced using the same equipment. If each design is printed identically, then the acoustic layer, spatial sensing layer, and materials verification layer may be applied to each individual print.

In a setting such as the one described for the case study in subsection 3.4.3, a base design may be modified for each print in order to adjust for biological parameters, etc. In this scenario, the user could train a known standard print and periodically test the printer for any malicious activity. This periodic test could include all three layers. Each specialized design, then, could be monitored using spatial and materials verification for real time and post production detection of malicious activity.

Finally, this verification scheme may be used in a scenario in which an end user sends a design to a third party to be printed. For the materials verification layer, she may send a specialized filament with embedded trackers to be used. If the object returns without the trackers or with trackers in the wrong locations, malicious activity may be detected. Also, using a secure live streaming connection, the user may receive data from the print in progress and perform any classification or analysis herself.

The experiments presented in this paper focus primarily on on the detection of subtle changes in the internal fill pattern. Therefore, it is logical that more significant changes such as holes in the fill pattern or changes in the overall design will be easily

detected.

Limitations. As with any verification schema, the system proposed here is not without limitations. The immediately obvious limitation is that the ability to detect a deviation from a training print decreases as the similarity to the print increases. However, drawn to its logical conclusion, this means that an attacker wishing to exploit this limitation would be forced to change the design in such a small way as to not affect its usefulness. Another limitation could be the need for a training print. This may be a minor issue in the mass production scheme described above. In a scenario such as the production of prosthetics, however, the periodic checks for malicious activity may be seen as time consuming. Finally, if a third party printing service implements these methods, some cost overhead will incur from the purchase of microphones, sensors, etc. However, these costs are relatively cheap considering that any major equipment such as a spectroscope or CT scanner would be in the domain of the end user.

3.6 Conclusion

Three layers of verification for AM are presented for a case in which either a control PC or printer firmware is compromised. Acoustic verification uses audio classification to determine whether a print matches a previously known print. Spatial verification provides a visualization of the print in real time along with data for frequency analysis of the printing process. Materials verification determines whether the correct materials were used and whether indicator patterns appear in the proper locations. Each layer is independent of firmware or a controller PC.

Acoustic and spatial verification are found to be useful for confirming the intended fill pattern and density in a print, and material verification is found to be most useful in determining that the correct material is used and that the design is free of tampering.

Future work will include improving the acoustic and spatial classification methods so that they work independently of human interaction and in real-time. Similarly, the materials verification methods presented in this paper could be tuned for domain-specific solutions to be more precise. This would facilitate automated materials verification

solutions.

Chapter 4

Formal Verification of Hybrid Controller Logic for Transient Stability in Hybrid Systems

4.1 Introduction

There has been an increased emphasis on modeling cyber-physical systems (CPS) as hybrid systems to prove their associated safety properties. The continuous dynamics of such systems are characterized by ordinary differential equations (ODEs) and are typically governed by discrete state transitions that modify the continuous evolution of the system. Proving that a safety property will hold true throughout the evolution of the entire system–i.e., that the system will not evolve into an unsafe state–is not a trivial task, especially when the ODE contains non-linear dynamics. The problem is further exacerbated when the dynamics contain transcendental functions such as the sine and cosine functions. Solutions to such ODEs are often not available due to the complexity.

Because the solutions to such ODEs are rarely attainable, alternative approaches are necessary in order to reason about the safety properties of such systems. An engineering approach may be to linearize the models such that solutions to the ODEs become feasible Hale and LaSalle (1963). However, such approaches may neglect critical aspects of the systems dynamics and may lead to false negatives in terms of reachability analysis of the safety properties. A more sound approach for such analyses is to reason about the ODEs without solving their initial value problems.

Previous approaches that prove properties of systems with non-linear ODEs without solving their initial value problems have fallen into two categories: those systems that soundly abstract the ODEs and perform a reachability analysis on the resulting abstraction, as well as those systems that perform a deductive verification based on reasoning

about the invariant properties of the continuous system. For the latter approach, a critical bottleneck has been the automatic generation of invariants that can sufficiently prove properties of such continuous systems. Previous works have provided significant leaps towards the automatic generation of invariant algebraic sets for polynomial vector fields that represent invariant properties of such hybrid systems Ghorbal and Platzer (2014) Sogokon et al. (2016) while preserving soundness. These invariant properties can then be used to develop formal proofs for such dynamical systems. However, for particular systems, the feasibility of such approaches depends on the ability to soundly abstract transcendental functions, e.g., trigonometric functions.

Deductive verification approaches typically handle transcendental functions by introducing fresh variables to the polynomial system such that non-linearities are eliminated Powers (1959). Such techniques have been referred to as recasting Savageau and Voit (1987) Papachristodoulou and Prajna (2005), polynomialization Kerner (1981), or differential axiomatization Platzer (2008). Although these abstractions are sound and sufficient to generate semi-algebraic invariants of the system, there are cases where the proof of the overall hybrid system requires access to the components that were abstracted away by such techniques.

In this chapter, we complement previous automatic deductive verification proof techniques to automatically reason about properties of the continuous systems that may have been previously abstracted away by the recasting of transcendental functions. Although the aforementioned fresh variables provide a sound abstraction of the transcendental functions, they may represent larger set of functions whose properties are not sufficient enough to prove certain properties of the original transcendental functions. Our approach narrows the number of possible solutions for the recasted polynomial continuous system to those solutions that include the transcendental functions. In particular, we leverage properties about the Taylor Series approximations of the abstracted transcendental functions in order to access properties of the previously abstracted transcendental functions.

As a motivating example and a relevant use case, we use deductive verification techniques to model and automatically verify the safety of a simplified electric power grid

system with respect to its transient stability. We use a semi-automated approach to generate the semi-algebraic invariants of the polynomialized model that must hold true throughout the evolution of the system. We then use our aforementioned approach to formally and automatically prove the hybrid system using a sound and relatively complete logic, differential-dynamic logic (d$\mathcal{L}$). In particular, we verify safety properties of the CPS model associated with a relay logic controller used for out-of-step protection of a generator. Out-of-step protection is the controlled islanding of power system components to mitigate disturbances in order to avoid widespread outages and equipment damageTziouvaras and Hou (2004). For our physical model, we formally model and verify the large-disturbance rotor angle stability of a single-machine to infinite bus system (SMIB), a simple model that typically serves as the starting point for transient stability analyses as well as various recent CPS attack modelsPasqualetti et al. (2012)Chen et al. (2014). As shown in these attack models, the physical evolution of the power system depends on the cyber-physical interaction with the associated cyber network topology. For this model, we analyze the safety property associated with the relay's critical clearing time of a three-phase fault on our SMIB system. In the event of a fault, the system must clear the fault within a critical clearing time period. Therefore, the timing constraints of programmable relay must abide to the generated safety requirements of the system. The invariant characterization of the system allowed us to provide a symbolic implicity relation that gives an accurate estimate for the critical clearing time of our SMIB system. However, as we will discuss in later sections, the critical clearing time is determined by information that is previously abstracted due to the recasting of the system's trigonometric functions. This is a preliminary work that serves as a basis for verifying larger and more complex power systems using the invariant properties of the physical equations. Our simple case study of opening and closing circuit breakers on a line can also be abstracted to more complex switching operations that have more discrete states, such as intelligent load shedding schemes.

Contributions. The approach presented in this chapter complements previous deductive verification approaches for systems governed by non-linear ordinary differential equations with mixed polynomial and trigonometric functions under semi-algebraic

evolution constraints whose proofs require semi-algebraic invariants. It works as a preliminary work to assess the how to automatically generate the safety constraints for a complex industrial control system such as the electric power grid. We provide the following main contributions:

- We present a semi-automatic and generalizable technique for recovering information of recasted transcendental functions that is necessary to prove properties of the overall system.
- We utilize Taylor series approximations to provide bounds on the fresh variables used in the polynomialization of the system such that they can soundly provide sufficient conditions to prove properties of the previously abstracted information.
- We automatically generate proof tactics that incorporate the recovered information in order to prove the aforementioned safety properties of the system.
- We present a use case study on the transient stability of a simplified electric power grid system. We specify and verify the safety of its associated hybrid systems model and verify our results against previous power systems analyses.
- We enumerate the limitations in the current state-of-the-art for automatic verification of such hybrid invariants.

This chapter is organized as follows. Section 4.2 provides preliminary information about transcendental functions as well as rotor angle stability. Section 4.3 presents the our formal approach using an SMIB case study. We discuss the limitations of our approach and the current state-of-the-art for automatic verification in Section 4.4, followed by a conclusion in Section 4.5.

4.2 Preliminaries

In this section we will present the background information required to understand the contributions of this paper. We first discuss a few concepts regarding the handling of non-linearities in ODEs with respect to transcendental functions as well as the formal

specification language used in this paper. We then present the electric power grid system that will be used as an ongoing example for the rest of the paper.

For the purpose of consistency, we will use the same notions of notation as previous related works Sogokon et al. (2016) for sets and formulas characterizing those sets. The notations will be used interchangeably, e.g., H will denote both a semi-algebraic set $H \subseteq \mathbb{R}^n$ and a formula H in first-order theory of real arithmetic with free variables in $x_1, ..., x_n$ that characterizes this set. Additionally, we will also only consider autonomous systems of polynomial differential equations under semi-algebraic evolution domain constraints, i.e., systems of the form:

$$\dot{x}_i = f_i(x), x \in H \subset \mathbb{R}^n \tag{4.1}$$

where $f_i(x) \in \mathbb{R}[x_1, ..., x_n]$ for $1 \leq i \leq n$ and the evolution domain constraint H is semi-algebraic. We will similarly use the concise vector notation of $\dot{x}_i = f(x)\&H$.

4.2.1 Recasting of Transcendental Functions

Transcendental functions such as trigonometric and logarithm functions introduce non-linearities in a system of ordinary differential equations Powers (1959). In addition to the elimination of non-linearities, such techniques do not compromise generality as time dependences can be transformed into autonomous systems by introducing fresh variables to model time evolution. However, there are cases where such transformations require additional conditions on the evolution domain constraint, initial conditions, as well as the proof tactics associated with a hybrid system. We will detail an example that will present such dilemmas.

4.2.2 Formal Specification

In this work, we use differential Dynamic Logic (d$\mathcal{L}$) Platzer (2010) as our first-order logic to model and formally verify hybrid systems. The d$\mathcal{L}$ approach comes with a formal programming language to model hybrid systems. The formalized models are called hybrid programs. The language extends the imperative constructs of Dynamic

Logic Harel et al. (2000) with additional constructs to encode constrained continuous evolutions. The model reads as a list of actions that the system has to perform to achieve its goals. The Dynamic Logic modal operators "[]" and $\langle\rangle$ are used to formally describe the behavioral properties the system has to verify. If α denotes a hybrid program, and ϕ and ψ are predicates, then the necessity proposition

$$\phi \rightarrow [\alpha]\psi$$

means "it is necessary that, if ϕ is initially satisfied, ψ holds true for all the states visited by executing the program α". One therefore sees how safety properties can be encoded for a model α.

Once the model has been specified as a hybrid program using d$\mathcal{L}$, an associated sequent calculus is used to symbolically compute the effects of hybrid programs and successively transform them into logical formulas describing these effects by structural decomposition. The set of propositional rules, rules for dynamic modalities, and quantifier rules are enumerated and defined in Platzer (2010).

4.2.3 Rotor Angle Stability of An Electric Power Grid System

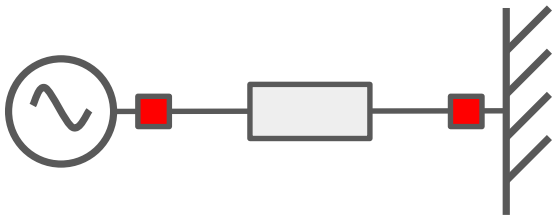


Figure 4.1: Single-Machine to Infinite Bus (SMIB)

Figure 4.1 depicts the SMIB system that will be modeled and verified in this paper. A synchronous machine, e.g., a generator, is connected to an infinite bus by a three-phase transmission line with a reactance X_e. At both ends of the transmission line, there are circuit breakers that can open the transmission line. As is typically done with the swing equation, a number of simplifications are made in order to make the equation useful in practiceAndersson (2004): the synchronous machine is modeled as a constant electric magnetic field behind the transient reactance; voltages and currents

are assumed to be symmetrical; the angular velocity is close to the nominal one; static models for the transmission lines are used; the mechanical input power for the prime mover is constant; the circuit breakers can open and close a line instantaneously; and all resistances in the synchronous machines, transmission lines and transformers are neglected. The angle of the electromagnetic field is assumed to coincide with the *rotor angle*, δ. The mechanical rotor angle depends on the balance between the mechanical torque due to the mechanical input power for the prime mover and the electromagnetic torque to the synchronous machine electrical output power. The mechanical rotor angle acceleration forms the basis for the swing equation in which the balance between the mechanical power acting on the rotor and the electrical power acting on the rotor determines the acceleration. This mechanical rotor angle is identical to the electrical angle between the phasors of the induced voltage and the terminal voltage. Therefore, the swing equation is typically expressed as follows:

$$\frac{2H}{\omega_0}\frac{d^2\delta}{dt^2} = P_M - P_E \tag{4.2}$$

where H is the inertia constant of the generator, ω_0 is the angular frequency of the generator, δ is the rotor angle, P_M is the mechanical input power and P_E is the electrical output power. In this case study, P_M is considered constant, while P_E is defined as follows:

$$P_E = P_{e,max}\sin(\delta) + D\dot{\delta}$$

where $P_{e,max}$ is the maximum power that can be transferred to the infinite bus and the product $\lambda\dot{\delta}$ represents the damping power of the system, P_d.

In the case of a fault, such as a short circuit of a line, P_E will go to zero, resulting in a constant acceleration of the rotor angle. When the fault is cleared, the system will behave according to the aforementioned swing equation with a reduced $P_{e,max}$, assuming the faulted line is opened. Ideally, the physics will decelerate the rotor angle enough to return the rotor angle to the left equilibrium point where P_M is equal to P_E. However, if the fault is not cleared within a certain period of time, the rotor angle will move past the the critical clearing angle required to restore the system to synchronism.

If the rotor angle increases, the value of P_E will decrease to the point where P_M is greater than P_E. Even if the fault is cleared, the acceleration will remain positive as the difference between P_M and P_E is positive, and the generator will fall *out-of-step*.

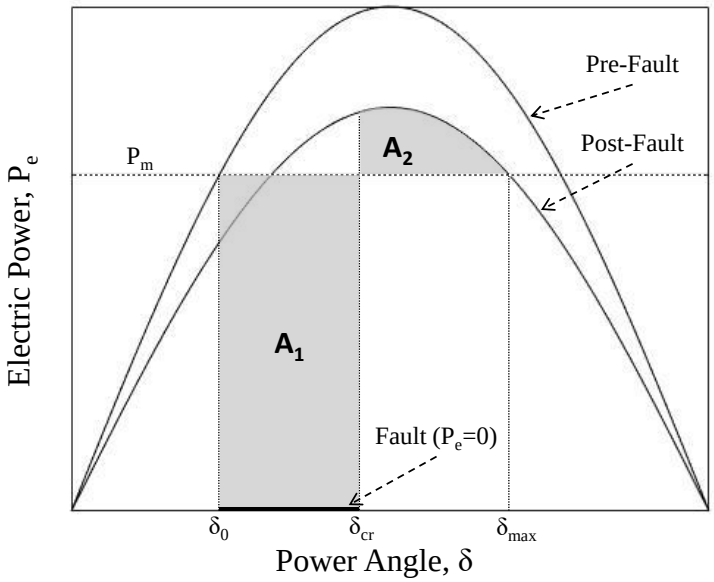


Figure 4.2: The swing equation plotted with the application of the equal area criterion after a disturbanceAndersson (2004)

The critical clearing time for a fault is typically computed by using the equal area criterion. Figure 4.2 shows a plot of swing equation dynamics. δ_0 shows the initial angle where the fault occurs, $\delta_c r$ is the clearing angle, and $\delta_m ax$ is the maximum rotor angle reached before the angle returns to the left equilibrium point. Our model assumes that the value of $P_{e,max}$ is reduced to a value that is less than the pre-fault value but still greater than P_M. After a fault is cleared, the total reactance will typically increase, and $P_{e,max}$ is inversely proportional to the reactance. The equal area criterion states that the system will be stable if the area under the curve during the faulted accelerating state, A_1, is equal to the area under the curve during the rotor angle's deceleration(retardation), A_2. The critical clearing angle can be calculated by integrating both areas and setting them equal to each other. Once the critical clearing angle is calculated, the critical clearing time for the fault can be derived as well. In this work, we will use the invariant properties of the physical equations to derive the clearing time bounds based on the given state of the system and compare them to the

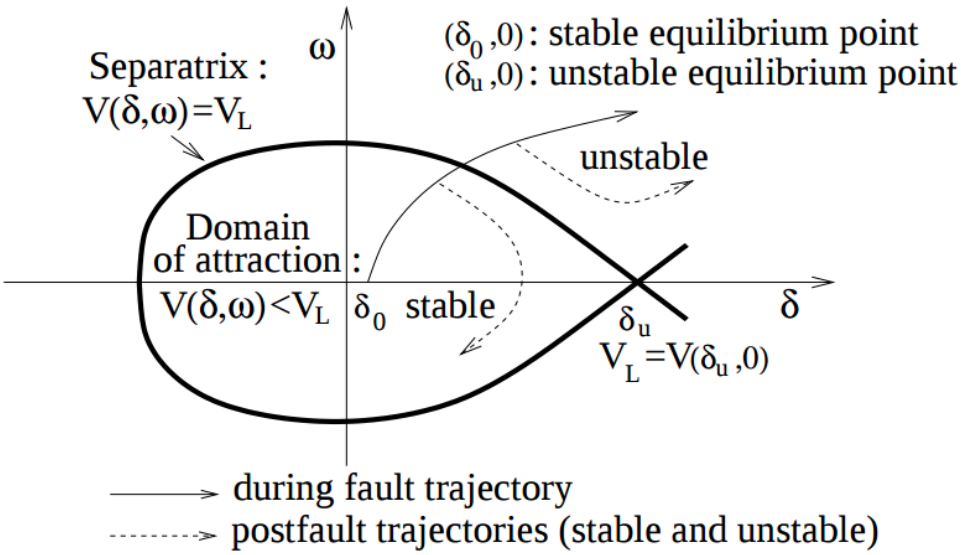


Figure 4.3: Phase-plane of SMIB system with Lyapunov V-functionPavella et al. (2012)

results of the equal area criterion calculation.

Other methods have been used to model the transient rotor angle stability. For the aforementioned system, the following Lyapunov function can be shown to fulfill the requirements of Lyapunov's stability theoremAndersson (2004):

$$V(\omega, \delta) = \frac{H}{\omega_0}\omega^2 + P_M(\delta_0 - \delta) + P_{e,max}(\cos(\delta_0) - \cos(\delta))$$

4.3 Recovering Properties of Recasted Transcendental Functions

In this section we will discuss in detail the motivating problem of our approach as well as the automatic techniques used to generate the sufficient conditions necessary to prove the safety properties of such systems. We first provide a detailed description of the non-faulted mode of the aforementioned SMIB electric power grid system as this mode introduces the conundrum of accessing previously abstracted properties for the sake of proving the associated safety properties of the system.

4.3.1 Case Study: Non-faulted SMIB Hybrid Program

We consider a SMIB model with two discrete modes: `normal` and `faulted`. Both modes are described by the swing equations (cf. Equation (4.2)) with different electrical output power P_E. For the `faulted` mode, $P_E = 0$, that is the motor is not producing any power. For the `normal` mode, $P_E = P_e \sin(\delta)$ for a fixed constant P_e that is assumed to only depend on the reactance of the line. This means that P_e may not be the same before and after a fault occurs but will be always greater than P_M for our model.

Remark 1. In the sequel, for the sake of simplicity, the scaling constant factor $\frac{2H}{\omega_0}$ is assumed to be the unit. This comes at no loss of generality and such a factor could be reintroduced at will whenever needed. More importantly, we neglect the damping power component of the electrical output power, $\lambda\dot{\delta}$. The damping power component is an abstract representation of all the damping power components. It only determines how fast a stable solution will converge to an equilibrium point and will therefore have little influence during the first swing after a fault. In the future, we plan to consider refined models including the damping factor.

Because the faulted mode does not depend on the transcendental function, $\sin(\delta)$, we initially are only concerned with proving the safety of the system in a non-faulted state. We model the system by the following hybrid program where Plant denotes the swing equations put in an algebraic, higher dimension, ordinary differential equation form. That is, the cosine and sine functions are substituted by additional fresh variables, namely $c := \cos(\delta)$ and $s := \sin(\delta)$. The algebraic rewriting is mandatory to eliminate non-linearities in the ODE as well as the fact that the theorem prover `KeymaeraX` does not support transcendental functions. This limitation is also imposed by the automated invariant generation tools we will be using. Using the pair (c, s), the rotor angle δ is entirely determined. It is thus assumed that δ is only accessible through its cosine and sine functions.

Definition 4.3.1 (Non-Faulted SMIB Hybrid Program)**.**

$$\alpha_n \equiv ((P_e := *; P_e > P_M?); \text{Plant}_{a=a_n})$$

$$\text{Plant} \equiv \left\{ \begin{array}{lcl} \dot{\delta} & = & \omega \\ \dot{\omega} & = & a \\ \dot{c} & = & -\omega s \\ \dot{s} & = & \omega c \end{array} \right. \quad \text{and} \quad a_n \equiv P_M - P_e s. \tag{4.3}$$

The sequence $(P_e := *; P_e > P_M?)$ allows the value of P_e to get updated before the execution of each normal mode while ensuring that it is always greater than P_M.

The swing equation is typically analyzed with an initial state at the equilibrium point of the normal mode. This is typically the point where P_M is equal to P_E. Given this notion, we would assume the initial conditions to be given by the following predicate:

$$\phi \equiv P_e > P_M > 0 \wedge \omega = 0 \wedge P_M - P_e s = 0$$
$$\wedge\, c > 0 \wedge c^2 + s^2 = 1 \wedge s_e = s \wedge c_e = -c$$

The extra pair (c_e, s_e) encodes δ_e, the position of the rotor angle at the unstable equilibrium of the system. It serves to describe the safety condition $\delta < \delta_e$ in terms of the sine and cosine functions:

$$\psi \equiv c < c_e \vee s > s_e$$

since $\delta \geq \delta_e$ (modulo π) if and only if $c \geq c_e \wedge s \leq s_e$.

Even with this encoding, the fresh variables c and s are still dependent on the initial values of δ in order to soundly abstract the sin and cos functions. The initial conditions for the δ should correspond to the associated with the initial state of the c and s variables. However, given the initial conditions, $\delta_0 = \arcsin(P_M/P_e)$, which will evaluate to a non-zero, real value. Intuitively, this calculation would provide a basis for a relatively accurate estimation of the system. However, because the values of sin and cos are infinite series, assigning δ to such a real value other than 0 or 1 will always be slightly incorrect by some infinite remainder. This co-dependence between

a transcendental function and its parameter variable is the critical conundrum that motivated this work. In the subsequent section, we will discuss how to automatically generate sufficient conditions such that the transcendental functions will be soundly recasted.

In spite of the lack of soundness, we can still generate semi-algebraic invariants based on the polynomialized system. In fact, we will see that the invariant generation is a prerequisite for the generation of evolution domain constraints.

Invariant Generation. We leverage recent effective methods to generate algebraic invariants for polynomial vector fields Ghorbal and Platzer (2014) to generate positive invariant regions for the polynomial ODE system of (4.3) for the normal mode. The method was able to generate the following invariant function:

$$p \equiv 2P_M\delta + 2P_e c - \omega^2 \tag{4.4}$$

It is easy to check that the time-derivative of p is identically zero. If one substitutes back the variable c by $\cos(\delta)$, then a natural positive invariant for the system can be derived from the above invariant function by exploiting its level sets, namely

$$2P_M\delta + 2P_e\cos(\delta) - \omega^2 \geq 2P_M\delta_e + 2P_e\cos(\delta_e), \tag{4.5}$$

where δ_e denotes the rotor angle at the unstable equilibrium which is entirely determined by P_M and P_e. In fact, one has

$$\delta_e \equiv \arccos\left(-\sqrt{1 - \frac{P_M^2}{P_{e,max}^2}}\right) \tag{4.6}$$

Such positive invariant region is however not semi-algebraic since it involves the cosine function. Thus, it cannot be exploited as is to discharge the safety of the system in `KeymaeraX`. To solve this problem, we resort to Taylor expansions to approximate the cosine function. For instance, using the 6th order approximation, one can exploit the

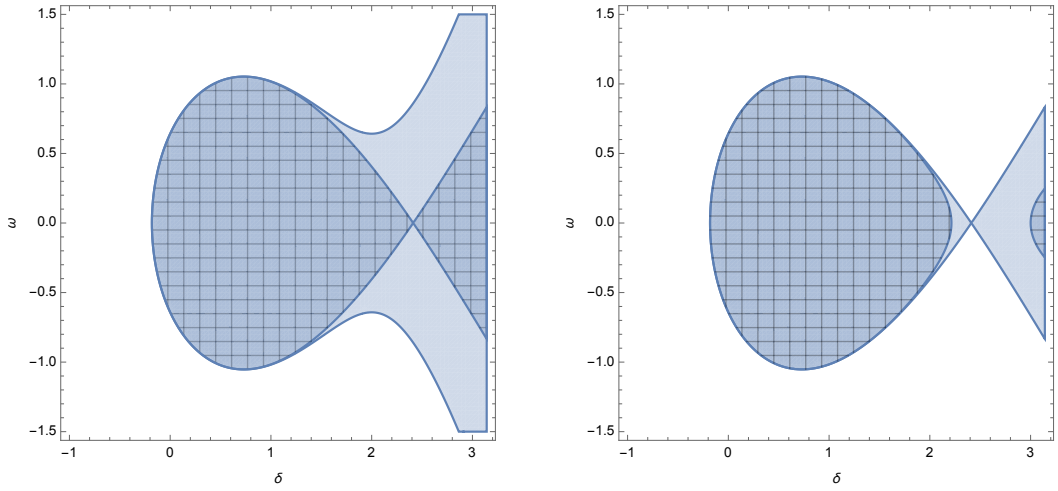


Figure 4.4: Approximation of an invariant region by Taylor expansions.

following inequality that holds true for all δ:

$$\cos(\delta) \geq 1 - \frac{\delta^2}{2!} + \frac{\delta^4}{4!} - \frac{\delta^6}{6!}, \tag{4.7}$$

To derive a positive invariant candidate involving only the rotor angle δ. In Fig.4.4, $P_M = 1$ and $P_e = \frac{3}{2}$, one can see that an expansion of order 4 (left figure) is not accurate, as it is much larger than actual invariant region (dashed), whereas an expansion of order 6 (dashed on the right figure) gives a tighter approximation.

Combining Equations (4.5) and (4.7), we get the following semi-algebraic invariant candidate:

$$2P_M\delta + 2P_e\left(1 - \frac{\delta^2}{2!} + \frac{\delta^4}{4!} - \frac{\delta^6}{6!}\right) - \omega^2 \geq 2P_M\delta_e + 2P_e\cos(\delta_e) \wedge \delta \leq \delta_e. \tag{4.8}$$

To check the invariance of the above region, we used the recent characterization of invariant semi-algebraic sets Liu et al. (2011a).

Figure 4.5 shows the plot of the invariant region for $P_M = 1$ and $P_e = \frac{3}{2}$. In that case, we get approximately $\delta \in [-0.182, 2.42]$ and $\omega \in [-1.053, 1.053]$.

These invariant properties can now be leveraged to reason about the system in spite of the initial lack of soundness.

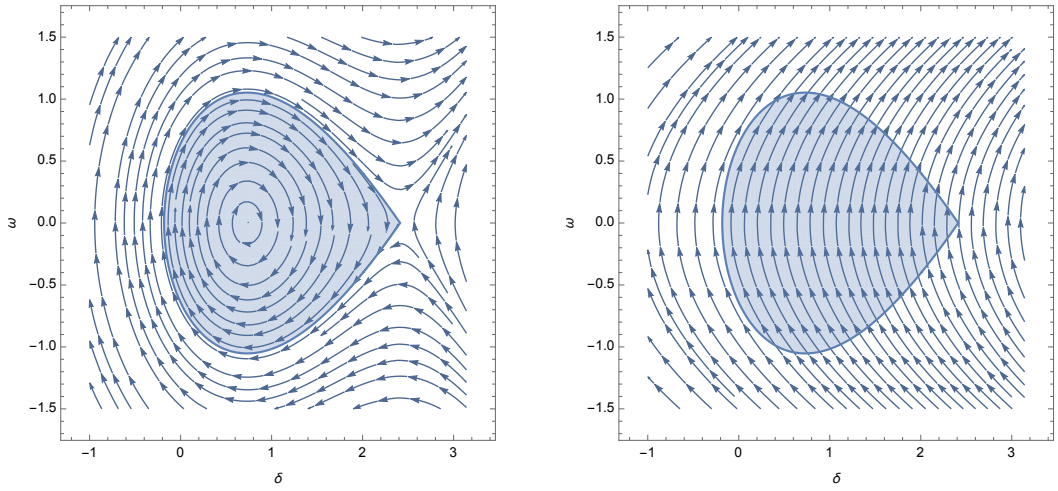


Figure 4.5: Normal (left) and faulted (right) flow with the invariant region.

4.3.2 Evolution Domain Constraint Generation Through Taylor Series Approximation

The intuitions gained by the invariant generation regarding the Taylor series approximation of cosine allowed us to provide a sound approach to automatically augment the evolution domain constraint of the Plant for the purpose of sound abstraction of the transcendental functions. By forcing a similar constraint on the evolution domain constraint of transcendental functions, we can narrow down the set of possible solutions to a set that includes solutions where the cosine and sine functions are among them.

Our solution automatically generates evolution domain constraints that hold true for the entire evolution of cosine by specifying a domain constraint by providing a bound on the fresh variable c based on the previous Taylor series approximation. Because the invariant generation process found that the Taylor series expansion of order 6 was necessary to prove that δ was bounded, we similarly generate a domain constraint that bounds the fresh variable c. Thus, the generated evolution domain constraint, H, will be the following:

$$H \equiv c \leq 1 - \frac{\delta^2}{2!} + \frac{\delta^4}{4!} - \frac{\delta^6}{6}$$

and the updated hybrid program for the SMIB normal mode will be

$$\alpha_n \equiv ((P_e := *; P_e > P_M?); \text{Plant}_{a=a_n} \,\&H)\,.$$

The evolution domain constraint property is true for this ODE on a manifold, and the cosine function is one of those solutions because the cosine function satisfies these conditions. This approach is as generalizable as the automatic invariant generation method previously presented. However, because we are using approximations in the evolution domain constraint, we need to generate sufficient initial conditions such that we soundly and precisely abstract the sine and cosine functions.

4.3.3 Automatic Generation of Sufficient Initial Conditions for Polynomialized Systems

To provide sound abstraction of the transcendental functions, we first discard the imprecisions of the previous initial conditions of the system, ϕ. These are replaced with initial values for δ, c, and s that we know to be precise, mathematical truths. The initial condition generation occurs in two stages, the first being the initialization of the fresh variables and associated dependences to values that preserve soundness, and the second being the provision of bounds with respect to the safety properties of the hybrid system.

Initialization of fresh variables. We know that for $\delta = 0$, the value of c and s will be 1 and 0, respectively. Although typical fault analysis of the swing equation has δ set to the equilibrium point, this assumes the system has already been in operation for some time such that δ will converge to the stable equilibrium point. Therefore, setting δ to 0 is a more appropriate value for this model since this is the value when the operation of the synchronous machines begin.

More generally, the initial values of fresh variables and their dependences that are recasting transcendental functions will depend on the dynamics of the system. Furthermore, the values will depend on domain-specific information. For instance, we know that δ can be set to 0 in this case, but there may be applications where this property is invalid. This can be easily automated if the operator provides domain specific

requirements for the initial conditions.

Provision of bounds on the safety properties. Up until this stage, we have soundly abstracted the transcendental functions by generating proper initial values for the associated state variables. However, we do not have a sound representation of the bounds associated with the safety properties of the hybrid system. For example, we have not provided a a sound representation of the unstable equilibrium point, δ_e for the SMIB system. We know that we have a geometric singularity at δ_e, and that δ_e is the point where we have an intersection if we had access to the cosine function. Thus, although we cannot assign δ_e to a particular value, we can provide upper and lower bounds for δ_e that approximates a point close to the singularity using the aforementioned Taylor series expansion. If we directly substitute δ_e into the the Taylor series approximation with respect to the previously generated invariant, we can then generate sound approximations for the upper and lower bounds of δ_e. The generated constraints after this substitution are as follows:

$$-2 * P_e * (-\frac{\delta_e^2}{2!} + \frac{\delta_e^4}{4!} - \frac{\delta_e^6}{6!}) + (-2 * \delta_e * P_M - \omega^2) \geq 0$$
$$\wedge 2 * P_e - \omega^2 < 2 * (1 - \delta_e^2/2 + de^4/24 - de^6/720) * Pe + 2 * de * Pm$$

Such an approach would be sufficient for models that hide the cosine and sine functions from the ODE, but this is not sufficient for cases where the invariant involves both the cosine and the angle. This co-dependence between the transcendental function and its dependent variable requires additional constraints on the safety properties to facilitate a proof of the system. The goal of the polynomialization is to project the ODE onto the a two-dimensional plane. In the case of the SMIB, we are projecting the system with δ and ω for its dimensions. Because we are using a Taylor approximation at order 6, we need extra conditions on δ_e so that δ and ω can be approximated on a two-dimensional plane. Therefore, we provide an extra constraint that ensures δ_e and ω are soundly approximated. After simplifications and a similar expansion of the sine function based on the same criterion as the Taylor series expansion of cosine, we generate the following formula

$$-120 + 120 * r + r^3 + r^5 = 0.$$

The roots of this equation will provide the final possible constraint for δ_e. However, only one of these roots is valid for the model under consideration. We first only consider the real roots of the system. Furthermore, as in the other initialization of fresh variables, we will need domain-specific information that defines what root of this equation to consider. For our model, we know that δ_e should be greater than the third root, which is approximately 3.68. We know this because we have domain knowledge about the right equilibrium point. Again, such validation requirements of the model can still be automated if given domain-specific requirements of the hybrid system under consideration. Therefore, we generate a sufficient initial condition that ensure we choose the correct root of the above system as well as a condition to ensure that δ_e is constrained by this root:

$$-120 + 120 * r + r^3 + r^5 = 0 \wedge r \geq 3 \wedge r \leq \delta_e$$

The constraint $r \geq 3$ is sufficient enough to ensure we are choosing the third root. However, different cases may require additional and more precise constraints in order to choose the correct root of the system.

The initial conditions, ϕ, of the SMIB hybrid program α_n can now be specified as follows:

$$\begin{aligned}
\phi \equiv\ & P_e > P_M > 0 \wedge \delta = 0 \wedge c = 1 \wedge s = 0 \\
& \wedge -2 * P_e * (-\frac{\delta_e^2}{2!} + \frac{\delta_e^4}{4!} - \frac{\delta_e^6}{6!}) + (-2 * \delta_e * P_M - \omega^2) \geq 0 \\
& \wedge 2 * P_e - \omega^2 < 2 * (1 - \frac{\delta_e^2}{2!} + \frac{\delta_e^4}{4!} - \frac{\delta_e^6}{6!}) * Pe + 2 * de * Pm \\
& \wedge -120 + 120 * r + r^3 + r^5 = 0 \wedge r \geq 3 \wedge r \leq \delta_e
\end{aligned}$$

Now that we have generated sufficient conditions to soundly recast this case of non-linear polynomial systems with transcendental functions, we can use the `KeymaeraX` theorem prover to prove the safety of our system as has been done in numerous related worksPlatzer and Quesel (2008)Platzer and Clarke (2009)Loos et al. (2011)Kouskoulas et al. (2013).

4.3.4 Proving the Safety of a Polynomialized System

The associated proof tactic for the safety of such a system given the sufficient conditions would not require any novel approaches. The main ingredients of the proof are the applications of the differential divide-and-conquer (DDC) Sogokon et al. (2016) and differential cut Platzer (2008) rules. As long as we have guaranteed the soundness of the polynomialized system, then we can safely exploit the sound proof rules associated with d$\mathcal{L}$.

4.4 Discussion and Limitations

Although this chapter presented an approach to the automatic generation of positive semi-algebraic invariants for complex CPS, we encountered two main difficulties for this particular model–which are common to interesting enough hybrid systems in general:

- The use of transcendental functions in a hybrid systems model. In particular, the difficulties that arise from the specific link between the angle and the trigonometric functions in the generated invariant.
- The generation of invariant for the entire hybrid model the includes the continuous dynamics.

To overcome the first difficulty, Taylor Series approximations proved to be an adequate approach for the proofs associated with such systems. This is a promising research direction for modeling more complex CPS, especially in the context of electric power grid systems.

For the second difficulty regarding the hybridness, we currently lack an automated method to properly propagate small invariant pieces and merge them into one “hybrid

invariant" that is sufficient enough for our safety condition. However, the preliminary results were a promising first step in which we automatically generated those local invariant pieces. Providing a generic compositional process will be a promising and challenging research direction.

4.5 Conclusion

In this paper, we presented a method that provides the preliminary results for the automatic proof of safety properties of systems governed by non-linear ordinary differential equations with mixed polynomial and trigonometric functions under semi-algebraic evolutions constraints whose proofs require semi-algebraic invariants and whose invariants have a co-dependence between a transcendental function and the parameters of the transcendental function. Depending on the use case, domain-specific information may be required to validate the assumptions regarding both the initial conditions as well as the derived properties of the associated safety properties presented in this paper. However, such information can be integrated in an automatic fashion.

We further evaluated our method on an electric power grid system whose continuous dynamics contained the aforementioned co-dependence between the transcendental function and its parameters. We used this case study as an ongoing example throughout the paper to exemplify how our methods can be automatically applied to prove the safety of a system. In particular, we prove properties about the transient stability of a cyber-physical power systems network using sound and relatively complete verification techniques. Our methods are as generalizable as the previous complementary works in the generation of semi-algebraic invariant sets. This work along with the use case study serves as a premise for more complex cyber-physical networks in power systems where the switching components in the cyber topology can alter the physical evolution of the system.

Chapter 5

Hybrid PLC Program Translation for Verification

5.1 Introduction

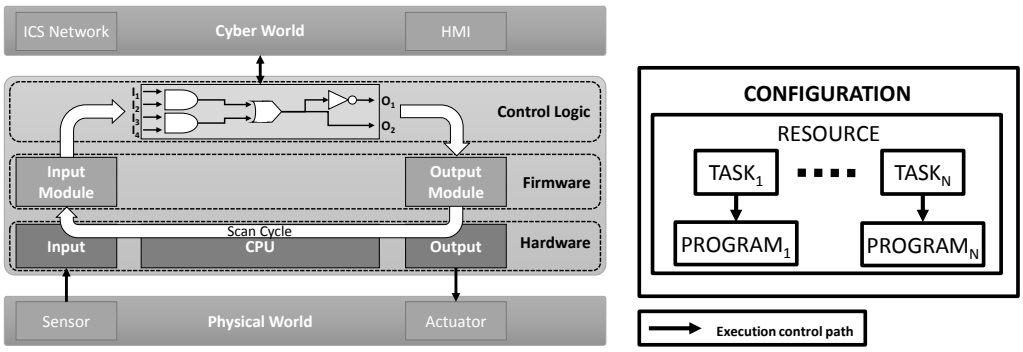
There has been an increased emphasis on the verification and validation of software used in embedded systems in the context of *industrial control systems* (ICS). ICS represent a class of cyber-physical systems (CPS) that provide monitoring and networked process control for safety-critical industrial environments, e.g., the electric power grid McGranaghan et al. (1993), railway safety abb, nuclear reactors Kesler (2011) and water treatment plants Manesis et al. (1998). A prominent choice of implementation platform for many ICS applications are programmable logic controllers (PLCs) that act as interfaces between the *cyber world*–i.e., the monitoring entities and process control–and the *physical world*–i.e., the underlying physical system which the ICS is sensing and actuating. Efforts to verify the correctness of PLC applications focus on the code that is loaded onto these controllers Moon (1994), Darvas et al. (2015b), Mader and Wupper (1999), Thapa et al. (2005). Existing methods are primarily based on on model checking of safety properties specified in modal temporal logics, e.g., Linear Temporal Logic (LTL) Gerth et al. (1995) and Computation Tree Logic (CTL) Clarke et al. (1986). However, since PLC code does not include a model of the system plant, such analyses are limited to discrete properties of the code instead of analyzing safety properties of the resulting physical behavior.

In this paper, we thus start from hybrid systems models of ICS, in which the discrete computations of controllers determine the continuous evolution of the underlying physical system. That way, correctness properties that consider both control decisions and physical evolution can be verified in the theorem prover KeYmaera X Fulton et al. (2015). The verified hybrid programs can then be translated systematically to PLC code

and executed as controllers. The reverse translation from PLC code to hybrid programs facilitates verifying existing controller code with respect to pre-defined models of the continuous evolution of the system.

In this paper, we present HyPLC, a tool that systematically translates verified hybrid systems models into PLC code and vice versa. The hybrid models are specified in differential dynamic logic, d$\mathcal{L}$Platzer (2008), which is a dynamic logic for hybrid systems expressed as *hybrid programs*. The translation from hybrid programs to PLC code generates deterministic implementations of the controller abstractions typically found in hybrid programs, which focus on capturing the safety-relevant decisions for verification purposes concisely with nondeterministic modeling concepts. Nondeterminism in hybrid programs can be beneficial for verification since nondeterministic models address a family of (control) programs with a single proof at once, but is detrimental to implementation with Structured Text (ST) programs on PLCs. Therefore, in this paper we focus on hybrid programs of a certain shape. The translation adopts the IEC 61131-3 standards for PLCs John and Tiegelkamp (2010). The translation from PLC code back to d$\mathcal{L}$and hybrid programs, implemented on top of the open-source MATIEC IEC 61131-3 compiler Sousa, provides a means of analyzing PLC code on pre-defined models of continuous evolution with the deductive verification techniques of KeYmaera X. Both directions yield a way of obtaining verified PLC code under the assumption that the compilation which we detail here is implemented correctly. We evaluated our tool on a water treatment testbed Mathur and Tippenhauer (2016) that consists of a distributed network of PLCs.

The rest of the paper is organized as follows. Section 5.2 provides background information necessary to understand the motivating problem as well as our solution. Section 5.3 introduces a systematic mapping of terms for both languages and describes how the semantics may be preserved. Similarly, Section5.4 and Section 5.5 describe the translation of formulas and programs, respectively. Section 5.6 presents our evaluation of HyPLC on different case studies. We discuss the limitations of HyPLC and conclude in Section 5.7.



(a) The PLC *scan cycle* in the context of ICS. (b) PLC Software Model.

Figure 5.1: The PLC architecture and software model based on the IEC 61131-3 standard John and Tiegelkamp (2010) considered in this paper. We consider only a single program per task running on a single resource configuration for simplicity. The full software model can be found in Appendix .3.

5.2 Preliminaries

In this section we introduce the preliminary information necessary to understand the underlying concepts of HyPLC. We first provide a brief overview of PLCs, including how they are integrated into ICS as well the associated programming languages and software model as defined by the IEC 61131-3 standard for PLCs John and Tiegelkamp (2010). We then discuss previous works in formal verification of PLC programs, followed by an overview of the dynamic logic and hybrid program notation used by HyPLC.

5.2.1 Programmable Logic Controllers

Part 3 of the IEC 61131 standards John and Tiegelkamp (2010) for PLCs specifies both the software architecture as well as the programming languages for the control programs that run on PLCs. We will provide the requisite knowledge for understanding the assumptions made by HyPLC.

PLCs in the context of ICS. Figure 5.1 shows how PLCs are integrated into ICS as well as an abstracted overview of the PLC *scan cycle*. Scan cycles are typical control-loop mechanisms for embedded systems.The PLC "scans" the input values coming from the physical world and processes this system state through the *control logic* of the PLC, which is essentially a reprogrammable digital logic circuit. The outputs of the control logic are then forwarded through the output modules of the PLC to the

physical world. HyPLC focuses on hybrid programs that mimic this control principle with time-triggered models–a common modeling pattern that is very expressible with hybrid programs.

Programming Languages. The IEC 61131-3 standard defines five programming languages to be used for control logic programs within PLCs: (1) ladder diagrams (LD), (2) function block diagrams (FBD), (3) sequential function charts (SFC), (4) instruction list (IL), and (5) structured text (ST).HyPLC focuses on bidirectional translation of the *Structured Text* (ST) language–a textual language similar to Pascal that can be augmented to subsume the other languages for formal verification purposes Darvas et al. (2017). Next, we recall the software execution model considered in this paper.

Software Execution Model. According to the IEC 61131-3 standard, a PLC hosts several *resources*, e.g., different CPU modules, which are assigned *tasks* for execution. Tasks may execute *programs* or *function blocks*, either periodically for a given interval or on events identified through the rising edge of a trigger value in memory–usually a boolean condition on a global variable. For multiple tasks, the task with the highest priority will preempt lower priority tasks. A program defines a program organization unit that performs a particular function for a task. A program is composed of a series of ST statements, which may include calls to function blocks. A PLC *function block* in turn contains a sequence of ST statements. Function blocks can be thought of as subprograms that are defined within programs and can be called directly by a program or task. The aforementioned organization for all components are defined within a *configuration* element that can define the resource allocation as well as the variable access paths, i.e., the communication channels between components. In this paper, we initially consider a single-resource configuration of a PLC that may have one or more tasks associated with a particular program, depicted in Figure 5.1. The full IEC 61131-3 software model can be found in Appendix .3.

5.2.2 Previous Works in PLC Programming Language Verification

There have been numerous works regarding the verification of PLC programming languages against safety properties. Rausch, *et al.* Rausch and Krogh (1998) modeled

PLC programs consisting only of boolean variables, single static assignment of variables, no special functions or function blocks, and no jumps except subroutines. Such an approach was an initial attempt to provide formal verification of "shallow" discrete properties of the system plant, i.e., properties that can be derived and verified in a finite amount of time–as opposed to "deep" correctness properties that hold true throughout the entire evolution of a hybrid cyber-physical system. Similarly, other approaches have been presented whose safety properties are specified and modeled using linear temporal logic (LTL) McLaughlin et al. (2014) Pavlovic et al. (2007) or by representing the system as a finite-state automaton Mertke and Frey (2001), Darvas et al. (2015a) Tapken and Dierks (1998). The formal verification of such systems is limited by the state-space exploration in the context of real-time systems. Model-checkers for such safety properties typically result in a state space explosion for complex systems with continuous time evolution. They do not benefit from the sound abstractions of models as in deductive verification techniques.

Conversely, there have been several works regarding the generation of PLC code based on the formal models of PLC code. PLCSpecif Darvas et al. (2015b) is a framework for generating PLC code based on finite-state automata represenations of the PLC. Although this framework provides a means of generating PLC code based on formally verified models, the formal verification has the aforementioned limitations of providing correctness guarantees for "shallow" discrete properties of the PLC code that can be verified in a finite amount of time. The approach presented by Sacha Sacha (2005) has similar limitations since it uses state machines to represent finite-state models of PLC code. Darvas , *et al.* also used PLCSpecif for conformance checking of PLC code against temporal properties Darvas et al. (2016). Flordal, *et al.* automatically generated PLC-code for robotic arms based on generated *zone* models to ensure the arms do not collide with each other as well as to prevent deadlock situations Flordal et al. (2007). The approach generates a finite-state model of the robot CPS environment that is then used to generate supervisory code within the PLC that controls the arm. The approach abstracts the PLC's discrete properties and does not incorporate the PLC's timing properties into the physical plant model. Furthermore, this is a domain-specific

approach for robot simulation environments and does not provide generalizability nor a means of formal verification of the initially generated finite-state models.

5.2.3 Differential Dynamic Logic and Hybrid Programs

HyPLC works on models that have been specified in differential dynamic logic (d$\mathcal{L}$) Platzer (2010), a logic that models hybrid systems and can be formally verified with a sound proof calculus. The formalized models that use d$\mathcal{L}$are referred to as *hybrid programs*. As with ST, we will recall the syntax and semantics of d$\mathcal{L}$and hybrid programs as needed throughout the course of this paper.

The Dynamic Logic modal operators [] and $\langle\rangle$ are used to formally describe the behavioral properties the system has to verify. If α denotes a hybrid program, and ϕ and ψ are predicates, then the necessity proposition

$$\phi \rightarrow [\alpha]\psi$$

means "it is necessary that, if ϕ is initially satisfied, ψ holds true for all the states after executing the program α". This way, safety properties can be encoded for a model α. A common modeling pattern in hybrid programs is

$$init \rightarrow [\{ctrl; plant\}^*]req,$$

where *init* represents the initial state of the system, *ctrl* describes the discrete control transitions of the system, *plant* defines the continuous plant of the system, and *req* is the safety property we are trying to prove. In this pattern, control and plant are repeated any number of times, as indicated with the nondeterministic repetition operator *. In addition to assuming verified hybrid programs, HyPLC assumes the models are valid with respect to the CPS. Methods such as ModelPlex Mitsch and Platzer (2016) have been proposed to ensure that the specified hybrid program is compliant with the real system execution.

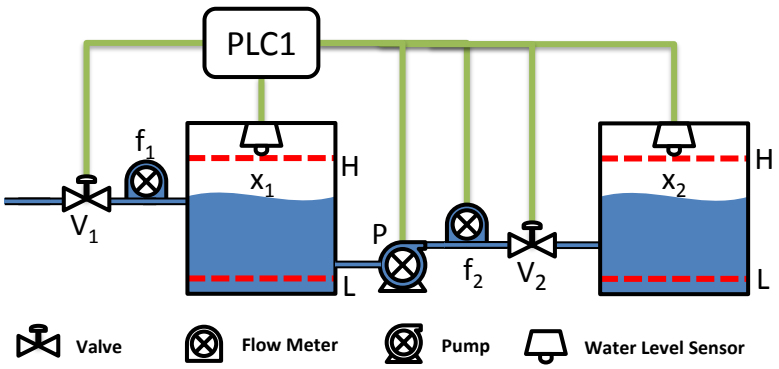


Figure 5.2: The first process control components for a water treatment testbed Mathur and Tippenhauer (2016).

5.2.4 Use Case: Water Treatment Testbed

As a running example for this paper, we will use a simple water tank component taken from the first of six control processes of a water treatment testbed Mathur and Tippenhauer (2016), depicted in Figure 5.2. This process is responsible for taking in water from a raw water source and feeding it into a tank. This water will then be pumped out into a second tank to be treated with chemicals. For this first process, the PLC is responsible for controlling the inflow of water for both tanks by opening or closing valves, V_1 and V_2, as well as the outflow of water to the second tank by running the pump, P. The PLC monitors a water level of both water tanks, x_1 and x_2, to ensure that V_1 and V_2, respectively, are closed before each respective tank overflows beyond an upper bound, H. Furthermore, the PLC responsible for protecting the outflow pump, P, by ensuring that the pump is not on if the water level of x_1 is below a lower threshold, L, or if the flow rate of the pump, f_2, is below a certain lower threshold, F_L(not pictured in the figure). Figure 5.3 shows the associated ST code for this model. This is a simplified representation of the actual PLC code that is loaded onto the PLC for a particular sample rate of T_{sample} for all the associated sensors. In this model, the flow rate for the incoming raw water, f_1, is not incorporated into the process control. The real system simply monitors the value of this flow rate without establishing a physical dependency. The upper limits of the water tank level, H_1 and H_2, and the lower limits, L_1 and L_2, actually represent trigger levels that are respectively below and above the actual safety thresholds, H_H and L_L. The trigger values were determined empirically. This simple

```
PROGRAM prog0
  VAR_INPUT
    x1 : REAL;
    x2 : REAL;
    f1 : REAL;
    f2 : REAL;
  END_VAR

  VAR_OUTPUT
    V1 : BOOL;
    V2 :BOOL;
    P : BOOL;
  END_VAR

  IF(x >= H1) THEN
    V1:=0;
  ELSIF(x <= L1) THEN
    V1:=1;
  END_IF;
  IF(x2 <= L2) THEN
    P := 1;
    V2 := 1;
  END_IF;
  IF(x1 <= LL OR f2 <= FL OR x2 >= H2) THEN
    P:= 0;
    V2 := 0;
  END_IF;
END_PROGRAM

CONFIGURATION Config0
RESOURCE Res0 ON PLC
TASK Main(INTERVAL:=T#1 s,PRIORITY:=0);
PROGRAM Inst0 WITH Main : prog0;
END_RESOURCE
END_CONFIGURATION
```

Figure 5.3: ST program for simplified process control of the system in Figure 5.2.

model will be used throughout the course of the paper to illustrate how an existing ST program can be systematically translated to the discrete control of a hybrid program and udpated if necessary to ensure the safe operation of the ICS.

5.3 Translation of Terms

In this section, we will define a birectional translation of the terms of both hybrid programs and ST for PLCs. The terms of ST are the leaf elements of ST expressions that represent the values stored in the PLC's memory and directly effect the sensing and actuation of the cyber-physical system for a particular context. As such, these values will need to be abstracted to represent the terms of an equivalent hybrid program. We will first discuss syntax of the terms for both languages and then define the semantic translations. **Notation.** We write $\mathbf{ST}(\theta_{HP})$ to express the compilation of a hybrid program term, θ_{HP}, to a structured text term, and we write $\mathbf{HP}(\theta_{ST}$ to represent

the translation of a structured text term, θ_{ST}, to a hybrid program term. This is the compilation notation that will also be used for the bidirectional translation of formulas and expressions.

5.3.1 Grammar Definitions

In order to provide a translation of terms for both languages that preserves the respective semantics, we first define the grammar for both languages.

Grammar of structured text terms. The terms of ST considered in this paper are defined by the grammar:

$$\theta, \eta ::= x \mid a \mid -\theta \mid \theta \sim \eta, \texttt{ where } \sim \in \{+, -, \cdot, /\}$$

and where $x \in V$ is an ST variable, and V is the subset of all ST variables of numeric elementary data types defined by the IEC 61131-3 standard that include integer, real, and bit string types. Similarly, the number literal, a, is restricted to the subset of data types.

Grammar of d$\mathcal{L}$terms. The terms of d$\mathcal{L}$and hybrid programs Platzer (2008) are defined by the grammar:

$$\theta, \eta ::= x \mid x' \mid \theta \sim \eta, \texttt{ where } \sim \in \{+, -, \cdot, /\}$$

and where $x \in V$ is a variable, and V is the set of all variables, $x' \in V'$ is a differential symbol, and f is a function symbol. The grammar allows the use of number literals as functions without arguments that are to be interpreted as the value they represent.

Generally, a direct syntax translation of the aforementioned grammars will be limited to those terms that can be directly represented in both contexts without the intiutions provided by the respective semantics. As a first step, we provide a systematic mapping of these terms that will be used to inductively define the translation of compositional terms, formulas, and expressions.

5.3.2 Systematic Mapping of Terms

Because of the recursive nature of the grammars, we will first derive a translation for the terminal expressions, referred to as atomic terms, and compose the other expressions accordingly.

Atomic terms. Atomic terms are not composed of other terms. For hybrid programs, an atomic term may be a variable, a number literal, or an uninterpreted functional symbol. For the sake of simplicity, we do not consider interpreted functions within hybrid programs as we want to focus on the core elements of discrete control.

HyPLC translates number literals and variables of hybrid programs, which are mathematical reals, to numbers and variables of data type `LReal` of the IEC 61131-3 standard. Note, that this translation does not necessarily preserves the semantics, since many implementations choose to implement `Real` with floating point numbers. In practice, this requires an appropriate encoding with floating point numbers (e.g, interval arithmetic) for soundness. We define the translation of a hybrid program number literal, $n \in \mathbb{R}$, as

$$\mathbf{ST}(n) \;=\; n_{LReal}$$

where n_{LReal} is the conversion of n to an LReal data type.

Conversely, because hybrid program variables are real-valued variables, the set of representable number literals in ST will be a subset of the representable number literals in hybrid programs. Therefore, we will have the direct translation of an ST number literal, n_{ST}, as

$$\mathbf{HP}(n_{ST}) \;=\; n_{ST}$$

We similarly perform the same conversion for variables as

$$\mathbf{ST}(x) \;=\; \mathbf{HP}(x) \;=\; x$$

A differential symbol x' is an independent variable associated with x and denotes the time-derivative of variable x when used in the context of differential equations $x' = \theta$. Differential symbols do not have a direct syntactic translation from hybrid programs to

ST. For the translation of ST to hybrid programs, differential symbols will be generated in the context of expressions as will be seen in section 5.5. If a differential symbol is used in a context that doesn't follow a particular syntactic pattern, the compiler will raise an error.

Proposition 1. The bi-directional translation rules of d$\mathcal{L}$ variables and literals and ST variables and literals preserves the semantics of terms for the respective languages.

Proof. For hybrid programs, a state is a mapping from variables V and differential symbol V' to $\mathbb{R}$, where the set of states is denoted as S. For ST programs, a context of an ST statement is denoted by σ, which is a function $\sigma : V \to D$ that assigns a value from pre-defined domains–in our case, `LReal`–to each defined variable. Provided that $LReal \in \mathbb{R}$ and given an appropriate encoding of floating point numbers to $\mathbb{R}$, we can assume that there will be a direct mapping of all states of σ to all states in S, i.e., for a correctly translated variable x, we have

$$\sigma(x) \equiv v(x) \texttt{ if } v \in S$$

Formally, the semantics for d$\mathcal{L}$ terms are defined Platzer (2015) as follows

$$[\![x]\!]^I v = v(x) \texttt{ for variable } x \in V$$

where for each interpretation I, the semantics of a term θ in a state $v \in S$ is its value in $\mathbb{R}$. For the interpretation of an ST variable value given a context σ, the operational semantics are defined Darvas et al. (2017) as follows

$$\frac{\sigma(v_1) = c_1}{(v_1, \sigma) \to_a c_1}.$$

Given the equivalence of states in S and states in σ, the interpretation of correctly translated variables in both directions will be equivalent as well, i.e.,

$$\sigma(ST(x)) \;=\; \sigma(x) \;=\; v(HP(x)) \;=\; v(x) \texttt{ if } v \in S.$$

□

Next, we inductively define the bi-directional translation for arithmetic operations.

Arithmetic operations. Arithmetic operations are similarly defined in an inductive fashion for the following operations: negation,addition (+), subtraction (−), multiplication (·), division (/), and exponentials. All of the arithmetic operations have the same syntax for both languages exponetials, i.e., for two terms, θ and η, we have

$$\begin{aligned} \mathbf{ST}(-(x)) &= -(\mathbf{ST}(x)); & \mathbf{HP}(-(x)) &= -(\mathbf{HP}(x)) \\ \mathbf{ST}(\theta \sim \eta) &= \mathbf{ST}(\theta) \sim \mathbf{ST}(\eta); & \mathbf{HP}(\theta \sim \eta) &= \mathbf{HP}(\theta) \sim \mathbf{HP}(\eta), \end{aligned}$$

where $\sim \in \{+, -, \cdot, /\}$. For exponentionals, the operators differ as follows:

$$\mathbf{ST}(\theta \hat{} \eta) = \mathbf{ST}(\theta) \star\star \mathbf{ST}(\eta); \ \mathbf{HP}(\theta \star\star \eta) = \mathbf{HP}(\theta) \hat{} \mathbf{HP}(\eta).$$

Proposition 2. The semantics of arithmetic operations will be preserved across the bi-directional translation for both languages.

Proof. The semantics for d$\mathcal{L}$ arithmetic operations with an operator $\sim$ for terms θ and η are inductively defined as follows.

$$\begin{aligned} [\![-\theta]\!]^I v &= \{v \in S : -[\![\theta]\!]^I v\} \\ [\![\theta \sim \eta]\!]^I v &= \{v \in S : [\![\theta]\!]^I \sim [\![\eta]\!]^I v\} \end{aligned}$$

Similarly, the semantics for arithmetic operations in ST with an operator $\sim$ for terms c_1 and c_2 are defined as follows.

$$\frac{(c_1, \sigma)}{((-c_1), \sigma) \to_a -c_1} \quad \frac{(c_1, \sigma) \to_a c_1 (c_2, \sigma) \to_a c_2}{((c_1 \sim c_2), \sigma) \to_a c_1 \sim c_2}$$

In both cases, the semantics are inductively defined by the arithmetic operations performed on the interpreted terms. The semantics of translated terms are preserved

by Proposition 1 in both translational directions. Because all arithmetic operations of hybrid programs and ST programs result in values in the domains $\mathbb{R}$ and $LReal$, respectively,and because we assume a proper encoding of both domains, there will be an equivalent mapping of arithmetic operations on translated terms in both translation directions as well. □

We next define how the translation definitions of terms can be leveraged to translate the *formulas* of both languages in both directions.

5.4 Translation of Formulas

In this paper, we use the term *formula* to encompass modality- and quantifier-free formulas of hybrid programs as well as conditional expressions of ST statements. As was done with the terms of each language, we will first discuss the syntax of the formulas considered for both languages.

5.4.1 Grammar Definitions

We define the grammar notation for each language considered in this paper as follows.
Grammar of ST formulas. Structured text formulas are used in conditional expressions defined by the IEC 61131-3 standard. Intuitively, these formulas those that return a boolean value. The grammar for ST conditional expressions,ϕ and ψ, is defined as follows

$$\phi, \psi ::= TRUE \mid FALSE \mid \theta \bowtie_{\texttt{ST}} \eta \mid NOT(\phi) \mid \phi \frown_{\texttt{ST}} \psi$$

$$\texttt{where} \bowtie_{\texttt{ST}} \in \{<, >, >=, <=, <>, =\},$$

$$\texttt{and} \frown_{\texttt{ST}} \in \{AND, OR, XOR\}$$

The values $TRUE$ and $FALSE$ represent the two boolean values a conditional expression can take upon evaluation, θ and η represent ST terms, the "$\bowtie_{\texttt{ST}}$" operator represents the set of relational operators used in ST, the "$\frown_{\texttt{ST}}$" operator represents the set of logical operators between two conditional expressions, and "NOT(ϕ)" is the

logical negation of a conditional expression.

Grammar of d$\mathcal{L}$ formulas. Formulas of hybrid programs specified in d$\mathcal{L}$ are defined as in first-order dynamic logic. The truncated grammar for modality- and quantifier-free formulas we consider in this paper is built using propositional connectives $\neg$, $\wedge$, $\vee$, $\rightarrow$, and $\leftrightarrow$ Platzer (2008). As such, the grammar for the d$\mathcal{L}$ formulas is given as follows:

$$\phi, \psi ::= true \mid false \mid \theta \bowtie_{\texttt{HP}} \eta \mid \neg\phi \mid \phi \frown_{\texttt{HP}} \psi,$$
$$\texttt{where} \bowtie_{\texttt{HP}} \in \{<, >, \geq, \leq, =\} \texttt{and} \frown_{\texttt{HP}} \in \{\wedge, \vee\}$$

and where ϕ, ψ are d$\mathcal{L}$ formulas, θ and η are d$\mathcal{L}$ terms, and x is a d$\mathcal{L}$ variable. Given these base grammars, we now present the systematic mapping and the associated correctness assumptions.

5.4.2 Systematic Mapping

As was done with the terms, the formulas similarly need to be translation by induction based on the atomic formulas.

Atomic formulas. As with atomic terms, atomic formulas are those formulas that are not composed of other formulas. In the context of both languages, these formulas can be either the logical *True* or the logical *False*. We do not consider predicate application formulas as we are not considering functions nor function applications in hybrid programs.

The compilation for the d$\mathcal{L}$ logical *true* and *false* formulas have equivalent expressions in ST, meaning that the logical result of a formula in hybrid programs is the same as the logical result of an equivalent conditional expression in ST, i.e.,

$$\mathbf{ST}(true) = TRUE \ \mathbf{HP}(TRUE) \quad = true$$
$$\mathbf{ST}(false) = FALSE \ \mathbf{HP}(FALSE) = false$$

The above translations simplify the minor discrepancies between the two grammars,

i.e., *true* and *false* can be represented as "0" or "1" in ST and the capitalization differs between the languages.

Proposition 3. The bi-directional translation of the logical *True* and logical *False* formulas for both ST expressions and hybrid programs preserves the respective semantics of each language.

Proof. As in Proposition 1 where there existed a mapping of number literals, the interpretation of the d$\mathcal{L}$ formulas *true* and *false* have a direct logical mapping to the ST formulas *TRUE* and *FALSE*, respectively. The logical evaluation of a translated boolean formula for a discrete state of a hybrid program is equivalent to the logical evaluation of a translated ST boolean formula in a context σ. □

The other types of atomic formulas are the comparison formulas that compare two terms. All comparisons in hybrid programs–equal to $(\theta = \eta)$, not equal to $(\theta \neq \eta)$, greater than $(\theta > \eta)$, greater than or equal to $(\theta \geq \eta)$, less than $(\theta < \eta)$, and less than or equal to $(\theta \leq \eta)$–have equivalent representations as ST conditional expressions if and only if the there is a correct translation of the two children d$\mathcal{L}$ terms. The converse is true for two children ST terms, i.e., for two terms, θ and η, we have

$$\begin{aligned}
&\mathbf{ST}(\theta = \eta) = \mathbf{ST}(\theta) = \mathbf{ST}(\eta); \quad &&\mathbf{HP}(\theta = \eta) \quad = \mathbf{HP}(\theta) = \mathbf{HP}(\eta)\\
&\mathbf{ST}(\theta \neq \eta) = \mathbf{ST}(\theta) <> \mathbf{ST}(\eta); &&\mathbf{HP}(\theta <> \eta) = \mathbf{HP}(\theta) \neq \mathbf{HP}(\eta)\\
&\mathbf{ST}(\theta > \eta) = \mathbf{ST}(\theta) > \mathbf{ST}(\eta); \quad &&\mathbf{HP}(\theta > \eta) \quad = \mathbf{HP}(\theta) > \mathbf{HP}(\eta)\\
&\mathbf{ST}(\theta \geq \eta) = \mathbf{ST}(\theta) \geq \mathbf{ST}(\eta); \quad &&\mathbf{HP}(\theta \geq \eta) \quad = \mathbf{HP}(\theta) \geq \mathbf{HP}(\eta)\\
&\mathbf{ST}(\theta < \eta) = \mathbf{ST}(\theta) < \mathbf{ST}(\eta); \quad &&\mathbf{HP}(\theta < \eta) \quad = \mathbf{HP}(\theta) < \mathbf{HP}(\eta)\\
&\mathbf{ST}(\theta \leq \eta) = \mathbf{ST}(\theta) \leq \mathbf{ST}(\eta); \quad &&\mathbf{HP}(\theta \leq \eta) \quad = \mathbf{HP}(\theta) \leq \mathbf{HP}(\eta)
\end{aligned}$$

Proposition 4. The systematic mapping of comparison formulas for both languages preserves the semantics of both languages.

Proof. This proof follows Proposition 1 and Proposition 2. Given the equivalence of a state in the ST context σ and a discrete state of a hybrid program, the translation

of comparison formulas in both directions will preserve the semantics inductively. Although the grammars of both languages have slight discrepencies with respect to the logical operators, there is direct mapping of the associated logical operations assuming the translation of the child terms preserves the semantics as in Proposition 1. □

The systematic of mapping of the atomic formulas for both languages allow us to define the mapping for the compositional formula types.

Logical formulas. The logical formulas represent the set of composite formulas that perform logical operations on one or more formulas. We first consider the logical negation of a formula. As with the arithmetic negation of a d$\mathcal{L}$ term, the logical negation of a d$\mathcal{L}$ formula is equivalent to the negation of conditional expression in ST as long as the translation of its child formula is correct, and vice versa:

$$\mathbf{ST}(\neg(\phi)) = NOT(\mathbf{ST}(\phi))\ \mathbf{HP}(NOT(\phi)) = \neg(\mathbf{HP}(\phi))$$

With the systematic mapping of the logical negation, we can similarly provide an inductive bi-translation for the set of logical connectives, $\bowtie$, between two formulas. For hybrid programs, the set of logical connectives consists of the logical conjunctions ($\phi \wedge \psi$), disjunctions ($\phi \vee \psi$), implications ($\phi \rightarrow \psi$), and bi-implication ($\phi \leftrightarrow \psi$). The conjunction and disjunctions of d$\mathcal{L}$formulas are easily translated to conjunctions and disjunctions of ST conditional expressions, i.e., for two formulas ϕ and ψ,

$$\mathbf{ST}(\phi \wedge \psi) = \mathbf{ST}(\phi)\ \texttt{AND}\ \mathbf{ST}(\psi);\ \mathbf{HP}(\phi\ \texttt{AND}\ \psi) = \mathbf{HP}(\phi) \wedge \mathbf{HP}(\psi)$$
$$\mathbf{ST}(\phi \vee \psi) = \mathbf{ST}(\phi)\ \texttt{OR}\ \mathbf{ST}(\psi);\quad \mathbf{HP}(\phi\ \texttt{OR}\ \psi)\ \ = \mathbf{HP}(\phi) \vee \mathbf{HP}(\psi)$$

Implications and bi-implications of d$\mathcal{L}$ formulas are rephrased in terms of conjunctions, disjunctions, and negation before translation as follows.

$$\mathbf{ST}(\phi \rightarrow \psi) = \mathbf{ST}(\neg\phi \vee \psi)$$
$$\mathbf{ST}(\phi \leftrightarrow \psi) = \mathbf{ST}(\neg\phi \wedge \neg\psi \vee \phi \wedge \psi).$$

Similarly, the XOR connective for ST conditional expressions is composed with an equivalent logical representation as follows.

$$\mathbf{HP}((\phi \texttt{XOR} \psi) = \mathbf{HP}(((\texttt{NOT}(\phi))\texttt{AND}\psi)\texttt{OR}(\phi\texttt{AND}(\texttt{NOT}(\psi))))$$

Proposition 5. The systematic mapping for logical formulas preserve the semantics of the respective languages if the translation of the child formulas preserve their respective semantics.

Proof. In both languages, the logical formulas are similarly interpreted in an inductive fashion. The semantics for a d$\mathcal{L}$ logical negation of a formula with an interpretation I and a corresponding set of states S are defined as follows:

$$[\![\neg\theta]\!]^I v = S \backslash [\![\theta]\!]^I$$

The semantics for a logical negation in ST is defined as follows

$$\frac{(c_1, \sigma)}{((\neg c_1), \sigma) \rightarrow_a \neg c_1}$$

As such, the semantics are preserved by induction based on the translation of the formula being logically negated in both contexts.

For the logical formulas with logical connectives, the semantics of d$\mathcal{L}$ formulas are similarly defined in an inductive fashion.

$$[\![\theta \bowtie_{\texttt{HP}} \eta]\!]^I v = v \in S : [\![\theta]\!]^I \bowtie_{\texttt{HP}} [\![\eta]\!]^I v$$

The semantics for ST logical formulas with logical connectives are defined as follows.

$$\frac{(c_1, \sigma) \rightarrow_a c_1 (c_2, \sigma) \rightarrow_a c_2}{((c_1 \bowtie_{\texttt{ST}} c_2), \sigma) \rightarrow_a c_1 \bowtie_{\texttt{ST}} c_2}$$

The semantics for each logical formula are preserved by induction following Proposition 5.4. □

As was previously mentioned, we are only providing a system mapping of modal- and quantifier-free d$\mathcal{L}$ formulas. If any quantifier formulas are encountered, a compiler error is raised. Modal formulas are associated with a program definition for hybrid programs. As such, they require a certain syntactic pattern as well as a systematic mapping for the associated programs that will be defined in the subsequent section.

5.5 Translation of Programs

In this paper we use the term *program* to encompass hybrid programs specified in d$\mathcal{L}$ and ST statements that compose the execution of a PLC scan cycle. As with the previous two sections, we will first provide the syntactic grammars for programs in both languages followed by the associated systematic mapping.

5.5.1 Grammar Definitions

We present the respective grammars for programs in each language.

Grammar of ST programs. ST programs refer to the sequence of statements defined by the IEC 61131-3 standard that compose entire ST programs. We will initially provide a base set of grammar components that can be used to express all types of programs as defined by the standard. The grammar we consider for ST statements α and β is defined as follows.

$$\alpha, \beta ::= x := \theta \mid IF(\psi)\ THEN\ \alpha\ ELSE\ \beta\ END\_IF; \mid \alpha; \beta$$

Where $x := \theta$ is assignment of an ST term θ to some variable, x, $IF\ (\psi)\ THEN\ \alpha\ ELSE\ \beta\ \ END\_IF;$ is a conditional statement where α is executed if ψ is true and executes β otherwise, and $\alpha; \beta$ sequential composition of ST programs where β executes only when α has finished its execution. While the structured grammar can support several other control structures such as finitely bounded loops and case-statements, these structures can be represented as a series of if-then-else statements and are generally implemented for ease of programmability. The d$\mathcal{L}$ grammar is composed in similar fashion.

Grammar of d$\mathcal{L}$ programs. The grammar for translatable d$\mathcal{L}$ hybrid programs is defined as follows.

$$\alpha, \beta ::= a \mid x := \theta \mid ?\psi \mid \alpha \cup \beta \mid \alpha; \beta$$

Where $x := \theta$ are assignments of a term θ to some variable x, $?\psi$ represents a test based on the formula ψ in the current state, and $\alpha \cup \beta$ represents a nondeterministic choice between programs α and β, Platzer (2008, 2015). Given these base grammars for the programs, we now present the systematic mapping and the associated correctness assumptions that will allow us to provide the complete bi-directional translation of hybrid programs and ST program and configuration elements. We then discuss how we translate the currently unhandled constructs such as nondeterministic repetition and differential assignments.

5.5.2 Systematic Mapping

Deterministic assignment. Assignments of terms to variables in hybrid programs represent the core of discrete state transitions for a hybrid system. In the context of PLC control, deterministic assignment $x := \theta$ will be interpreted as actuation in ST. Hence, generation of an ST assignment statement, $\mathbf{ST}(x := \theta)$, requires a few restrictions on which a variable is being assigned to as well as on the term θ. HyPLC checks that assignments between variables resolve to the same type, i.e., the data types of the generated ST configuration must agree. As with the model checking that is performed on hybrid programs to verify that a particular safety requirement holds, an ST compiler would enforce type-checking for the generated configuration.

HyPLC also ensures that only output variables are being assigned values. This provides an additional model sanity check to ensure that only variables associated with actuation, not sensing, are being written[1]. Having these provisions, the compiled assignment will be generated as follows, where I denotes the set of input variables and

[1] A simpler approach would be to always declare ST variables as `IN_OUT` I/O types so that they can act as both input and output, but we want to provide the developer with the ability to enforce strict I/O assignments.

O refers to the set of output variables. Variables that are in $I \cap O$ are I/O variables.

$$\mathbf{ST}(x := \theta, I, O) = \mathbf{ST}(x) \mathbf{:=} \mathbf{ST}(\theta); \qquad \text{if } x \notin I \text{ else ERROR}$$

As shown in the formulation above, ST actually uses the same syntax for assignments, including a semi-colon to separate sequential statements. The translation of variable assignments in ST to discrete assignments in hybrid programs does not require any of the aforementioned constriants as the grammar for d$\mathcal{L}$ does not include input/output specification, i.e.,

$$\mathbf{HP}(x := \theta) = \mathbf{HP}(x) := \mathbf{HP}(\theta);$$

Proposition 6. The semantics of determistic assignment is preserved for both languages.

Proof. For both languages, the dynamic semantics of the child terms for each program are preserved by induction based on the previously discussed preservation of semantics. To prove that the semantics of the overall assignment programs are preserved across both translations, we must focus on the transition semantics of the programs from one discrete state to the next. For hybrid programs, an interpretation of a hybrid program α is semantically interpreted as a binary transition $[\![\alpha]\!]^I \subseteq SxS$ on states. As such, the transition semantics for d$\mathcal{L}$ discrete assignments are defined as follows.

$$[\![x := \theta]\!]^I v = \{(v, v^r{}_x : r = [\![\theta]\!]^I v\} = \{(v, \omega) : \omega = v \; except [\![x]\!]^I \omega = [\![\theta]\!]^I v\}$$

Similarly, we define the transition semantics for ST programs where we semantically interpret an assignment as the binary transition from an initial context σ_0 that transitions to a subsequent context σ_1 that determines the values of the physical outputs and initial values of the retained variables for a subsequent PLC scan cycle, i.e.,

$$\frac{(e_1, \sigma) \rightarrow_a c_1}{(\langle v_1 := e_1; \rangle, \sigma) \rightarrow (\langle \mathbf{skip}; \rangle, \sigma[v_1 \mapsto c_1])}$$

where **skip** denotes the end of code for this scan cycle. For translation of ST assignments to d$\mathcal{L}$ discrete assignments, the semantics are preserved by induction based on the notion that a translated ST variable will take on the value of a translated ST term after transitions from the current state to the next. The inverse translation is true as well based on the same equivalence of discrete-state transitions for both languages. □

Nondeterministic assignment. Nondeterministic assignment only exists in the context of d$\mathcal{L}$ hybrid programs. HyPLC interprets nondeterministic assignments ($x := *$) as sensor inputs, since nondeterministic assignments themselves are not supported in ST. As such, nondeterministic assignments are omitted during translation, as indicated with the `skip` operation:

$$\mathbf{ST}(x := *, I, O) = \texttt{skip} \qquad \text{if } x \in I \text{ else ERROR}$$

Based on our strict I/O type enforcement, we can ensure that there will be no undefined behavior. The `ERROR` statement represents the abortion of a compilation process if a violation is detected. For instance, the d$\mathcal{L}$ expression "`x:=x+1; x:=*`" would be invalid since the variable x must have an exclusive input/output type. We would therefore report an error and abort the compilation for this expression.

Revisiting the structured code presented in Figure 5.3, the water tank example follows this patterns: it uses nondeterministic assignment to model the water inflow/outflow sensors f_n, and deterministic assignment to model the valve control output variables V_n. The systematic mapping of input variables to nondeterministic assignment does not necessarily preserve the semantics of assigning a nondeterministic value to a d$\mathcal{L}$ variable, HyPLC preserves the implicit nondeterminism expected from sensor inputs. Having defined the systematic mapping of deterministic and nondeterministic assignments, we can now use these atomic programs to define the bi-directional translation of compositional programs.

Sequential composition programs. The sequential composition of two hybrid programs α and β states that the hybrid program β starts executing after α has finished,

meaning that β never starts if the program α does not terminate. Because ST statements are composed in an identical manner, semantic translation of sequential hybrid programs is straightforward. An ST statement can only begin execution once the previous ST statement has completed. Therefore, we have the following translations.

$$\mathbf{ST}(\alpha;\beta) = \mathbf{ST}(\alpha);\mathbf{ST}(\beta)\ \mathbf{HP}(\alpha;\beta) = \mathbf{HP}(\alpha);\mathbf{HP}(\beta)$$

Proposition 7. The bi-directional translation of sequentially composed programs preserves the semantics of the respective languages.

Proof. The semantics of the interpretation of a sequentially composed program are preserved by induction for both cases as the syntactic mapping is identical. However, as with other programs, we must ensure that the transitional semantics are preserved as well. The transitional semantics for composed hybrid program is as follows.

$$[\![\alpha;\beta]\!]^I = [\![\alpha]\!]^I \circ [\![\beta]\!]^I = \{(v,\omega) : (v,\mu) \in [\![\alpha]\!]^I, (\mu,\omega) \in [\![\beta]\!]^I\}$$

Similarly, the transitional semantics for a sequentially composed ST program is as follows.

$$\frac{(\langle s_1;\rangle,\sigma) \to (\langle \mathbf{skip};\rangle,\sigma')}{(\langle s_1; s_2;\rangle,\sigma) \to (\langle s_2;\rangle,\sigma')}$$

In both cases, the subsequent context or state is not allowed to execute until the prior context or state has completed execution. The semantics of the child programs are preserved by induction. □

Remark 2 (ST Task Execution Timing)**.** The execution of a series of statements with respect to sequential composition assumes that the statements execute atomically, which is defined in the transition semantics of hybrid programs. We do not model the preemption of higher priority tasks as the modeling of the PLC's task scheduling is outside of the scope of the paper and is left for future research directions. HyPLC assumes that the developer designs a hybrid system such that a system with multiple tasks is designed such that (1) the execution time of a highest priority task is significantly less

than its period and that (2) the total execution of all tasks is significantly less than the period of the lowest priority tasks Automation (2018). However, as we will discuss in later subsections, we have instrumented HyPLC to retain the priority assignments based on the design of the hybrid program.

Conditional programs. Standalone tests for hybrid programs ($?\phi$) are not directly translated to ST programs, since a failed test requires backtracking to other program options. Similar to nondeterministic assignments, we therefore allow tests to occur only in specific places in the beginning of the branches of nondeterministic choices. Furthermore, only specific cases of nondeterministic choice in which a choice is preceded by a particular test that decides whether to execute a branch or not. As such, a nondeterministic choice between hybrid programs α and β executes either hybrid program and is resolved by favoring execution of α over β in an if-then-else statement. To avoid backtracking, we require the shape $?\phi;\alpha$ on the first branch. The associated translation is defined as follows.

$$\mathbf{ST}(?\phi;\alpha;\cup\beta) = \begin{cases} IF\big(\mathbf{ST}(\phi)\big) \\ THEN\big(\mathbf{ST}(\alpha)\big) \\ ELSE\big(\mathbf{ST}(\beta)\big) \\ END\_IF; \end{cases} \tag{5.1}$$

HyPLC enforces the coupling of tests with non-determnistic choice by raising a compilation error upon encountering a stand-alone test or a non-determistic choice between programs where at least one of the programs is not prefixed with a test. Because we only consider loop-free semantics, we avoid having to enforce backtracking for deep tests that may exist in α or β. Instead, the tests will simply be compiled as nested conditional programs. The same restrictions will apply when translating ST conditional program statements to hybrid programs, i.e.,

$$\mathbf{HP}(\texttt{IF}\ \ (\psi)\ \ \texttt{THEN}\ \ \alpha\ \ \texttt{ELSE}\ \ \beta) \equiv ?\mathbf{HP}(\psi);\ \mathbf{HP}(\alpha);\ \cup\ ?\mathbf{HP}(\neg\psi);\ \mathbf{HP}(\beta);$$

Notice that for the translation of an ST if-then-else statement to a hybrid program, we must enforce the negation of the condition, ψ, on the non-determistic branch that

includes β.

Remark 3 (False-equivalence of `FALSE`)**.** There are instances where hybrid programs involving $False$ formulas cannot be semantically translated to ST. For instance, a test $?False$ on every execution path of a program would terminate the program without runs, which is not a very useful controller. We therefore restrict our translation to validated models that do not have these impasses.

Remark 4 (Model compliance guarantee)**.** The aforementioned assumption where nondeterministic choices are each preceded by tests of complementary formulas would be guaranteed by model compliance checking methods such as Modelplex Mitsch and Platzer (2016). For instance, in the first IF-THEN-ELSE statement in Figure 5.3 (lines 15-19) consists of a nested IF-THEN-ELSE sequence. HyPLC converts these lines to the following hybrid program:

$$?x_1 >= H_1 \& x_1 > L_1; V_1 := 0; \cup ?x_1 <= L_1 \& x_1 < H_1; V_1 := 1;$$

As you can see, for the first conditional formula $?x_1 >= H_1$, it's complementary test, $x_1 < H_1$, is included in the second nondeterministic choice. If the complementary case was not included, the hybrid program would be allowed to arbitrarily choose a path of execution regardless of the value of the variable x_1. It would represent nondeterministic execution of the discrete control of the translated PLC code, which is not correct.

The d$\mathcal{L}$ grammar has been augmented to include an IF-THEN-ELSE statement by construct that enforces these restrictions Quesel et al. (2016). In the sequel, we will discuss the overall translation of hybrid programs and ST programs that include all configuration elements.

5.5.3 Cyclic Control Configuration

Nondeterministic repetition, α^*, of a hybrid program, α, generates an ST program's cyclic configuration. Although the nondeterministic repetition operator of hybrid programs provides a loop structure, additional constraints on timing are required in order

to reintroduce determinism to the PLC's cyclic control sensing and actuation, i.e., HyPLC expects an input hybrid program to be a *time-triggered model*. A typical modeling pattern for time-triggered controllers in hybrid programs is $\{ctrl; t := 0; \{..., t' = 1 \& t \leq \epsilon\}\}^*$, where t is a clock variable, *ctrl* is the discrete control of the system, and ϵ is the upper bound evolution domain constraint for the clock variable. Such a program structure allows us to model time-triggered control. In this pattern, the evolution domain constraint $t \leq \epsilon$ of the differential equation system determines the execution interval of the controller *ctrl*. Because PLC task needs to be executed for a particular time, ϵ, HyPLC requires an input hybrid program to be a time-triggered model to take the exact following shape

$$init \rightarrow [ctrl;\ t := 0; \{t' = 1, t \leq \epsilon \& Q\} *]req,$$

where t is a fresh clock variable assigned to the PLC task, ϵ is the cyclic execution interval for a particular task that is provided by the auxiliary file, and Q is the evolution domain constraint for the plant of the model. HyPLC will only translate the *ctrl* portion of the hybrid program. For a single task, we initially assume that ϵ includes the execution timing for a PLC scan cycle, i.e., for the translation a hybrid program with the aforementioned shape we have

$$\begin{aligned} \mathbf{ST}(init \rightarrow [\{ctrl; plant\}^* req]) = \texttt{Task}(\mathbf{ST}(ctrl), \epsilon), \\ \text{where } plant \equiv t := 0; \{t' = 1, t \leq \epsilon \& Q\} \end{aligned}$$

and $\texttt{Task}(\alpha,\epsilon)$ is a shorthand tuple defining a task[2] that executes the translated discrete control, *ctrl*, cyclically with an interval ϵ. Similarly, we define the translation of a

[2]A task is being used here to abstract the other configuration components of an ST program, i.e., Configurations and Resources. We assume only one configuration and one resource at a time in this paper for a single PLC.

structured text program α with a single task and execution time of ϵ as

$$\mathbf{HP}(Task(\alpha, \epsilon)) = init \rightarrow [\mathbf{HP}(\alpha); plant]$$

$$\text{where } plant \equiv t := 0; \{t' = 1, t \leq \epsilon \& Q\}.$$

We next extend our translations to include multiple tasks in the context of a PLC scan cycle.

```
1  PROGRAM prog0
2    //prog0 code
3  END_PROGRAM
4  PROGRAM prog1
5    //prog1 code
6  END_PROGRAM
7  ...
8  PROGRAM progN
9    //progN code
10 END_PROGRAM
11
12 CONFIGURATION Config0
13 RESOURCE Res0 ON PLC
14 TASK Task0(INTERVAL:=T#Interval0 s,PRIORITY:=0);
15 TASK Task1(INTERVAL:=T#Interval1 s,PRIORITY:=0);
16 ...
17 TASK TaskN(INTERVAL:=T#IntervalN s, PRIORITY:=0);
18 PROGRAM Inst0 WITH Task0 : prog0;
19 PROGRAM Inst1 WITH Task1 : prog1;
20 ...
21 PROGRAM InstN WITH TaskN : progN;
22 END_RESOURCE
23 END_CONFIGURATION
```

Figure 5.4: Configuration for multiple ST tasks.

Extension: translation of multiple tasks. For multiple tasks, we consider a single configuration of a PLC with a single resource that has a one-to-one mapping of task configurations to ST programs. Figure 5.3 shows a single progrmam, `prog0`, for a single task, `Main`. We can similarly define other tasks with their associated programs as shown in Figure 5.4, where $\texttt{Interval}_n$ defines the interval ϵ for each $task_n$ that will execute a program $prog_n$. To provide a translation of such a configuration with multiple ST tasks to a hybrid program, HyPLC designates a fresh clock variable[3], t_n, for every task that executes on a particular interval, ϵ_n. Furthermore, the task execution clocks are complemented with an additional clock variable, t_{plc}, that represents the scan cycle

[3]The clock variables are assigned a `CLK` type in the aforementioned auxiliary file.

timing of the PLC. This defines how often the PLC will check to see if the any of the clock variables have elapsed. As such, the translation of multiple ST task configurations to a hybrid program will take on the following shape

$$
\begin{aligned}
&\mathbf{HP}(Task_0(\alpha_0,\epsilon_0),Task_1(\alpha_1,\epsilon_1),\ldots,Task_n(\alpha_n,\epsilon_n)) = \\
&\qquad init \rightarrow [tasks; t_{plc} := 0; plant]^* req; \\
&\text{where } tasks \equiv ?t_0 \geq \epsilon_1; ctrl_0; t_0 := 0; \cup ?t_0 < \epsilon_0; \\
&\qquad\qquad ?t_1 \geq \epsilon_1; ctrl_1; t_1 := 0; \cup ?t_1 < \epsilon_1; \\
&\qquad\qquad \ldots \\
&\qquad\qquad ?t_n \geq \epsilon_n; ctrl_n; t_n := 0; \cup ?t_n < \epsilon_n; \\
&\text{and } plant \equiv \; t'_{plc} = 1 \wedge t'_1 = 1 \wedge t'_2 = 1 \ldots \wedge t'_n = 1 \& t_{plc} < \epsilon_{plc} \& Q.
\end{aligned}
$$

As before, the translation in both directions will require this format for both an output and input hybrid program in HyPLC. We now discuss optional extensions of HyPLC.

Remark 5 (PLC Scan Cycle Time Bound)**.** The aforementioned approach requires that the programmer specify a bound T_{plc} for the scan cycle clock variable, t_{plc}. This timer represents how quickly the PLC can check to see if time- or event-triggered task has to be executed or not. Although the execution of instructions can be nondeterministic, the execution time bound can be estimated, for example, with the approach in Wilhelm (2005).

Remark 6 (Optional Extension: Task Priority)**.** Because we assume atomicity of the ST task execution, the priorities of the tasks are irrelevant. However, if a developer designs a hybrid system such that the priority of the tasks need to be maintained, the previous approach can be augmented to simply assign the tasks an incremental priority number in the order in which they occur in the hybrid program, i.e., the sequential ordering of the task in the hybrid program will determine its priority.

Now that we have provided the systematic mappings that will be used by HyPLC,

we will now evaluate the tool on a real system.

5.6 Evaluation

HyPLC was implemented as two module extensions for the `KeymaeraX` tool: one for each translational direction. For the compilation of hybrid program to ST, the aforementioned systematic mapping rules were implemented on top of the exisiting lexical analyzer of the `KeymaeraX` tool. Given the abstract syntax tree of a hybrid program, HyPLC generates the associated ST code based on the aforementioned generation rules. The implementation was written in ∼700 LoC. Similarly, the module for the compilation of an ST program to a hybrid program was implemeneted on top of the lexical analysis provided by the MATIEC IEC 61131-3 compiler Sousa. The compiler itself already provides modules that compile ST programs to either C code or other languages provided by the IEC standard. As such, we utilized the same APIs to implement the generation rules given the abstract syntax tree of an ST program. The module was implemented in `C++` with ∼1000 LoC.

We next present how HyPLC was evaluated against the aforementioned water treatment testbed.

5.6.1 Use Case: Water Treatment Testbed

We first translate the PLC code from the water treatment testbed shown in Figure 5.3. Our results will show that this implementation is unsafe. We then update the generated hybrid program with the necessary parameters to guarantee the safety of the ICS. Finally, we generate the PLC code that propagates these safety guarantees.

Checking the Safety Existing PLC Code

In order to generate the associated hybrid program of the water treatment testbed, we need to provide the continuous plant, *plant*, of the ICS as well as the constraints, *init*, that intialize the discrete states of the system variables. These will be combined with the compiled *ctrl* of the ICS that provides the discrete-state transitions of the system.

Finally, we define the safety requirement, req, that ensures that the water tank levels will remain within their upper and lower thresholds.

$$
\begin{aligned}
&init \Rightarrow [\{ctrl; plant\&H\}*](req)\\
&init \equiv L_1 \leq x_1 \bigwedge x_1 \leq H_1 \bigwedge L_2 \leq x_2 \bigwedge x_2 \leq H_2 \bigwedge V_1 = 0 \bigwedge V_2 = 0 \bigwedge f_1 = 0\\
&\quad \bigwedge f_2 = 0 \bigwedge P = 0 \bigwedge t_{plc} = 0 \bigwedge t_{sample} = 0 \bigwedge T_{sample} > T_{plc} \bigwedge T_{plc} >= 0\\
&\quad \bigwedge F_L > 0 \bigwedge L_L < L_1 \bigwedge L_L < L_2 \bigwedge L_1 < H_1 \bigwedge L_2 < H_2 \bigwedge H_1 < H_H \bigwedge H_2 < H_H\\
&ctrl \equiv \{f_1 := *; f_2 := *; //\text{Sensors}\\
&\quad ?(t_{sample} \geq T_{sample});\\
&\quad \{?x_1 \geq H_1 \bigwedge x_1 > L_1; V_1 := 0; \cup ?x_1 \leq L_1 \bigwedge x_1 < H_1; V1 := 1;\}\\
&\quad \{?x_2 \leq L_2; P := 1; V_2 := 1; \cup ?x_2 > L_2;\}\\
&\quad \{?(x_1 < L_L \vee f_2 F_L \vee x_2 > H_2); P := 0; V_2 := 0; \cup ?!(x_1 < L_L \vee f_2 F_L \vee x_2 > H_2);\}\\
&\quad t_{sample} := 0;\}\\
&\quad \cup ?(t_{sample} < T_{sample});\}\\
&\quad t_{plc} := 0;\\
&plant \equiv x_1' = V_1 * f_1 - V_2 * P * f_2, x_2' = V_2 * P * f_2, t_{plc}' = 1, t_{sample}' = 1\\
&H \equiv t_{plc} \leq T_{plc} \bigwedge x_1 \geq 0 \bigwedge x_2 \geq 0 \bigwedge f_1 \geq 0 \bigwedge f_2 \geq 0\\
&req \equiv L_L \leq x_1 \bigwedge x_1 \leq H_H \bigwedge L_L \leq x_2 \bigwedge x_2 \leq H_H
\end{aligned}
$$

Figure 5.5: Hybrid program generated by HyPLC. This is a compilation of the PLC code in Figure 5.3.

Figure 5.5 shows the full hybrid program generated by HyPLC that incorporates both the translated ST code as well as the continuous dynamics of the water treatment testbed. Intuitively, this model cannot be proven as there are not constraints on the flowrates, f_1 and f_2, of the system in this model, nor do the guards on actuation depend on these sensor values that might enforce such constraints. This means that for this model, there is not way of proving that a flowrate sensor would have an infinitely large value. We confirmed this after loading the model into `KeymaeraX`and being left with open goals on the constraints of the flowrate. The next step will be to modify this model to guarantee the model will be safe.

$$init \Rightarrow [\{ctrl; plant\&H\}*](req)$$
$$ctrl \equiv \{f_1 := *; f_2 := *; //\text{Sensors}$$
$$?(t_{sample} \geq T_{sample});$$
$$\{?((f_1 - f_2) < \frac{L_L - x_1}{T_{sample} + T_plc})$$
$$\wedge ((f_1 - f_2) > \frac{H_H - x_1}{T_{sample} + T_plc});$$
$$V_1 := 0;$$
$$\{?((f_1 - f_2) \geq \frac{L_L - x_1}{T_{sample} + T_plc})$$
$$\wedge ((f_1 - f_2) \leq \frac{H_H - x_1}{T_{sample} + T_plc});$$
$$V1 := 1;\}$$
$$?((f_2) > \frac{H_H - x_2}{T_{sample} + T_plc});$$
$$P := 0; V_2 := 0;$$
$$?((f_2) \leq \frac{H_H - x_2}{T_{sample} + T_plc});$$
$$P := 1; V_2 := 1;\}$$
$$t_{sample} := 0;\}$$
$$\cup?(t_{sample} < T_{sample});\}$$
$$t_{plc} := 0;$$

(a) Hybrid program with safe *ctrl*.

```
1   IF(((f1-f2)<((LL-x1)/(
       Tsample+Tplc)))) AND
        (((f1-f2)>((HH-x1)
       /(Tsample+Tplc))))
       THEN
2     V1:=0; ELSE
3   IF(((f1-f2)>=((LL-x1)/(
       Tsample+Tplc)))) AND
        (((f1-f2)<=((HH-x1)
       /(Tsample+Tplc))))
       THEN
4     V:=1;
5   END_IF;
6   END_IF;
7   IF(((f2)>((HH-x2)/(
       Tsample+Tplc))))
       THEN
8     V2:=0; P:= 0; ELSE
9   IF(((f2)>((HH-x2)/(
       Tsample+Tplc))))
       THEN
10    V2:=1; P:= 1;
11  END_IF;
12  END_IF;
```

(b) Updated ST code fragment.

Figure 5.6: The hybrid program with and updated *ctrl* that is guaranteed to be safe and the associated ST translation code fragment (excluding variable and configuration declarations).

Generating Safe PLC Code

The hybrid program was updated to reflect a safe system that constrained the flowrates by modifying the guard values on the discrete control. Figure 5.6 shows the updated hybrid program that was proven to be safe with `KeymaeraX`. Once verified, HyPLC generates the associated PLC code also depicted in Figure 5.6.

Comparison on real-world data. To illustrate the safety guarantees of our system,

we developed a monitor for the sensor and actuation values of the water treatment testbed and analyzed 4 days worth of sensor data Goh et al. (2016). The monitor was based on the verified model showing in Figure 5.6. The monitor found there to be no violations of the safety constraints given the current state of the code. This due to the fact that the trigger values of the system, i.e., H_1, H_2, L_1 and L_2, were set cautiously based on empirical data. Intuitively, we can reason about the constraints on the flow rate of the valves. The specifications for the manual show the flow rate to be rated for max value of about $2.6\frac{m^3}{h}$. During our evaluation, we observed a max value of about $2.75\frac{m^3}{h}$. Using the state estimation formulas with the associated conversion constants calculated in a previous work Ahmed et al. (2016), we found the trigger thresholds to satisfy our safety guards based on the flow rate that were generated by the updated hybrid program. This study allowed us to not generate safe PLC code, but to also formally reason about unsafe PLC code that has empirically proven to be safe.

5.7 Conclusion

In this paper, we formalize a systematic mapping between safety-critical code utilized in industrial control systems (ICS) and the discrete control of hybrid programs specified in differential dynamic logic (d$\mathcal{L}$). We present HyPLC, a tool that provides the bi-directional translation of code loaded onto programmable logic controllers (PLCs) to and from hybrid programs specified in d$\mathcal{L}$ to provide safety guarantees for "deep" correctness properties of the PLC code in the context of the cyber-physical ICS. We evaluated HyPLC on a real water treatment testbed, demonstrating how HyPLC can be utilized to both verify the safety of existing PLC code as well as generate correct PLC code given a verified hybrid program. This work serves as a foundation for pragmatic verification of PLC code as well as to understand the safety implications of a particular implementation given complex cyber-physical interdependencies.

Chapter 6

Control Behavior Integrity for Distributed Cyber-Physical Systems

6.1 Introduction

Industrial control system (ICS) are used in a multitude of control systems across several applications of industrial sectors and critical infrastructures, including electric power transmission and distribution, oil and natural gas production, refinery operations, water treatment systems, wastewater collection systems, as well as pipeline transport systems Stouffer et al. (2011). ICS typically consist of interconnected embedded systems, called programmable logic controllers (PLCs). In a distributed ICS, multiple PLCs jointly control a physical process or the physical environment. Using a series of sensors and actuators, PLCs can monitor the physical system's state and control the system behavior. This makes the correct functioning of PLCs crucial for the correct and safe operation of these systems.

This critical role of the PLCs makes them a valuable target for adversaries aiming to interfere with any of these systems Adepu et al. (2017). Past incidences show that such attacks are applied in practice, often remaining undetected over a long period of time. Examples include the infamous Stuxnet worm Falliere et al. (2010) against Iranian nuclear uranium enrichment facilities as well as the BlackEnergy crimeware F-Secure Labs (2016) against the Ukranian train railway and electricity power industries. These attacks demonstrate impressively that targeted attacks on critical infrastructure can evade traditional cybersecurity detection and cause catastrophic failures with substantive impact. The discovery of Duqu Chien et al. (2011) and Havex Rrushi et al. show that such attacks are not isolated cases as they infected ICS in more than eight countries. Nation-state ICS malware have typically either targeted the control programs of

PLCs or the central control infrastructure (operator workstations). However, academic research has demonstrated even more sophisticated attacks against ICS and PLCs that can circumvent existing defense mechanisms by manipulating the PLC's firmware and incorporating physics-aware models into the attack code Garcia et al. (2017b).

A comprehensive defense against ICS attacks needs to protect against various attack vectors. (1) The software determining a PLC's behavior could be replaced by a malicious program Falliere et al. (2010); Klick et al. (2015); Brüggemann and Spenneberg (2015). Updating the PLC control program over the network is an intended functionality of PLCs to allow central management. However, the control program can also be manipulated if an attacker gains physical access to a PLC.(2) The PLC firmware (which includes the OS) could be manipulated/replaced either via the network or through physical access Garcia et al. (2017b); Basnight et al. (2013). (3) The attack can exploit a memory corruption vulnerability (e.g., buffer overflow Schuett (2014)) in the PLC's control programs and/or firmware for code-injection or to launch run-time attacks such as return-oriented programming (ROP) Roemer et al. (2012), to manipulate a PLC's behavior. (4) Memory corruption vulnerabilities can be exploited to launch data-only attacks Hu et al. (2016) against a PLC to manipulate its behavior. For instance, the initiation of a trigger-response may be inhibited by manipulating the associated control parameters Quarta et al. (2017), e.g., a threshold value that determines whether the action must be started.

For all above enumerated attack vectors, a common goal of the adversary is to modify the physical behavior of the system. As long as the attacked device behaves correctly the overall system will continue to operate correctly. Therefore, the ultimate goal of the attacker will always be to change a device's behavior.

Previous works in defending against ICS attacks focus only on a subset of the above listed attack vectors. These approaches can be generally categorized into two categories: defenses that focus on verifying the integrity of the software running on a PLC and defenses that verify the behavior of the overall ICS based on models that abstract control decisions of the PLC software. In the former case, PLC-based verification solutions typically cannot account for attacks which replace and/or modify either the application

layer programs or the underlying firmware. For instance, ECFI Abbasi et al. (2017) provides protection against run-time attacks targeting PLC control programs, but does not protect against data-only attacks nor maliciously modified/replaced control programs or firmware. Orpheus Cheng et al. (2017) monitors the behavior of a device's control program based on the invoked system calls. Attacks are detected based on a finite-state machine (FSM) representing the control program's benign system call behavior. Orpheus' behavior monitor is placed inside the device's OS, hence, a compromised OS can disable and circumvent its protection mechanism.

Similarly, solutions that enforce compliance via state estimation Adepu and Mathur (2016a) or cyber-physical access control Etigowni et al. (2016b) from within the PLC could be circumvented as well. Zeus Han et al. (2017) uses side-channel analysis to verify the control flow of programs running on a PLC, but cannot defend against firmware modifications nor sensor data attacks. By extension, offline, static analysis of control programs being loaded onto PLCs McLaughlin et al. (2014) Darvas et al. (2015a) provides even less run-time guarantees. For ICS-based verification techniques, it has been shown that state estimation can be used to infer the control commands issued by distributed controllers Etigowni et al. or to detect false data injection attacks Liu et al. (2011b) based on the sensor data. Such protection mechanisms may be circumvented via physics-aware attacks Garcia et al. (2017b) Garcia et al. (2014). Further, supervised machine learning has been used to characterize physical invariants of the CPS Chen et al. (2018). However, such an approach depends on the training data to include all corner cases of the system execution and is not based on the control flow of the software.

We present Scadman, the first *control behavior integrity* (CBI) solution for distributed industrial control systems. Unlike previous state estimation approaches Scadman does not abstract the behavior of the cyber-components (i.e., PLCs). Instead, Scadman *precisely* simulates the state of all PLCs. By monitoring the input and output behavior of the entire ICS, Scadman can detect inconsistencies within the actions of PLCs. To enable a global view of the entire ICS, a consolidated control program of all PLCs in the system must be generated to resolve functional dependencies between individual programs. The consolidated control program in conjunction with a physical

state estimator is used to determine a set of acceptable states. For that, SCADMAN needs means to analyze which control-flow paths are valid given the current system state. Based on this context-aware control-flow path analysis, SCADMAN determines benign resulting states. Comparing the set of benign states against the reported sensor readings and actuation commands from the ICS allows SCADMAN to detect anomalies in the system behavior. This makes SCADMAN agnostic to the attack technique used to cause a PLC to deviate from its intended behavior and makes SCADMAN a powerful tool to protect ICS against a wide range of attack vectors.

We evaluated SCADMAN on a real ICS, the Water Treatment Testbed SWaT.[1] SWaT is the quasi-standard for security research in the context of ICS, as shown by the huge number of previous works evaluated on SWaT Junejo and Yau (2016); Lin et al. (2018); Chen et al. (2018, 2016); Goh et al. (2017); Kong et al. (2016); Wang et al. (2017b,a); Inoue et al. (2017); Umer et al. (2017); Pal et al. (2017). Simulation-based evaluation does not provide a viable option for SCADMAN. In general, simulations are based on models of an ICS similar to SCADMAN. Hence, such an evaluation would validate the accuracy of our models against the model of the simulator–leading to no meaningful results.

We make the following contributions:

- We present SCADMAN, the first control behavior integrity system for distributed ICS based on a model comprising cyber *and* physical components.
- Our solution does not require any changes to the hardware or software of the PLCs, making it independent of the PLC manufacturers. Furthermore, leaving the PLCs unmodified is important for safety certifications to remain valid in the presence of SCADMAN.
- We provide an automated solution that allows SCADMAN to consolidate the control programs of all PLCs in an ICS. This allows us to comprehensively simulate the control behavior of the entire system, which is important for detecting inconsistent behavior across the borders of individual PLCs.

[1] https://itrust.sutd.edu.sg/research/testbeds/secure-water-treatment-swat/

- We implemented SCADMAN using the MATIEC compiler from the OpenPLC project for automated generation of the consolidated PLC control program and LLVM for instrumentation of the consolidated PLC control program.
- We evaluated SCADMAN using real data from the SWaT water treatment ICS testbed. We show that we can detect all attacks executed on SWaT so far.

The rest of the paper is structured as follows. First, we provide background on the most relevant topics related to our work (industrial control systems, control-flow integrity and cyber-physical system modeling) in section 6.2. We define the assumptions and system model underlying our work in section 6.3. In section 6.4 we explain the design and main ideas of SCADMAN. We detail on our implementation in section 6.5. In section 6.6 we discuss SCADMAN's security for various attack scenarios and present our evaluation results in section 6.7. Relevant related work is discussed in section 6.8. section 6.9 proposes future work directions, while section 6.10 concludes.

6.2 Background

In this section we first provide background on industrial control systems (ICS) in general. Afterwards we introduce two concepts–control-flow integrity (CFI) and cyber-physical systems modeling–which have been used in the past in an attempt to secure ICS, however, none of them are sufficient to solve this challenge.

Industrial Control Systems. Programmable logic controllers (PLC) are cyber-physical systems that are used to control industrial appliances. PLCs feature input and output modules, which translate physical inputs – in most cases currents on a wire – into digital values and vice versa, to interact with the physical appliances like sensors and actuators.

PLCs can convert sensor readings into digital values, process the readings with the built-in computing unit, and forward the outputs to actuators to manipulate the physical world. Based on the available information about the system state, a PLC calculates the next actuations to steer the system towards a desired state. The program running

on the PLC, called *control logic*, determines the control algorithm used to decide actuations. The control logic program(s) of a PLC are Turing complete and programmable using the development environments provided by the PLC manufacturers. The target system state towards which the PLC is working can be fixed in the control logic or could be set dynamically over the network by the ICS operator.

Control logic programs can be loaded onto PLCs and run on top of a privileged software layer like a real-time operating system (RTOS). This privileged software layer contained in the PLC's firmware provides services to the control logic programs (e.g., networking, storage) and manages the programs' updates and execution. The control logic programs are executed repeatedly in fixed intervals, called *scan cycles*. Furthermore, in a distributed ICS, the physical process is jointly controlled by multiple PLCs. To do so, PLCs are usually connected through a computer network, allowing them to share sensor readings or internal states.

The PLCs in an ICS are usually managed and monitored through central management systems, called *Supervisory Control and Data Acquisition* (SCADA). Typical components of a SCADA system are *historians*, which are databases logging data from all control devices in the ICS, *human machine interfaces* (HMI), which allow an operator to interactively control the system, operator workstations also allowing interactive control as well as PLC reprogramming, and IT infrastructure like servers that connect the ICS to systems such as a supply chain management system.

Control-Flow Integrity. Control-flow integrity (CFI) is a defense mechanism against run-time attacks. Modern run-time attacks do not inject or modify the code of a system. Instead, they reuse the existing code by hijacking the control flow of a program in order to cause unintended, malicious program behavior Roemer et al. (2012). These attacks have been demonstrated on various platforms and devices, including embedded architectures like ARM Kornau (2009), SPARC Buchanan et al. (2008) and Atmel AVR Francillon and Castelluccia (2008b).

CFI enforces that a program's control flow does not deviate from the developer-intended flow. The integrity of the program flow is ensured by validating for each control-flow decision if the executed path lies within the program's control-flow graph Abadi

et al. (2005, 2009).

In the context of ICS, the guarantees provided by CFI are not sufficient. In particular, data-only attacks like data-oriented programming (DOP) Hu et al. (2016) pose a severe threat to PLCs. Simple modifications like changing a threshold value can have catastrophic consequences, e.g., in the attack against a steel-mill, the blast furnace cloud not be turned off due to compromised controllers, resulting in massive damage.[2]

Cyber-Physical Systems Modeling. ICS comprise a class of cyber-physical systems that can be modeled as *hybrid systems*, or systems whose continuous evolution (physical equations) evolve based on the discrete-state transitions (controller actuations) of the system Antsaklis et al. (1993).

For instance, in Figure 6.1 a simplified example of two PLCs controlling the mixing and filling of colors is shown. Four input colors are mixed and filled into cans. The input of each color is controlled by PLC1, which controls the respective valves. PLC2 controls the conveyor belt, using a scale to determine when the current can is full and the next one has to be placed under the mixer. The pseudo code and control-flow graph shows the relation between the actions of PLC1 and the readings of PLC2, i.e., the operations of PLC1 determine the physical behavior observed by PLC2. In such a hybrid system, the closing of the valve is a discrete event. However, the time required to fill a single can will increase gradually (evolve continuously).

In the context of ICS, state estimation techniques have been leveraged to model physical dynamics for particular discrete events of the system Davis et al. (2015) Etigowni et al. (2016b). However, the actuation of the associated controlling devices are typically abstracted to simplify the complexity of the model, neglecting the underlying control-flow behavior of any running programs. This simplification opens up these systems to motivated adversaries that exploit such abstractionsKang et al. (2016) to launch stealthy attacks.

[2] https://www.wired.com/2015/01/german-steel-mill-hack-destruction/

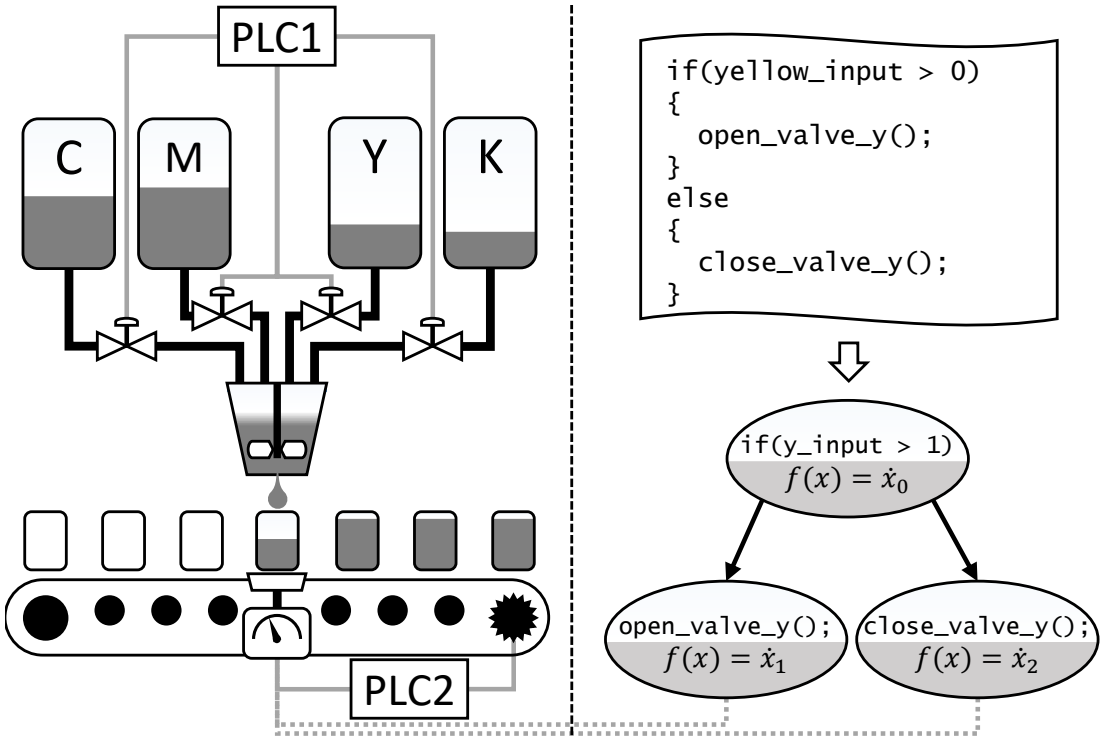


Figure 6.1: Simplified Industrial Control System, two PLCs control the mixing of four input colors–cyan (C), magenta (M), yellow (Y) and black (K for Key)–and filling into cans. PLC1's control actions on the valves of the individual color tanks will have an effect on the filling rate measured by PLC2.

6.3 Models and Assumptions

In this section we present the system model and adversary model considered in this paper.

6.3.1 System Model

In this paper, we consider large distributed industrial control systems (ICS) with a centralized monitoring system (SCADA). This is the predominant system design Stouffer et al. (2011) for large scale industrial plants. The ICS consists of networked controllers (PLCs) that jointly control a (complex) physical process, where the actions of the individual PLCs are interdependent. In particular, actuations initiated by one PLC effect the system state which will be represented in the sensor readings of other PLCs. This means that all PLCs are indirectly connected with each other through the physical

dynamics of the controlled physical system.[3]

Each PLC is connected to its own local network of sensors and actuators. These sensors and actuators are directly interfacing with the physical system, and associated discrete sampled values are accessed by the PLCs.

In addition to the indirect connection between PLCs, all components of a distributed ICS are also connected explicitly. That said, all PLCs are connected to each other and to SCADA over a computer network, e.g., Ethernet. By means of the computer network, input and output data of all PLCs are reported to the SCADA system, where data is recorded in a historian database and can be viewed by the operator.

6.3.2 Adversary Model

The adversary's goal is to cause misbehavior of the ICS while remaining undetected. The behavior of the system refers to the actions that influence the physical process controlled by the ICS. In particular, the adversary alters control commands sent to physical appliances (actuators) that can change the state of the physical process. Passive attacks that do not alter the system behavior, e.g., attacks that ex-filtrate data, are out of scope in this work.

ICS usually have built-in safety functions that will be triggered by rapid changes of the system state or control commands that set system parameters far outside of the valid range. Therefore, the adversary needs to make sure to not trigger these safety mechanisms. Similarly, ICS are usually monitored by a human operator. This precludes naïve attacks like denial-of-service (DoS) on devices or the network. The adversary has to make sure that her manipulations do not cause suspicion on the operator's side Garcia et al. (2017b), i.e., the attack needs to be stealthy.

Number of Compromised PLCs: The adversary can compromise one (or a small subset) of PLCs in a distributed ICS. The attacker in our adversary model has knowledge about the attacked system, so we have to assume a compromised PLC will report

[3]Note, the system does not need to be "fully" connected, i.e., the actuations of one PLC are not required to be observed by all other PLCs.

legitimate sensor values. Furthermore, we assume that the attacker is not able to compromise all PLCs in the ICS. We argue that this is a realistic assumption due to different reasons. For instance, in a geographically distributed ICS, the adversary might have physical access to some PLCs in a remote station of the plant. This is relevant if the attacker compromises PLCs via physical access, e.g., by updating the control logic via USB, replacing storage media like an SD-Memory card, or through debugging interfaces like JTAG Garcia et al. (2017c). Furthermore, PLCs often have physical switches that deactivate the remote update functionality, i.e., an adversary has to have physical access to a PLC before being able to modify its software remotely. Other reasons why an adversary cannot compromise all PLCs of a plant include systems which consist of heterogeneous PLCs, i.e., PLCs with different hardware or firmware versions, different models, or even from different vendors. If the adversary has knowledge about a vulnerability in one of the PLC variants, he can compromise these but not the other PLCs of the system. Also, PLCs might be isolated in different network segments, exposing only a subset to a remote attacker.

We assume that the adversary has complete control over the compromised PLC, i.e., she can compromise the firmware and the control logic of the PLC. The adversary can gain control over a PLC leveraging static or dynamic attack techniques. In a static attack, the adversary replaces the software (firmware or control logic) of a PLC, e.g., via a malicious software update. Dynamic attacks are based on injecting new code at run-time, manipulating the behavior of existing code by means of return-oriented programming (ROP) Roemer et al. (2012), or data-oriented programming (DOP) Hu et al. (2016).

The adversary cannot compromise the system components that configure and execute Scadman. In particular, we assume that Scadman itself is not compromised.

For the sake of simplicity we consider network attacks out of scope. We assume a secure, i.e., integrity protected and authenticated channel between the controllers and Scadman. We discuss network attacks and defense mechanisms for such settings without secure channels in subsection 6.6.3.

6.4 Our Design

Before we describe our SCADMAN design and framework, we discuss important challenges that we had to tackle for SCADMAN.

6.4.1 Challenges

ICS are usually centrally managed, hence, remote control of PLCs is an important feature. However, the ability to remotely update and reconfigure PLC control programs and firmware opens up a large attack surface, which has been exploited by many real-world attacks in the past Falliere et al. (2010); F-Secure Labs (2016). Although the general concepts to prevent an adversary from misusing update functionalities to install malware on a device are well-known (e.g., digital signatures), they cannot be easily integrated into existing legacy systems. As such, security solutions for ICS need to consider legacy systems that are still vulnerable. Due to the long-living nature of ICS, legacy systems remain vulnerable for a long time, possibly decades.

Another important limitation in ICS stems from closed source, proprietary software. Control software, firmware, and compilers are usually manufacturer specific and cannot be modified by the customer. Thus, modification of the software running *on* the PLC is not feasible as it would require cooperation of the manufacturer.

Modifications of the PLC software also can lead to undesirable implications that will hinder adoption in practice. Safety and reliability are paramount in ICS. Hence, all modifications that could impact them are unlikely to be adapted. In particular, in systems that require safety certification modifications of the PLC software would void them, i.e., solutions that rely on the modification of control-components cannot be used in highly-sensitive environments.

6.4.2 Scadman Design

The goal of SCADMAN is to ensure the correct behavior of a distributed industrial control system (ICS). The correct behavior can be violated by different types of attacks,

as discussed before in subsection 6.3.2. As a result, SCADMAN must provide a general mechanism that can counter all possible attacks that result in an incorrect control behavior of the system. Control behavior includes any action taken by any of the *programmable logic-controller* (PLC)s that modifies the overall system state. We consider the control behavior as correct, if it fits the behavior intended by the system operator. An attack can result in incorrect control behavior, when the attacker makes one of the components perform a different action than was intended. For example, a PLC is intended to close a valve when a certain threshold is reached. The attacker then forces the PLC to keep the valve open, contrary to the original programming of the PLC. SCADMAN ensures the *control behavior integrity* (CBI) of an ICS. A violation of CBI is a deviation from the intended behavior of any of the PLCs within the ICS.

The system can also deviate from the intended state for other reasons like faults, e.g., a faulty sensor reporting incorrect values. SCADMAN can also detect these situations, allowing the operator to repair the system.

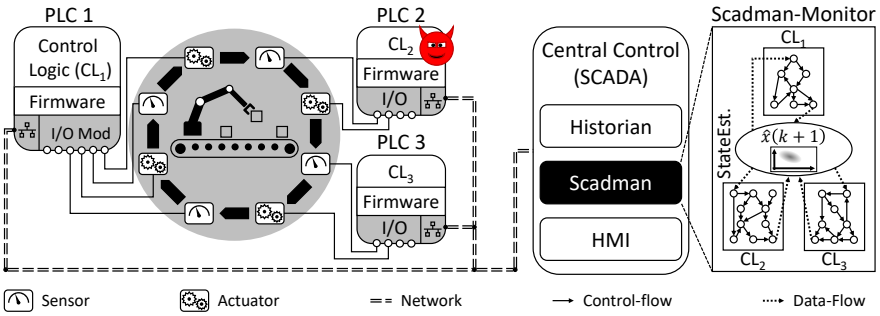


Figure 6.2: SCADMAN system overview and architecture. The central control (SCADA) is extended with a component called SCADMAN-MONITOR that monitors the behavior of the distributed ICS and can detect compromised controllers in the system.

Figure 6.2 shows the concept of SCADMAN. In a distributed ICS, multiple PLCs interact independently with a physical process. However, the actions of one PLC influence the overall state of the physical system. This is reflected in the sensor readings of other PLCs that are not under control of the adversary. We exploit this interdependency to detect the misbehavior of a compromised PLC. For example, a compromised PLC cannot stealthily open a valve because a second trustworthy PLC would measure the change in inflow. This discrepancy between expected sensor readings and the actual system state is used by SCADMAN to detect deviations in the control behavior.

Scadman-Monitor. All PLCs report their actuation commands and sensor readings to a central entity, which we call SCADMAN-MONITOR. Note that centrally reporting and logging all operations is very common in ICS Stouffer et al. (2011), e.g., for reporting to HMI components. Based on the retrieved data, the SCADMAN-MONITOR will subsequently check whether any of the PLCs have been deviating from the intended behavior. This check requires two components of the SCADMAN-MONITOR. (1) A consolidated control logic code of *all* PLCs, and (2) a model of the physical process allowing SCADMAN-MONITOR to determine the interdependencies of the PLCs' inputs and outputs.

SCADMAN generates a consolidated control logic which combines the control logic of *all* PLCs into a single large program that represents the entire control actions of the ICS. This code is executed on the SCADMAN-MONITOR to determine valid actions of the PLCs. Based on the current state of the overall system and the model of the physical system, SCADMAN dynamically derives the legitimate control-flow paths through the PLC code in a physics-aware manner. This means, SCADMAN does not accept all possible, benign control-flow paths in the control logic's control-flow graph as valid, as is the case with CFI Abadi et al. (2005, 2009), but only those that are valid at any given time in the current state of the *cyber physical system* (CPS). This approach limits the set of allowed control-flow paths and thus the adversary's actions.

The physical process model allows the SCADMAN-MONITOR to estimate the influence of control commands sent by one PLC on the expected sensor readings. As the adversary cannot influence the physical model of the system (the laws of physics cannot be altered), an inconsistency between actuation commands and sensor readings implies that either the PLC controlling the actuation or the PLC controlling the sensors must behave and/or report incorrectly. SCADMAN can tolerate imprecise and incomplete models. The model quality largely determines the detection precision. However, an imprecise model, noisy sensors, and other factors impacting the state estimation are handled by SCADMAN as described in more details in subsection 6.5.3.

6.4.3 Scadman Framework

Our SCADMAN framework leverages a compiler-based approach to automatically generate the SCADMAN-MONITOR process–allowing for the validation of the behavior of a distributed ICS. The SCADMAN-MONITOR is a program that interacts with a simulated physical system and allows SCADMAN to calculate the expected state of the overall ICS. Our framework receives two inputs to generate the SCADMAN-MONITOR, (1) the PLC control logic codes of all PLCs in the system, and (2) a model of the physical process of the ICS.

SCADMAN uses these inputs to generate a consolidated model of the entire ICS. First, the control logic codes of the individual PLCs are combined into a single control program. In the subsequent section, we define how the functional interdependencies between the individual PLCs are resolved. Second, the control program is compiled and instrumented such that all interactions with the I/O ports of a PLC are detected, intercepted, and redirected to the physical simulation interface. Third, the instrumented control program and the model of the physical system are combined into the SCADMAN-MONITOR.

For every physical world interaction that would occur on a real PLC, the SCADMAN-MONITOR performs a check comparing its internal state with the sensor values reported by the PLCs. The SCADMAN-MONITOR utilizes the consolidated control program in conjunction with the physical state estimated by the physical model to calculate the expected state of the system.

Figure 6.3 shows the abstract operation of SCADMAN-MONITOR. At run-time, our SCADMAN-MONITOR executes in parallel with the ICS and validates the behavior of the system. The operation of SCADMAN-MONITOR is based on scan cycles with three phases: (1) the current system state is read via the sensors, (2) the control program executes, and (3) the computed actuation commands are applied to the actuators.

SCADMAN-MONITOR estimates the state of the system for the next scan cycle based on the last actuation commands sent to the actuators. The state estimator determines a set of possible states–in Figure 6.3 the state variables S_1, S_2 and S_3 correspond to

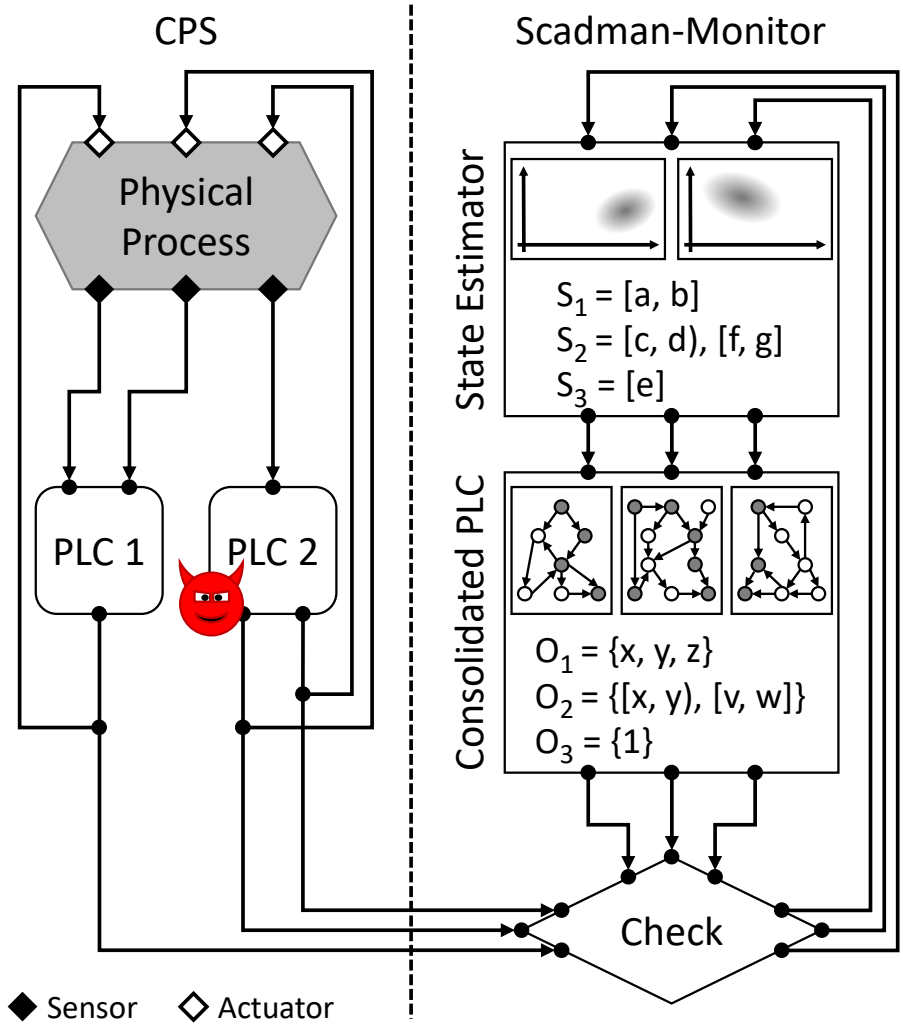


Figure 6.3: SCADMAN scan cycle. For each interaction of a PLC with the physical process, the SCADMAN-MONITOR performs the corresponding operations using the state estimated based on the system's past events. Afterwards, consistency between both sides is checked, and an alarm is raised when a deviation is detected.

one sensor of the system–are calculated. The values of S_1 and S_2 cannot be estimated precisely. Instead an interval of possible values is predicted. In parallel, the physical process evolves based on the last actuation commands.

In the real CPS the current state is read by the PLCs and their control programs are executed. SCADMAN-MONITOR executes the consolidated control program on the estimated system state. For variables where the system state cannot be precisely estimated, SCADMAN-MONITOR has to execute all possible control-flow paths for the calculated interval.

In the CPS the PLCss output their calculated actuation commands. The SCADMAN-MONITOR has generated a set of actuation commands as well. However, due to the execution of multiple possible control-flow paths, the set of resulting commands can be larger than the set produced by the real PLCs.

As long as the real PLCs executed only benign code following benign control-flow paths, the set of actuation commands they produced must be a subset of the actuation commands predicted by SCADMAN-MONITOR. If this condition is violated, SCADMAN-MONITOR has detected a behavior violation in the system and triggers and alert.

6.5 Scadman Implementation

SCADMAN introduces the SCADMAN-MONITOR process, which is responsible for receiving all sensor values and actuations, and validating them using the consolidated PLC code and state estimation. We propose a generic approach to build the SCADMAN-MONITOR. We will first discuss how SCADMAN consolidates the control programs for a distributed PLC network along with the necessary assumptions for timing and functional correctness. We then demonstrate how the consolidated code will be compiled and instrumented into an executable that can receive the state of the ICS network, e.g., the state of the sensors, as input for each scan cycle and update the estimated state of the system. We also introduce a novel approach, so-called error-margin multi-execution, that allows SCADMAN to account for falsely predicted actuations in case the model of the physical system deviates from the real system. Finally, we describe how these components of SCADMAN can be combined with the physical state estimation to detect any compromised components in the ICS.

Example. For the purpose of clarity, we will provide a simplified representation for a single process control for two PLCs from the system in Figure 6.1. Figure 6.4 shows two PLC programs, `plc1` and `plc2` that are consolidated by SCADMAN into a single PLC representation, `master`. PLC1 is responsible for controlling a valve, `YellowValve`, associated with the yellow color dispenser. The valve will open if an input amount is greater than 0. PLC2 is responsible for moving the conveyor belt if the current can is

```
PROGRAM plc1
  VAR_INPUT
    YellowAmount : REAL;
  END_VAR
  VAR_IN_OUT
    YellowValve : BOOL;
  END_VAR
  IF (YellowAmount > 0) THEN
    YellowValve := 1;
  ELSE
    YellowValve := 0;
  END_IF;
END_PROGRAM
CONFIGURATION Config0
  RESOURCE Res0 ON PLC
    TASK Main(INTERVAL := T#1s,
              PRIORITY := 0);
    PROGRAM Inst0 WITH Main : plc1;
  END_RESOURCE
END_CONFIGURATION
```

```
PROGRAM plc2
  VAR_INPUT
    CanWeight : REAL;
    YellowValve :  BOOL;
  END_VAR
  VAR_IN_OUT
    ConveyorMove : BOOL;
  END_VAR
  IF (CanWeight > 100.0
      AND NOT(YellowValve)) THEN
    ConveyorMove := 1;
  ELSE
   ConveyorMove := 0;
  END_IF;
END_PROGRAM
CONFIGURATION Config2
  RESOURCE Res0 ON PLC
    TASK Main(INTERVAL := T#1s,
              PRIORITY := 0);
    PROGRAM Inst0 WITH Main : plc2;
  END_RESOURCE
END_CONFIGURATION
```

```
(* Master PLC Code *)
PROGRAM master

  (* combined  variables *)

  VAR_INPUT
    YellowAmount : REAL;
    CanWeight : REAL;
  END_VAR

  VAR_IN_OUT
    YellowValve : BOOL;
    ConveyorMove : BOOL;
  END_VAR


  (* plc1 code*)

  IF (YellowAmount > 0) THEN
    YellowValve := 1;
  ELSE
    YellowValve := 0;
  END_IF;


  (*plc2 code*)

  IF (CanWeight > 100.0
      AND NOT(Filling)) THEN
    ConveyorMove := 1;
  ELSE
   ConveyorMove := 0;
  END_IF;
END_PROGRAM

(* master  configuration*)
CONFIGURATION MasterConfig
  RESOURCE Res0 ON PLC
    TASK Main(INTERVAL := T#1s,
              PRIORITY := 0);
    PROGRAM Inst0 WITH Main: master;
  END_RESOURCE
END_CONFIGURATION
```

Figure 6.4: Code consolidation for two PLCs with respect to a single process of the paint mixing plant in Figure 6.1. The left 2 programs, `plc1` and `plc2` are merged into a `master` PLC code.

full and if the `YellowValve` is not open. Descriptive variable names have been used in the code. This example will be used to explain each component of the implementation.

6.5.1 PLC-Code Consolidation

The premise of generating a SCADMAN-MONITOR representation is to first merge the control program code of all the ICS PLC's into a single PLC program representation. This consolidated representation is necessary for two reasons. First, in order to monitor the distributed processes, we need access to all of the system parameters in order

to successfully simulate the physical model of the overall system, i.e., we cannot simulate the physics of a process with partial sensor data. In theory, the consolidation would not be necessary for a subset of the PLCs that do not have any cyber-physical interdependencies. However, these dependencies are difficult to derive manually. As such, SCADMAN automatically generates models that incorporate these cyber-physical interdependencies as long as the distributed system conforms to the assumptions required to ensure functional and timing correctness, which are discussed at the end of this subsection.

In this paper, we consider PLC control programs that conform to the IEC 61131 standard John and Tiegelkamp (2010). According to the standard, programs are typically composed of three types of programming organisation units (POUs): programs, functions, and function blocks. A *program* is the "main program" of the PLC that includes I/O assignments, variable definitions, and access paths. A *function* is a programming block that returns a value given input and output variables in a similar vein to function definitions for other procedural programming languages such as C. A *function block* is a data structure that has the same functionality as a function but retains the associated values in memory across executions. As such, the code consolidation process will append all of the function and function block definitions and merging the main PLC program of each PLC. This allows us to retrieve the state of all sensors and actuators of the ICS, feed the values through this consolidated representation, and observe how the actuators are updated. Figure 6.4 illustrates how the main programs of two PLC programs will be merged.

However, it is common practice to define different components for a single PLC control program using different programming languages. The IEC 61131 standard enumerates five programming languages: (1) ladder diagrams (LD) – a graphical programming language to design logic circuits, (2) sequential function charts (SFC) – another graphical programming language to define sequential state operations, (3) function block diagrams (FBD) – a graphical representation of function blocks, (4) instruction lists (IL) – an assembly-like textual programming language, and (5) structured text (ST) – a textual programming language similar to Pascal. The heterogeneity of a PLC program

significantly increases the complexity of any form of static code analysis as compilation and simulation rules would have to be defined for each language. As such, SCADMAN first converts all programs to a single programming language representation. Previous works have formally proven that the structured text (ST) programming language can be used to represent the other four languages Darvas et al. and therefore serves as our base programming language for the SCADMAN-MONITOR.

We will now discuss the correctness of our consolidation process and the necessary assumptions.

Timing correctness. The correctness of consolidating the control logic of all PLC's depends on the required sampling time of the ICS. PLC tasks can be executed either continuously or periodically for some interval. For a distributed network of N PLC's that are configured to run programs at varying sample times, $T_{sample}(i)$ for a PLC$_i$, the consolidation is valid if and only if the sum of the execution times, $T_{execution}(i)$ of all controller programs is less than the smallest task interval, i.e.,

$$\sum_{i=1}^{N} T_{execution}(i) < \min_{\forall i \in N} T_{sample}(i).$$

For continuously executing PLC configurations–i.e., event-driven control–the cumulative scanning time must be less than the shortest duration time of an input or an output signal Moon (1994). The continuous scan cycle time of a PLC ranges from microseconds to tenths of a second. However, because SCADMAN is implemented on a standard computer with substantially more computing power than typical PLC's, the "scan cycle" for SCADMAN's consolidated PLC code is much faster and, hence, the only bottleneck is the sampling time over network communication. In the system shown in Figure 6.4, the programs `plc1` and `plc2` are shown to have the same execution interval timing, which is reflected in the consolidated `master` program.

With respect to clock drift between PLC's, we assume that the design of the overall ICS accounts for clock drift as such a hindrance would be a pre-existing condition.

Functional correctness. We also consider the functional correctness of combining multiple control logic programs sequentially into a single control logic program. The PLC's scan cycle can be abstracted into three components: the scanning of the inputs, the propagation of the inputs through a logic circuit, and the updating of all associated

outputs at the end of the scan cycle. The SCADMAN-MONITOR program combines the scanning of inputs and updating of outputs for all PLC programs as these actions are atomic in nature. Previous work has shown that separate processes update the values of inputs/outputs to/from memory independent of the scan cycle process Garcia et al. (2017b).

Because the process of propagating the inputs through a logic circuit to update the associated outputs is parallel in nature across PLCs, we can inductively claim the correctness of merging based on the timing correctness of our assumptions. Furthermore, the ordering of the merging process is arbitrary as any unsatisfied dependencies or race conditions amongst PLCs would be a pre-existing nuisance in the design of the system. For instance, for the system in Figure 6.4, the ordering of `plc1` and `plc2` is arbitrary in the context of the `master` program. If there was a race condition and/or ordering dependency where both programs were writing to the actuator `YellowValve`, this would be a flaw by design of the overall ICS.

6.5.2 Compilation and Instrumentation

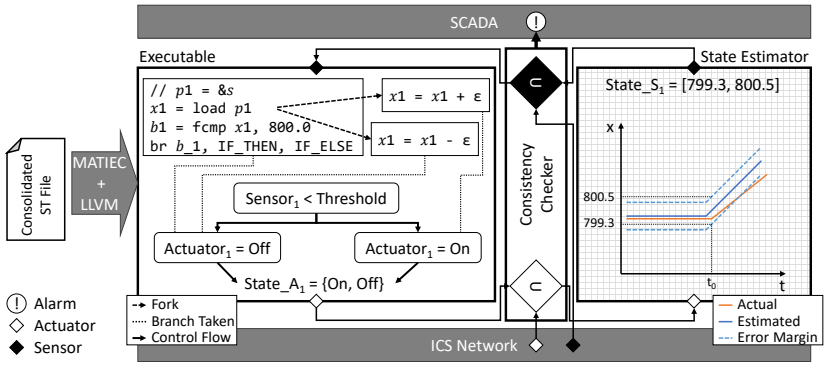


Figure 6.5: SCADMAN implementation overview. The consolidated PLC code is compiled in combination with MATIEC and LLVM to an executable. The ICS network feeds sensor values into SCADMAN-MONITOR to simulate the PLC scan cycle and check for consistency. The updated outputs are fed from the executable to the state estimation as well. Any deviations in the expected behavior is alerted to the SCADA monitor.

To compute the updated values of actuators at the end of a scan cycle, we need to execute the consolidated PLC code given the state of the sensors. We integrate the consolidated control code into the SCADMAN-MONITOR and record the actuations, performed by the control code. Figure 6.5 provides an overview of the implementation

and shows how the consolidated code is incorporated into the SCADMAN-MONITOR. We use a modified version of the MATIEC compiler[4] to compile the consolidated PLC code to C code. We modified the MATIEC compiler to automatically generate functions that allow easy access and modification of the internal state of the generated C code. This is used by the SCADMAN-MONITOR process to simulate access to sensor values and actuators.

As a second step, we compile the C code into an intermediate representation using the LLVM Lattner and Adve (2004) compiler framework. Operating on the LLVM intermediate code allows us to perform analysis and instrumentation without having to analyze structured text or C code directly. We perform instrumentation on the generated LLVM intermediate code to introduce a execution mode that draws from ideas of symbolic execution and interval arithmetic, which we call *error-margin multi-execution*. This is used to reduce the number of false positives by introducing an error to accessed sensor values. We discuss details of this approach in the following subsection.

The final step is to produce a runnable executable. We compile the instrumented PLC code to native code and link it with a support library, providing various utility functions. The C code generated by the MATIEC compiler is intended to be linked to a userspace driver that implements hardware access. Instead we link the generated code with our framework such that no hardware access is needed and is replaced with interaction with the state estimation and attack detection.

6.5.3 Error-margin Multi-execution

The model of a physical system may deviate from the real system due to various reasons. For instance, a physical processes might evolve slower or faster than expected in the model. These minor differences between the model-based estimated state and the real system state can lead to inconsistencies that SCADMAN would incorrectly report as an attack. These false positives can occur, for instance, if the PLCs perform an actuation depending on whether a sensor value is above or below a threshold, as shown in the

[4] https://github.com/thiagoralves/OpenPLC_v2/

example in Figure 6.5. In the real system, the sensor might be above the threshold, while in the simulated physical system, the sensor is still below the threshold. In this case, the SCADMAN-MONITOR performs an actuation, while the real PLC does not, or vice versa. For instance, this can happen when the weight of a can increases at a slightly higher rate than estimated by the physical system model. The reason for this can be a valve–which controls the inflow of a color into a can–may not necessarily close within one scan cycle. In our system model, the actuations are assumed to be immediate, i.e., the valve will be closed and our system model will reflect an inflow rate of 0. In reality it may only be partially closed with a nonzero inflow rate. These slight deviations will be propagated to the associated physical model.

To tackle this problem and reduce false positives, we check whether the SCADMAN-MONITOR behaves differently in terms of actuation assuming an error in the sensor readings. We introduce error-margin multi-execution to detect differences in actuation. First, we define error-margins for sensors. Second, we detect whether a PLC performs different actions, when executed with an error applied to the sensor value. A difference in the performed actions, are only observed when the PLC is taking a different control-flow through the program execution. Therefore, we need to detect whether the control-flow of the PLC depends on a sensor value (cf. code snippet in Figure 6.5).

We define an error-margin, $\pm\epsilon$, for each of the sensors. We then check whether the SCADMAN-MONITOR performs different actuations when applying $\pm\epsilon$ to the sensor reading, which we denote as s. Using interval arithmetic, one can propagate the error-margin through the executed program. However, if a branching condition depends on the sensor value, possibly two branches must be executed if the decision is inconclusive. For example, the branching condition is $(s < N)$, then the execution could take both branches if $s + \epsilon \geq N$ and $s - \epsilon < N$. Therefore, we need to execute multiple paths through the control program. Symbolic execution would allow us to use symbolic sensor values and constrain them into the error-margin and execute multiple paths at once. However, current symbolic execution engines have known limitations when it comes to solving constraints for floating point operations Liew et al. (2017). Typically sensor values are represented as floating point types. To overcome this limitation we introduce

multi-execution, that operates solely on concrete floating point values within the error-margin applied and can execute multiple branches in parallel.

We integrate this error application into the consolidated PLC code simulation at the LLVM level. Whenever a conditional branch instruction depends on a sensor value s, we introduce instrumentation that forks the execution of the PLC code. In one fork we continue without an error, so $s' = s$. In the second and third fork we continue with the upper bound of the error-margin $s' = s + \epsilon$ and the lower-bound $s' = s - \epsilon$, respectively. Using *only* the upper and lower bound of the error interval $[s - \epsilon, s + \epsilon]$ is not sufficient. To be able to evaluate equality comparisons we need to also continue one fork of the code on s (without applying any error). At the end of the scan cycle we merge all forks and continue without any error applied in the next scan cycle. We create $\mathcal{O}(3^{\#sensors})$ forks per scan cycle. While this is a significant overhead in the worst case, in practice it will be less of a problem. We can use several optimizations in practice to reduce the number of concurrent forks. For example, if two forks take the same control-flow path, we can stop executing one of the two forks. Most basic blocks have only two outgoing edges, therefore we can usually kill one of the forks directly after they have taken the branch. In fact, we do not need to use the multi-execution approach until we detect a discrepancy in the actuations. We can then re-execute the scan cycle in multi-execution mode to get a more accurate result on the actuation.

Instead of producing one value for an actuator we now get a set of values for each of the actuators. If the actuation of the real system is not in the set which is reported by the consolidated PLC code we detected a inconsistency which is beyond the errors-margins and report an attack. Incorporating state estimation errors allows Scadman to minimize false positives that would arise with slight errors in the model of the physical system.

To detect whether a branch condition depends on a sensor value, we perform backwards data-flow analysis, starting from the condition of the branch. We use the *single static assignment* (SSA) of LLVM intermediate code to perform intra-procedural data-flow analysis. Inter-procedural analysis is not implemented in our current prototype as the code generated by MATIEC does not require inter-procedural data-flow analysis for

most sensors. We search the resulting data-flow graph for load instructions that read sensor values. Because the load instruction is contained in the backwards data-flow graph, we know it will affect the branching condition and must be instrumented to incorporate the check for forking the process based on the given error-margins.

Our error-margin multi-execution introduces some imprecision into the system. An attacker might try to exploit this imprecision to evade detection, we discuss this scenario in more detail in section 6.6.

6.5.4 Attack Detection

To detect attacks, the SCADMAN-MONITOR performs two steps, where the results from the n-th scan cycle are used to predict and verify the $n+1$-th scan cycle of the system. First the SCADMAN-MONITOR compares sensor values received in scan cycle n with the sensor values estimated based on the inputs from scan cycle $n-1$. When the received senor values are verified, i.e., fall within the set of predicted values, the SCADMAN-MONITOR uses them as input to execute the consolidated PLC code. This results in a set of acceptable actuation operations for scan cycle n. SCADMAN-MONITOR compares this set against actuation operations reported by the real PLCs. If the actuations of the real PLCs are verified correctly they serve as input to the state estimator of the physical system, which will predict the sensor values for the next scan cycle $n+1$.

Incomplete data. SCADMAN can also be used on system that cannot provide complete data or which involve (sub)process for which no accurate state estimation is possible. This can be due to various reasons, e.g., if a sensor state depends on human interactions with the system the SCADMAN-MONITOR cannot predict the state of that sensor in a meaningful way. However, the state of other sensors and actuators of the system that are not directly influenced by such external influence can still be validated by SCADMAN.

6.6 Security Considerations

In this section we consider different kind of attacks in the context of ICS and how SCADMAN can detect them. According to our adversary model (cf. section 6.3) we

consider a subset of PLCs to be compromised, i.e., k out of n PLCs are compromised, where $k < n$.

Afterwards we will discuss attacks scenarios that go beyond our adversary model and show that SCADMAN is valuable in these scenarios as well.

6.6.1 k-out-of-n Compromised PLCs

As discussed before, for various reasons the attack might have compromised a subset of PLCs in an ICS. The adversary aims to act stealthy, hence, we assume the adversary lets the compromised PLCs report sensor readings and actuation commands that meet the expectations of the operator as well as SCADMAN. However, the reported values will *not* match the expected values relative to the values reported by the non-compromised PLCs.

In particular, as long as one PLC that is physically interconnected with the compromised PLCs reports correct values, a discrepancy will emerge. SCADMAN will detect this discrepancy and will raise an alarm. While SCADMAN will not be able to identify which PLCs are compromised, it can still warn the operator, who can then start an in-depth investigation on the system. In the next section we evaluated SCADMAN on a large set of ICS attacks implemented for the SWaT–a real ICS for research–and show that it can detect all of these attacks.

Slow Evolving Attacks SCADMAN is based on a closed loop approach where, for each scan cycle, the system is analyzed for anomalies. The system continues when no anomaly is detected. By continuing, the current state of the system is accepted as benign and serves as the basis for estimating the system's future state. An adversary could try to exploit this scenario by *slowly* pushing the system towards a false state. On each iteration, the adversary would manipulate the system within the error margins of SCADMAN. However, ICS are usually designed to include safety measures programmed into the PLCs that prevent the system from being steered to an unsafe state. While the attack can slowly modify the system within the safety boundaries of the system without being detected, the system cannot be pushed to an unsafe state. SCADMAN

would detect any deviations in the control flow path of the PLC, e.g., if an an adversary pushes the system outside of the safety boundaries enforced by a safety check within the control flow of the original PLC program. The best the adversary can do is to leverage the simulation error margin used by SCADMAN to get the system slightly outside of its safety boundaries. However, the safety boundaries are usually chosen such that the system remains safe even in the presence of small errors, e.g., due to sensor measurement noise.

6.6.2 All PLCs Compromised

SCADMAN provides security based on the assumption that physical interdependencies of controllers (PLCs) enable the detection of misbehavior. In the simple case that the entire system is controlled by a single PLC, an adversary would control all of the data (e.g., sensor readings) available to SCADMAN if the PLC is compromised. Hence, an intelligent adversary can provide a consistent view of the system Garcia et al. (2017b) towards SCADMAN and remain undetected.

For distributed ICS, the adversary needs to control *all* PLCs to provide a coherent view of *all* actuation commands and *all* sensor readings reported to SCADMAN-MONITOR. This means the adversary has to simulate the expected behavior of the *entire* system and synchronizes the actions of all PLCs. While this might be feasible for very simple and static ICS, the attacker's limited resources (PLCs have limited computation power and memory) significantly aggravate the complexity of stealthy attacks for dynamic ICS.

6.6.3 Network Attacks

In this work we assume a secure channel between SCADMAN-MONITOR and the PLC, i.e., an adversary cannot launch network attacks by impersonating other devices. However, some legacy systems do not provide secure network channels, in which cases the adversary might try to overcome SCADMAN by manipulating network packages.

The adversary can either try to manipulate or suppress network packages of other PLCs, i.e., PLCs not controlled by the adversary that would reveal the adversary's

behavior manipulations. This is not possible in commonly used switched networks, i.e., network packages of on an un-compromised PLC will never be routed to a compromised PLC but directly to SCADMAN-MONITOR.

The second option for the adversary is to impersonate another PLC, i.e., by sending packages to SCADMAN-MONITOR pretending to originate from an uncompromised PLC, e.g., by modifying the source IP address of a package. However, as discussed before, the adversary cannot suppress packages sent by the benign PLC, hence, SCADMAN-MONITOR will receive both types of packages: those with benign sensor reading and those with manipulated values reported by the adversary. This mixture of input values will lead to inconsistencies, which will trigger a security alarm of SCADMAN.

6.7 Evaluations

In this section, we provide an overview of our experimental evaluation of SCADMAN. We first introduce the water treatment testbed SWaT[5] that we used for our evaluation. We then discuss how SCADMAN implements code consolidation for the proprietary PLCs used by the testbed. Afterwards we describe the choice of physical state estimation equations used in our attack detection for this testbed. Finally, we evaluate SCADMAN against a set of attacks for the testbed that were enumerated by previous works.

6.7.1 Evaluation Environment and Dataset

The study reported here was conducted on data from a real distributed industrial control system, the Secure Water Treatment plant (SWaT). SWaT is the quasi-standard for evaluating ICS security solutions used by many researchers in the past Junejo and Yau (2016); Lin et al. (2018); Chen et al. (2018, 2016); Goh et al. (2017); Kong et al. (2016); Wang et al. (2017b,a); Inoue et al. (2017); Umer et al. (2017); Pal et al. (2017). SWaT

[5] https://itrust.sutd.edu.sg/research/testbeds/secure-water-treatment-swat/

is used by industry for testing and evaluating.[6] [7][8]

SWaT is a 6-stage water treatment plant, where the sub-process of each stage is controlled by an individual PLC. In total, SWaT contains 68 sensors and actuators; some actuators serve as standbys and are intended to be used only when the primary actuator fails. A more detailed description of the SWaT plant is provided in section .1.

Dataset. We evaluated SCADMAN using data generated by the SWaT plant iTrust. The dataset includes both normal operations to evaluate the false positive rate of SCADMAN as well as attacks to evaluate the detection performance of SCADMAN. The attack data were generated independently of our work, modeled after Adepu et al. Adepu and Mathur (2016c,d) and was used in previous works to evaluate ICS security solutions Goh et al. (2017).

The dataset contains data collected during seven days of continuous operation of SWaT. It contains 496, 800 data points, each point representing the system state using 53 features of the system, e.g., sensor values and actuator states.

A more detailed description of the dataset as well as some sample data is provided in section .2.

6.7.2 Scadman State Estimation

We now describe how we evaluated each component of SCADMAN's implementation for the SWaT use case in order to generate the cyber-physical state estimator.

SWaT PLC code consolidation. The SWaT testbed consists of six Allen Bradley PLCs using proprietary code and development tools. Each PLC was programmed individually using the proprietary Rockwell Automation Studio 5000 development environment. In order to perform the code consolidation for each PLC, we first needed

[6] https://www.sgcybersecurity.com/securityarticle/securityarticle/deployment-of-kaspersky-s-industrial-cybersecurity-kics-solution-leveraging-on-itrust-s-test-bed

[7] http://www.upgrademag.com/web/2018/01/22/kaspersky-lab-deploys-industrial-cybersecurity-solutions-leverages-on-itrust-test-bed/

[8] https://labs.mwrinfosecurity.com/blog/offensive-ics-exploitation-a-technical-description/

to translate the heterogeneously programmed controller projects to a single IEC 61131-3 standard structured text format. To do so, we extracted the L5X project files for each PLC Automation (2016 howpublished = `http://literature.rockwellautomation.com/idc/groups/literature/documents/rm/1756-rm084_-en-p.pdf`). The L5X format is an XML format used for importing/exporting projects to and from the Studio 5000 environment. We then built a translation tool, L5X2IEC, using an existing `l5x` Python library[9] that provides accessors for the XML elements within the L5X files. Although the Allen Bradley PLC programming languages conform to the IEC standards, we needed to provide translations for the proprietary extensions of each language.

For a detailed description of the transformation process please refer to section .3.

Physical state estimation SCADMAN provides physical state estimation only for sensors whose sensor and actuation dependencies are satisfied. For instance, we cannot predict the value of a water tank if we do not have access to the corresponding flow rate sensor. Similarly, we only modeled processes whose dynamics were based on physical characteristics. We did not model any of the chemical processes of the plant. However, we were able to model a subset of the associated valves and pumps for these processes. As such, we provide generic physical state estimators for the water tank level sensors, the flow rate level sensors, as well as the status indicators for pumps and valves. However, models for other components of the system can be added in future work.

For water tanks, we used the same estimation and threshold values provided in prior analyses of the SWaT testbed Adepu and Mathur (2016a). Based on previous analyses of state estimation for the SWaT testbed Adepu and Mathur (2016a,b), we use the following closed-loop state estimation models:

$$TankLevel = TankLevel + (Inflow - Outflow) * F_c.$$

Where `Inflow` and `Outflow` are the inflow/outflow rates of the tank and F_c is a conversion constant for the flow rate.

[9] `https://pypi.python.org/pypi/l5x/1.2`

For flow rates, we derived a closed-loop model that incorporates any actuators that may open/close the flow of water:

$$FlowRate = FlowRate * \prod_{n=1}^{N} Actuator_n$$

Where $Actuator_n$ represents any pump or value whose value is 0 (for *off*) or 1 (for *on*). We use invariants from the prior study of the SWaT testbed Adepu and Mathur (2016b). These invariants capture the state of the system at any point of time. Each model is then invoked automatically when a particular variable needs to be estimated. In addition to providing generic models for these subsystems, the models for the binary values of the actuator states are automatically generated by our SCADMAN-MONITOR executable.

6.7.3 Attack Detection

We were able to successfully detect all attacks enumerated in the attack data set. We further evaluated SCADMAN against the record-and-replay attacks enumerated in a previous case study, where sensor values were recorded and replayed back to the HMI to spoof sensor values as was done in the Stuxnet malware Adepu and Mathur (2016a). For the non-optimized evaluation, SCADMAN had 0 false negatives with a very low false positive rate of 0.36% for the nominal water tank level deviation threshold in the implementation *without* multi-execution. The false positives were due to the cases mentioned in section 6.5, where an actuator may open/close a tick too early or too late based on our estimated sensor values.

False positive pruning. All false positives were pruned by our error-margin multi-execution implementation for the nominal water tank level deviation threshold. We show the associated ROC curves of varying water tank level deviation thresholds for both the normal execution and the error-margin multi-execution in Figure 6.6. False positives only exist for very small threshold values, i.e., a threshold value that is less than 1mm for the water tank level will obviously result in some false positive rate. The nominal threshold values were based on the threshold values used for state estimation in a previous work Adepu and Mathur (2016a). The nominal threshold value of 5mm

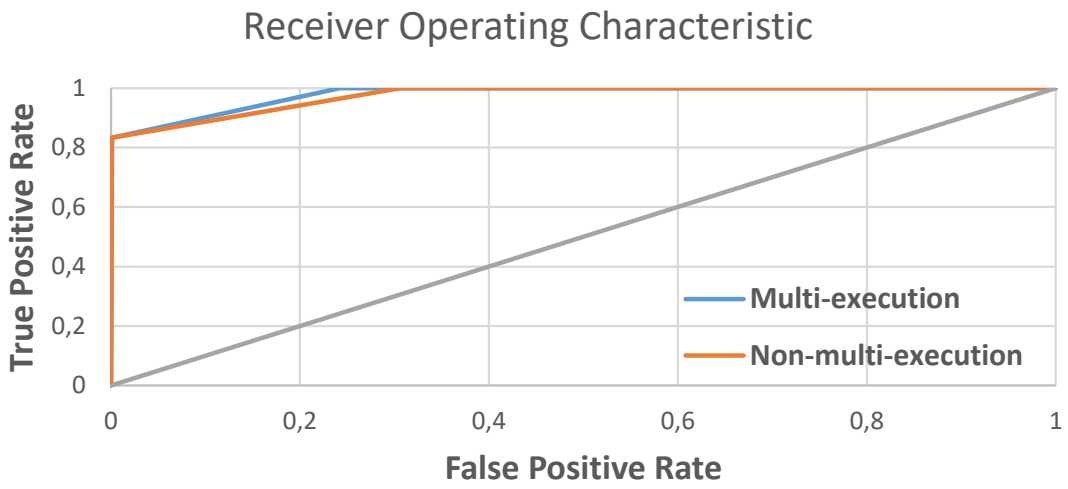


Figure 6.6: ROC curve for attack detection across varying water tank deviation thresholds for both single and multi-execution analysis.

obtained from our ROC curve confirmed the choice in the previous work.

Performance. We performed our evaluation on a system equipped with an Intel Core i7-4710MQ Processor at $2.50\,GHz$, $16\,GB$ of RAM running Linux v4.4.0-112-generic. Running SCADMAN on the entire dataset of seven days took 30 hours for the single threaded deviation-checking, and 51 hours for the multi-threaded error-margin multi-execution using our current prototype implementation that is not optimized for performance. This shows that SCADMAN can "keep up" when running in parallel with the real system even on a desktop-grade computer.

Memory. The memory usage of SCADMAN was $36\,MB$ on average with a peak memory requirement of $149\,MB$, with multi-execution turned off. This shows that SCADMAN can be used to constantly monitor the system behavior using standard server equipment.

Communication. By default, all PLCs communicate with the SCADA system to display the operational process data and to store the operational data in a historian. SCADMAN can retrieve its data from the historian causing no communication overhead in the PLC network.

6.8 Related Work

The previous works on ICS security can be categorized into cyber-physical security mechanisms that are implemented within the ICS controller to enforce code integrity

and monitoring solutions that abstract the control of PLCs to verify the overall cyber-physical system.

Internal CPS Control Security. ECFI Abbasi et al. (2017) provides a control-flow integrity (CFI) solution for PLCs, where the code running on the PLC is instrumented to validate whether indirect branches follow a legitimate path in the control-flow graph (CFG). SCADMAN does not need any modifications of the code running on the PLC. In contrast, it monitors the overall behavior of the PLC reflecting its entire software (including the OS). Furthermore, SCADMAN provides context sensitive control-flow checking, i.e., the set of allowed CFG paths is further restricted based on the current system state.

Control-Flow Attestation (C-FLAT) enables a prover device to attest the exact control-flow path of an executed program to a remote verifier Abera et al. (2016). However, it cannot be applied to existing systems that do not have the necessary hardware security extensions such as the ARM TrustZone. PyCRA Shoukry et al. (2015) uses a physical challenge-response authentication to protect active sensing systems against cyber physical attacks. PyCRA's focus on active sensors, hence it is not applicable to passive sensors nor to actuators, both of which are also common in ICS. Orpheus Cheng et al. (2017) monitors the behavior of a program based on executed system calls and checks whether a system call is legitimate in the given context. The decision is made based on a finite-state machine (FSM) representing the programs system call behavior, i.e., system calls are only allowed to be executed in sequences for which valid transitions exist within the FSM. Orpheus requires a FSM of the monitored system, which needs to be constructed in a learning phase. SCADMAN does not require such a model of the overall system but only models for the individual subprocesses of the physical system. Also, Orpheus performs detection on the device and relies on an un-compromised OS and that physical event reports are untampered. SCADMAN does not require any modifications to the monitored devices and does not require a trusted channel to input sensors.

Zeus Han et al. (2017) monitors the control flow of a PLC control program by monitoring the electromagnetic emissions side channels of the PLC by a neural network

model. Such a defense does not protect against data attacks and can further be circumvented via firmware modification attacks. Furthermore, Zeus cannot account for verifying other the other networked components as in the SCADMAN framework.

State estimation has been used within the PLCs to detect if any of the invariant properties of the system have been violated Adepu and Mathur (2016a,b); Adepu et al. (2016). This enforcement resides in the application layer of a single PLC, which can be circumvented if the PLC is compromised. Furthermore, the physical invariants and their dependencies are specified manually. SCADMAN automatically enforces the checking of discrete-state transitions by analyzing the consolidated PLC code. Similarly, on-device runtime verification has been proposed for PLCs with coupled hypervisors Garcia et al. (2016). The hypervisor resides above the firmware and relies on the integrity of the PLC control logic.

TSV McLaughlin et al. (2014) verifies the integrity of any program being loaded onto a PLC by lifting the associated binary to an intermediate language to symbolically execute the program and verify that it is not violating any of the provided infrastructural safety requirements. The safety requirements are enforced within the PLC by extension of the guarantees provided by TSV. Similarly, PLCVerif Darvas et al. (2015a) provides a framework for checking safety properties of PLC code against finite state automata. These solutions are offline analyses that do not provide any runtime guarantees and only verify the control logic application code.

External CPS Control Security. Previous works have proposed means of detecting stealthy attacks in the context of ICS. David et. al. Urbina et al. (2016) reported on limiting the impact of stealthy attacks on industrial control systems. Liu et .al. Liu et al. (2009, 2011c) presented false data injection attacks against state estimation in electric power grids. This work is implemented mainly in a simulation environment, where they are considering stealthy attacks on smart meters. In SCADMAN, we also consider stealthy attacks on multiple sensors and actuators on real-time operational data.

Yuqi et. al. Chen et al. (2018, 2016) proposed an approach for learning physical invariants that combines machine learning with ideas from mutation testing. Initial

models are learned using support vector machines. These learned models are used for code attestation and identifying standard network attacks. Configuration based intrusion detection system have also been proposed for Advanced Metering Infrastructure Ali and Al-Shaer (2013). The AMI behavior is modeled using event logs collected at smart meters. Event logs are modeled using Markov chains and linear temporal logic for the verification of specifications. However, such models depend on the completeness of the training data set used for the learned models. A water control system was modeled using an autoregressive model in order to monitor physics of the system Hadžiosmanović et al. (2014). For distributed systems with complex cyber-physical interdependencies, it is infeasible to assume all discrete states of the system will be traversed. SCADMAN automatically contains a discrete-state model of the entire ICS and depends only on the accuracy of the physical state estimation. In a similar vein, the idea of detecting attacks by monitoring physics Paul et al. (2014); Choudhari et al. (2013) of the ICS by using invariants has been applied. However, in these instances the invariants were derived manually based on domain knowledge. SCADMAN automatically derives these cyber-physical invariants and significantly reduces the probability of human error during the modelling phase of complex systems.

6.9 Future Work

In this section we propose possible extensions of SCADMAN to improve its security and functionalities.

Simulation interval. The current implementation of SCADMAN uses a closed loop approach where the system state s after each scan cycle is serving as the basis for the next round. However, SCADMAN can be extended to use state s only after n scan cycles. This means that the state estimation and multi-execution performed by SCADMAN must cover n scan cycles, which could lead to larger errors in the state estimation, which in turn could impact the error-margin multi-execution of SCADMAN negatively. However, this approach can make slowly evolving attacks (see section 6.6) even more complicated, further increasing the security of SCADMAN.

Automated invariant generation. SCADMAN cannot only serve as a security solution but can also help improve the functional correctness and safety of an ICS. Our modeling framework can be used to determine interdependencies of system variables. This information is useful when programming an ICS as it helps to identify conditions and safety checks that need to be included in the PLCs for the ICS to operate correctly.

6.10 Conclusions and Summary

Industrial control systems (ICS) are ubiquitous and increasingly deployed in critical infrastructures. In fact, recent large-scale cyber attacks (e.g., Stuxnet, BlackEnergy, Duqu to name a few) exploit vulnerabilities in these systems. Building a generic defense mechanism against the various ICS attack flavors is highly challenging. However, we observe that all these attacks influence the physics of these devices. As a result, we developed SCADMAN, a system that preserves the *Control Behavior Integrity* (CBI) of distributed cyber-physical systems. SCADMAN provides real time monitoring for intrusion detection and sensor fault detection by maintaining a cyber-physical state estimation of the system based on a novel control code consolidation generation as well as state estimation equations of the physical processes. SCADMAN enforces the correctness of individual controllers in the system by verifying the actuation values being sent from the PLCs as well as the associated changes that propagated through the physical dynamics of the system. We evaluated SCADMAN against an enumerated set of attacks on a real water treatment testbed. Our results show that we can detect a wide range of attacks in a timely fashion with zero false positives for nominal threshold values.

Chapter 7

Conclusion

The promising preliminary results of this dissertation will motivate more scalable and robust solutions for the compositional modeling and verification of embedded cyber-physical systems. This dissertation narrowed the gap between practical and theoretical approaches in complex CPS.

The cyber-physical rootkit, Harvey, presented a practical security vulnerability assessment of PLCs that leverages the physics of the system. This work emphasized how a PLC could be attacked if an attacker has access to the physical model of a system as well as the ability to compromise a PLC's firmware. As such, this dissertation then presented how the physical side-channel properties of an ICS can be leveraged to provide additional layers of security and verification of the overall process in the context of additive manufacturing. Although this work provided practical measures for verifying 3D-printed models, the solution was domain-specific and covers a particular subset of ICS. The dissertation therefore simulatenously presents more theoretical approaches to understand the feasibility and limitations of formal verification methods in the context of embedded systems security.

The dissertation evaluated the feasibility of extracting safety properties for PLCs in the context of a complex CPS. In particular, the presented work focused on a simplified model of a complex CPS–a single-machine to infinite bus (SMIB) electric power grid system. The model was specified in a sound and relatively-complete dynamic logic, d$\mathcal{L}$, that is used for modelling hybrid systems. First, the work showed how these models require a coarse-grained, high-level abstraction of the discrete control provided by the PLCs. Second, this was the first work that attempted to specify a power grid system in d$\mathcal{L}$. Providing safety guarantees for even the simplest of power systems proved to

be extremely complex, and this work will motivate future research directions in the automation of the physical invariant extraction process for complex and hybrid CPS.

Although this work provided a future direction for modeling complex and hybrid CPS, the high-level abstraction of the associated controller treats the PLC as a black-box for actuation. As such, this dissertation presented the HyPLC tool that provides a systematic mapping of PLC code to actuation that can be modeled by a hybrid program specified in d$\mathcal{L}$. This bi-directional translation allows a programmer to not only verify the safety of existing PLC code, but to also generate PLC code from a verified hybrid program based on the IEC 61131-3 standard. This work was a first step in providing compositional verification of ICS in the context of complex ICS and will also motivate future research that soundly integrates formal methods into CPS security analyses.

Finally, this dissertation presented Scadman to provide a more practical approach to the security of distributed ICS. Scadman uses physical state estimation to determine if a sensor or actuator value in a distributed ICS is behaving abnormally. It uses the control flow of the software to change the physical state estimation model being used, i.e., a change in an actuator value in the PLC code will correspond to a change in the physics of the system. Scadman proved to be an effective intrusion-dection system for distributed ICS, and will motivate future research that emphasizes the physical evolution of a CPS with respect to the control flow of the software.

Bibliography

Abb launches new pluto programmable logic controller for rail safety applications. URL http://www.abb.com/cawp/seitp202/fa405fb9803dd9eac1258035002f53c0.aspx.

IEEE standard test access port and boundary scan architecture. *IEEE Std. 1149.1-2001*, 2001.

Federal energy regulatory commission (ferc): Optimal power flow and formulation papers. http://www.ferc.gov/industries/electric/indus-act/market-planning/opf-papers.asp, 2010.

Arconic strengthens 3d printing collaboration with airbus. *http://advancedmanufacturing.org/arconic-airbus-3d-printing-collaboration/*, Dec 2016. URL http://advancedmanufacturing.org/arconic-airbus-3d-printing-collaboration/.

Hardware meets software in advanced manufacturing. *https://www.ge.com/stories/hardware-meets-software-advanced-manufacturing*, 2017. URL https://www.ge.com/stories/hardware-meets-software-advanced-manufacturing.

Knee replacement implant materials. *https://bonesmart.org/knee/knee-replacement-implant-materials/*, 2017. URL https://bonesmart.org/knee/knee-replacement-implant-materials/.

Natural machines: The makers of foodini - a 3d food printer making all types of fresh, nutritious foods. *http://www.naturalmachines.com/*, 2017. URL http://www.naturalmachines.com/.

M. Abadi, M. Budiu, U. Erlingsson, and J. Ligatti. Control-flow Integrity. In *Conference on Computer and Communications Security*, CCS, 2005.

M. Abadi, M. Budiu, U. Erlingsson, and J. Ligatti. Control-flow Integrity Principles, Implementations, and Applications. *ACM Trans. Inf. Syst. Secur.*, 13(1), Nov. 2009.

A. Abbasi and M. Hashemi. Ghost in the plc: Designing an undetectable programmable logic controller rootkit via pin control attack. 2016.

A. Abbasi, T. Holz, E. Zambon, and S. Etalle. ECFI: Asynchronous Control Flow Integrity for Programmable Logic Controllers. In *Proceedings of the 33rd Annual Computer Security Applications Conference*, ACSAC, 2017.

T. Abera, N. Asokan, L. Davi, J.-E. Ekberg, T. Nyman, A. Paverd, A.-R. Sadeghi, and G. Tsudik. C-FLAT: Control-Flow Attestation for Embedded Systems Software. In *Conference on Computer and Communications Security*, CCS, 2016.

S. Adepu and A. Mathur. Using process invariants to detect cyber attacks on a water treatment system. In *IFIP International Information Security and Privacy Conference*, pages 91–104. Springer, 2016a.

S. Adepu and A. Mathur. Distributed detection of single-stage multipoint cyber attacks in a water treatment plant. In *Proceedings of the 11th ACM Asia Conference on Computer and Communications Security*, pages 449–460, New York, NY, May 2016b. ACM.

S. Adepu and A. Mathur. An investigation into the response of a water treatment system to cyber attacks. In *Proceedings of the 17th IEEE High Assurance Systems Engineering Symposium, Orlando*, pages 141–148, January 2016c.

S. Adepu and A. Mathur. Generalized attacker and attack models for Cyber-Physical Systems. In *Proceedings of the 40th Annual International Computers, Software & Applications Conference, Atlanta, USA*, pages 283–292, Washington, D.C., USA, June 2016d. IEEE.

S. Adepu, S. Shrivastava, and A. Mathur. Argus: An orthogonal defense framework to protect public infrastructure against cyber-physical attacks. *IEEE Internet Computing*, 20(5):38–45, 2016.

S. Adepu, G. Mishra, and A. Mathur. Access control in water distribution networks: A case study. In *2017 IEEE International Conference on Software Quality, Reliability and Security (QRS)*, pages 184–191, 2017.

C. M. Ahmed, S. Adepu, and A. Mathur. Limitations of state estimation based cyber attack detection schemes in industrial control systems. In *Smart City Security and Privacy Workshop (SCSP-W), 2016*, pages 1–5. IEEE, 2016.

S. Akin and A. Kovscek. Computed tomography in petroleum engineering research. *Geological Society, London, Special Publications*, 215(1):23–38, 2003.

M. Q. Ali and E. Al-Shaer. Configuration-based ids for advanced metering infrastructure. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, CCS '13, pages 451–462, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2477-9. doi: 10.1145/2508859.2516745. URL http://doi.acm.org/10.1145/2508859.2516745.

S. Amin, X. Litrico, S. S. Sastry, and A. M. Bayen. Stealthy deception attacks on water scada systems. In *Proceedings of the 13th ACM International Conference on Hybrid Systems: Computation and Control*, HSCC, 2010.

S. Amini, H. Mohsenian-Rad, and F. Pasqualetti. Dynamic load altering attacks in smart grid. In *Innovative Smart Grid Technologies Conference (ISGT), 2015 IEEE Power & Energy Society*. IEEE, 2015.

G. Andersson. Modelling and analysis of electric power systems. *EEH-Power Systems Laboratory, Swiss Federal Institute of Technology (ETH), Zürich, Switzerland*, 2004.

P. J. Antsaklis, J. A. Stiver, and M. Lemmon. Hybrid system modeling and autonomous control systems. In *Hybrid systems*, pages 366–392. Springer, 1993.

W. Arbaugh, D. Farber, and J. Smith. A secure and reliable bootstrap architecture. In *IEEE Symposium on Security and Privacy*, 1997.

R. Automation. Logix5000 controllers generals instructions reference manual, 2016 howpublished = http://literature.rockwellautomation.com/idc/groups/literature/documents/rm/1756-rm003_-en-p.pdf .

R. Automation. Logix5000 controllers import/exports, 2016 howpublished = http://literature.rockwellautomation.com/idc/groups/literature/documents/rm/1756-rm084_-en-p.pdf .

R. Automation. *Logix5000 Controllers, Tasks, Programs, and Routines*. July 2018.

Avery Li-Chun Wang. An industrial strength audio search algorithm.

L. Aylmore. Use of computer-assisted tomography in studying water movement around plant roots. *Advances in Agronomy*, 49:1–54, 1993.

M. Backes, M. Drmuth, S. Gerling, M. Pinkal, and C. Sporleder. Acoustic side-channel attacks on printers. In *Proceedings of the 19th USENIX Conference on Security*, USENIX Security'10, pages 20–20. USENIX Association. URL http://dl.acm.org/citation.cfm?id=1929820.1929847.

Z. Basnight, J. Butts, J. Lopez, and T. Dube. Firmware modification attacks on programmable logic controllers. *International Journal of Critical Infrastructure Protection*, 2013.

D. Beresford. Exploiting Siemens Simatic S7 PLCs. In *Black Hat USA 2011*, Black Hat USA '11.

B. Berman. 3-d printing: The new industrial revolution. 55(2):155–162. ISSN 0007-6813. doi: 10.1016/j.bushor.2011.11.003. URL https://www.sciencedirect.com/science/article/pii/S0007681311001790.

W. Bolton. *Programmable logic controllers*. Newnes, 2015.

S. Bose, S. H. Low, T. Teeraratkul, and B. Hassibi. Equivalent relaxations of optimal power flow. *Automatic Control, IEEE Transactions on*, 2015.

F. Brasser, B. El Mahjoub, A.-R. Sadeghi, C. Wachsmann, and P. Koeberl. TyTAN: Tiny Trust Anchor for Tiny Devices. In *Proceedings of the 52nd Annual Design Automation Conference*, DAC, 2015.

M. Brüggemann and R. Spenneberg. Plc-blaster der virus im industrienetz. https://events.ccc.de/congress/2015/Fahrplan/events/7229.html, 2015.

E. Buchanan, R. Roemer, H. Shacham, and S. Savage. When Good Instructions Go Bad: Generalizing Return-oriented Programming to RISC. In *ACM Conference on Computer and Communications Security*, CCS, 2008.

A. Campion and P. Kambhampati. Surface-enhanced raman scattering. *Chemical Society Reviews*, 27(4):241–250, 1998.

B. Chen, K. L. Butler-Purry, S. Nuthalapati, and D. Kundur. Network delay caused by cyber attacks on svc and its impact on transient stability of smart grids. In *2014 IEEE PES General Meeting— Conference & Exposition*, pages 1–5. IEEE, 2014.

Y. Chen, C. M. Poskitt, and J. Sun. Towards learning and verifying invariants of cyber-physical systems by code mutation. In *Proc. International Symposium on Formal Methods (FM 2016)*, volume 9995 of *LNCS*, pages 155–163. Springer, 2016.

Y. Chen, C. M. Poskitt, and J. Sun. Learning from mutants: Using code mutation to learn and monitor invariants of a cyber-physical system. In *Proc. IEEE Symposium on Security and Privacy (S&P 2018)*. IEEE Computer Society, 2018. To appear.

L. Cheng, K. Tian, and D. D. Yao. Orpheus: Enforcing cyber-physical execution semantics to defend against data-oriented attacks. 2017.

S. Cheung, B. Dutertre, M. Fong, U. Lindqvist, K. Skinner, and A. Valdes. Using model-based intrusion detection for scada networks. In *Proceedings of the SCADA Security Scientific Symposium*, Miami Beach, Florida, jan 2007.

S. R. Chhetri, A. Canedo, and M. A. Al Faruque. Kcad: Kinetic cyber attack detection method for cyber-physical additive manufacturing systems. In *Proceedings of the 35th International Conference on Computer-Aided Design*, page 74. ACM, 2016.

E. Chien, L. OMurchu, and N. Falliere. W32.Duqu - The precursor to the next Stuxnet. Technical report, Symantic Security Response, 2011.

K. B. Chien, E. Makridakis, and R. N. Shah. Three-dimensional printing of soy protein scaffolds for tissue regeneration. *Tissue Engineering Part C: Methods*, 19(6):417–426, 2012.

A. Choudhari, H. Ramaprasad, T. Paul, J. W. Kimball, M. Zawodniok, B. McMillin, and S. Chellappan. Stability of a cyber-physical smart grid system using cooperating invariants. In *2013 IEEE 37th Annual Computer Software and Applications Conference*, pages 760–769, 2013.

R. Christie. Power systems test case archive. *Electrical Engineering dept., University of Washington*, 2000.

E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 8(2):244–263, 1986.

V. Cnudde and M. N. Boone. High-resolution x-ray computed tomography in geosciences: A review of the current technology and applications. *Earth-Science Reviews*, 123:1–17, 2013.

D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk. Internet x.509 public key infrastructure certificate and certificate revocation list (crl) profile. https://tools.ietf.org/html/rfc5280, 2008.

A. J. Crosby and J.-Y. Lee. Polymer nanocomposites: the nano effect on mechanical properties. *Polymer reviews*, 47(2):217–229, 2007.

D. Darvas, I. Majzik, and E. B. Viñuela. Generic representation of plc programming languages for formal verification. In *Proc. of the 23rd PhD Mini-Symposium*, pages 6–9.

D. Darvas, E. Blanco Vinuela, and B. Fernández Adiego. Plcverif: A tool to verify plc programs based on model checking techniques. 2015a.

D. Darvas, E. Blanco Vinuela, and I. Majzik. A formal specification method for plc-based applications. 2015b.

D. Darvas, I. Majzik, and E. B. Viñuela. Conformance checking for programmable logic controller programs and specifications. In *Industrial Embedded Systems (SIES), 2016 11th IEEE Symposium on*, pages 1–8. IEEE, 2016.

D. Darvas, I. Majzik, and E. B. Viñuela. Plc program translation for verification purposes. *Periodica Polytechnica. Electrical Engineering and Computer Science*, 61 (2):151, 2017.

A. Davies, . Feb. 28, and . 6. A swedish automaker is using 3d printing to make the world's fastest car. URL http://www.businessinsider.com/koenigsegg-one1-comes-with-3d-printed-parts-2014-2.

K. R. Davis, C. M. Davis, S. A. Zonouz, R. B. Bobba, R. Berthier, L. Garcia, and P. W. Sauer. A cyber-physical modeling and assessment framework for power grid infrastructures. *IEEE Transactions on Smart Grid*, 6(5):2464–2475, 2015.

K. E. Defrawy, A. Francillon, D. Perito, and G. Tsudik. SMART: Secure and Minimal Architecture for (Establishing Dynamic) Root of Trust. In *NDSS*, 2012.

H. Dommel and W. Tinney. Optimal power flow solutions. *IEEE Transactions on Power Apparatus and Systems*, 1968.

K. T. Erickson. Programmable logic controllers. Institute of Electrical and Electronics Engineers, 1996.

S. Etigowni, M. Cintuglu, M. Kazerooni, S. Hossain, P. Sun, K. Davis, O. Mohammed, and S. Zonouz. Cyber-Air-Gapped Detection of Controller Attacks through Physical Interdependencies.

S. Etigowni, D. Tian, G. Hernandez, K. Butler, and S. Zonouz. Cpac: Mitigating attacks against critical infrastructure with cyber-physical access control. In *Annual Computer Security Applications Conference (ACSAC)*, 2016a.

S. Etigowni, D. J. Tian, G. Hernandez, S. Zonouz, and K. Butler. CPAC: securing critical infrastructure with cyber-physical access control. In *Proceedings of the 32nd Annual Conference on Computer Security Applications*, pages 139–152. ACM, 2016b.

European Network and Information Security Agency (ENISA). Protecting industrial control systems–recommendations for Europe and Member States. https://www.enisa.europa.eu/, 2011.

F-Secure Labs. BLACKENERGY and QUEDAGH: The convergence of crimeware and APT attacks, 2016.

N. Falliere, L. O. Murchu, and E. Chien. W32.Stuxnet Dossier. Technical report, Symantic Security Response, 2010.

J. FitzPatrick and M. King. Nsa playset: Jtag implants, 2015.

M. Fleischmann, P. J. Hendra, and A. J. McQuillan. Raman spectra of pyridine adsorbed at a silver electrode. *Chemical Physics Letters*, 26(2):163–166, 1974.

H. Flordal, M. Fabian, K. Åkesson, and D. Spensieri. Automatic model generation and plc-code implementation for interlocking policies in industrial robot cells. *Control Engineering Practice*, 15(11):1416–1426, 2007.

D. Formby, P. Srinivasan, A. Leonard, J. Rogers, and R. Beyah. Who's in control of your control system? device fingerprinting for cyber-physical systems. In *Proceedings of the Network and Distributed System Security (NDSS) Symposium*, 2016.

A. Francillon and C. Castelluccia. Code injection attacks on harvard-architecture devices. In *Proceedings of the 15th ACM Conference on Computer and Communications Security*, CCS '08, 2008a.

A. Francillon and C. Castelluccia. Code Injection Attacks on Harvard-architecture Devices. In *ACM Conference on Computer and Communications Security*, CCS, 2008b.

S.-Y. Fu, X.-Q. Feng, B. Lauke, and Y.-W. Mai. Effects of particle size, particle/matrix interface adhesion and particle loading on mechanical properties of particulate–polymer composites. *Composites Part B: Engineering*, 39(6):933–961, 2008.

N. Fulton, S. Mitsch, J.-D. Quesel, M. Völp, and A. Platzer. Keymaera x: an axiomatic tactical theorem prover for hybrid systems. In *International Conference on Automated Deduction*, pages 527–538. Springer, 2015.

L. Garcia, H. Senyondo, S. McLaughlin, and S. Zonouz. Covert channel communication through physical interdependencies in cyber-physical infrastructures. In *Smart Grid Communications (SmartGridComm), 2014 IEEE International Conference on*, pages 952–957. IEEE, 2014.

L. Garcia, S. Zonouz, D. Wei, and L. P. de Aguiar. Detecting plc control corruption via on-device runtime verification. In *Resilience Week (RWS), 2016*, pages 67–72. IEEE, 2016.

L. Garcia, F. Brasser, M. H. Cintuglu, A.-R. Sadeghi, O. Mohammed, and S. A. Zonouz. Hey, my malware knows physics! attacking plcs with physical model aware rootkit. In *24th Annual Network & Distributed System Security Symposium (NDSS)*, Feb. 2017a.

L. Garcia, F. Brasser, M. H. Cintuglu, A.-R. Sadeghi, O. Mohammed, and S. A. Zonouz. Hey, my malware knows physics! attacking plcs with physical model aware rootkit. In *24th Annual Network & Distributed System Security Symposium (NDSS)*, 2017b.

L. A. Garcia, F. Brasser, M. Cintuglu, A.-R. Sadeghi, O. Mohammed, and S. A. Zonouz. Hey, my malware knows physics! attacking plcs with physical model aware rootkit.

In *Proceedings of the Network and Distributed System Security (NDSS) Symposium*, NDSS, 2017c.

R. Gerth, D. Peled, M. Y. Vardi, and P. Wolper. Simple on-the-fly automatic verification of linear temporal logic. In *Protocol Specification, Testing and Verification XV*, pages 3–18. Springer, 1995.

K. Ghorbal and A. Platzer. Characterizing algebraic invariants by differential radical invariants. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 279–294. Springer, 2014.

J. Goh, S. Adepu, K. N. Junejo, and A. Mathur. A Dataset to Support Research in the Design of Secure Water Treatment Systems. In *The 11th International Conference on Critical Information Infrastructures Security (CRITIS)*, pages 1–13, New York, USA, October 2016. Springer.

J. Goh, S. Adepu, M. Tan, and Z. S. Lee. Anomaly Detection in Cyber Physical Systems Using Recurrent Neural Networks. In *2017 IEEE 18th International Symposium on High Assurance Systems Engineering (HASE)*, pages 140–145, Washington D.C., USA, Jan 2017. IEEE.

A. Gómez-Expósito, A. J. Conejo, and C. Cañizares. *Electric energy systems: analysis and operation*. CRC Press, 2016.

M. Guri, Y. Solewicz, A. Daidakulov, and Y. Elovici. Fansmitter: Acoustic data exfiltration from (speakerless) air-gapped computers. URL http://arxiv.org/abs/1606.05915.

D. Hadžiosmanović, R. Sommer, E. Zambon, and P. H. Hartel. Through the eye of the PLC: Semantic security monitoring for industrial processes. In *Proceedings of the 30th Annual Computer Security Applications Conference*, pages 126–135. ACM, 2014.

J. Hainfeld, D. Slatkin, T. Focella, and H. Smilowitz. Gold nanoparticles: a new x-ray contrast agent. *The British journal of radiology*, 2014.

J. K. Hale and J. P. LaSalle. Differential equations: linearity vs. nonlinearity. *SIAM Review*, 5(3):249–272, 1963.

Y. Han, S. Etigowni, H. Liu, S. Zonouz, and A. Petropulu. Watch Me, but Don't Touch Me! Contactless Control Flow Monitoring via Electromagnetic Emanations. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1095–1108. ACM, 2017.

D. Harel, J. Tiuryn, and D. Kozen. *Dynamic Logic*. MIT Press, Cambridge, MA, USA, 2000. ISBN 0262082896.

A. Hay, D. Cid, and R. Bray. *OSSEC Host-Based Intrusion Detection Guide*. Syngress, 2008.

G. Hernandez, O. Arias, D. Buentello, and Y. Jin. Smart nest thermostat: A smart spy in your home. In *BlackHat USA*, 2014.

J. Hicks. FDA approved 3d printed drug available in the US. URL http://www.forbes.com/sites/jenniferhicks/2016/03/22/fda-approved-3d-printed-drug-available-in-the-us/.

A. Hojjati, A. Adhikari, K. Struckmann, E. Chou, T. N. Tho Nguyen, K. Madan, M. S. Winslett, C. A. Gunter, and W. P. King. Leave your phone at the door: Side channels that reveal factory floor secrets. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 883–894. ACM. ISBN 978-1-4503-4139-4. doi: 10.1145/2976749.2978323. URL http://doi.acm.org/10.1145/2976749.2978323.

H. Hu, S. Shinde, S. Adrian, Z. L. Chua, P. Saxena, and Z. Liang. Data-Oriented Programming: On the Expressiveness of Non-control Data Attacks. In *IEEE Symposium on Security and Privacy*, S&P, 2016.

X. Huang, I. H. El-Sayed, W. Qian, and M. A. El-Sayed. Cancer cells assemble and align gold nanorods conjugated to antibodies to produce highly enhanced, sharp, and polarized surface raman spectra: a potential cancer diagnostic marker. *Nano letters*, 7(6):1591–1597, 2007.

J. Inoue, Y. Yamagata, Y. Chen, C. M. Poskitt, and J. Sun. Anomaly detection for a water treatment system using unsupervised machine learning. In *2017 IEEE International Conference on Data Mining Workshops, ICDM Workshops 2017, New Orleans, LA, USA, November 18-21*, pages 1058–1065, 2017.

iTrust. Dataset and models. https://itrust.sutd.edu.sg/dataset/.

G. D. Janaki Ram, Y. Yang, and B. E. Stucker. Effect of process parameters on bond formation during ultrasonic consolidation of aluminum alloy 3003. 25(3):221–238. ISSN 0278-6125. doi: 10.1016/S0278-6125(07)80011-2. URL http://www.sciencedirect.com/science/article/pii/S0278612507800112.

F. Jeff. SpaceX unveils its 21st century spaceship. URL http://www.newspacejournal.com/2014/05/30/spacex-unveils-its-21st-century-spaceship/.

K.-H. John and M. Tiegelkamp. *IEC 61131-3: programming industrial automation systems: concepts and programming languages, requirements for programming systems, decision-making aids*. Springer Science & Business Media, 2010.

K. N. Junejo and D. K. Yau. Data driven physical modelling for intrusion detection in cyber physical systems. In *Proceedings of the Singapore Cyber-Security Conference (SG-CRC)*, pages 43–57, 2016.

A. C. Kak and M. Slaney. *Principles of computerized tomographic imaging*. SIAM, 2001.

E. Kang, S. Adepu, D. Jackson, and A. P. Mathur. Model-based security analysis of a water treatment system. In *Proceedings of the 2Nd International Workshop on Software Engineering for Smart Cyber-Physical Systems*, SEsCPS '16, 2016.

E. H. Kerner. Universal formats for nonlinear ordinary differential systems. *Journal of Mathematical Physics*, 22(7):1366–1371, 1981.

B. Kesler. The vulnerability of nuclear facilities to cyber attack; strategic insights: Spring 2010. *Strategic Insights, Spring 2011*, 2011.

D. Kilgus, J. Moreland, G. Finerman, T. Funahashi, and J. Tipton. Catastrophic wear of tibial polyethylene inserts. (273):223–231. ISSN 0009-921X.

A. Kleinman and A. Wool. Accurate modeling of the siemens s7 scada protocol for intrusion detection and digital forensics. *The Journal of Digital Forensics, Security and Law: JDFSL*, 2014.

J. Klick, S. Lau, D. Marzin, J.-O. Malchow, and V. Roth. Internet-facing plcs - a new back orifice. In *Black Hat USA*, 2015.

K. Kneipp, Y. Wang, H. Kneipp, L. T. Perelman, I. Itzkan, R. R. Dasari, and M. S. Feld. Single molecule detection using surface-enhanced raman scattering (sers). *Physical review letters*, 78(9):1667, 1997.

P. Kong, Y. Li, X. Chen, J. Sun, M. Sun, and J. Wang. Towards concolic testing for hybrid systems. In *FM 2016: Formal Methods*, pages 460–478, 2016.

T. Kornau. Return Oriented Programming for the ARM Architecture. Technical report, 2009.

Y. Kouskoulas, D. Renshaw, A. Platzer, and P. Kazanzides. Certifying the safe design of a virtual fixture control algorithm for a surgical robot. In *Proceedings of the 16th international conference on Hybrid systems: computation and control*, pages 263–272. ACM, 2013.

C. Lattner and V. Adve. LLVM: A Compilation Framework for Lifelong Program Analysis & Transformation. In *Proceedings of the 2004 International Symposium on Code Generation and Optimization (CGO'04)*, Palo Alto, California, Mar 2004.

T. Le, G. Salles-Loustau, L. Najafizadeh, M. Javanmard, and S. Zonouz. Secure point-of-care medical diagnostics via trusted sensing and cyto-coded passwords. In *Dependable Systems and Networks (DSN), 2016 46th Annual IEEE/IFIP International Conference on*, pages 583–594. IEEE, 2016.

E. C. Le Ru, M. Meyer, and P. G. Etchegoin. Proof of single-molecule sensitivity in surface enhanced raman scattering (sers) by means of a two-analyte technique. *The journal of physical chemistry B*, 110(4):1944–1948, 2006.

R. L. Lemaster, L. Lu, and S. Jackson. The use of process monitoring techniques on a CNC wood router. part 2. use of a vibration accelerometer to monitor tool wear and workpiece quality. 50(9):59–64. ISSN 00157473. URL http://search.proquest.com/docview/214622388/abstract/AF151E1F83B2490BPQ/1.

D. Liew, D. Schemmel, C. Cadar, A. F. Donaldson, R. Zähl, and K. Wehrle. Floating-point Symbolic Execution: A Case Study in N-version Programming. In *Proceedings of the 32Nd IEEE/ACM International Conference on Automated Software Engineering*, ASE 2017, pages 601–612, Piscataway, NJ, USA, 2017. IEEE Press.

Q. Lin, S. Adepu, S. Verwer, and A. Mathur. Tabor: A graphical model-based approach for anomaly detection in industrial control systems. 2018.

D. Lin-Vien, N. B. Colthup, W. G. Fateley, and J. G. Grasselli. *The handbook of infrared and Raman characteristic frequencies of organic molecules*. Elsevier, 1991.

H. Liu and T. J. Webster. Mechanical properties of dispersed ceramic nanoparticles in polymer composites for orthopedic applications. *Int J Nanomedicine*, 5:299–313, 2010.

J. Liu, N. Zhan, and H. Zhao. Computing semi-algebraic invariants for polynomial dynamical systems. In *EMSOFT*, pages 97–106. ACM, 2011a.

Y. Liu, P. Ning, and M. Reiter. False data injection attacks against state estimation in electric power grids. In *Proceedings of the 16th ACM Conference on Computer and Communications Security*, pages 21–32, 2009.

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

Y. Liu, P. Ning, and M. K. Reiter. False data injection attacks against state estimation in electric power grids. *ACM Transactions on Information and System Security (TISSEC)*, 14(1):13, 2011c.

S. M. Loos, A. Platzer, and L. Nistor. Adaptive cruise control: Hybrid, distributed, and now formally verified. In *International Symposium on Formal Methods*, pages 42–56. Springer, 2011.

A. Mader and H. Wupper. Timed automaton models for simple programmable logic controllers. In *Real-Time Systems, 1999. Proceedings of the 11th Euromicro Conference on*, pages 106–113. IEEE, 1999.

S. Manesis, D. Sapidis, and R. King. Intelligent control of wastewater treatment plants. *Artificial Intelligence in Engineering*, 12(3):275–281, 1998.

A. P. Mathur and N. O. Tippenhauer. Swat: A water treatment testbed for research and training on ics security. In *Cyber-physical Systems for Smart Water Networks (CySWater), 2016 International Workshop on*, pages 31–36. IEEE, 2016.

M. F. McGranaghan, D. R. Mueller, and M. J. Samotyj. Voltage sags in industrial systems. *IEEE Transactions on industry applications*, 29(2):397–403, 1993.

S. McLaughlin and S. Zonouz. Controller-aware false data injection against programmable logic controllers. In *IEEE SmartGridComm*, 2014.

S. E. McLaughlin, S. A. Zonouz, D. J. Pohly, and P. D. McDaniel. A trusted safety verifier for process controller code. In *20th Annual Network & Distributed System Security Symposium (NDSS)*, volume 14, 2014.

T. Mertke and G. Frey. Formal verification of plc programs generated from signal interpreted petri nets. In *Systems, Man, and Cybernetics, 2001 IEEE International Conference on*, volume 4, pages 2700–2705. IEEE, 2001.

A. M. Michaels, M. Nirmal, and L. Brus. Surface enhanced raman spectroscopy of individual rhodamine 6g molecules on large ag nanocrystals. *Journal of the American Chemical Society*, 121(43):9932–9939, 1999.

S. Mitsch and A. Platzer. Modelplex: Verified runtime validation of verified cyber-physical system models. *Formal Methods in System Design*, 49(1-2):33–74, 2016.

I. Moon. Modeling programmable logic controllers for logic verification. *IEEE Control Systems*, 14(2):53–59, 1994.

J. Mulder, M. Schwartz, M. Berg, J. R. Van Houten, J. Mario, M. A. K. Urrea, A. A. Clements, and J. Jacob. Weaselboard: Zero-day exploit detection for programmable logic controllers. Technical report, tech. report SAND2013-8274, Sandia Natl Laboratories, 2013.

S. Nie and S. R. Emory. Probing single molecules and single nanoparticles by surface-enhanced raman scattering. *science*, 275(5303):1102–1106, 1997.

B. Nikoobakht and M. A. El-Sayed. Surface-enhanced raman scattering studies on aggregated gold nanorods. *The Journal of Physical Chemistry A*, 107(18):3372–3378, 2003.

N. I. of Standards and Technology. FIPS 180-4, Secure Hash Standard, Federal Information Processing Standard (FIPS). Technical report, 2012.

OPC Foundation. Open platform communication foundation. https://opcfoundation.org/, 2015.

C. J. Orendorff, L. Gearheart, N. R. Jana, and C. J. Murphy. Aspect ratio dependence on surface enhanced raman scattering using silver and gold nanorod substrates. *Physical Chemistry Chemical Physics*, 8(1):165–170, 2006.

K. Pal, S. Adepu, and J. Goh. Effectiveness of association rules mining for invariants generation in cyber-physical systems. In *18th IEEE International Symposium on High Assurance Systems Engineering, HASE 2017, Singapore, January 12-14, 2017*, pages 124–127, 2017.

A. Papachristodoulou and S. Prajna. Analysis of non-polynomial systems using the sum of squares decomposition. *Positive polynomials in control*, pages 580–580, 2005.

J. Parker Jr, D. Feldman, and M. Ashkin. Raman scattering by silicon and germanium. *Physical Review*, 155(3):712, 1967.

F. Pasqualetti, F. Dörfler, and F. Bullo. Cyber-physical attacks in power networks: Models, fundamental limitations and monitor design. In *IEEE CDC and ECC*. IEEE, 2011.

F. Pasqualetti, F. Dörfler, and F. Bullo. Cyber-physical security via geometric control: Distributed monitoring and malicious attacks. In *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, pages 3418–3425. IEEE, 2012.

T. Paul, J. W. Kimball, M. Zawodniok, T. P. Roth, B. McMillin, and S. Chellappan. Unified invariants for cyber-physical switched system stability. *IEEE Transactions on Smart Grid*, 5(1):112–120, 2014.

M. Pavella, D. Ernst, and D. Ruiz-Vega. *Transient stability of power systems: a unified approach to assessment and control*. Springer Science & Business Media, 2012.

O. Pavlovic, R. Pinger, and M. Kollmann. Automated formal verification of plc programs written in il. In *Conference on Automated Deduction (CADE)*, pages 152–163, 2007.

A. Platzer. Differential dynamic logic for hybrid systems. *Journal of Automated Reasoning*, 41(2):143–189, 2008.

A. Platzer. *Logical analysis of hybrid systems: proving theorems for complex dynamics*. Springer Science & Business Media, 2010.

A. Platzer. A uniform substitution calculus for differential dynamic logic. In *International Conference on Automated Deduction*, pages 467–481. Springer, 2015.

A. Platzer and E. M. Clarke. Formal verification of curved flight collision avoidance maneuvers: A case study. In *International Symposium on Formal Methods*, pages 547–562. Springer, 2009.

A. Platzer and J.-D. Quesel. Logical verification and systematic parametric analysis in train control. In *International Workshop on Hybrid Systems: Computation and Control*, pages 646–649. Springer, 2008.

J. E. Powers. Elimination of special functions from differential equations. *Communications of the ACM*, 2(3):3–4, 1959.

D. Qi and A. J. Berger. Quantitative concentration measurements of creatinine dissolved in water and urine using raman spectroscopy and a liquid core optical fiber. *Journal of biomedical optics*, 10(3):031115–0311159, 2005.

X. Qian, X.-H. Peng, D. O. Ansari, Q. Yin-Goen, G. Z. Chen, D. M. Shin, L. Yang, A. N. Young, M. D. Wang, and S. Nie. In vivo tumor targeting and spectroscopic detection with surface-enhanced raman nanoparticle tags. *Nature biotechnology*, 26 (1):83–90, 2008.

D. Quarta, M. Pogliani, M. Polino, F. Maggi, A. M. Zanchettin, and S. Zanero. An Experimental Security Analysis of an Industrial Robot Controller. In *IEEE Symposium on Security and Privacy*, S&P, 2017.

J.-D. Quesel, S. Mitsch, S. Loos, N. Aréchiga, and A. Platzer. How to model and prove hybrid systems with keymaera: a tutorial on safety. *International Journal on Software Tools for Technology Transfer*, 18(1):67–91, 2016.

M. Rausch and B. H. Krogh. Formal verification of plc programs. In *American Control Conference, 1998. Proceedings of the 1998*, volume 1, pages 234–238. IEEE, 1998.

J. Reeves. *Autoscopy Jr.: Intrusion Detection for Embedded Control Systems Dartmouth Computer Science Technical Report TR2011-704 A Thesis*. PhD thesis, DARTMOUTH COLLEGE Hanover, New Hampshire, 2011.

H. Richter, Z. Wang, and L. Ley. The one phonon raman spectrum in microcrystalline silicon. *Solid State Communications*, 39(5):625–629, 1981.

Rick Smith. 8 Hot 3D Printing Trends To Watch In 2016. http://www.forbes.com/sites/ricksmith/2016/01/12/8-hot-3d-printing-trends-to-watch-in-2016/, 2016.

R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 1983.

Rockwell Automation. Rockwell automation download center. http://compatibility.rockwellautomation.com/Pages/MultiProductDownload.aspx?famID=4&Keyword=Controller&crumb=112, 2015.

R. Roemer, E. Buchanan, H. Shacham, and S. Savage. Return-Oriented Programming: Systems, Languages, and Applications. *ACM Trans. Info. & System Security*, 15(1), Mar. 2012.

J. Rrushi, H. Farhangi, C. Howey, K. Carmichael, and J. Dabell. A quantitative evaluation of the target selection of havex ics malware plugin.

K. Sacha. Automatic code generation for plc controllers. In *International Conference on Computer Safety, Reliability, and Security*, pages 303–316. Springer, 2005.

M. A. Savageau and E. O. Voit. Recasting nonlinear differential equations as s-systems: a canonical nonlinear form. *Mathematical biosciences*, 87(1):83–115, 1987.

C. Schmidler. Knee joint anatomy, function and problems. *http://www.healthpages.org/anatomy-function/knee-joint-structure-function-problems/*, Dec 2016.

C. D. Schuett. Programmable logic controller modification attacks for use in detection analysis. Technical report, DTIC Document, 2014.

Y. Shoukry, P. Martin, Y. Yona, S. N. Diggavi, and M. B. Srivastava. Pycra: Physical challenge-response authentication for active sensors under spoofing attacks. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-6, 2015*, pages 1004–1015. ACM, 2015.

A. Sogokon, K. Ghorbal, P. B. Jackson, and A. Platzer. A method for invariant generation for polynomial continuous systems. In *International Conference on Verification, Model Checking, and Abstract Interpretation*, pages 268–288. Springer, 2016.

C. Song, F. Lin, Z. Ba, K. Ren, C. Zhou, and W. Xu. My smartphone knows what you print: Exploring smartphone-based side-channel attacks against 3d printers. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 895–907. ACM. ISBN 978-1-4503-4139-4. doi: 10.1145/2976749.2978300. URL http://doi.acm.org/10.1145/2976749.2978300.

M. d. Sousa. Matiec-iec 61131-3 compiler, 2014. *URL https://bitbucket. org/mjsousa/-matiec*.

O. Stava, J. Vanek, B. Benes, N. Carr, and R. Měch. Stress relief: improving structural strength of 3d printable objects. *ACM Transactions on Graphics (TOG)*, 31(4):48, 2012.

P. L. Stiles, J. A. Dieringer, N. C. Shah, and R. P. Van Duyne. Surface-enhanced raman spectroscopy. *Annu. Rev. Anal. Chem.*, 1:601–626, 2008.

K. Stouffer, J. Falco, and K. Scarfone. Guide to industrial control systems (ics) security. *NIST special publication*, 800(82):16–16, 2011.

C. J. Strachan, T. Rades, K. C. Gordon, and J. Rantanen. Raman spectroscopy for quantitative analysis of pharmaceutical solids. *Journal of pharmacy and pharmacology*, 59(2):179–192, 2007.

R. Strackx, F. Piessens, and B. Preneel. Efficient isolation of trusted subsystems in embedded systems. In *Security and Privacy in Communication Networks*, volume 50 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*. Springer, 2010.

L. Sturm, C. Williams, J. Camelio, J. White, and R. Parker. Cyber-physical vunerabilities in additive manufacturing systems. *Context*, 7(2014):8, 2014.

Symantic. Windows Rootkit Overview. Technical report, Symantic Security Response, 2005.

J. Tapken and H. Dierks. Moby/plcgraphical development of plc-automata. In *International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems*, pages 311–314. Springer, 1998.

TechNavio. Global Industrial Control Systems (ICS) Security Market 2014-2018. http://www.technavio.com/report/global-industrial-control-systems-ics-security-market%C2%A02014-2018, 2014.

P. A. Temple and C. Hathaway. Multiphonon raman spectrum of silicon. *Physical Review B*, 7(8):3685, 1973.

Texas Instruments . Stellaris LM4F120H5QR ROM User's Guide. www.ti.com/lit/ug/spmu245a/spmu245a, 2011-2013.

Texas Instruments. Stellaris LM3S2793 Microcontroller Data Sheet. www.ti.com/lit/gpn/lm3s2793, 2007-2014.

D. Thapa, S. Dangol, and G.-N. Wang. Transformation from petri nets model to programmable logic controller using one-to-one mapping technique. In *Computational Intelligence for Modelling, Control and Automation, 2005 and International Conference on Intelligent Agents, Web Technologies and Internet Commerce, International Conference on*, volume 2, pages 228–233. IEEE, 2005.

D. A. Tziouvaras and D. Hou. Out-of-step protection fundamentals and advancements. In *Protective Relay Engineers, 2004 57th Annual Conference for*, pages 282–307. IEEE, 2004.

M. A. Umer, A. Mathur, K. N. Junejo, and S. Adepu. Integrating design and data centric approaches to generate invariants for distributed attack detection. In *Proceedings of the 2017 Workshop on Cyber-Physical Systems Security and PrivaCy, Dallas, TX, USA, November 3*, pages 131–136, 2017.

D. I. Urbina, J. A. Giraldo, A. A. Cardenas, N. O. Tippenhauer, J. Valente, M. Faisal, J. Ruths, R. Candell, and H. Sandberg. Limiting the impact of stealthy attacks on industrial control systems. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 1092–1105. ACM, 2016. ISBN 978-1-4503-4139-4.

P.-J. T. K. Vandekerckhove, M. G. Teeter, D. D. R. Naudie, J. L. Howard, S. J. MacDonald, and B. A. Lanting. the impact of coronal plane alignment on polyethylene wear and damage in total knee replacement: a retrieval study. ISSN 0883-5403. doi: 10.1016/j.arth.2016.12.048. URL http://www.sciencedirect.com/science/article/pii/S0883540316309342.

G. Veneziani, E. Corrêa, M. Potiens, and L. Campos. Attenuation coefficient determination of printed abs and pla samples in diagnostic radiology standard beams. In *Journal of Physics: Conference Series*, volume 733, page 012088. IOP Publishing, 2016.

J. Wang, X. Chen, J. Sun, and S. Qin. Improving probability estimation through active probabilistic model learning. In *Formal Methods and Software Engineering*, pages 379–395, 2017a.

J. Wang, J. Sun, Q. Yuan, and J. Pang. Should we learn probabilistic models for model checking? a new approach and an empirical study. In *International Conference on Fundamental Approaches to Software Engineering*, pages 3–21. Springer, 2017b.

R. Wilhelm. Determining bounds on execution times. *Embedded Systems Handbook*, 2, 2005.

D. Will. Dejavu; available at https://github.com/worldveil/dejavu, 2017.

T. Wohlers. *Wohlers Report 2015: 3D printing and additive manufacturing state of the industry; annual worldwide progress report*. Wohlers Associates, 2015.

C. L. Wu, M. Q. Zhang, M. Z. Rong, and K. Friedrich. Tensile performance improvement of low nanoparticles filled-polypropylene composites. *Composites Science and Technology*, 62(10):1327–1340, 2002.

L. Xie, Y. Mo, and B. Sinopoli. False data injection attacks in electricity markets. In *Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on*. IEEE, 2010.

M. Yampolskiy, A. Skjellum, M. Kretzschmar, R. A. Overfelt, K. R. Sloan, and A. Yasinsac. Using 3d printers as weapons. 14:58–71. ISSN 1874-5482. doi: 10.1016/j.ijcip.2015.12.004. URL http://www.sciencedirect.com/science/article/pii/S1874548215300330.

D. Yang, A. Usynin, and J. W. Hines. Anomaly-based intrusion detection for scada systems, 2006.

B. Zhu and S. Sastry. SCADA-specific intrusion detection/prevention systems: A survey and taxonomy. In *Proceedings of the First Workshop on Secure Control Systems*, 2010.

Q. Zhu, R. G. Quivey, and A. J. Berger. Raman spectroscopic measurement of relative concentrations in mixtures of oral bacteria. *Applied spectroscopy*, 61(11):1233–1237, 2007.

S. Zonouz, K. M. Rogers, R. Berthier, R. B. Bobba, W. H. Sanders, and T. J. Overbye. Scpse: Security-oriented cyber-physical state estimation for power grid critical infrastructures. *IEEE Transactions on Smart Grid*, 2012.

S. Zonouz, C. M. Davis, K. R. Davis, R. Berthier, R. B. Bobba, and W. H. Sanders. SOCCA: A security-oriented cyber-physical contingency analysis in power infrastructures. *Smart Grid, IEEE Transactions on*, 2014a.

S. Zonouz, J. Rrushi, and S. McLaughlin. Detecting industrial control malware using automated plc code analytics. *Security & Privacy, IEEE*, 2014b.

APPENDIX

.1 Raman Spectroscopy Measurements

Figure 1 shows the Raman spectroscopy measurements of 3D printed disks of Raman scattering enhancers gold nanorods (GNRs), and Diethylthiatricarbocyanine iodide (DTTCI) embedded in acrylonitrile butadiene styrene (ABS) filament.

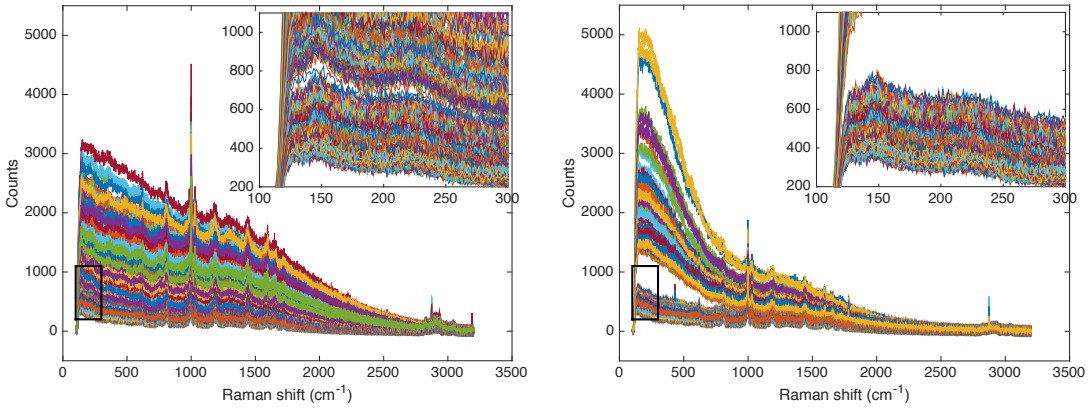


Figure 1: (a) Raman spectra GNRs embedded ABS filament. The GNRs amplifies Raman scattering of ABS. Inset figure shows the separation between the blank ABS and GNRs embedded ABS Raman spectra. (b) Raman spectra of ABS and DTTCI embedded ABS filaments. Large separation is due to the large quantity of enhancer embedded in ABS filament.

.2 Detailed Results of Acoustic Classification on Tibial Knee Prosthetic

Tibial Knee Prosthetic Classificatin, Trained with Rectilinear Fill, 60% Density									
	60% Rectilinear Fill		20% Honeycomb Fill			60% Rectilinear Fill		20% Honeycomb Fill	
Index Value	Classification Result	Confidence	Classification Result	Confidence	Index Value	Classification Result	Confidence	Classification Result	Confidence
0	Taz6Tibia_Rectilinear_60_T(0)	132	Taz6Tibia_Rectilinear_60_T(0)	137	43	Taz6Tibia_Rectilinear_60_T(43)	57	Taz6Tibia_Rectilinear_60_T(53)	7
1	Taz6Tibia_Rectilinear_60_T(1)	80	Taz6Tibia_Rectilinear_60_T(1)	19	44	Taz6Tibia_Rectilinear_60_T(44)	70	Taz6Tibia_Rectilinear_60_T(5)	7
2	Taz6Tibia_Rectilinear_60_T(2)	117	Taz6Tibia_Rectilinear_60_T(3)	10	45	Taz6Tibia_Rectilinear_60_T(45)	31	Taz6Tibia_Rectilinear_60_T(55)	8
3	Taz6Tibia_Rectilinear_60_T(3)	108	Taz6Tibia_Rectilinear_60_T(3)	19	46	Taz6Tibia_Rectilinear_60_T(46)	53	Taz6Tibia_Rectilinear_60_T(58)	12
4	Taz6Tibia_Rectilinear_60_T(4)	133	Taz6Tibia_Rectilinear_60_T(4)	18	47	Taz6Tibia_Rectilinear_60_T(47)	28	Taz6Tibia_Rectilinear_60_T(36)	9
5	Taz6Tibia_Rectilinear_60_T(5)	178	Taz6Tibia_Rectilinear_60_T(5)	45	48	Taz6Tibia_Rectilinear_60_T(48)	29	Taz6Tibia_Rectilinear_60_T(17)	10
6	Taz6Tibia_Rectilinear_60_T(6)	61	Taz6Tibia_Rectilinear_60_T(33)	13	49	Taz6Tibia_Rectilinear_60_T(49)	23	Taz6Tibia_Rectilinear_60_T(61)	15
7	Taz6Tibia_Rectilinear_60_T(7)	107	Taz6Tibia_Rectilinear_60_T(12)	9	50	Taz6Tibia_Rectilinear_60_T(50)	41	Taz6Tibia_Rectilinear_60_T(62)	27
8	Taz6Tibia_Rectilinear_60_T(8)	114	Taz6Tibia_Rectilinear_60_T(10)	28	51	Taz6Tibia_Rectilinear_60_T(51)	67	Taz6Tibia_Rectilinear_60_T(63)	15
9	Taz6Tibia_Rectilinear_60_T(9)	189	Taz6Tibia_Rectilinear_60_T(13)	14	52	Taz6Tibia_Rectilinear_60_T(52)	31	Taz6Tibia_Rectilinear_60_T(63)	14
10	Taz6Tibia_Rectilinear_60_T(10)	136	Taz6Tibia_Rectilinear_60_T(13)	45	53	Taz6Tibia_Rectilinear_60_T(53)	23	Taz6Tibia_Rectilinear_60_T(64)	16
11	Taz6Tibia_Rectilinear_60_T(11)	189	Taz6Tibia_Rectilinear_60_T(19)	10	54	Taz6Tibia_Rectilinear_60_T(54)	25	Taz6Tibia_Rectilinear_60_T(66)	33
12	Taz6Tibia_Rectilinear_60_T(12)	194	Taz6Tibia_Rectilinear_60_T(19)	11	55	Taz6Tibia_Rectilinear_60_T(55)	49	Taz6Tibia_Rectilinear_60_T(0)	10
13	Taz6Tibia_Rectilinear_60_T(13)	178	Taz6Tibia_Rectilinear_60_T(16)	72	56	Taz6Tibia_Rectilinear_60_T(56)	31	Taz6Tibia_Rectilinear_60_T(68)	7
14	Taz6Tibia_Rectilinear_60_T(14)	128	Taz6Tibia_Rectilinear_60_T(16)	15	57	Taz6Tibia_Rectilinear_60_T(57)	35	Taz6Tibia_Rectilinear_60_T(68)	17
15	Taz6Tibia_Rectilinear_60_T(15)	204	Taz6Tibia_Rectilinear_60_T(18)	47	58	Taz6Tibia_Rectilinear_60_T(58)	43	Taz6Tibia_Rectilinear_60_T(10)	15
16	Taz6Tibia_Rectilinear_60_T(16)	203	Taz6Tibia_Rectilinear_60_T(15)	14	59	Taz6Tibia_Rectilinear_60_T(59)	49	Taz6Tibia_Rectilinear_60_T(71)	10
17	Taz6Tibia_Rectilinear_60_T(17)	120	Taz6Tibia_Rectilinear_60_T(20)	67	60	Taz6Tibia_Rectilinear_60_T(60)	36	Taz6Tibia_Rectilinear_60_T(71)	83
18	Taz6Tibia_Rectilinear_60_T(18)	147	Taz6Tibia_Rectilinear_60_T(24)	9	61	Taz6Tibia_Rectilinear_60_T(61)	32	Taz6Tibia_Rectilinear_60_T(72)	68
19	Taz6Tibia_Rectilinear_60_T(19)	71	Taz6Tibia_Rectilinear_60_T(27)	10	62	Taz6Tibia_Rectilinear_60_T(62)	31	Taz6Tibia_Rectilinear_60_T(73)	38
20	Taz6Tibia_Rectilinear_60_T(20)	67	Taz6Tibia_Rectilinear_60_T(23)	37	63	Taz6Tibia_Rectilinear_60_T(63)	36	Taz6Tibia_Rectilinear_60_T(74)	14
21	Taz6Tibia_Rectilinear_60_T(21)	99	Taz6Tibia_Rectilinear_60_T(24)	27	64	Taz6Tibia_Rectilinear_60_T(64)	42	Taz6Tibia_Rectilinear_60_T(32)	9
22	Taz6Tibia_Rectilinear_60_T(22)	99	Taz6Tibia_Rectilinear_60_T(32)	12	65	Taz6Tibia_Rectilinear_60_T(65)	46	Taz6Tibia_Rectilinear_60_T(84)	10
23	Taz6Tibia_Rectilinear_60_T(23)	115	Taz6Tibia_Rectilinear_60_T(27)	23	66	Taz6Tibia_Rectilinear_60_T(66)	31	Taz6Tibia_Rectilinear_60_T(84)	10
24	Taz6Tibia_Rectilinear_60_T(24)	70	Taz6Tibia_Rectilinear_60_T(27)	20	67	Taz6Tibia_Rectilinear_60_T(67)	19	Taz6Tibia_Rectilinear_60_T(80)	13
25	Taz6Tibia_Rectilinear_60_T(25)	100	Taz6Tibia_Rectilinear_60_T(25)	11	68	Taz6Tibia_Rectilinear_60_T(68)	18	Taz6Tibia_Rectilinear_60_T(84)	9
26	Taz6Tibia_Rectilinear_60_T(26)	58	Taz6Tibia_Rectilinear_60_T(30)	20	69	Taz6Tibia_Rectilinear_60_T(69)	21	Taz6Tibia_Rectilinear_60_T(30)	7
27	Taz6Tibia_Rectilinear_60_T(27)	41	Taz6Tibia_Rectilinear_60_T(32)	19	70	Taz6Tibia_Rectilinear_60_T(70)	34	Taz6Tibia_Rectilinear_60_T(82)	8
28	Taz6Tibia_Rectilinear_60_T(28)	49	Taz6Tibia_Rectilinear_60_T(33)	14	71	Taz6Tibia_Rectilinear_60_T(71)	70	Taz6Tibia_Rectilinear_60_T(5)	16
29	Taz6Tibia_Rectilinear_60_T(29)	60	Taz6Tibia_Rectilinear_60_T(34)	44	72	Taz6Tibia_Rectilinear_60_T(72)	96	Taz6Tibia_Rectilinear_60_T(10)	11
30	Taz6Tibia_Rectilinear_60_T(30)	93	Taz6Tibia_Rectilinear_60_T(35)	11	73	Taz6Tibia_Rectilinear_60_T(73)	46		
31	Taz6Tibia_Rectilinear_60_T(31)	78	Taz6Tibia_Rectilinear_60_T(35)	34	74	Taz6Tibia_Rectilinear_60_T(74)	36		
32	Taz6Tibia_Rectilinear_60_T(32)	60	Taz6Tibia_Rectilinear_60_T(10)	10	75	Taz6Tibia_Rectilinear_60_T(75)	38		
33	Taz6Tibia_Rectilinear_60_T(33)	53	Taz6Tibia_Rectilinear_60_T(38)	12					

.3 IEC 61131-3 Full Software Model

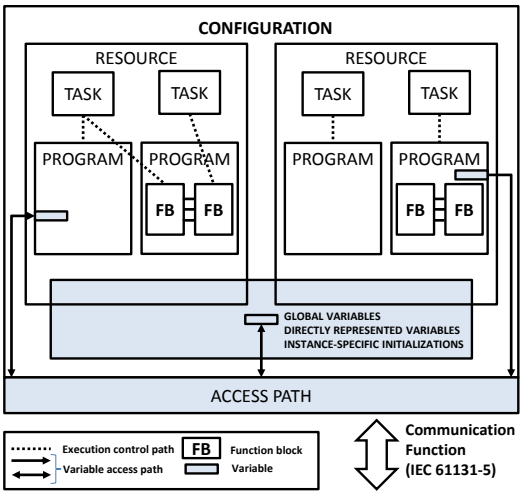


Figure 2: Full software model of PLCs based on IEC 61131-3 standard John and Tiegelkamp (2010).

.1 Secure Water Treatment Plant (SWaT)

SWaT is a 6-stage water treatment plant that produces 5 gallons/minute of treated water. The plant can operate non-stop 24/7 in fully autonomous mode. The six sub-processes in the plant, one corresponding to each stage, are shown in Figure 3. Each sub-process is controlled by a Programmable Logic Controller (PLC).

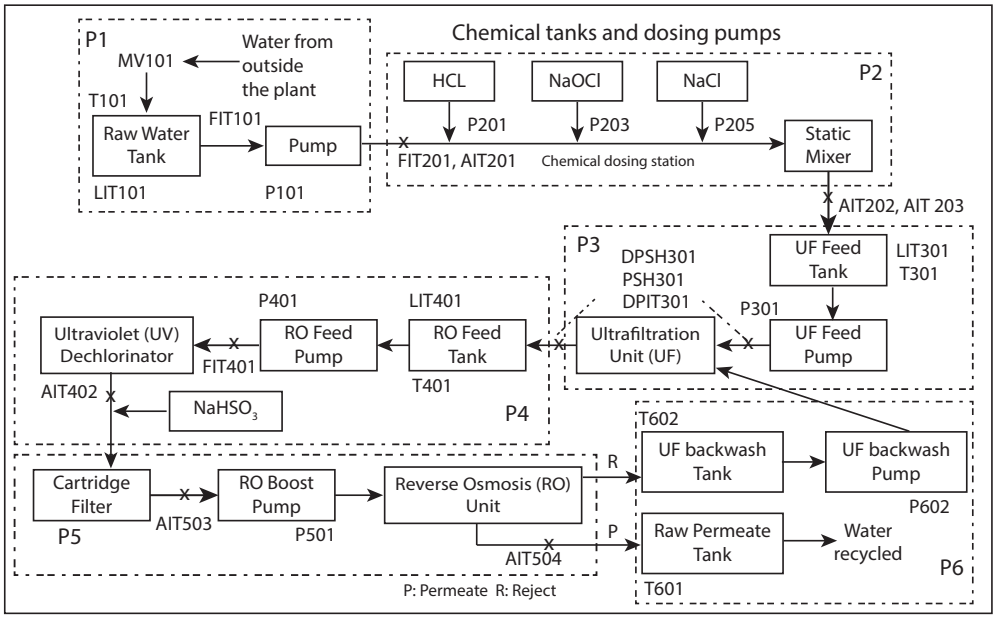


Figure 3: Sub-processes in a 6-stage water treatment plant operation in iTrust). **PLC**: Px: for stage x. **Sensors**: LITxxx: water level sensor; AITxxx: chemical property analyzer; DPITxxx: Differential pressure sensor. **Actuators**: Pxxx: pump; MVxxx: Motorized valve; PSHxxx: High pressure switch.

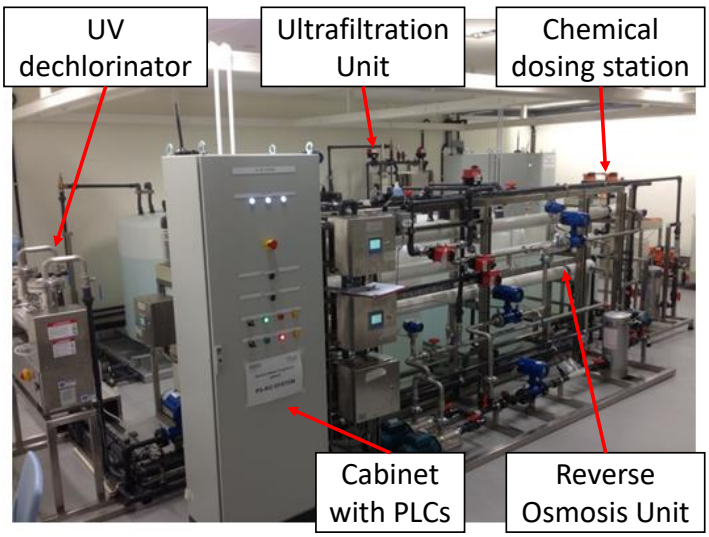


Figure 4: Sub-processes in a 6-stage water treatment plant operation in iTrust

.

Sensors and actuators: SWaT contains a total of 68 sensors and actuators; not all are shown in Figure 3. Some actuators serve as standbys and are intended to be used only when the primary actuator fails. For example, pump P102 is always in standby mode and is used only when pump P101 fails.

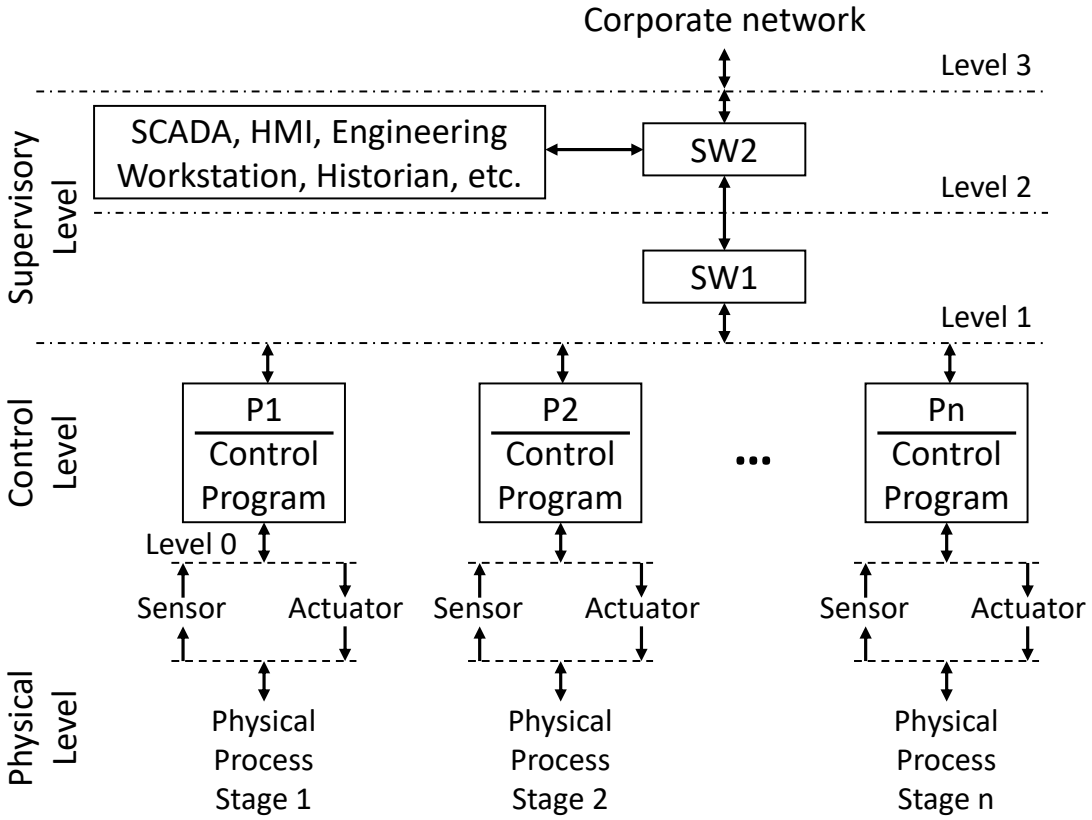


Figure 5: Architecture of the control portion of a CPS. P_1, P_2,...,P_n denote PLCs.

SWaT operation: Operation of the plant is initiated by an operator at the SCADA workstation and, when needed, can be controlled. State information can be viewed at the workstation or at the HMI, and is recorded in the historian. Process anomaly detectors, i.e. monitors, developed by researchers have been installed in SWaT. Detectors generate visual alerts and send messages to the operator. All alerts generated by the monitors, i.e. coded invariants, are recorded in the historian. SWaT can be attacked by compromising its communications network at all levels as well as directly by accessing the PLCs, the SCADA workstation, and the HMI. Physical attacks are feasible in SWaT through several means such as by replacing or removing sensors, disconnecting wires between sensors/actuators and the PLCs, removing power to one or more actuators, and so on.

Plant supervision and control: A Supervisory Control and Data Acquisition (SCADA) workstation is located in the plant control room. Data or control access to nearly all plant components is available via this workstation. A plant operator can view process state and set process parameters via the workstation. A Human Machine Interface (HMI) is also located inside the plant room and can be used to view process state and set parameters. Control code can be loaded into each PLC via the workstation. A historian is available for recording process state as well as network packet flows at

preset time intervals.

Communications: A multi-layer network enables communications (as shown in Figure 5) across all components of SWaT. The ring network at each stage at level 0 enables PLCs to communicate with sensors and actuators at the corresponding stage. A star network at level 1 enables communications across PLCs, SCADA, HMI and the historian. Both wired and wireless options are available at level 1 and also for communications with the sensors at level 0. PLCs communicate with each other through the L1 network, and with centralized Supervisory Control and Data Acquisition (SCADA) system and Human-Machine Interface (HMI), through the Level 2 network.

.2 SWaT Dataset

The SWaT data set Goh et al. (2016) is a publicly available dataset that has been used in several projects Goh et al. (2017). The time stamped dataset consists of 7 days of normal operation of the SWaT testbed as well as 4 days in which several attacks were evaluated for different components and physical processes in the testbed.

The dataset is stored in a CSV file which contains the time stamped values for all sensor data of the testbed as well as whether a particular attack was implemented or not. Additional data is provided that indicates the nature of each attack to provide a ground truth for our intrusion detection. The sensor data indicates the states of various plant components including tanks, valves, pumps, and meters, as well as data on chemical properties including pH, conductivity, and the Oxidation Reduction Potential (ORP).

Table 1: Sample data from SWaT dataset.

Timestamp	**FIT101**	**LIT101**	**MV101**	**P101**	**FIT201**
22/12/2015 4:00:00 PM	2.470294	261.5804	2	2	2.471278
22/12/2015 4:00:01 PM	2.457163	261.1879	2	2	2.468587
22/12/2015 4:00:02 PM	2.439548	260.9131	2	2	2.467305
27/12/2015 3:45:59 AM	2.471575	538.8619	2	1	0.007432801
27/12/2015 3:46:00 AM	2.458764	539.5684	2	1	0.00076891
27/12/2015 3:45:59 AM	2.471575	538.8619	2	1	0.007432801
28/12/2015 4:30:30 AM	0	812.8069	1	1	0
28/12/2015 4:30:31 AM	0	812.6106	1	1	0
28/12/2015 4:30:32 AM	0	812.6106	1	1	0

FIT: 0 $\Longrightarrow$ no flow; MVxxx: 2 $\Longrightarrow$ *OPEN*; 1 $\Longrightarrow$ *CLOSED*; PVxxx: 2 $\Longrightarrow$ *ON*; 1 $\Longrightarrow$ *OFF*.

For illustration, a few rows and columns, extracted from SWaT data, are included in Table 1. Data in the first row of the table was recorded on Dec 22, 2015 at 4pm. It indicates that valve MV101 is *OPEN* (2) and pump P101 is *ON* (2). The inflow and outflow rates into and from tank T101 as indicated by FIT101 and FIT102, respectively, are around 2.47. The nearly same inflow and outflow rates are consistent with the water level in T101 which hovers around 261 as indicated by LIT101.

The dataset consists of seven days of normal continuous operation and four days with attack. A total of thirty-six attacks were conducted during the four days. The attacks generated in the dataset were modeled after Adepu et al. Adepu and Mathur (2016c,d). The attack model considers the intent space of an attacker for any given CPS. As listed in Table 2, the attack duration depends on the kind of attack. Some attacks occur consecutively within a 10 minutes gap of one other, while some of the

Table 2: Sample attack descriptions in SWaT dataset

Type of attack	Attack ID	Attacker's Intent	Start state of System	Description of Attack
Single Stage Single Point Attacks (SSSP)	A1	Overflow Tank	MV-101 is closed	Open MV-101
Single Stage Multi Point Attacks (SSMP)	A2	Tank overflow	MV-101 is open; LIT-101 between L and H	Keep MV-101 on continuously; Value of LIT-101 set as 700 mm
Multi Stage Single Point Attacks (MSSP)	A3	Underflow tank in P1; Overflow tank in P3	P-101 is off; P102 is on; LIT-301 is between L and H	P-101 is turned on continuously; Set value of LIT-301 as 801 mm
Multi Stage Multi Point Attacks (MSMP)	A4	Underflow tank in P1; Overflow tank in P3	P-101 is off; MV-101 is off; MV201 is off; LIT-101 is between L and H;,LIT-301 is between L and H	Turn P-101 on continuously; Turn MV-201 on continuously; Set value of LIT-101 as 700 mm;

attacks are performed while leaving time for the system to stabilize. All the attacks are carried out by fooling the Programmable Logic Controller (PLC) at each process into believing the sensor information it is being sent is genuine, in other words, spoofing the values.

.3 Automated PLC Code Consolidation

The SWaT testbed consists of programs written in structured text, ladder diagrams, as well as function block diagrams. Although we only implemented translations for these languages, the tool would be easily extensible to support other languages and/or proprietary formats.

Ladder diagram translation. Ladder diagrams consist of a set of rungs that are executed sequentially. Each rung is typically composed of an initial rung condition followed by actions to take if the rung condition is true. As such, the conditional instructions that were followed by instructions and/or functions were translated to if-then-else statements. All instructions and/or functions were translated based on their equivalent structured text expressions provided by Allen Bradley Automation (2016 howpublished = http://literature.rockwellautomation.com/idc/groups/literature/documents/rm/1756-rm003_-en-p.pdf). The main routine for all PLCs were written as ladder diagrams. All subsequent routines were invoked via the jump-to-subroutine instruction (JSR).

Function block diagram translation. Function block diagrams typically consist of a block of structured text content and a set of inputs/outputs that need to be *wired* to variables from the calling routine. As such, declarations for each function block were extracted from L5X file and translated to function block declarations as specified by the IEC 61131-3 standard. Whenever a call to the function block diagram is encountered during the processing of a routine, the wiring configuration is generated to set up the parameters for a function call. The call to the function is inlined sequentially in the calling routine as a structured text statement.

Structured text conformance. Any structured text routines–including structured

text routines generated by our aforementioned translations–needed to be examined to ensure that all variable and function references are resolved. The Allen Bradley programming languages contain several built-in functions and data structures that needed to be defined based on their functional description Automation (2016 howpublished = http://literature.rockwellautomation.com/idc/groups/literature/documents/rm/1756-rm003_-en-p.pdf). In the current implementation our tool provides translations for those functions/structures that were used by the SWaT system's PLC's.

.4 Acronyms

ICS	Industrial Control System
LIT	Level Indicator and Transmitter
FIT	Flow Indicator and Transmitter
DPIT	Differential Pressure Indicator and Transmitter
AIT	Analyzer Indicator and Transmitter
HMI	Human Machine Interface
MITM	Man In The Middle
MV	Motorized valve
UF	Ultrafiltration
RO	Reverse Osmosis
PLC	Programmable Logic Controller
SCADA	Supervisory Control and Data Acquisition
SWaT	Secure Water Treatment