SECURING SAFETY-CRITICAL SYSTEMS USING PHYSICAL AND CONTROL INVARIANTS

By

SRIHARSHA ETIGOWNI

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

Saman Zonouz

and approved by

_____

_____

_____

_____

New Brunswick, New Jersey

January, 2020

ABSTRACT OF THE DISSERTATION

Securing Safety-Critical Systems using Physical and Control Invariants

by Sriharsha Etigowni

Dissertation Director:

Saman Zonouz

The critical infrastructures such as electrical power systems, telecommunication, transportation systems, and safety systems in vehicles and avionics comprise some critical safety systems. Safety critical systems use real-time control, including the software used in the design of physical systems and structures, whose failure can have a life-threatening impact. Failure or malfunctioning of such systems will lead to damage to the equipment/property, cause serious injuries or death to people. Since the safety-critical systems use embedded systems running software on them, they are prone to kinetic cyber attacks. Kinetic cyber attacks are a class of cyber attacks that can cause physical damage, injury or death solely through the exploitation of vulnerabilities on the systems. Most of the safety-critical systems are cyber-physical systems. The main targets of kinetic cyber attacks are cyber-physical systems due to there tight coordination between the computational and physical systems.

This thesis provides security solutions using both the cyber as well as physical space together which are tailored for cyber-physical systems. This thesis provides cyber-physical security assessments and solutions by considering the interdependencies between cyber and physical worlds. We leverage the physical and control invariants for security assessment, control

flow monitoring and verification purposes at different levels of abstraction in safety-critical systems. The physical invariants of a cyber-physical system is the laws of physics which will not change and the control invariants are the control algorithm which do not change during there operations. Some of the physical and control invariants used in this thesis for securing critical systems are electromagnetic emanation due to noise in digital circuits, flight dynamics for UAV's and power flow equations, swing equations for the electrical power system.

First, we introduce a novel approach to vulnerability assessment in *critical infrastructures* by cyber-physical interdependency. We provide an attack synthesis method for power grids, which is analogous to the penetration testing in cybersecurity. Second, to prevent attacks against the control logic used in controllers, we provide a runtime verification solution by leveraging the physical and control invariants of the system. Control logic is a part of a software program that controls the operations of the program. The number of states that a system can be is represented by state variables. The system state space is the set of possible configurations of the system. The state space increases exponentially with the increase in state variables. The proposed verification technique can solve problems such as state space explosion when used on cyber-physical systems. The above-mentioned control logic verification technique has short-come in detecting firmware level malware such as physics aware rootkits. Hence, we provide a separate solution of contactless side channel control flow monitoring technique by receiving the electromagnetic emanations from the PLC. Finally, we introduce cyber-physical access control considering the cyber and physical interdependencies. Cyber-physical access control makes decisions to grant or reject access to an authenticated subject based on what he is authorized to access. Cyber-physical access control is proposed to prevent the system from entering an unsafe state. Apart from these defensive solutions, we also provide the defensive solution earlier in the pipeline, the manufacturing process of the physical system used in safety critical systems.

Acknowledgements

I would firstly like to thank my doctoral committee for there valuable suggestions and helping me to improve and produce my thesis.

I am thankful to Prof Saman Zonouz my adviser, who was by my side supporting, having confidence in me and guiding me during the highs and lows in my research.

I would also like to thank people from my lab Gabriel, Luis, Pengfei, Mingbo and my collaborators Shamina, Maryam, Kate, Kevin, Sizhuang, Zhenqi with who I worked, enjoyed after work and during conferences.

My friends who made me feel like family Nesar, Amogh, Dileep, Spoorti, Harsh, Dhruval, Chethan, Prashant, Sandhya, Ajoy, Anish, Aarti, Ashika, Varad for there support and making my life easier and fun during my PhD. I have not listed everyone's name since the list is huge, but you all are on my mind.

Finally, I am greatly thankful to US National Science Foundation for the grant without which this work would not be possible. This work is supported in part by the US National Science Foundation under grant numbers CNS-1446471, CNS-1453046.

Dedication

This thesis is dedicated to my parents and my brother.

For their endless love, support and encouragement

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Department of Homeland Security has identified 16 critical infrastructure sectors which are vital to the country and any destruction or attack on such systems will have drastic effects on the security, economic, and public health [60] of a country. Since the attacks on these critical infrastructures could cause massive destruction, they are targeted by the attackers. These cyber attacks are the most dangerous weapons that a country has to protect against in the cyber-connected world.

The discovery of the Stuxnet worm, a malicious software that replicates itself in order to spread to other computers in 2010 proved that the cyber attacks on critical infrastructures are one of the critical problems a country has to defend against. Stuxnet attacked Iran's nuclear enrichment facility to sabotage the program. These cyber worms could sabotage the technological growth of a country. After Stuxnet, there were many attacks on critical infrastructures such water treatment plant and electrical grids. In 2015, attack on Kemuri Water Company as referred by Verizon [207], the attackers modified the chemicals in the treated tap water which demonstrated that attacks could sabotage the public health of a country. These attacks on critical infrastructure can be used as biological weapons. Ukraine's power plant attack [191] from a mutant of the blackenergy malware in December 2015 demonstrates that such attacks could sabotage a country economically.

The critical infrastructure mainly consists of *Supervisory control and data acquisition* (SCADA) systems [36], *Distributed control systems* (DCS) [39], and *Programmable Logic Controllers* (PLC) [35]. SCADA systems are used for monitoring and controlling underlying applications. DCS's are used to collect inputs from sensors and send outputs to actuators such as motor drives. PLC's are industrial grade embedded systems for controlling physical systems. The use of legacy systems with known vulnerabilities in safety-critical systems is one of the reasons for

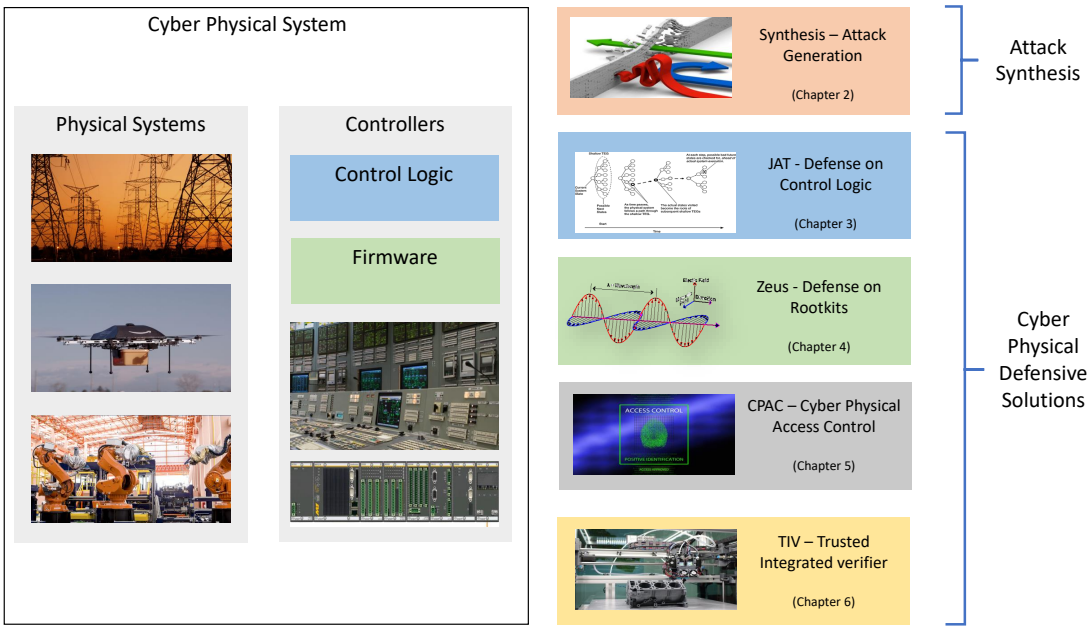


Figure 1.1: Overview of thesis

the attacks. The Kemuri Water Company was using 1980's IBM Application System (AS/400) server [96]. Most of the OT and IT operations were running on AS400 and the company was using AS400 as its SCADA platform. This system was controlling hundred's of PLC's which were controlling water district's valve and flow control application. Since the company was using a legacy system and the attacker had access to the vulnerabilities of that legacy system, it was easier to perform the attack.

This thesis focuses on securing safety-critical systems such as electric power systems and transportation system. The overview of the thesis is shown in Figure 1.1. The physical systems in this thesis mainly focus on power grids, drones, and 3D printers. These physical systems are controlled by either PLC's or other microcontrollers/processors (cyber domain). This thesis answers the following questions:

- Can an attacker drive the system into an unsafe state by sequentially changing the outputs of actuators without being detected by the protection systems?
- Can an operator protect against controller attacks on control logic and firmware that drives the system into an unsafe state?
- Can we limit the capabilities of remote and insider attacks, e.g., disgruntled employees,

by restricting their capabilities in a similar vein to cyber access control from cybersecurity?

To assess the security of the critical infrastructure such as power grid, we wanted to determine if an attacker sabotage the entire power system by gaining control over a subset of controllers used. We introduce a novel approach to vulnerability assessment in safety-critical systems by cyber-physical interdependency. We provide an attack synthesis method for power grids that is analogous to the penetration testing in cybersecurity. Attacks were synthesized algorithmically using hybrid dynamics of the power grid. The attacker can sequentially modify the output to the actuators to drive the system into an unsafe state without being detected by traditional protection systems such as circuit breakers. In the best case scenario, the attack can bring down the complete power system leading to blackouts.

Since most of the safety-critical systems use embedded systems to control the physical systems, the control logic running on those embedded systems are critical for the safety and security of the systems. Control logic is a part of a software program that controls the operations of the program. The control logic should be verified so that it does not violate any safety requirements leading to damage of life and property. Hence, we provide a runtime control logic verification solution for the attack against control logic on a cyber-physical system. Previous verification techniques proposed (e.g., TSV [128] and HACMS [149]) were limited to verification against safety properties before uploading the control logic. Since the solution was static verification, the runtime safety violations can be evaded. This control logic verification also causes state space explosion for larger control logic's due to exploring all the program execution paths. On the other hand, existing dynamic execution monitoring solutions (e.g., Avatar [221] and WeaselBoard [137]) notify operators about incidents that have just occurred or are about to occur, and hence do not leave enough of a time buffer for effective manual or automated response and recovery. Hence, we provide *just ahead of time* verification solution by predicting the future control paths of the control logic. Just ahead of time verification will provide a time buffer for the solution or controlling operator to react ahead of time and prevent the system entering into an unsafe state. Since the solution only verifies the control logic till a certain time horizon (just a few steps ahead of time), it reduces the risk of state space explosion of the control logic program's execution paths. By leveraging the physical properties of

the systems, we propose verification techniques that can be used without state space explosion when used on cyber-physical systems.

Although the verification technique provides protection against the attacks on control logic, they are inadequate to protect against physics aware rootkits [79]. Traditional software-based detection systems cause performance overhead and will affect the real-time of the systems. Additionally, the addition of software probes will increase the attack surface. Hence, we provide a contactless side channel control flow monitoring technique by receiving the electromagnetic emanations from the PLC. During the PLC code execution, the processor clock frequency and switching of the underlying CMOS devices along with the power regulation board result in the change of electric current in the PLC circuitry. The current produces a time-varying magnetic field that interacts with the electric field leading to an electromagnetic (EM) wave. The components on the PLC's printed circuit board (PCB) act as antennas to transmit the EM wave generated. This EM waves received by the EM probe are compared with the legitimate control flow of the program that was fingerprinted through a secure medium before the program was deployed on the system. This provides control flow integrity of the program running on the embedded controller. Since this solution does not require any changes to the current running systems, it can be applied to legacy systems as well.

By the above solutions, we detect controllers against attacks on the control flow of the running programs and protect against any violation of the safety requirements. However, these solutions would not protect against insider attacks or remote attacks by spoofing. The insider attacks could be due to lack of knowledge or disgruntle employees. The systems operators are authenticated to change the control logic or the actuator values of a system if required. However, a disgruntled employee or a remote attacker spoofing as an employee can upload a malicious control logic or change the value of the actuators to drive the system into unsafe states. Hence, to restrict the capabilities of the operators, we introduce cyber-physical access control considering the cyber and physical interdependencies for safety-critical systems.

The above solutions focused on verifying the safety and security of the current and future states of safety-critical systems. However, can we provide such guarantees at a much earlier

stage in the pipeline, i.e., at the time of manufacturing of such safety-critical systems? Especially since now such objects used in safety-critical systems can be 3D printed. In the manufacturing industry, to manufacture any components they employ either subtractive manufacturing or additive manufacturing. In both, the scenario's, the design files describe the manufactured objects. Attacks on these design files will lead to a malicious manufactured object which could cause serious damage to property and human life. Previous studies [28] have shown that by attacking design file by adding voids to the critical locations of a propeller, they were able to crash a drone flying mid-air. Therefore, it is equally important to protect against attacks on the input files such as design files which are feed to the controllers. We introduce TIV framework to detect such voids in the STL file before the manufacturing to reduce wastage to time and resources. TIV focuses on security solution to detect attacks against the critical inputs to the controllers such as designs used for describing the 3D printed objects. This security solution will check the structural integrity of the designs before they are printed. Hence, only permitting designs that are benign and will not fail during there normal operational conditions.

The preliminary results are as follows.

Algorithmic Attack Synthesis using Hybrid Dynamics of Power Grid Critical Infrastructures. Automated vulnerability assessment and exploit generation for computing systems have been explored for decades. However, these approaches are incomplete in assessing industrial control systems, where networks of computing devices and physical processes interact for safety-critical missions. We present an attack synthesis algorithm against such cyber-physical electricity grids in Chapter 2. The algorithm explores both discrete network configurations and continuous dynamics of the plant's embedded control system to search for attack strategies that evade detection with conventional monitors. The algorithm enabling this exploration is rooted in recent developments in the *hybrid system* verification research: it effectively approximates the behavior of the system for a set of possible attacks by computing sensitivity of the system's response to variations in the attack parameters. For parts of the attack space, the proposed algorithm can infer whether or not there exists a feasible attack that avoids triggering protection measures such as relays and steady-state monitors. The algorithm can take into account constraints on the attack space such as the power system topology and the set of controllers across the plant that can be compromised without detection. With a proof-of-concept prototype, we

demonstrate the synthesis of transient attacks in several typical electricity grids and analyze the robustness of the synthesized attacks to perturbations in the network parameters.

Crystal (ball): I Look at Physics and Predict Control Flow! Just-Ahead-Of-Time Controller Recovery. Recent major attacks against unmanned aerial vehicles (UAV) and their controller software necessitate domain-specific cyber-physical security protection. Existing offline formal methods for (untrusted) controller code verification usually face state-explosion. On the other hand, runtime monitors for cyber-physical UAVs often lead to too-late notifications about unsafe states that makes timely safe operation recovery impossible.

In Chapter 3, we present Crystal, a just-ahead-of-time control flow predictor and proactive recovery for UAVs. Crystal monitors the execution state of the flight controller and predicts the future control flows ahead of time-based on the UAV's physical dynamics. Crystal deploys the operator's countermeasures proactively in case of an upcoming unsafe state. Crystal's just-ahead-of-time model checking explores the future control flows in parallel ahead of the UAV's actual operation by some time margin. The introduced time margin enables Crystal to accommodate operator's feedback latency by the time the actual execution reaches to the identified unsafe state. Crystal periodically queries the controller's execution state. It emulates the UAV physical dynamical model and predicts future sensor measurements (controller inputs) and upcoming feasible controller's execution paths. This drives Crystal's model-checking exploration away from unreachable future states. Crystal's selective model checking saves computational time to stay ahead of execution by concentrating on relevant upcoming control flows only. This eliminates the state-explosion problem in traditional offline formal methods. We evaluated a multi-threaded prototype of Crystal between the control station server and the UAV. Crystal was able to predict upcoming hazardous states caused by the third-party controller program and proactively restored the safe states successfully with minimal overhead.

Watch Me, but Don't Touch Me! Contactless Control Flow Monitoring via Electromagnetic Emanations. Trustworthy operation of industrial control systems depends on secure and real-time code execution on the embedded programmable logic controllers (PLCs). The controllers monitor and control the critical infrastructures, such as electric power grids and health-care platforms, and continuously report back the system status to human operators. In Chapter 4 , we present Zeus, a contactless embedded controller security monitor to ensure its

execution control flow integrity. Zeus leverages the electromagnetic emission by the PLC circuitry during the execution of the controller programs. Zeus's contactless execution tracking enables non-intrusive monitoring of security-critical controllers with tight real-time constraints. Those devices often cannot tolerate the cost and performance overhead that comes with additional traditional hardware or software monitoring modules. Furthermore, Zeus provides an air-gap between the monitor (trusted computing base) and the target (potentially compromised) PLC. This eliminates the possibility of the monitor infection by the same attack vectors.

Zeus monitors for control flow integrity of the PLC program execution. Zeus monitors the communications between the human-machine interface and the PLC, and captures the control logic binary uploads to the PLC. Zeus exercises its feasible execution paths, and fingerprints their emissions using an external electromagnetic sensor. Zeus trains a neural network for legitimate PLC executions, and uses it at runtime to identify the control flow based on PLC's electromagnetic emissions. We implemented Zeus on a commercial Allen Bradley PLC, which is widely used in industry, and evaluated it on real-world control program executions. Zeus was able to distinguish between different legitimate and malicious executions with 98.9% accuracy and with zero overhead on PLC execution by design.

CPAC: Securing Critical Infrastructure with Cyber-Physical Access Control. Critical infrastructure such as the power grid has become increasingly complex. The addition of computing elements to traditional physical components increases complexity and hampers insight into how elements in the system interact with each other. The result is an infrastructure where operational mistakes, some of which cannot be distinguished from attacks, are more difficult to prevent and have greater potential impact, such as leaking sensitive information to the operator or attacker.

In Chapter 5, we present CPAC, a *cyber-physical access control* solution to manage complexity and mitigate threats in cyber-physical environments, with a focus on the electrical smart grid. CPAC uses information flow analysis based on mathematical models of the physical grid to generate policies enforced through verifiable logic. At the device side, CPAC combines symbolic execution with lightweight dynamic execution monitoring to allow non-intrusive taint analysis on programmable logic controllers in realtime. These components work together to provide a realtime view of all system elements, and allow for more robust and finer-grained

protections than any previous solution to securing the grid.

We implement a prototype of CPAC using Bachmann PLCs and evaluate several real-world incidents that demonstrate its scalability and effectiveness. The policy checking for a nation-wide grid is less than 150 *ms*, faster than existing solutions. We additionally show that CPAC can analyze potential component failures for arbitrary component failures, far beyond the capabilities of currently deployed systems. CPAC thus provides a solution to secure the modern smart grid from operator mistakes or insider attacks, maintain operational privacy, and support $N-x$ contingencies.

TIV: A Trusted Integrity Verifier for STL Files in Additive Manufacturing. STereoLithography (STL) files describe the geometry of objects to be printed in additive manufacturing. Previous studies have shown that the STL files that describe functional objects can be attacked such that the objects appear normal during inspection, but fail during operation. Such attacks lead to damage to systems that use the objects and possibly loss of life. The detection of any defects caused due to the attacks nowadays is limited to the quality control process after the objects are manufactured.

We present a *Trusted Integrity Verifier* (TIV) to detect such attacks on 3D printed objects in the early stage of the manufacturing process. These type of new attacks cannot be detected by traditional software security mechanisms since they only focus on the printers and do not consider the inputs (STL design files) to the printer. Early detection of attacks prevents from printing malicious objects resulting in saving time, resources and manufacturing efforts. TIV detects malicious STL files using multidisciplinary approaches unlike the traditional integrity verification techniques. TIV leverages computer vision techniques (flood fill) to identify the internal defects such as voids. Some of these features could be from the design and some could be due to the attack. To differentiate the malicious features from the design features, TIV implements a numerical methods-based approach (Finite Element Analysis). TIV uses Finite Element Analysis to differentiate the malicious features from the design features by calculating the loading bearing mechanical stress on the objects. These mechanical stresses are compared to the safety operational conditions to determine if the printed object will break or fail during its normal operation.

To illustrate TIV's generality and scalability, we conducted a large-scale analysis on 16,000

real-world 3D print STL files. TIV verified the STL files successfully as either safe or malicious with high accuracy of 92% for object classification and 96.5% for void detection.

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

Chapter 2

ALGORITHMIC ATTACK SYNTHESIS USING HYBRID DYNAMICS OF POWER GRID CRITICAL INFRASTRUCTURES

1 Introduction

Trustworthy operation of the nation's critical infrastructure like the electricity grid requires effective cybersecurity and power system protections simultaneously. Ideas from cybersecurity research have been extensively deployed to keep adversaries out of the critical plants and control systems. However, cybersecurity solutions alone are inadequate for safe-guarding cyber-physical systems where software is used to monitor and control networks of complex physical processes.

Recent security incidents presage this new class of vulnerabilities. One well-known example is the Stuxnet worm, which targeted Siemens industrial software used to control nuclear fuel processing plants. The worm exploited several complicated cyber attack vectors, including four Windows zero-day vulnerabilities and logic controller exploits. It ultimately sabotaged and destroyed an Iranian facility by introducing malicious control inputs to actuators controlling uranium centrifuges. Understanding the scale and sophistication of this attack has led to mandatory governmental regulations embodied in the North American Electric Reliability Corporation's Critical Infrastructure Protection (NERC-CIP) regulations [145] that is now widely adopted in the industry.

The most recent CIP (v5 - enforceable on April 1, 2016) has significantly increased strict cybersecurity requirements in order to prevent adversarial power grid incidents by terrorists and targeted nation-state intruders. One of the major CIP standards for electricity grid protection is that the grid's real-time operation should always comply with "N-1" contingency resilience requirements. That is, given a power system with N components (for example transmission lines), the system must be able to tolerate any single component failure such as an overflown

line outage possibly caused by malicious control inputs. Non-compliant utilities are required to increase their system redundancy to improve tolerance, otherwise, they risk financial penalties imposed by the government. To implement the requirements, power utilities across the nation have been deploying various automation solutions such as protection relays to detect incidents, e.g., line current overflows, and clear the faults through the opening of circuit breakers. The utilities have developed regular procedures such as contingency analyses to periodically validate their system resilience against failures. The existing contingency analysis algorithms that are used in practice nowadays leverage power system to perform *steady-state* analysis of potential incidents and their potential consequences. Those solutions completely ignore non-steady-state or transient dynamics of the system. Thus far, these protections are largely believed to provide safety against the accidental component failures and malicious attacks such as Stuxnet that target the steady-state system dynamics.

We show that NERC-CIP cyber security requirements can be vulnerable to a new class of attacks, and hence inadequate for protecting the national electricity grid from dynamic transient exploits. Not only do we show the existence of transient attacks in systems that are compliant with NERC-CIP regulations, we present an algorithmic approach for effectively synthesizing such attacks from reasonably available information about power networks. These synthesized attacks against CIP compliant power networks can cause global system-level collapse or instability. As attackers are becoming more concerted and sophisticated, transient attacks should be taken into consideration for the design of future protection mechanisms and our analysis suggests conceptual methods for the security evaluation of power networks.

The synthesis procedure leverages *hybrid models* of the electricity grid that incorporate not only the continuous, nonlinear dynamics of the electrical quantities like currents, voltages, power, and phase angles, but also discrete dynamics of the changes in the network topology brought about by opening and closing of relays and changes in power consumption (loads). The synthesized attacks consist of a sequence of malicious control actions that ensure that all possible behaviors of the system remain undetected by the conventional steady-state monitors but ultimately destabilize the system.

Our goal is to automatically synthesize such destabilizing attack over a specified bounded time horizon such that the deployed CIP compliant protection mechanisms are not triggered.

Owing to the capabilities of the adversary, it turns out that the space of possible attacks can be naturally parameterized. The adversary can switch a few compromised relays (opening/closing power lines) or she can inject power at a few of the compromised buses. These actions can be stitched together to construct more complex attack sequences. Nevertheless, there are only a finite (and in fact small) number of network topologies that the adversary can realize. Although the set of possible power injection signals are infinite, they can also be parameterized via a family of curves or proportional–integral–derivative (PID) control signals. With a parameterized attack space, the proposed algorithm leverages state-of-the-art formal verification algorithms to effectively search this space for successful attacks. The key to this approach is to be able to quickly eliminate parts of the attack space that *cannot* produce successful attacks—either because they are detected or because they are unsuccessful in making the system unstable. The algorithm eliminates sets of unsuccessful attacks by first simulating one potential attack with specific parameter values and checking many other similar potential attacks, by computing on-the-fly the sensitivity of the system's behavior to changes in the attack parameters. If all attacks with a single topology are eliminated, then the algorithm proceeds to check attacks that string together several topology switches with different power injections in each.

Contributions. The contributions of this chapter are as follows: *i)* we design and demonstrate the first transient dynamics attack against critical electricity grid infrastructures, and show that the current governmental regulatory protections are rendered insufficient in practice; *ii)* we introduce an automatic attack synthesis framework that implements reachability-based synthesis to generate transient attacks for a given network while meeting realistic constraints on the attack space; and *iii)* we implemented a proof-of-concept prototype to demonstrate the feasibility of transient attacks synthesis against NERC CIP-compliant topologies.

The remainder of this chapter is organized as follows: Section 2 describes and justifies the threat model used in the chapter and gives an overview of the proposed approach. Section 3 describes the hybrid models of the power network, protections, and attacks. Section 4 presents the attack synthesis problem formulation. Section 5 describes our results and demonstration of transient attacks. Section 6 shows a case study transient attack against a power system that is NERC CIP N-1 compliant. Section 7 reviews the past related work. Section 8 concludes the chapter.

2 Overview

In this section, we justify the attack model and the architecture used throughout this chapter based on the current state of the electricity grid, its operations, and its protection mechanisms.

Threat model. Like most of the past work on power grid attacks [122], we assume that the attackers know the configuration of target power network. We also assume that the attackers have access to necessary controllable actuation points to manipulate few system parameters such as a power generator governor valve (input mechanical power to the generator) and circuit breakers.

2.1 Electricity Grid Security

Electricity power grid consists of electric power generators, loads such as factories, where the power gets consumed, and the transmission lines to transfer the generated power to the consumption sites (loads). As an interconnected network, the power grid connects a variety of electric generators together with a host of users across a large geographical area. Generally, the power system transient dynamics originate from its power generators (synchronous machines). The generation AC power flows across the grid based on the Kirchhoff laws—power system equations. Any change to the system parameters or its network topology will change the equations and the power flow values will update accordingly. The changes to the grid are performed through installed actuators and are normally used by the power operators to control and ensure the safe operation of the grid. Examples of typical actuators on the power system include *(i)* circuit breakers that are physical devices controlled by computer-enabled relays, and could connect/disconnect a power system asset such as a generator to/from the rest of the grid interconnect; *(ii)* generator governor steam valve controller that defines how much mechanical torque should be fed to the generator and hence power to the rest of the grid. Hence in case of load (power consumption) increase, the valve should open further to compensate for the consumed power, and maintain the load-generation balance; *(iii)* the DC current value controller that defines how much current should flow through the field windings within a generator. Higher field currents increase the generator's output voltage, and hence this actuator could be used for power grid's global voltage control and prevent a wide-area voltage collapse. In a

real-world power system, there are many actuators deployed, and they could be maliciously accessed remotely through cyber attacks, and leveraged to change the power system parameters to cause a large-scale blackout.

For reliability purposes, redundant paths and lines are provided so that power can be routed from any power plant to any customers, through a variety of routes that is resilient against failures such as a line outage. A SCADA network is usually used to monitor and control the power system and devices in a geographical area. The SCADA network is connected to the physical power system components through distributed sensors and actuators. The daily operation of the power grid follows typical closed-loop monitoring and control paradigm. Sensors send noisy measurements to the SCADA network, where the state estimation servers process the received measurement data points. The state estimation's objective is to filter out the noise and calculate precise estimates of the various power system variables, i.e., power bus voltages. The power operators look at these estimates and decide on proper control actions that are sent to the actuators. For instance, if the calculated power system estimates indicate that there is more power consumption than the generation in the grid, the operator increases the generation set-points on one (or more) of the generators to maintain the load-generation balance. Traditional power grids often have protection measures deployed to maintain reliable system operations against accidental and natural events and failures such as a broken line.

Due to the close and constant interaction of the physical power system with the SCADA network, the electricity grid is referred to as a *cyber-physical* infrastructure. The emerging advanced SCADA equipment improves cyber interconnections among various points (e.g., actuators) of the system from remote sites. Although this simplifies the grid's monitoring and control significantly, improving SCADA technologies also vastly inflates the grid's attack surface. The adversaries launch the attack through cyber-side penetration to gain sufficient control on the cyber assets such as actuators like power system relays that are often used to (dis)connect a transmission line. The final and more important step of the attack is to leverage the compromised actuators to cause a physical damage to the power system. For instance, disconnecting a transmission line to a large neighborhood may result in a temporal power outage in the area. There have been several successful security attacks against the electricity grid within the United

States [6]. In fact, the number of reported major security attacks against the critical infrastructures has increased from 9 incidents in 2009 to 257 incidents in 2013, 28X within 4 years.

Due to the rising number of attacks, both the industry and government have taken initiatives to protect the electricity grid on both the cyber and the physical sides. Initial efforts involved incorporation of traditional IT cyber security mechanisms into SCADA networks. The direct adoption did not always result in suitable outcomes such as unacceptable packet transfer latencies for control inputs, which often have strict end-to-end time requirements. Later endeavors included more domain-specific mechanism deployments such as Digital Bond's Snort signatures for power control network DNP3 protocol. Following the cyber-side security protection improvements, however, there has been little progress on power-side solutions against non-accidental incidents and malicious adversaries that could lead to misoperation or instability.

NERC-CIP contingency requirements. Consequently, NERC developed 412-page CIP v5 requirements, and Federal energy regulation commission (FERC) approved the requirements on November 22, 2013. The NERC website says "[the requirements] represent significant progress in mitigating cyber risks to the bulk power system". One of the key requirements is the $N-1$ *contingency* compliance. NERC defines a power system contingency as an unexpected failure or outage of a system component, such as a generator, transmission line, circuit breaker, switch or other electrical element [145]. NERC CIP N-1 contingency requirement mandates that grid must operate safely given any single contingency. To enforce the requirements, power utilities run contingency analysis algorithms on the grid's current topology and state every few minutes to ensure that it can tolerate any single contingency. All of the deployed real-world algorithms make use of steady-state system models such as (rarely) non-linear AC power flow models or (almost always) linearized approximate DC models for speed up. None of these models consider the transient dynamics of the system. We answer the following question in this chapter: is it possible that an electricity grid is reported NERC CIP-compliant after state-of-the-art contingency analyses, while a (possibly sophisticated) attack still destabilizes it?

2.2 Overview of Synthesized Attacks

We show that the NERC-CIP compliant systems are vulnerable against a new class of attacks that leverage the transient dynamics of the power system. These attacks can be launched with

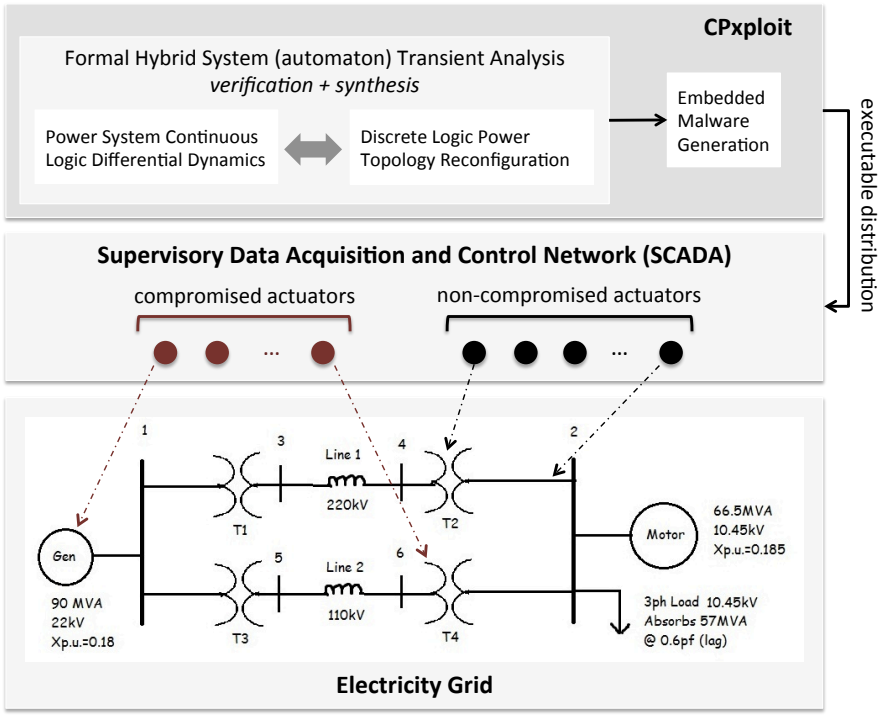


Figure 2.1: High-Level architecture for an attack.

the threat models discussed above. The attack's core idea is to manipulate the system in such a way that its fine-grained (fast) transient dynamics exceeds the safety zone, while the system's steady-state converged values resides within NERC CIP requirement limits. Our algorithm achieves its objectives through a controlled violation of the *synchrony condition* [26] that refers to the condition when both the AC current frequency and phase for all generators within the electricity grid are aligned. Malicious loss of synchrony in the grid can lead to blackouts. The frequency of a generator is directly related to the speed (angular velocity) of its internal rotor that converts its input mechanical power (e.g., by steam turbines) to electrical power. The generated electrical power is then injected into the grid. The input mechanical power magnitude can be adjusted by the generator's governor valve controller. Consequently, a malicious governor valve controller could manipulate the rotor speed, and hence the system frequency that could possibly perturb the grid's synchrony condition.

Figure 2.1 shows the high-level architecture of an attack and how its components are logically interconnected. CIP requirements are based on the widely-used simplifying assumption that the system transients die (fade) quickly and it is focused on detecting anomalies in the

steady-state values of the system. Normal transients are caused by abrupt discrete incidents, such as a transmission line outage, and do indeed meet this assumption. As we show, maliciously triggered transients, however, can be amplified significantly if one is followed by another in a quick succession. The synthesis algorithm explores the space of such transient attacks and (if possible) finds one that cannot be detected by CIP requirements that only consider the steady state values.

The algorithm searches for attacks over all possible sequence of network topology changes that the adversary can bring about. For each possible sequence, the algorithm explores the space of possible attacks that can drive the system to an *Unsafe* state that violates the synchrony condition and avoids the states *Detected* that can be detected by the conventional (CIP) monitors. This exploration is performed by over-approximating the system's behavior (reachable states) under a set of attack injections. This computation of reachable states pushes the state-of-the-art in formal verification for hybrid models (see Section 7 for an overview). A new class of verification algorithms [61, 64] compute reachable sets of the system by simulating individual behaviors and computing the local sensitivity of that execution to small perturbations to the states and inputs to the system. In this chapter, we extend this approach to analyze the sensitivity with respect to the attack parameters. Since the algorithm is based on over-approximations, it is *sound* in the sense that it can infer that certain range of parameter values and certain topologies *cannot* give rise to a successful attack. For example, for several networks, we show that CIP regulations effectively protect against attacks on the system with no discrete topological modifications. This enables the algorithm to move the search quickly to a different part of the attack space or a different (possibly longer) sequence of topology switches.

Finally, we comment on the advantages of synthesizing attacks. Generally, one could use simulation-based strategy to search for an attack. However, simulation-based approaches alone are incomplete; they cannot guarantee the absence of an attack even for small (but compact) sets of system parameters. Our formal approach using sensitivity computation enables us to eliminate (possibly large) sets of potential attacks as infeasible and therefore helps focus the search on interesting parts of the attack parameter space. Our choice was also influenced by the finding that CIP compliance and its protection measures significantly reduce the attack search space. It makes it significantly harder to realize an attack such that it can achieve its

adversarial objectives without triggering CIP's mandatory protection measures. Therefore, the proposed exploration for a feasible attack recipe includes the corner cases that are not often hit by simulation methods quickly in practice.

3 Transient Attacks on Power Networks

We describe typical dynamical models for monitoring and control of power networks, and then we proceed to develop the core technique for synthesizing transient attacks.

3.1 Power Network Dynamics

A power network consists of several buses. Each bus serves as an electrical interconnection point for generators, loads, and transmission lines to other buses. A network with three buses is illustrated as Figure 2.4a. The relationship among the currents and voltages in an N-bus network is given by the Kirchoff's law

$$\vec{I} = Y \cdot \vec{V},$$

where $\vec{I} = [I_1, \ldots, I_N]$ and $\vec{V} = [I_1, \ldots, I_N]$ are complex vectors corresponding to the currents the voltages in the buses, and Y is the $N \times N$ matrix of complex numbers called the *Nodal admittance matrix* and it captures the electrical parameters of the network. The *admittance* (reciprocal of the *impedance*) between the ith and the kth bus is the complex number $Y_{ik} = G_{ik} + jB_{ik}$. Its real part G_{ik} is called the *conductance* and its imaginary part B_{ik} is called the *susceptance*. The second relationship is given by the power flow equation: $S_i = V_i \cdot I_i^*$ for each bus i. By rewriting $\vec{I}$ as $Y.\vec{V}$ and comparing the real power P_i and the *reactive* or imaginary part of power Q_i at each bus, we get

$$P_i = \sum_{k=1}^{N} |V_i||V_k|(G_{ik}\cos(\delta_i - \delta_k) + B_{ik}\sin(\delta_i - \delta_k)), \tag{2.1}$$

$$Q_i = \sum_{k=1}^{N} |V_i||V_k|(G_{ik}\sin(\delta_i - \delta_k) - B_{ik}\cos(\delta_i - \delta_k)), \tag{2.2}$$

where δ_i is the phase angle of the complex voltage V_i. For an N-bus network this gives $2N$ equations with $4N$ unknowns, namely, $|V_i|$, P_i, Q_i, and δ_i for each bus i. If we fix $2N$ of these

unknowns (e.g., based on real-time sensor measurements), we can solve these equations and obtain the *steady state* values of power, reactive power, the voltage and the phase angle.

It is worth mentioning here that there are three different kinds of buses in a power system: *i)* a *generator bus* connects power generation equipment which fixes the real power P and the voltage $|V|$; *ii)* a *load bus* connects power consumers which fixes the real (P) and reactive power (Q); and *iii)* a *slack bus* connects to a *slack generator* that balances the active and reactive power in the system and it fixes the voltage $|V|$ and the phase angle θ at the bus. In a real power system with these three types of buses, each bus fixes two of the four unknowns, and therefore, we can solve Equation 2.1 and Equation 2.2 to obtain the steady state values of all the quantities.

In Section 4, we will consider power networks subject to changes in the input/output power, voltage, as well as in the network topology. When the topology of the network changes the admittance matrix Y changes, which in turn changes the steady state values of voltage, power, and phase of one or more buses as determined by Equations (2.1)-(2.2). Similarly, when the power at a load bus changes suddenly, then the steady state values at all the other buses may be affected. Between the time when the change occurs and the time when new steady state values are reached (if at all), the network is said to be in *transient state.* The thesis of this chapter is that currently implemented NERC-CIP monitoring approaches leave power systems fundamentally vulnerable to attacks that exploit the transient dynamics of the system.

Finally, we describe the differential equations that govern the transient behavior of each generator. Each generator k has two state variables, namely the synchronous phase angle θ_k and the angular velocity ω_k. Given the bus quantities δ_k and V_k from the above, the dynamics of generator k is given by

$$\dot{\theta}_k = \omega_k \tag{2.3}$$

$$\dot{\omega}_k = C_1(P_M - C_2|V_k|\sin(\theta_k - \delta_k) - C_3\omega), \tag{2.4}$$

where C_1, C_2, and C_3 are constant parameters of the generator and its circuitry; P_M is the input mechanical power to the generator (e.g., by a steam turbine). We denote the term

$$P_E := C_2|V_k|\sin(\theta_k - \delta_k)$$

as the output electrical power generated by the generator. Given a power system topology and steady state values of the bus voltage and phase angle, the output power is a function of the machine phase angle θ_k. The output current is derived by dividing the output power by the voltage $I_k = C_2 \sin(\theta_k - \delta_k)$. Equation 2.3 merely says that the rate of change of the phase angle θ_k is the angular speed ω_k. Equation 2.4 says that the rate of change of angular speed depends (according to this nonlinear function) on the mechanical power input P_M, the phase angle θ_k, the angular velocity ω, and the steady state voltage $|V_k|$ and rotor angle δ_k of the bus it connects to.

To keep the power system stable, the phase angle has to satisfy the *synchrony condition*, i.e., keep $\theta_k \leq a_{crit}$, where $a_{crit} = \pi/2 + \delta_k$ is the *critical angle* [113]. If the generator is operating below the critical angle, it is dynamically stable, meaning it can tolerate the environmental disturbance and maintain the power output. However, if the phase angle is above the critical angle, the generator's behavior goes out of synch with even little disturbances. We will further justify this statement in Section 6 with experiments on a high-fidelity power system simulator. Throughout this chapter, we study attackers that attempt to violate the synchrony condition and drive the phase angle larger than the critical angle without violating the steady-state NERC-CIP conditions.

3.2 Attack Protection: Relays and Monitors

The most widely deployed protections in power networks to enforce NERC-CIP requirements and protect against attacks are *relays* and *steady state monitors*. A relay has two parts: a sensor for detecting over-current conditions, and an actuator for electrically disconnecting power lines. Each sensor detects over-current condition on the line between a generator and a specific bus. A relay on the k-th bus registers an *over-current event* if $I_k = C_2 \sin(\theta_k - \delta_k)$ exceeds a critical current value I_{crit} (transmission line capacity) for at least a certain threshold duration c_{oc} of time. In contrast, steady state monitors check whether the steady state power output of a generator P_E deviates significantly from a desired value P_{ref} for some duration of time. These monitors detect a deviation if $P_E(t)$ is not within $\pm P_{tol}$ of P_{ref} over an interval of time of duration c_{pd} (here P_{tol} is a constant tolerance parameter).

Indeed, these protections are adequate against a broad class of attacks. One simple attack is

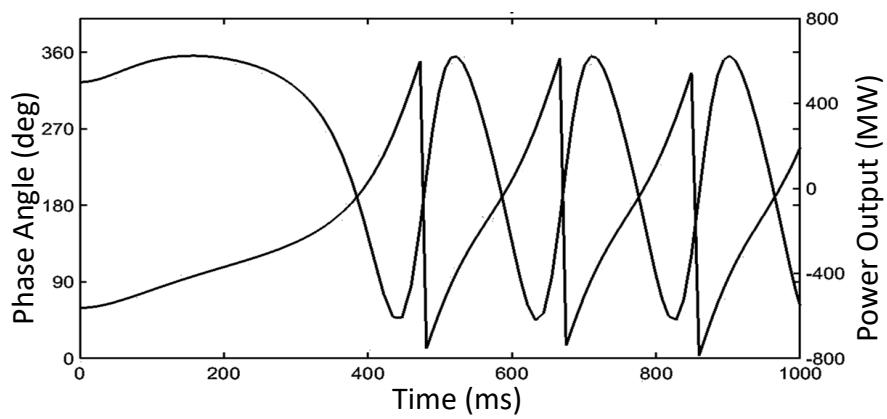


Figure 2.2: Attack on maximum input mechanical power. The lines capture the trajectories of phase angle and output power with constant input $P_M = 7$ p.u.

to set the input mechanical power P_M to a constant high value (e.g., via opening the steam turbine valves) such that the phase angle of the machine will increase and enter the asynchronous region exceeding the critical threshold value. The corresponding trajectories of the phase angle and the output power are illustrated in Figure 2.2. Since the phase angle is not stabilized, the output power also oscillates wildly; therefore, this attack can be caught by an appropriate steady state monitor. A more sophisticated attack would use a time-varying input power signal P_M. The attacker first computes a reference angle $\theta_r = \pi - \sin^{-1}\frac{P_{ref}}{C_2|V_k|} + \delta_k$, such that it violates the synchrony condition ($\theta_r > a_{crit}$) and it produces the steady state power output equals P_{ref}. Then, the attacker picks the PID constants $k_p, k_i, k_d \geq 0$ and computes the input mechanical power at each time $t \geq 0$ as

$$P_M(t) = k_p e(t) + k_i \int_0^t e(s)ds - k_d \omega(t), \tag{2.5}$$

where $e(t) = \theta_r - \theta_k(t)$ is the difference of the current phase angle and the reference angle. This attack can drive the phase angle to the asynchronous value θ_r while ensuring that the output power P_E to remains within the desired range. Hence it remains undetected by the steady state monitors. An example of how such an insidious attack may proceed is shown in Figure 2.3, where after the transient, the machine phase angle converges to $\theta_r = 133°$ and the output current converges to 5 p.u. (per unit). If the generator is equipped with a relay with critical current $I_{crit} = 5.5$ p.u., the relay will be triggered between the two vertical dashed lines in the plot for around 200 ms. If the critical over-current duration $c_{oc} \leq 200$ms, then the attack

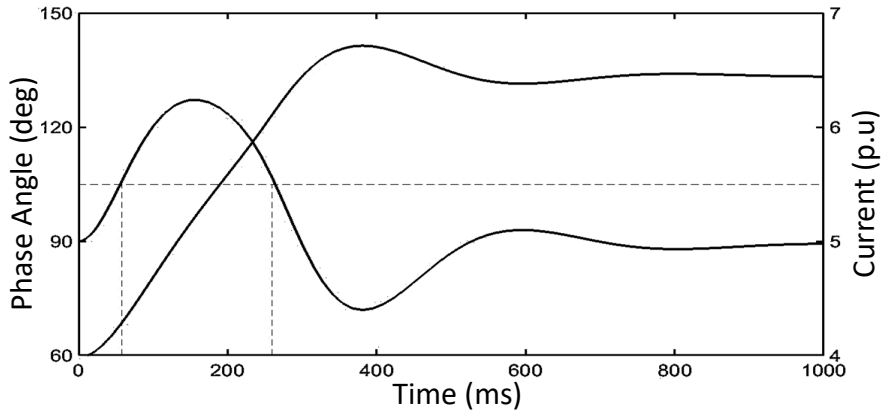


Figure 2.3: Attack on PID control. The solid lines capture the trajectories of phase angle and current. The critical current of the relay is 5.5 p.u as illustrated by the horizontal dashed line.

will be detected.

We observed that the relays and steady state monitors are able to detect some simple attacks such as maximum mechanical input attacks or PID attacks. In the following sections, we discuss how to synthesize a valid transient attack automatically that evades the deployed monitors by carefully composing the simple attacks.

4 The Attack Synthesis Problem

We present a formal model[1] for power networks as hybrid automata. We then use the model to synthesize and generate transient attacks automatically.

4.1 Formal Hybrid Model

As we saw in the previous section, the transient behavior of a power network can be modeled by several sets of differential equations—one for each configuration of the network topology and the bus parameters. These differential equations define the *trajectories* of the physical state of the system over time intervals. Together with these trajectories, the complete system description requires a set of *transitions* that define the abrupt changes in the network configuration, e.g., a relay opening. After every such transition, the power system transients evolve according to its dynamic model of differential equations. It has now become standard to model such systems

[1]This work was completely done by Zhenqi Huang from University of Illinois and added in my thesis to understand the complete picture of the solution.

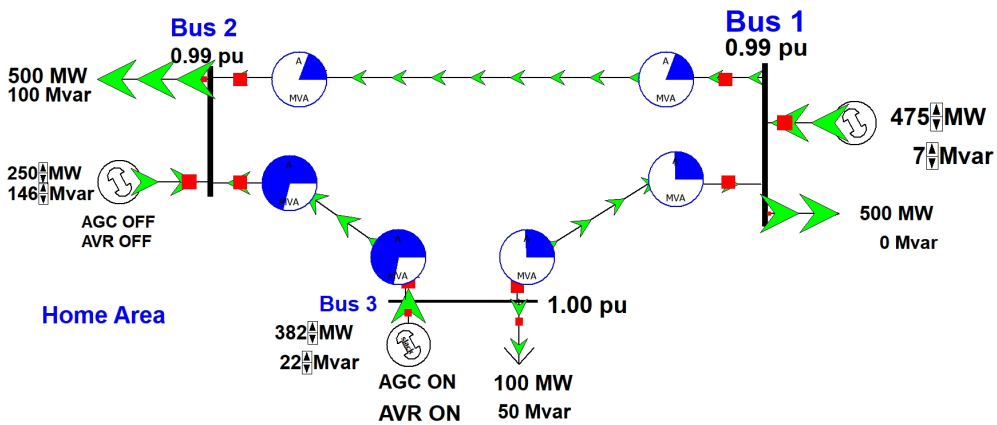
combining differential equations and transitions as a *hybrid automaton* [111, 131]. A hybrid automaton is a state machine with both discrete and continuous state variables. The discrete state variable typically captures software (cyber) states and the system configurations and their changes are specified by discrete transitions rules similar to if-then statements, e.g., a relay's open/close conditions. While the continuous variables capture physical variables (for example, voltage and phase angles) and their evolution is specified by differential equations. For the sake of precision, we give a formal definition of a hybrid automaton.

Definition 1. *A* hybrid automaton (HA) **A** *is a 6-tuple* $\langle Var, Q, U, \Theta, D, T \rangle$*, where*

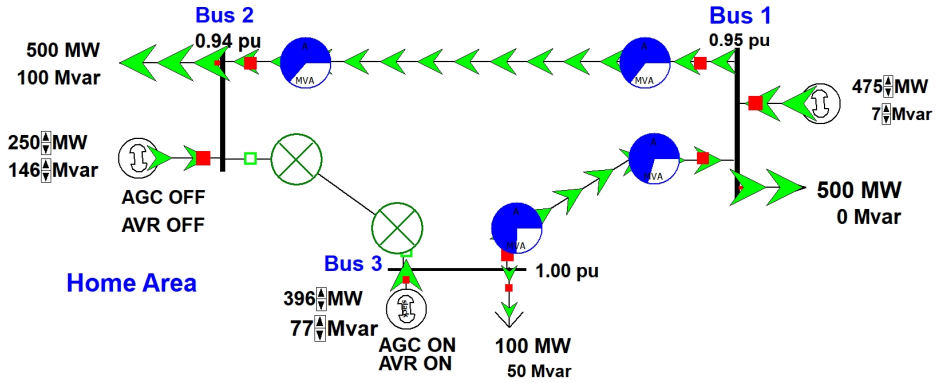
1. *Var is a variable set consisting of i) X, a set of real-valued continuous state variables; ii) mode, a single discrete variable that takes value in the finite set denoted by L.*
2. $Q = \mathbb{R}^{|X|} \times L$ *is the state space of the system; it corresponds to all possible valuations of the continuous variables in X and the discrete variable mode.*
3. *U is a set of real-valued continuous input variables.*
4. $\Theta \subseteq Q$ *is the set of initial states,*
5. $D \subseteq Q \times Q$ *is the set of possible discrete transitions.*
6. *T is the trajectory set. Each trajectory* $\tau \in T$ *is a function* $\tau : [0, t_1] \to Q$ *mapping the time interval* $[0, t_1]$ *to the states.*

For a transition $(q, q') \in D$, it is standard to write $q \to q'$; q is called the pre-state and q' is called the post-state of the transition. For a trajectory $\tau \in T$ and a time point t in the interval $[0, t_1]$, $\tau(t)$ is the state of the system at time t. For a particular variable $x \in X$, the valuation of that variable at $\tau(t)$ is denoted by $\tau(t).x$. A hybrid automaton with no inputs, $U = \emptyset$, is said to be *closed*. Otherwise, it is *open*.

Next we instantiate the above definition to model a specific power network. There are software tools [24, 124, 64] that automatically convert engineering models constructed in different commercial tools like Simulink, LabView, and Mathematica to the hybrid automata. Consider the 3-bus power network shown in Figure 2.4. For the sake of illustration, we consider two different topologies of the network: topology 1 with the line between buses 2 and 3 closed and



(a) Topology 1. The line between bus 2 and 3 is closed. Less power on other two lines. The complex voltage at bus 2 is $0.99\angle -5°$ p.u.



(b) Topology 2. The line between bus 2 and 3 is open. More power on other two lines. The complex voltage of bus 2 is $0.94\angle -17°$ p.u.

Figure 2.4: Two topologies of a 3-Bus power system

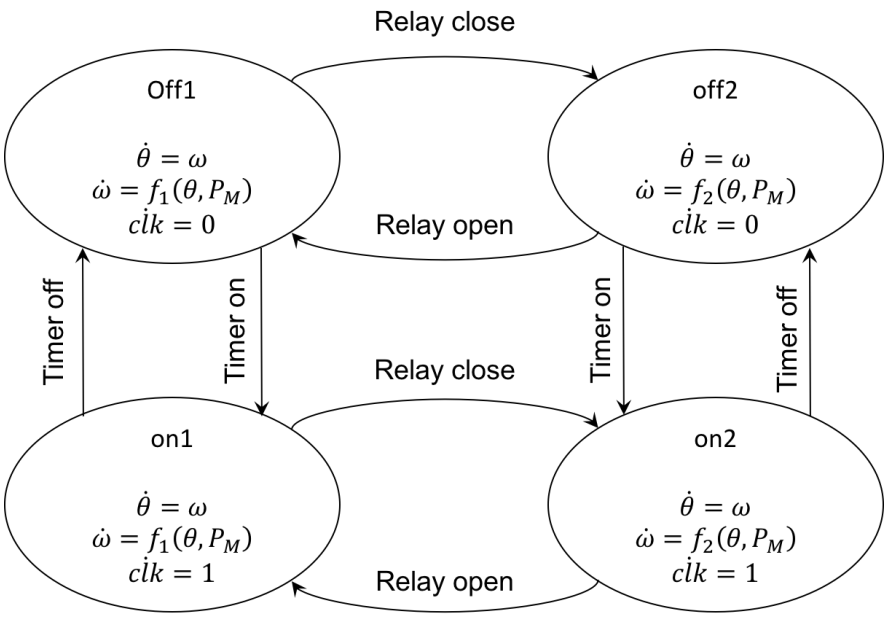


Figure 2.5: Schematic diagram of the HA modeling power network with two topologies. The four modes and their differential equations are shown in the circles. The arrows show the discrete transitions.

topology 2 with the line open. These two topologies and their corresponding steady state values for voltages, phase angles, etc., give rise to two configurations. With each topology $i \in \{1,2\}$, the steady state voltage $|V_{ik}|$ and phase angle δ_{ik} at bus the kth bus are determined by the admittance matrix of the ith topology. With those quantities fixed, the dynamics for the generator at bus k, is given by Equations (2.3)-(2.4):

$$\begin{aligned} \dot{\theta}_k &= \omega_k \\ \dot{\omega}_k &= C_1(P_M - C_2|V_{ik}|\sin(\theta_k - \delta_{ik}) - C_3\omega_k). \end{aligned}$$

For bus 2, P_M is the mechanical power input at the bus and for the other buses this quantity is set to zero. These two sets of differential equations for the two topologies are represented by two functions $[\dot{\theta}, \dot{\omega}] = f_i(\theta, \omega, P_M)$, where $i \in \{1,2\}$.

A schematic representation of the corresponding hybrid automaton is shown in Figure 2.5. Its components are specified as follows: (1) The set of continuous variables X consists of θ_k and ω_k for each of the N generators and a timer variable *clk* for the relay. The discrete *mode* variable can take values from the set $\{\text{on1}, \text{off1}, \text{on2}, \text{off2}\}$; $mode = \text{on1}$ (respectively $mode = \text{off1}$) and

corresponds to the configuration where the topology 1 is active and the relay timer (*clk*) is on (and off). (2) The set of states $Q = \mathbb{R}^{2N} \times \{\text{on1}, \text{off1}, \text{on2}, \text{off2}\}$. (3) The single continuous input variable P_M corresponds to the mechanical power input (4) The mode is switched from offi to oni, with topology $i \in \{1,2\}$, if the current $C_2 \sin(\theta - \delta_i)$ goes above a threshold I_{crit} and switched back otherwise. The switch between the two topology is assumed to be controlled by the attacker (5) The set of trajectories T is defined by the differential equations. Specifically, a trajectory $\tau : [0,T] \to Q$ is a valid trajectory if and only if it is a trajectory for one of the four modes. For example, if it is a trajectory of mode off1 then at each time $t \in [0,T]$ along the trajectory, (i) $\tau(t).mode = \text{off1}$ remains constant, (ii) $\tau(t).clk = \tau(0).clk$, i.e., the relay timer remains constant, and (iii) $\tau(t).\theta$ and $\tau(t).\omega$ are solutions of the differential equation $[\dot{\theta}, \dot{\omega}] = f_1(\theta, \omega)$ for some input signal for the mechanical power P_M. The trajectories for the other modes are defined analogously. For the on1 and on2, at each time $t \in [0,T]$ along any trajectory $\tau(t).clk = \tau(0).clk + t$, i.e., the relay timer measures time elapsed.

4.2 Attack Synthesis Problem

To present the attack synthesis problem precisely, we define the semantics of hybrid models. The semantics of a hybrid automaton model is given in terms of its runs or execution. An *execution* of a hybrid automaton $\mathbf{A}$ is a finite sequence of trajectories $\alpha = \tau_0, \tau_1, ..., \tau_m$, such that each τ_i is a valid trajectory of the automaton. It is a solution of the differential equations corresponding to one of the modes—which in power networks correspond to a particular configuration. For any pair of consecutive trajectories $\tau_i : [0,t_1] \to Q$ and $\tau_{i+1} : [0,t_2] \to Q$ in the sequence α, there must be a valid transition from the last state of τ_i to the first state of τ_{i+1}, i.e., $\tau_i(t_1) \to \tau_{i+1}(0)$. Finally, a sequence of trajectories meeting the above two criterion is deemed an execution only if the first state of $\tau_0(0)$ is one of the initial states of the automaton.

A state $q \in Q$ of $\mathbf{A}$ is said to be *reachable* if there is some execution α that arrives at it. The set of all the reachable states of the automaton $\mathbf{A}$, written as $Reach_{\mathbf{A}}$, plays an important role in analysis and verification of automata. For instance, for some specified set of bad states $U \subseteq Q$, if $Reach_{\mathbf{A}} \cap U = \emptyset$, then it follows that the system is safe with respect to U. If we can find a set of input signals for which $Reach_{\mathbf{A}} \subseteq U$ then it establishes that the system always ends up in a bad state. An invariant for $\mathbf{A}$ is any over-approximation of $Reach_{\mathbf{A}}$. Thus, a fundamental problem

in the analysis of automaton models is to compute or approximate $Reach_{\mathbf{A}}$. In Section 4.3, we give a brief overview of the related work in this area.

A hybrid automaton is non-deterministic in general. It may have multiple initial states, more than one transition may occur from a given state, and even more than one trajectory may emanate with different inputs (for example, the input power P_M) from a given state (and mode). On the one hand, these different sources of non-deterministic choice make the models expressive. They enable us to capture uncertainties in the model parameters, measurements, and the environment. On the other hand, non-determinism makes it difficult to design algorithms for precise approximation of the set of reachable states $Reach_{\mathbf{A}}$.

We state the transient attack synthesis problem using reachability. First, suppose the attacker's input signal is generated as a PID control input as given in Equation (2.5) parameterized by the constants k_p, k_i, and k_d ranging over a parameter space $\mathcal{P}$. This assumption makes sense because (a) PID is the predominant method for generating control signals, and (b) a broad class of continuous functions can be approximated as PID signals. To find a transient attack on the hybrid automaton $\mathbf{A}$, we would like to find PID parameters $\langle k_p, k_i, k_d \rangle \in \mathcal{P}$ such that there exists a time $t \leq T$ when $Reach_{\mathbf{A}}(t) \subseteq Unsafe$ while for all $t \leq T$, $Reach_{\mathbf{A}}(t) \cap Detected = \emptyset$. Here *Unsafe* is the unsafe ansynchronous state ($\theta_k > \pi/2 + \delta_{ik}$), *Detected* is the set of states that can be detected by either the relay or the steady state monitor, and $Reach_{\mathbf{A}}(t)$ is the set of states reached by the system at time t before the time horizon of analysis T.

If the search for the parameters is successful, then the system is guaranteed to enter the bad state *Unsafe* within the time horizon T while evading detection. On the other hand, if the search terminates without finding appropriate parameters, then we can conclude that there is no such attack at least within the parameter space $\mathcal{P}$.

4.3 Algorithm for Attack Synthesis

In this section, we sketch the main idea underlying the automatic attack synthesis algorithm. The key is to parameterize the space of attacks and then search for successful attacks by eliminating parts of the parameter space that cannot yield attacks. This elimination procedure is at the heart of our synthesis algorithm and it combines numerical simulations with formal static analysis of the models. In what follows, we first discuss the synthesis of attacks consisting of

a single attack mode defined by a single configuration of the network and a single set of attack parameters. In Section 5 we discuss how more complex attacks consisting of a sequence of configurations and attack parameters can be synthesized using the same principle.

In the preceding example, for instance, we chose to define the space of parameters $\mathcal{P}$ as the values of the PID gains k_p, k_i, and k_d and also the timing of the topology switches. For a given hybrid model $\mathbf{A}$, let us fix the mode or configuration of the network to be i. Now if we also fix a specific value of these parameters $p \in \mathcal{P}$, arbitrarily, then that defines a unique trajectory $\tau_p \in T$ that satisfies the differential equations of the hybrid automaton.

We can simulate this system to infer if this arbitrary trajectory τ_p, $p \in \mathcal{P}$ yields a successful attack. If it does not, however, we learn very little as there are infinitely many other choices in $\mathcal{P}$. This is where we use sensitivity and simulation-based analysis technique from the formal methods literature that has been quite successful in verifying industrial scale control systems [61, 64]. For a time $t \geq 0$, compact set $P \subseteq \mathcal{P}$ of parameter values, let us denote by $Reach_{\mathbf{A}}(P,t)$ the set of states that are reachable at time t with parameter values in chosen from the set P.

The idea is to generalize from this one choice $p \in \mathcal{P}$ of parameter values, a much larger set of values $P \subseteq \mathcal{P}$ that is also guaranteed have no attack. We can then eliminate P from the search and move on to a different part of $\mathcal{P}$. We formalize this generalization using *discrepancy* used in [63, 97]:

Definition 2. *A function $\mathcal{P}^2 \times \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is the* discrepancy function *of* $\mathbf{A}$ *if*

1. *for any pair of parameter values $p, p' \in \mathcal{P}$:*

$$|\tau_p(t) - \tau_{p'}(t)| \leq \beta(|p - p'|, t), \textit{and} \tag{2.6}$$

2. *for any $t \geq 0$, $\beta(|p - p'|, t)$ converges to 0 as $p \to p'$. Here $|\cdot|$ is the standard ∞-norm of Euclidean space*

Discrepancy function β upper bounds the sensitivity of the trajectories τ_p and $\tau_{p'}$ to the changes in the parameters as a function of time. Several algorithms for computing the discrepancy functions have been presented in literature [63, 61]. By computing (or simulating)

Algorithm 1: Transient attack synthesis.

```
1  input A, β, Unsafe, Detected, T_max, 𝒫;
2  S ← Partition(𝒫);
3  while S ≠ ∅, for ⟨p_i, ε_i⟩ ∈ S do
4      α ← Simulate(A, p_i, T_max);
5      R ← Bloat(α, p_i, ε_i, β);
6      if ∃ t ≤ T_max R(t) ⊆ Unsafe and ∀ t ≤ T_max R(t) ∩ Detected = ∅ then
7          return (Success, p)
8      else if ∀ t ≤ T_max R(t) ∩ U = ∅ or ∃ t ≤ T_max R(t) ⊆ Detected then
9          S ← S \ ⟨p_i, ε_i⟩;
10     else
11         S ← S ∪ Refine(p_i, ε_i);
12     end
13 end
14 return (Fail, ⊥)
```

one of the trajectories $\tau_p(t)$ and enlarging it by the factor $\beta(\varepsilon_1, t)$, we can compute all the trajectories—and the states they reach—resulting from changes in parameters. In summary, discrepancy functions can be used to compute over-approximations of $Reach_{\mathbf{A}}(P,t)$ just from simulations of the system.

The algorithm for attack synthesis using this principle is shown in algorithm 1. The input to the algorithm is the automaton model $\mathbf{A}$, the discrepancy function(s) for the different modes, the unsafe set *Unsafe* (in power networks this is the asynchronous state), the detectable set *Detected*, the time horizon T_{max}, and the set of parameters $\mathcal{P}$. The algorithm maintains a data structure S which is a *cover* of the parameter space $\mathcal{P}$. Concretely, S is a collection of pairs $\{\langle p_i, \varepsilon_i\rangle\}_i$, such that the compact parameter space $\mathcal{P}$ is contained in $\cup_i B(p_i, \varepsilon_i)$, the union of the ε_i-balls[2] around the p_i's. More generally, the cover S could have balls of different sizes.

For each parameter value $\langle p_i, \varepsilon_i\rangle \in S$, the algorithm first simulates the system with parameter p_i to generate one execution α_{p_i} of duration T_{max} (line 4). It then uses the discrepancy functions to compute the set of states R by bloating α (line 5). From the property of discrepancy functions and the argument presented above we know that R is an over-approximation of the set of reachable state $Reach_{\mathbf{A}}(B(p_i, \varepsilon_i), T_{max})$.

Next in line 7 the algorithm checks if there exists a time t at which the reach set $R(t)$

[2]The set $B(x, \varepsilon)$ is the ball of radius ε centered at x, i.e., $\{y \mid |y - x| \leq \varepsilon\}$.

from the parameter range $B(p_i, \varepsilon_i)$ is contained in the unsafe set *Unsafe* and for each $t \leq T_{max}$, $R(t) \cap Detected = \emptyset$. If this condition holds, then the parameter p_i yields a successful attack—one that reaches *Unsafe* while avoiding *Detected*. Having found an attack with p_i then the algorithm terminates. If there is no proof of successful attack from $B(p_i, \varepsilon_i)$ then the algorithm checks (line 9) if (a) for all $t \leq T_{max}$ $R(t)$ is disjoint from *Unsafe*, i.e., it does not make the system unsafe or (b) there is a time $t \leq T_{max}$ at which the system is detected, i.e., $R(t) \subseteq Detected$. In either case, we can infer that none of the parameter values in $B(p_i, \varepsilon_i)$ gives a successful attack, and therefore, it eliminates $\langle p_i, \varepsilon_i \rangle$ from the set S. Finally if none of the above conditions hold, i.e., the algorithm is not able to infer conclusively if the set $B(p_i, \varepsilon_i)$ gives a successful attack or not, then this set is partitioned more finely for future consideration using the *Refine* function. Once all the elements in the set S are removed (in line 9), the algorithm concludes that there is no attack for U possible from the set of parameters that is $\mathcal{P}$.

In the above, we discussed the attack synthesis algorithm using discrepancy functions, and we did not delve into the details of how *Bloat* works with discrete transitions or switches between different topologies and configurations. Our implementation of attack synthesis for this chapter does handle the transitions and it uses hybrid models [64].

4.4 Switched Transient Attack Parameter Space

A switched attack (attack with transitions) consists of a sequence of *attack modes* $A_0, A_1, \ldots, A_{n-1}$ and a sequence of time points $t_0, t_1, \ldots, t_n$. Attack A_i is active over the time interval $[t_i, t_{i+1})$. Each attack mode A_i records, as before, a network topology and the mechanical input signal P_M over the interval. Thus, synthesizing a switched attack of length n is the same as identifying the n topologies, the switching times, and the parameters defining the input signal (P_M) in each of the attack modes. In each attack mode i, the input signal P_M can either be some constant $(constPM_i)$ or follow a PID law as before:

$$P_M(t) = kp_i e(t) + ki_i \int_0^t e(s)ds - kd_i \omega(t), \tag{2.7}$$

where $e(t)$ is the difference between the reference phase angle and current phase angle. In summary, a program for executing the switched attack would look like the pseudocode in Figure 2

Algorithm 2: template of attack mode A_i

```
if Time ∈ [t_i, t_{i+1}] then
    Topology ← top_i;
    if usePID_i = 1 then
        P_M ← PID(kp_i, ki_i, kd_i, State);
    else
        P_M ← constPM_i;
    end
end
```

with all the parameters are synthesized from our algorithm.

Otherwise, the mechanical input P_M is set to a constant $constPM_i$. The attack synthesis problem is reduced to assigning proper values to parameters t_i, top_i, $usePID_i$, kp_i, ki_i, kd_i, and $constPM_i$.

To synthesize all the parameters in an attack sequence, we use Algorithm 1 as a subroutine. Notice that the there are only a finite (typically small) number of choices for the topology top_i depending on the number of relays and lines the adversary can have access to. Thus, we use a brute force search for all possible choices of top_i. Then for each sequence of choices for the topologies, we run Algorithm 1 to check if there exists a valuation of the other parameters that produce an undetectable attack.

5 Evaluations

In Section 3.2, we discussed two attacks by which the mechanical input (P_M) to one of the generators in the network were controlled to make the system unstable and we showed how these attacks are detected by the conventional protection mechanisms. Now, equipped with the attack synthesis algorithm of Section 4.3, we present synthesized attacks that evade detection with relays and steady state monitors. In addition to the mechanical input, now we endow the attacker with limited capability in changing the topology of the network by opening or closing one of the lines in the network.

5.1 Application to 3-Bus Power Network

As the first illustration of a synthesized transient attack that is undetected by the relays and the steady state monitors, we revisit the 3 bus System of Figure 2.4. We consider the scenario where the attacker can control the mechanical input to the generator at bus 2 and also switch the topology between 1 and 2. The machine is equipped with a relay and a steady state monitor. The critical current of the relay is $I_{crit} = 3.5$ p.u. and the over-current duration is $c_{oc} = 175$ ms. The steady state monitor uses a reference power $P_{ref} = 250$ MW with tolerance $P_{tol} = 20$ MW. The mechanical input has to take values in $[0, 700]$ MW.

The attack can be made arbitrarily complicated by having more attack modes. Here, we study switched attacks with two attack modes. Our experiments show that the class of attack is rich enough for finding valid transient attacks on the power system.

One such attack is shown in Figure 2.6a. A constant input of $P_M = 700$ MW for 154 ms is applied to topology 1 in the first attack mode, and then in the second attack mode, P_M is generated as a PID input with parameters $k_p = 6, k_i = 0.1, k_d = 0.2$ for topology 2. The resulting transient behaviors of the phase angle and the output power are illustrated in Figure 2.6b. The asynchronous states *Unsafe* are those with phase angle greater than the critical angle 72°. It is clear that the attack drives the system to *Unsafe*.

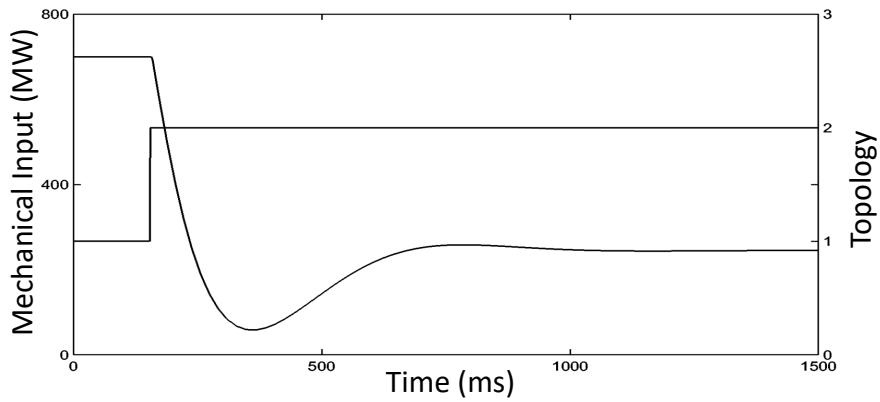
Since the output power is derived by multiplying the output current by the bus voltage $|V_i|$, the relay will be on if the output power is above $|V_i| \cdot I_{crit}$. The thresholds for both topologies are plotted as red dashed line segments in Figure 2.6b. We observe that the relay is on for 167 ms. Thus the attack is not detected by the relay. Moreover, after around 600 ms, the output power stabilized within the range $[230, 270]$ MW, such that the attack is not detected by the steady state monitor.

5.2 Attack Generalization

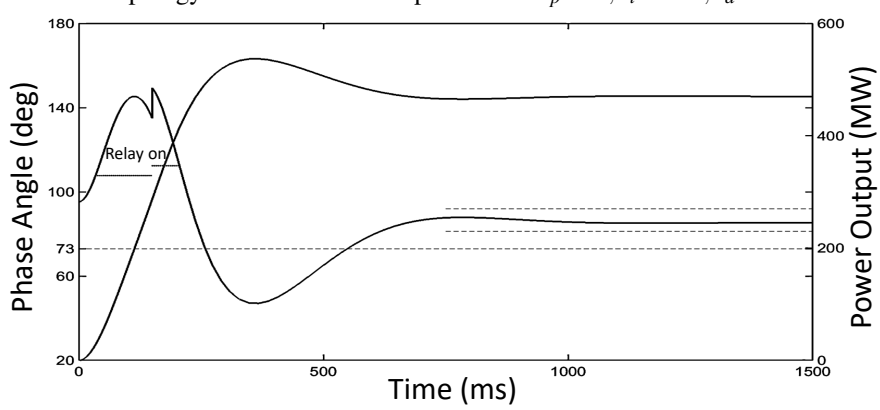
We applied the same analysis to a couple of different power systems including the Western system coordinating council (WSCC) 9-bus power system[3].

The relay parameters are $I_{crit} = 2.25$ p.u. and $c_{oc} = 175$ ms. $P_{ref} = 163$ MW, $P_{tol} = 20$

[3] Available at http://publish.illinois.edu/smartergrid/wscc-9-bus-system/

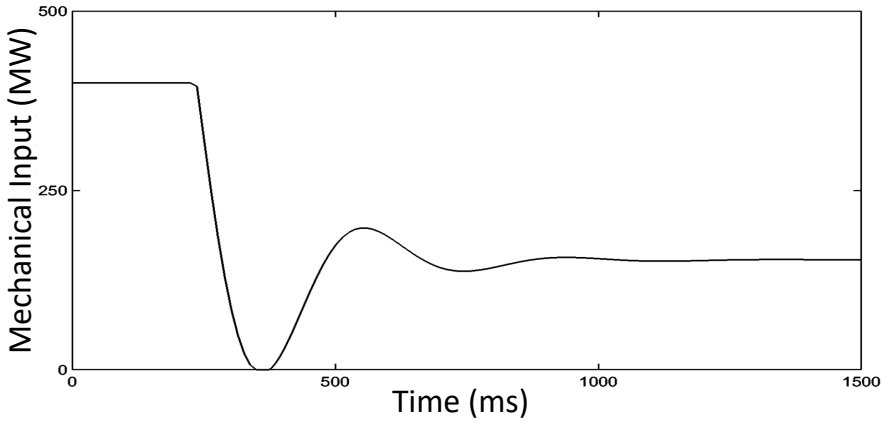


(a) First apply a constant input power 700 MW on topology 1 for 154 ms and then switch to topology 2 with PID control parameters $k_p = 6, k_i = 0.1, k_d = 0.2$.

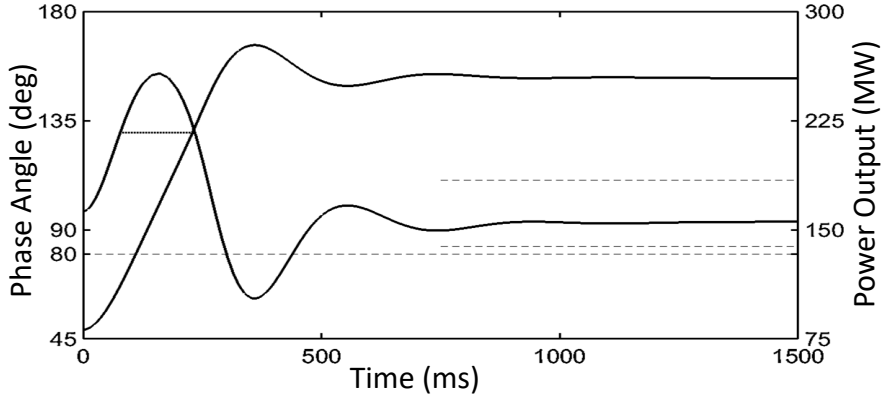


(b) The phase angle converges to 145° and the power output converges to 250MW. The power output has a discrete jump at time 154ms because of the topology change.

Figure 2.6: An attack to a 3 bus system. The critical angle of machine 2 is 72°.



(a) First apply a constant input 400 MW for 154 ms. Then switched to a PID controller with $k_p = 7, k_i = 1.2, k_d = 0$.



(b) The phase angle converges to 152° and the power output converges to 163 MW.

Figure 2.7: An attack to a 9-bus system

MW. We successfully synthesize an attack as illustrated in Figure 2.7. The input mechanical power is plotted in Figure 2.7a: In the first 237 ms, the mechanical input is set to 400 MW, and afterwards the attacker use a PID control with $k_p = 7, k_i = 1.2, k_d = 0$. The resulting phase angle and power out are illustrated in Figure 2.7b, where the relay is on for 159 ms and the power output converges to 163 MW.

The experiments suggest that the proposed algorithm can be applied to general power networks and produce sound attacks. Even though there is a dramatic difference between the work topologies, power flow conditions, sensor parameters and input constraints of the 3 bus and 9 bus systems, the attacks follow the same pattern. That is, first attack mode applies a large mechanical power and the second mode is a PID control law. This pattern makes intuitive sense. A maximum mechanical input drives the phase angle to *Unsafe* with minimized duration of the relay on. Then a PID law guarantees the output power to stabilize without triggering the steady state monitor. This observation suggests that by applying our algorithm we could recognize attack patterns to the system.

5.3 Robustness of Attacks

So far our attack synthesis approach assumed perfect knowledge about the model of the system. In reality, however, detailed models may not be available to the attacker and the network parameters like loads, line admittances, and generator inputs fluctuate over time. Since our synthesis approach not only computes a single attack (with a single set of parameters) but it computes a set of parameter values $B(p_i, \varepsilon_i)$ that produce a successful attack, we expect that these attacks enjoy some degree of robustness. That is, even if the network parameters change to some degree, the same attack will continue to produce similar qualitative results, i.e., evade detection and drive the system to an unsafe asynchronous state. In this section, we put this hypothesis to test by perturbing several of the network parameters and subjecting it to the same attack as the one synthesized in Section 5.1.

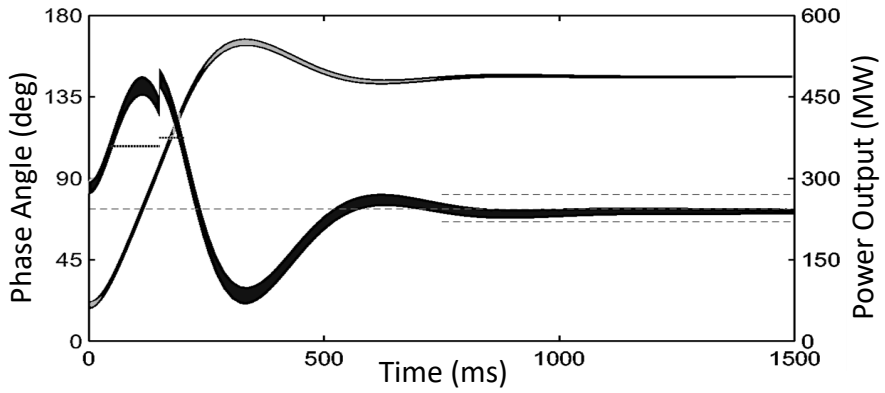
As we discussed in Section 3, the change in the load parameters and the admittance matrix results in different bus voltages and phase angles. These are obtained by solving the power flow equations (2.1)-(2.2). The bus voltages and phase angles directly affect the dynamics of the generator (2.4).

For example, consider the same system topology as illustrated in Figure 2.4b. If we perturb the power demand at bus 2 from 500 MW to 550 MW and the line admittances from 5.5-j18.3 p.u. to 5-15j p.u., then the voltage of bus 2 will change from $0.99\angle -5^\circ$ p.u. to $0.98\angle -5.8^\circ$ p.u., which alters the differential equation of machine 2.

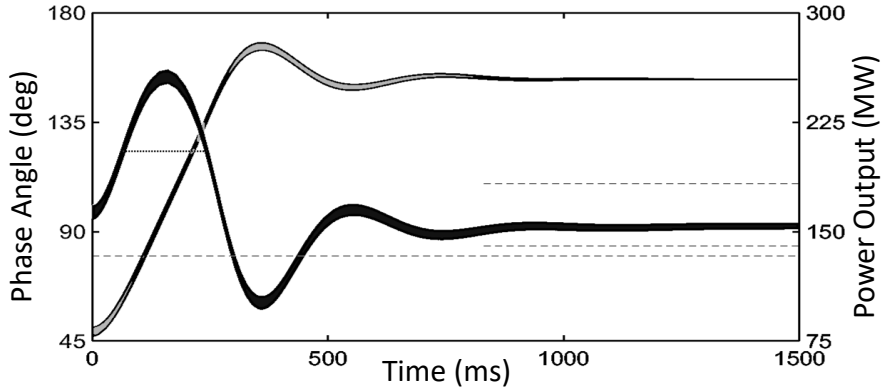
In this section, we allow the voltage of bus 2 has a ± 0.2 p.u. perturbation, phase angle has a $\pm 1^\circ$ perturbation. With this level of uncertainty in the parameters, we compute the reachable states of the system under the same attack. The computation uses a similar reachability algorithm as presented in Section 4.3 and the result is that the reach set satisfies the same properties as a successful attack: it avoids *Detected* and eventually reaches the *Unsafe* set. The reach set of the phase angle and power output of generator 2 are plotted in Figure 2.8a. We observed that the set of the reachable phase angle is eventually contained by the *Unsafe* (72° or above). By examining the reachable states, we observe that the relay is on for at most 172 ms and the power output converges to 250 MW in all cases. That is, the reach set does not intersect with the detectable states *Detected*. A similar robustness property is proved for the attack to the 9 bus system as shown in Figure 2.8b. These experimental results suggest that the attacker can synthesize an attack without knowing the precise model of the system.

6 Transient Attacks vs. NERC CIP-Compliance

Following the attacks synthesized automatically in the previous section, we implemented transient attacks against a realistic version of the 9-bus system with NERC-CIP mandated protection in the PowerWorld simulator [148]. PowerWorld is the de facto simulator for power system operation and control. Its models are much more detailed, higher dimensional, and complicated with many more parameters than the ones used in models of Section 3.1. For example, these models allow the modeler to choose various IEEE standard wiring configurations for the stators and rotors in the electrical machines. Our goal for this section is to study the mechanics of the progression of transient attacks, despite the NERC-CIP protections, in these realistic models. In the future, we hope to undertake the considerably harder problem of automatically synthesizing attacks for PowerWorld models.



(a) Robustness of attack to 3 bus system.



(b) Robustness of attack to 9 bus system

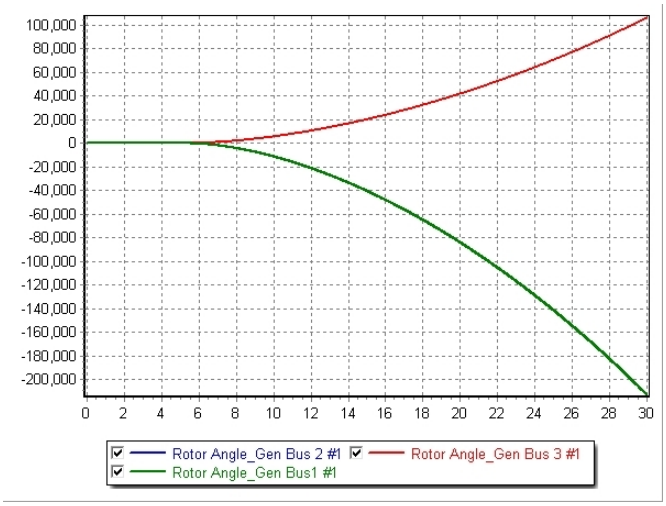
Figure 2.8: Robustness of the attack. The bus voltage and phase angle have a ± 0.2 and $\pm 1°$ uncertainty respectively. The reachable states of the phase angle and output power with the same attack. The reachable states do not intersect with *Detected* and are contained in *Unsafe* eventually.

The attacks caused global instability against power systems that comply with the NERC-CIP N-1 contingency requirements. PowerWorld reported no violation as the result of NERC-CIP steady-state contingency that includes transmission line outages, transformer outages, and the power generator outages. The transient attack involved a sequence of discrete switches (transitions) changing the network topology, which in turn altered the system's continuous dynamics.

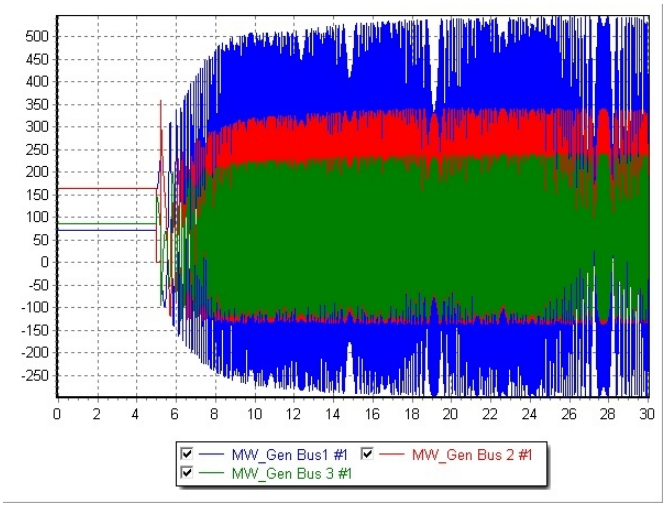
Each successful attack is a sequence of relay switches changing the topology of the power network. One attack, for example, starts at second 5 and opens the breaker on the bus 2, and waits for 1.18 seconds before re-closing it. Figure 2.9a shows the generators' rotor angles that fall out of step once the line gets re-closed. Intuitively speaking, the attack is successful because of how the generators operate. While the relay is open, the generator on bus 2 is isolated, that is, its output electrical power becomes zero. This is while its input mechanical power from the governor steam valve stays the same as before and is positive. This turns the power generator into an energy buffer, where its input mechanical power is a constant positive value while its output electrical power is zero. The buffered energy gets stored by the generator through its accelerated rotor, and hence its rotor's angular velocity increases. The rotor's angular velocity is directly related to the generator's generated electrical power frequency if it is connected to the rest of the grid (when its electrical power is not zero).

When the attacker connects the generator back to the grid by re-closing the line, its accelerated rotor will cause a frequency disturbance to the rest of the grid. Other adjacent generators attempt[4] to absorb the target generator's high frequency by increasing their rotor's angular velocity. Given the power system's transient swing equations [20], there is a limit to every generator's rotor angle beyond which the generator cannot re-stabilize itself whatsoever, which is the critical angle introduced in Section 3.1. In the attack above, the wait caused the generator on bus 2 to gain so much acceleration and angular velocity that it was not able to recover the system stability by absorbing its increased frequency while at the same time not triggering the over-current relays upon reconnection. Figure 2.9b shows the electrical power output of the generators. As the graph shows, the generators' output power oscillate out of control due to

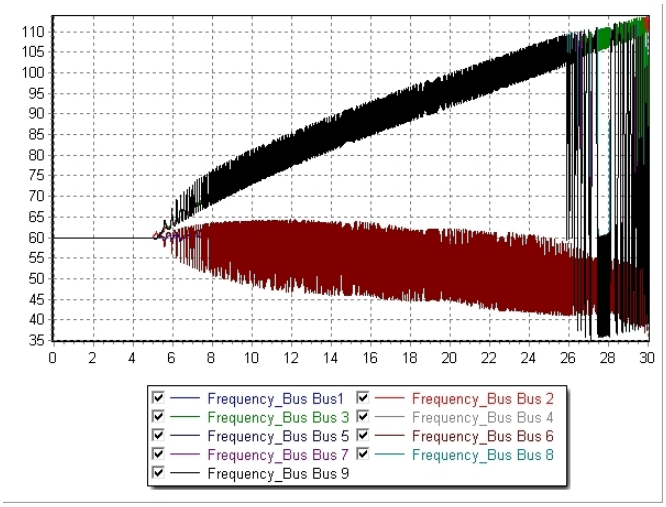
[4]This is due to the physics of the system and there is no explicit controller involved.



(a) Generator rotor angle falls out of step (destabilizes).



(b) Generator's output electrical power becomes unstable.



(c) Bus frequency cannot regain stability.

Figure 2.9: Successful Transient Attack Analysis against a Power Generator

the system's raised frequency. The system is unable to stabilize itself and the grid experiences a global collapse leading to a blackout. Figure 2.9c shows the system frequency for individual power generators, and how the system loses its synchrony (as discussed in Section 2.2) as the result of subsequent discrete state transitions that even return to its original continuous dynamics.

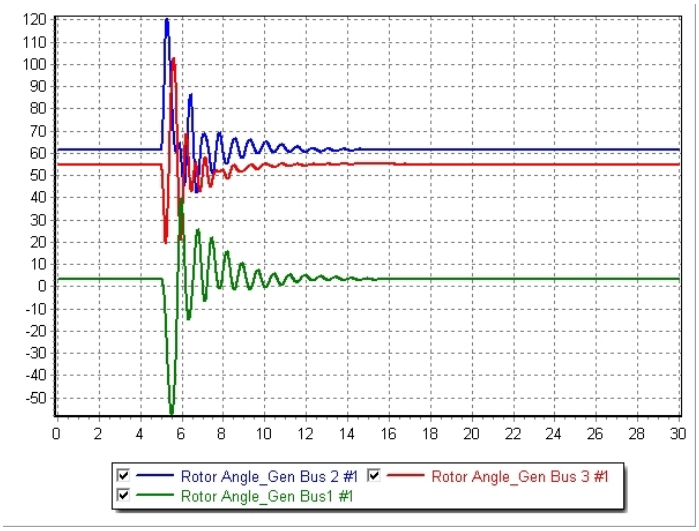
In Section 5.3, we discussed that synthesized attacks have a certain degree of robustness with respect to system parameters due to the continuity of the dynamics. Next, we show with a typical example that how there are limits to this robustness. For example, if we change the timing of the above attack sequence so that then the relay on bus 2 closes after 1.17 seconds of waiting (0.01 seconds earlier) then the critical angle is reached and the attack is unsuccessful. First, in Figure 2.10a we show the evolution of the rotor angles for the power system generators under this benign sequence of switches (unsuccessful attack). The power system stabilizes and converges a safe steady-state value. Figure 2.10b and Figure 2.10c show the electrical power outputs for the power generators (MW) and the transient frequency dynamics of the power system buses, both stabilizing approximately 11 seconds after the switches.

Practical implications. In summary, these experiments suggest that switched transient attacks that evade NERC-CIP protection are indeed feasible in realistic power systems, and therefore, transient attacks and their protection should be considered by the governmental regulatory agencies, power system operators, and security researchers in the field. Secondly, it is possible to quantitative estimate robustness of synthesized attacks with respect to different network and attack parameters, and these measures may be used as metrics for evaluating the security of power networks.

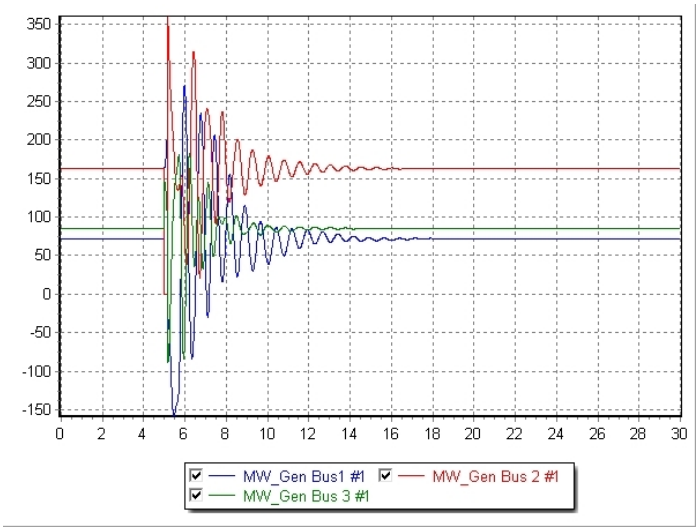
7 Related Work

Since the past real-world critical infrastructure attacks, there has been an increasing number of security protection solutions proposed. We review the most related work.

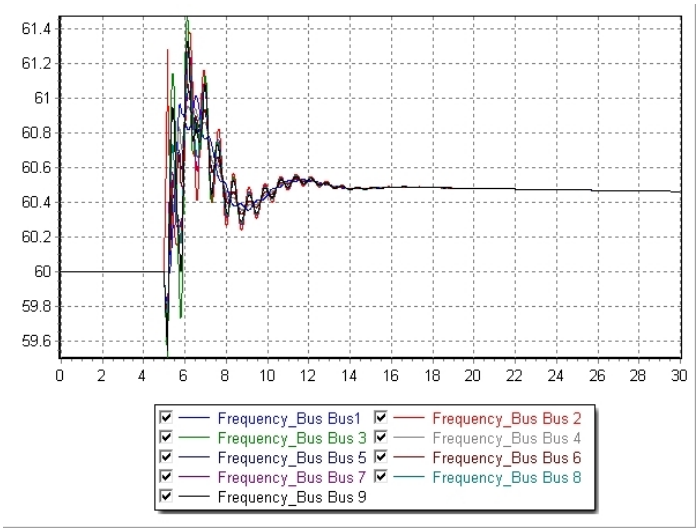
As a fundamental power system monitoring tool, state estimation is the process of fitting power sensor data to a system model and determining the current system state , e.g., using weighted least squares [174]. The estimated state is then used in stability analysis [85] through



(a) Generator rotor angle stabilizes safely after a sequence of relay switches.



(b) Generator's output electrical power stabilizes.



(c) Power bus frequency stabilizes safely.

Figure 2.10: Unsuccessful Transient Attack Analysis against a Power Generator

solving nonlinear AC or linear DC power flow equations for a series of "what if" scenarios, or contingency analysis [209, 85] that investigate the potential power system state in the case of an event, e.g., a generator outage. Almost all the current solutions, e.g., contingency analyses, do not consider the cyber-side controllers and/or take into account adversarial settings, and hence those solutions miss maliciously induced topological errors in modern cyber-physical infrastructures. Additionally, power system stability analysis concentrates on continuous dynamics only, and does not fully consider the possibility of subsequent discrete logic events in the system.

Recently, cyber security solutions have been proposed to harden critical infrastructures. These include practical best-effort techniques such as regulatory compliance such as attack tree analysis , NIST guidelines [187], and perimeter protection recommendations [117]. These approaches have been confirmed to be insufficient by the past major security incidents [6], and recently discovered fundamental security flaws in power grid control devices [204] and popular human machine interfaces (HMIs) from major vendors. From an adversarial viewpoint, the past cyber attacks are mostly not physics-aware and do not complete the attack path by sending malicious control inputs to the underlying physical plant components. The very few real-world security incidents with physical impact [72], however, use manually crafted malicious control parameters such as setting them to an unsafe high value like in Stuxnet. Those trivial strategies are to be addressed by NERC-CIP regulations [145] that mandate local safety measure deployment to protect unsafe component operational points.

One specific related line of research is false data injection (FDI) attacks [122] that have been explored over the past few years. FDI assumes compromised set of sensors and make them send corrupted measurements to electricity grid control centers to mislead the state estimation procedures. The authors propose a system *observability* [122] analysis to determine the required minimal subset of compromised sensors to evade the electricity grid's bad data detection algorithms [123]. FDI's focus domain differs completely from our objective. Rather than sensor measurement corruption, we concentrate on malicious control inputs to the plant and by performing formal *controllability* analysis of the electricity grid for attack feasibility assessment. Additionally, [122] leverage a linear DC model of the grid for designing attacks, and hence ignores all the system non-linearities. The attack also ignores the discrete logic-based

incidents, such as relay openings/closings that occur frequently in the grid. Using steady-state models, FDI [122] cannot evade NERC-CIP compliant protection schemes.

Automatic verification and specifically reachability analysis of hybrid models has enjoyed sustained attention from researchers in computer science and control theory for over three decades. The latest generation of tools from this research include Flow* [48], C2E2 [64], and Breach [61] that can approximate bounded time reach sets of nonlinear hybrid models. We refer the interested reader to the proceedings of the hybrid systems conference for recent developments [65]. The approach used in this chapter in the transient attack synthesis algorithm is comparable to the analytical approaches used in Breach and C2E2 that use static analysis methods for computing sensitivity or discrepancy measures of the model.

Algorithmic control synthesis, in contrast, is still in its infancy (see, for example, [91, 55]). Inductive synthesis is considered in [98, 108]. For the synthesis of a provably correct controller, the community adopts reachability-based approach [62, 59]. Particularly, a reachability-based approach similar to ours for parameter synthesis has been presented in [62].

8 Conclusions and Mitigations

In this chapter, we presented an automated cyber-physical attack synthesis algorithm. Unlike previous work on electricity grid security analysis, the algorithm makes a complete use of both discrete and continuous dynamics of the system simultaneously. Through its formal hybrid system analyses, the algorithm demonstrates that the most recent governmental electricity grid cybersecurity NERC-CIP requirements can fall short in protecting our national grid against malicious adversarial parties. Our experimental results show that the use of non-steady-state system transient dynamics enables attackers design recipes that are not feasible using existing state-of-the-art attacks, which NERC-CIP is designed to prevent. The solution to mitigate the proposed transient attacks would be: *(i)* NERC-CIP should have transient-aware contingency analysis enforcement along with the N-1 steady-state contingencies; *(ii)* more fine-grained monitoring of system transients using more advanced sensors such as phasor measurement units (PMU) is required; *(iii)* using algorithmic attack tool as a contingency analysis technique would guarantee the system security against transient attacks.

Chapter 3

JUST-AHEAD-OF-TIME CONTROLLER RECOVERY

1 Introduction

The use of unmanned aerial vehicles (UAVs) has been increasing in many mission-critical settings such as surveillance, delivery systems and military applications [25]. Consequently, they are becoming attractive targets for malicious penetrations leading to physical damage. Recent attacks took control of the drones remotely [161, 110, 169] and hacked RQ-170 Sentinel built by Lockheed Martin [170]. AnonSec team [3] obtained partial control over the global hawk used by NASA in 2016.

Secure operation of next-generation cyber-physical systems, specifically unmanned aerial vehicles will require effective *scalable formal verification* capabilities. Current static formal verification methods (e.g., TSV [128] and HACMS [149]) analyze the system model in an offline manner, often facing state space explosion, and hence do not scale up to large-scale cyber-physical systems. On the other hand, existing dynamic execution monitoring solutions (e.g., Avatar [221]) notify operators about incidents that have just occurred or are about to occur, and hence do not leave enough of a time buffer for effective manual or automated response and recovery.

To take best of both worlds, we present Crystal that leverages *Just-Ahead-of-Time (JAT)* verification (Figure 3.1). Crystal stays ahead of the actual system state by an arbitrary time buffer. Crystal is a nonintrusive formal verifier for drone controller programs. It speculatively checks in real-time whether the program execution may drive the drone towards any unsafe state. Drone's get there controller program inputs from the sensors, e.g., inertial measurement unit (IMU) or GPS and sent the output commands to drone's actuators, e.g., propeller motors. Crystal executes the control logic symbolically and calculates possible input-to-output mapping (sensor measurements to actuation commands) of the controller program. Crystal closes the

loop on the physical channel by calculating how sensor measurements are determined based on the previous actuation commands and physical drone's dynamic evolution (i.e., actuation commands to sensor measurements). Crystal estimates drone's state (e.g., location and orientation) and deploys a new symbolic execution of the drone's physical dynamics to calculate its input-to-output mappings.

Given the flight control unit and physical system coupling, Crystal *pipes* its findings of IO mappings of the cyber controller and physical dynamics together to create a full closed-loop model of the cyber-physical system. The model captures all the interdependencies between cyber and physical components and is used for Crystal's just-ahead-of-time formal verification. Crystal does not generate the complete system model one-time due to its (very) large size and instead relies on local exploration and model checking of the upcoming future symbolic states up to a finite horizon based on the current system state. This enables Crystal to stay just a few steps ahead of native execution and avoids exhaustively considering all states. Crystal periodically synchronizes the model exploration process with the native execution through communication with the drone's processor and obtaining concrete values of flight controller program variables. The synchronization step allows Crystal to refine the model exploration and not explore the states that will definitely not be reached through the native execution. The model generation process continues to stay ahead of native execution and explore's the states that have not yet been reached.

Crystal inspects each upcoming state and checks its symbolic variable values to determine whether the state *could* be unsafe under a specific concretization. If the drone's native execution is about to enter the unsafe state according to its upcoming concrete input values Crystal will notify the operator about potentially upcoming unsafe state's and requests for recommended recovery response. Crystal expects to receive the operator's response before the native execution catches up. This waiting time denotes how far Crystal's JAT exploration leads the native execution and could be adjusted initially. Crystal deploys the recommended countermeasure if the unsafe state is actually realized, and caches it to not involve the operator again for future similar situations.

The contributions of this chapter are as follows:

- We present a scalable formal verification technique, just-ahead-of-time verification, for

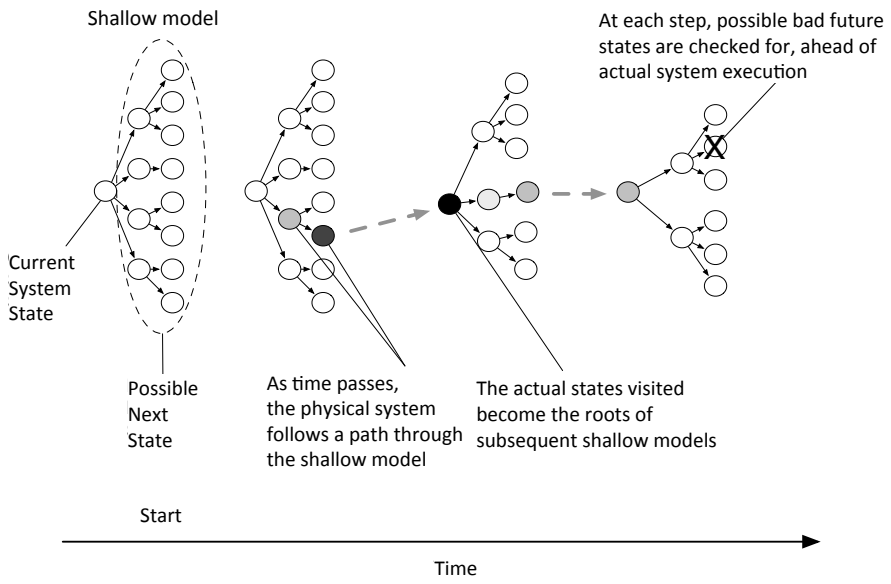


Figure 3.1: Just-Ahead-Of-Time Verification

complex UAV platforms that eliminates state explosion problem and leaves time for the operator to select an appropriate countermeasure strategy.

- We present a cyber-physical symbolic execution framework that leverages programming language analysis and enhanced drone state estimation techniques to create cyber-physical models for formal verification purposes.

- We present a predictive hybrid model with data-driven and system knowledge model of the drone's physical dynamics using a neural network and extended Kalman filter (EKF) that takes actuation commands (flight controller outputs) as inputs and outputs the predicted sensor data (controller's next input).

This chapter is organized as follows. Section 2 gives an introduction to drones dynamics and a previous related work [128]. Section 3 lays out our assumed threat model and gives an overview of CRYSTAL's architecture. Section 4 explains the predictive model based on a hybrid approach of a neural network and EKF model to formulate the drone physical dynamics. Section 5 describes how the drone's controller code and physical dynamical formulations are integrated into a unified cyber-physical model. Section 6 explains the real-time formal verification and recovery using the cyber-physical models. Section 7 describes our prototype implementation and evaluation results. Section 8 covers the related work, and Section 9 concludes

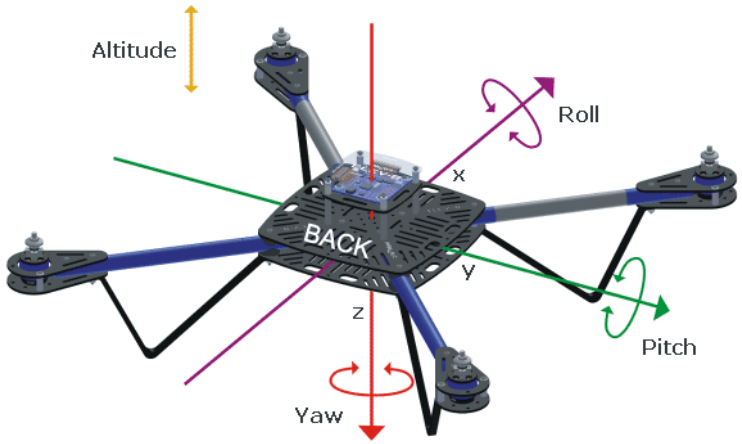


Figure 3.2: Drone's pitch, roll, and yaw

the chapter.

2 Background

2.1 Drone Flight Dynamics

Since the motion of UAV is in three-dimensional space, it can be controlled along three axes. The altitude of the drone is proportional to and controlled by the thrust produced by the propellers from the four motors. For instance, the thrust on all the motors should be the same to move the drone just in the z-axis. In order for the drone to hover (stay at a fixed location in air), all the motors should produce a thrust to neutralize gravity. To balance the rotational torque produced by the motors and to increase the stability of the drone, motors on the opposite direction rotate in the clockwise direction and the ones adjacent to these rotate in the counter-clockwise direction. The rotation of the drone along the x-axis is called *roll*, along the y-axis is called *pitch*, and along the z-axis is called *yaw* (Figure 3.2).

The drone's left-right motion can be controlled by changing its roll. If the drone has to move towards left (right), the thrust on the right (left) motors is increased. Similarly, the drone's front-back motion can be controlled by changing the pitch and changing the thrust on front or back motors. Drone's direction along the z-axis is controlled by changing the yaw. If the drone has to rotate clockwise, the thrust on the motors rotating counterclockwise has to be reduced. The aforementioned parameters allow defining the drone's state notion as a six-entry vector of its

location (x,y,z), and orientation (roll, pitch, yaw). Given the drone's current state and the latest issued actuation commands to the motors, the drone's next state can be calculated based on its physical dynamic models. In this chapter, we used a combination of extended Kalman filter and data-driven model of the drone's physical dynamics to predict its future states.

2.2 Offline Controller Code Verification

Due to model error and random disturbances, drones obtain the aid of flight control unit to constantly monitor and regulate the flight operations. The controller repeatedly takes measurements of the process state and feeds these to its software *control logic* to determine what changes need to be made to keep the process on course. This sense-execute-actuate loop is called the *scan cycle* and may occur many times per second.

In most commercial-grade flight control units, the control logic can be modified remotely, exposing the threat of malicious logic injection. Authentication required for modifying flight control logic is often weak or does not exist, e.g., no authentication for Bitcraze [33]. Hard-coded backdoors are a common industry practice [165]. Modification of a flight control logic grants the attacker complete control of the physical drone operation. As demonstrated by the well-known Stuxnet attack [72], the malicious control logic can also forge sensor data reported to human operators, thus hiding the first signals of malicious behavior.

We review the most related recent work on offline controller code verification [128]. The trusted safety verifier (TSV) [128] is interposed between controllers (i.e., programmable logic controllers - PLCs) and the control network (i.e., supervisory control and data acquisition - SCADA). Any controller-bound code must be formally verified against a set of safety properties. The safety properties are stated in linear temporal logic (LTL) that allows for the description of temporal properties such as causal relationships, and guarantees of eventual progress [83]. An LTL property contains a set of atomic propositions that are non-temporal property typically stated in propositional logic, e.g., $\texttt{landed} = (\texttt{altitude} = 0) \wedge (\texttt{drone\_velocity} = 0)$. The LTL property then combines these atomic propositions using temporal connectors to describe relationships over time, e.g., $\texttt{safe\_landing} = \neg\texttt{motors\_off}$ **until** $\texttt{landed}$.

Cyber-physical system controllers typically follow synchronous execution paradigm as a sequence of fixed-time sense-process-actuate epochs called *scan cycles*. TSV performs a mixed

concrete and symbolic execution of an ILIL program to produce a *symbolic scan cycle.* A symbolic scan cycle represents every possible execution of a single scan cycle of the controller program. It is a mapping from path constraints over controller input variables (sensor measurements) to symbolic values for PLC output variables (actuation commands). For a single entry in the mapping, if the input variables satisfy the path constraint, then the output variables will have the corresponding symbolic value.

To model the PLC execution's subsequent scan cycles, TSV combines successive symbolic scan cycles into the Temporal Execution Graph (TEG). The TEG is a tree that represents a nondeterministic execution of a controller program for some fixed number of scan cycles. For example, a TEG of depth three would represent all possible executions of the program for three successive scan cycles. The TEG is later checked against the safety requirements to determine if the controller code could ever produce unsafe outputs.

2.3 Limitation of Existing Solutions

By design, offline verification solutions [128, 155, 112] should complete before the code is allowed to run on the controller. Due to the exhaustive exploration of *all* possible states and too many possibilities, existing formal solutions face state-space explosion and do not scale up to real-world complex cyber-physical systems [128].

Additionally, TSV's analysis of subsequence scan cycles consider all controller inputs (sensor measurements) as free variables that can take on *any* value. This results in too-pessimistic outcomes (i.e., the code rarely satisfies all safety properties) and deteriorates the solution scalability. The physical dynamics of the drone determines the sensor values given its recent actuation commands. For instance, the actuation command "increase the propellers rotation speed" results in an increase in altitude sensor value. As discussed later, the models of the physical dynamics along with the controller code models can be used to predict and investigate the drone's future behavior.

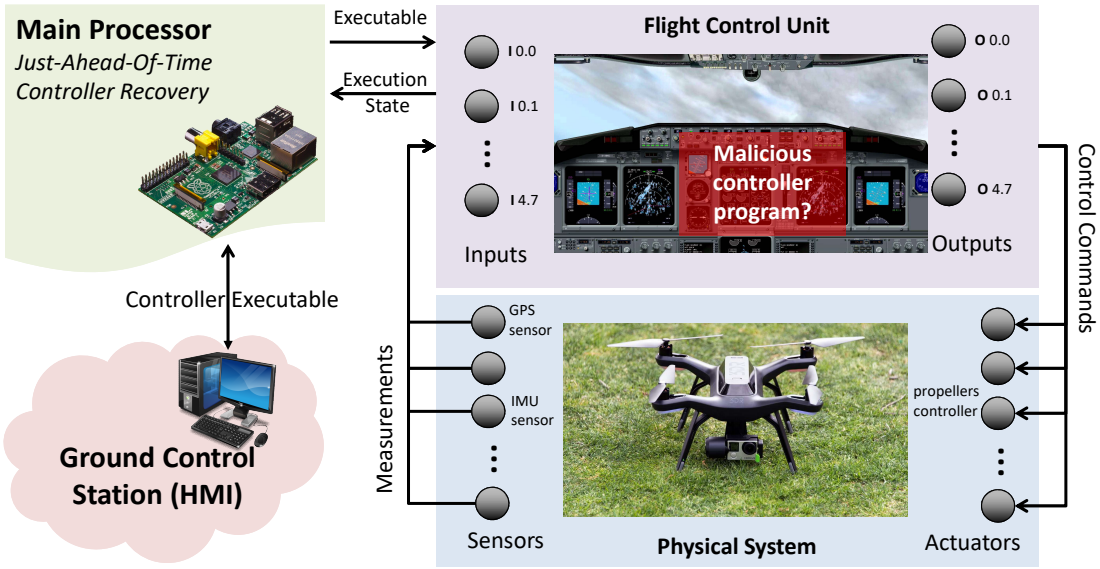


Figure 3.3: Crystal's High-Level Architecture

3 Overview

3.1 Threat Model

The threat to the UAVs can be from one or more of the following attack vectors [139]: physical (attack on sensors, actuators), cyber (firmware, controller software, guidance and navigation algorithms) and communication (radio link to ground control station). One of the most prominent causes for recent drone security failures is remote cyber attacks [161, 110] that Crystal aims to protect against. We assume that the underlying software stack (e.g., operating system or firmware) and hardware are trusted, while the control logic, guidance and navigation algorithms on the drone's flight control processor can be malicious. By ahead-of-time verifying that control logic will not violate safety properties, Crystal protects against arbitrary control logic injection on the controller. Crystal does not defend against physical attacks since they are arbitrary. Crystal does not defend against sensor channel attacks, where sensor data is forged [122, 179]. In such a case, the control logic may behave exactly as intended, but on false sensor data. Such attacks are outside the scope of this chapter and must be addressed by improved state estimation techniques [122].

3.2 Crystal Architecture

Figure 5.2 shows Crystal's context and its interactions with other control assets. Drones typically follow a dual processor architecture. The flight control unit processor takes care of the real-time sense-process-actuate executions; it receives sensor measurements, runs its control logic, and sends the output values to the motors. The main processor interacts with other peripheral devices, e.g., to communicate with the ground control station. All the control logic updates to flight control unit from the external world go through the main processor. Crystal runs on the main processor to monitor each control logic update and its execution on the flight control unit processor.

3.3 Safety Requirement Definition

Before Crystal's setup, the drone operator needs to define a set of high-level safety properties for the underlying physical dynamics. The safety requirements do not have to correspond to specific actuator outputs and could be defined for global system parameters. For instance, if the controller has to maintain the propeller speeds around a fixed set-point, the safety requirements can be defined to limit the whole drone's acceleration. Crystal's consideration of the physical system dynamics enables automated correlation of global system parameters to individual actuation commands.

Previous protection solutions (e.g., TSV [128]) fully ignore the physical dynamics of the underlying system, and hence pose strong requirements for the operators: *i)* flight dynamics expertise: all safety requirements should be defined for only the target flight control unit's output values; hence, in the example above, the operators have to analyze the flight dynamics to determine what flight control unit's output values (propeller set-points) would cause unacceptable accelerations of the quadcopter; *ii)* tedious human involvement: the operators should redefine/update the safety requirements every time the parameters of the components change due to external environmental factors, which occurs often in practice.

3.4 Predictive Flight Modeling

Crystal enables hybrid cyber-physical symbolic execution by complementing flight control code analysis [128] through its predictive modeling of the underlying flight physical dynamics. The drone's physical state (i.e., location and orientation) is estimated and predicted using a hybrid approach of neural network and EKF that is already trained and configured to formulate the flight dynamics. The predictive trained hybrid model will estimate the values of the sensors in the future given the upcoming actuation commands. The upcoming actuation commands are calculated by Crystal's analysis of the drone's controller code. During the drone's operation, the hybrid predictive model adaptively corrects itself using a sliding window on the most recent sensor values. Later in the chapter, we extend to multiple parallel actuation commands (e.g., power control of the drone's four propeller motors), where the actuation is modeled as a symbolic vector (Section 4).

3.5 Just-Ahead-of-Time Verification

Crystal is deployed on the drone's main processor and intercepts every control logic updates bound to the flight control unit processor. Crystal disassembles the binary and starts the dynamic conversion of the resulting source to its corresponding finite state automaton. During the conversion (model exploration), Crystal implements on-the-fly formal model checking in parallel to ensure that the reachable states are safe and do not violate the safety requirements.

In practical situations, the conversion does not complete within a reasonable time due to the state explosion problem. After a predefined waiting time (so-called *time margin* t_m which could be varied) and most likely before it completes its formal verification, Crystal uploads the control logic on the flight control unit. The control logic starts its native on-device execution, while Crystal, in parallel, is exploring and verifying its future states, called just-ahead-of-time (JAT) verification. The future states being explored are initially ahead of the native execution by t_m. Higher the value of t_m lower is the accuracy.

Assuming that JAT's speed is not lower than the flight control unit's native execution, Crystal maintains the time margin. That is any state $\mathcal{S}$ visited by JAT at time $\mathcal{T}$ will not be visited

by the native execution sooner than $\mathcal{T} + t_m$. Otherwise, if Crystal gets behind the actual execution, Crystal's outputs would be useless, because the operators would get notified about the *actual* adversarial consequences of the malicious control logic before Crystal could analyze them symbolically. It is noteworthy that JAT exercises all the paths, while the native execution goes through a single path only. Therefore, the native execution may never visit that specific state $\mathcal{S}$, because it may take a different execution trace.

The time margin between the symbolic model exploration and the flight control unit's actual execution enables the operators or automated response systems to take the time deciding upon an appropriate recovery strategy in case an unsafe state is visited during the future state speculations. We will not focus on the automated response selection, and consider it outside the scope of this chapter.

4 Drone Physics Modeling

To ensure effective JAT drone safety monitoring, Crystal must be able to quickly model and reason about its flight's physical dynamics, it's current and future behaviors when the upcoming actuation command sequences are given. The model outcomes are later merged with flight control code analysis to analyze the controller code execution impact on the flight dynamical operation and drone safety. Crystal uses the hybrid model for its formal JAT verification to determine if the controller execution can drive the drone towards unsafe situations.

4.1 Normal Operation Mode Physical Modeling

The estimation of the future sensor values during normal operation mode is computed using an extended Kalman filter (EKF). The extended Kalman filter is a nonlinear state estimation algorithm that uses data containing noise and inaccuracies collected from a series of observations over time and estimates the unknown states. The states of the system are the physical parameters(acceleration, altitude, pitch, roll, yaw) which are obtained by inertial measurement unit sensors on the drone. The observations are the values given to the actuators like the PWM signal to the motors of the drones. The EKF is one the popular technique used for future sensor data estimation for a nonlinear system like drones. The EKF linearizes the flight dynamics

equations at each step and applies Kalman filter technique on the linear system. The system equations which are required for the algorithm are derived from the physical dynamics hence in order to use the EKF algorithm for state estimation, the system dynamics (flight dynamics) has to be known. Detailed equations of the model are described in Section 4. The estimation of sensor data K steps ahead of time is given by Equation 11

$$\hat{x}(n+k|n+k-1) = f(\hat{x}(n+k-1|n+k-1)), u(n+k-1)) \tag{3.1}$$

The values for (K - m) step are evaluated based on (K - m - 2) estimation. The EKF predicts the sensor values K steps ahead of time and then updates it if the error is large after every scan of the sensor data and updates equations to minimize the error from the actual sensor data.

4.2 Failure Mode Data Driven Modeling

Crystal leverages a data-driven physical flight dynamics modeling using neural networks for the estimation of sensor values during abnormal conditions such as failure modes. The trained model's inputs are the drone's current state (sensor measurements) and upcoming sequence of actuation commands calculated by the flight controller code analysis. The neural network outputs the next sequence of drone's flight dynamical state (location and orientation), and hence the upcoming sequence of sensor measurements.

Since the sensor data is collected from the physical system, the data is continuous and has some fixed slew rate with respect to the actuator actions. The sensor data varies continuously and does not change abruptly although it is collected in a discrete manner. For instance, during the takeoff, the drone's altitude increases from 0 to some $\mathcal{H} \gg 0$ gradually and does not jump $0 \rightarrow \mathcal{H}$ instantaneously. The change rate (e.g., $\mathcal{T}$ per second) is limited by the drone's physical limitations, e.g., maximum thrust by its motors. Since the sensor data is continuous, Crystal leverages the spatial features of the measurements by using convolutional neural networks. The model uses convolutional layer, pooling layer, flattening layer and a dense layer. The applications of different layers are explained below.

Convolutional layer. It is the core of the neural network. Crystal obtains the sensor data from the flight controller memory and uses them as inputs to the convolutional layer through an input layer. Each convolutional layer has four learnable filters and the window size is fifty. In our experiments, we empirically found these numbers to optimize the trade-off between hard-to-train accurate huge networks and inaccurate easy-to-train small models. Each filter performs a convolution (dot product) between the drone input sensor data and the values of the filter data.

The filters are learned to activate when they observe specific features on input sensor data such as increasing, decreasing or fluctuating between values based on the time and state of the drone dynamics. These dot product from a filter and input data produces outputs called activation map. Since Crystal uses four filters, it produces four activation maps. These maps are stacked to produce output volume that feeds the next layer. The rectifier linear units are used along with the convolutional layers as the activation function.

Pooling layer. It is periodically inserted in between the convolutional layers reduce the over-fitting by control reduction of the spatial size. In convolutional layers, the window slides through all the drone sensor data in the strides of one. So, it has a lot of spatial data which leads to large spatial size and high computation apart from the overfitting. Crystal uses pooling layers to reduce the spatial size and hence over-fitting. It makes use of one-dimensional pooling layer after each convolutional layers.

Flatten layer. Crystal uses this later to flatten the activation maps and get a single dimensional data. Flatten layer is used to merge all the four activation maps to a single dimensional activation map.

Dense layer. These layers are connected to all the activations from the flattening layer to the output layer. The dense layer is a conventional neural network with weights and biases.

The drone sensor data has a lot of spatial locality in nature due to the continuity of the underlying flight physical dynamics and how the drone's physical state evolves over time. Therefore, the neural network just uses previous finite $\mathcal{N}$ sensor data samples to predict the future sensor data i.e it does not depend on the whole flight history. Similar to the EKF even the neural network model predicts K steps ahead of time, based on the (K - 1) prediction. During the

runtime, the predicted (K - 1) steps are continuously compared against the actual values from the sensors and updates the values.

4.3 Full Flight Operation mode

The system knowledge is essential for the prediction of sensor data by using EKF. In most of the scenario's the system model is not complete due to missing few modeling parameters due to lack of complete knowledge, approximating few parameters during modeling or due extreme nonlinearities. The system model parameters will also not hold true over the lifetime of a system due to the changes caused to the parameters by wear and tear or damage, replacement of old or damaged part with newer once. Apart from the difficulties of modeling of the system, EKF also takes few iterations to converge from the initial estimated state values by correcting itself based on the noise parameters. If the change on sensor values are random and sporadic then EKF takes many iterations to correct itself whereas the neural network takes only a single pass. Due to the linearization of the system equations in each step, if the system is modeled incorrectly or if the initial state estimation is wrong then the algorithm quickly diverges. On the other hand, the neural network does not know the system dynamics initially and has to be trained and re-designed using a large set of data considering different scenarios. The neural network's model performs well if the scenario's which are trained but not so well on a newer strange scenario. Since the neural networks lack the physical system knowledge, they are weaker in self-adaptability compared to the EKF. Hence a hybrid approach is used leveraging the advantages [173] of both the models depending on the sensor data.

The hybrid estimation model is used at runtime for Crystal's JAT verification to predict future drone states (location, orientation) over next several scan cycles and sensor data that feed the controller's future executions. The time margin between the current concrete execution and the predicted sensor data allows for Crystal's ahead-of-time analysis and timely notification of the operators about potentially upcoming unsafe states. Crystal collects upcoming actual sensor measurements (coming from actual execution) and use them to correct the predicted data. This feedback loop enables Crystal to update the hybrid model about the drone's actual physical dynamical evolution, and improve the next predicted sensor measurements.

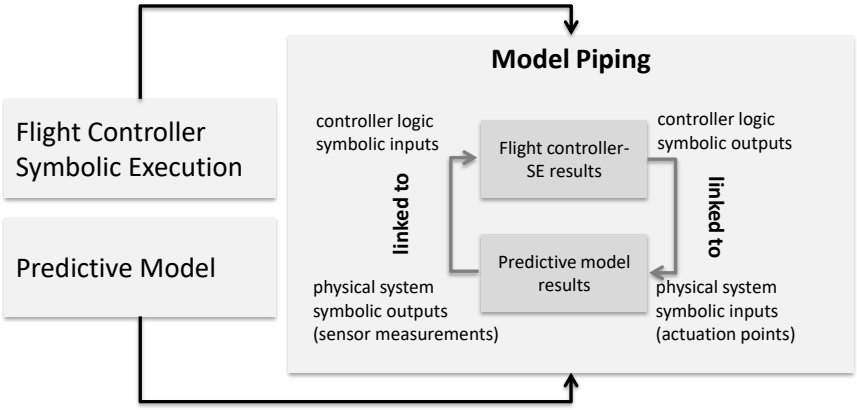


Figure 3.4: Hybrid Cyber-Physical Symbolic Execution

5 Cyber-Physical Security Modeling

Crystal combines the outcomes of the controller code symbolic execution and the flight physics models (i.e., neural network). The resulting hybrid symbolic model captures the dynamics of both the cyber flight controller and the underlying physical flight dynamics. This allows Crystal to analyze how the (malicious) controller code execution affects the drone's physical operation and safety status.

The flight controller and physical dynamics are interconnected through sensing and actuation channels (Figure 3.4). The flight control unit's outputs feed actuators on the drone, and the flight control unit's inputs are fed by the drone's sensors. Our calculated symbolic hybrid model formulates how the flight control unit's outputs and inputs are interdependent based on the flight dynamics. The flight controller code models encode how actuation commands are calculated based on sensor measurements. On the other hand, physics models (EKF and neural network) close the loop and determine the next sensor measurements based on the actuation commands.

Crystal generates the hybrid models through the following four phases. First, Crystal translates the drone global safety requirements to the flight control unit's actuation output constraints automatically. Second, Crystal analyzes the flight control unit's code to determine the primary constraints on its inputs that guarantee the above-mentioned output constraints if the code executes. Third, Crystal emulates the flight dynamics symbolically using the above-mentioned flight control unit's output constraints and calculates a secondary set of constraints on the flight

control unit's inputs. Finally, Crystal uses formal theorem provers to prove the drone safety; Crystal verifies that the flight control unit's input value space defined by the secondary constraints is a subset of the input space defined by the primary constraints. Otherwise, there will be flight control unit's inputs that violate its output constraints and hence the drone's safety requirements. In this section, we explain these four steps in more details using a running example.

drone requirements→controller output constraints. Crystal requires the *global* safety requirements for the drone's physical dynamics. The global requirements impose constraints on the parameters that are often different from those directly controlled by the flight control unit's outputs. For instance, the global requirements may state "the drone should not accelerate on z-axis by more than $\mathcal{X}ms^{-2}$ " while the flight control unit's output controls the amount of power fed to the propeller motors. The z-axis acceleration depends on the propellers power feed according to the flight dynamics of the drone. Crystal uses its drone physics models (Section 4) to calculate the flight control unit's output constraints given the drone's global safety requirements. Crystal uses the predictive model along with the hybrid symbolic execution to calculate the range of the flight control unit's output values that ensure the global requirements are satisfied.

Flight controller output constraints→controller's primary input constraints. Flight controller code produces actuation outputs given its sensor data inputs. Crystal calculates the ranges of the controller inputs that ensure output values satisfying the constraints calculated in step a. Crystal executes the control logic symbolically and computes the corresponding input constraints and symbolic output values for feasible execution paths. Crystal calculates the controller input constraints as the logical conjunction of the path condition predicates and the calculated controller output constraints (step a). Consequently, Crystal obtains input predicates for feasible control logic execution. The calculated input constraints represent the permissible drone sensor value ranges to ensure that the controller's actuation outputs will not cause the drone's global safety property violations.

Controller output constraints→secondary input constraints. In step b, we discussed how Crystal transforms the controller output constraints to its input constraints by analyzing the cyber channel (i.e., the controller code). In this step, Crystal performs the same transformation (the controller output to input constraints) but considering the physical channel (i.e., flight dynamics). The controller's actuation outputs lead to the drone's physical state change (dynamical evolution) indicated later by the updated sensor measurements back to the controller. Crystal assumes the controller outputs comply with the constraints (calculated in step a) and emulates the flight dynamics and its evolution over time using the trained neural networks. Crystal calculates the resulting constraints on the upcoming sensor measurements. We call the calculated controller input constraints *secondary* constraints to differentiate them from the constraints calculated in step b.

Formal proof of the drone safety. So far, we have calculated two sets of flight controller input constraints: *i)* the primary constraints for the controller code that ensure the flight controller execution (cyber dynamics) will generate outputs that do not violate the drone safety, and *ii)* the secondary constraints that, if violated, indicate the controller outputs must have taken on values (based on the flight dynamics) that do not satisfy the global safety requirements. To guarantee the compliance with the global requirements, the value space defined by the secondary constraints should be a subset of the space defined by the primary constraints; otherwise, the controller code execution could possibly result in outputs that violate the global safety requirements. Consequently, Crystal checks for the above-mentioned relationship. Crystal negates the primary constraint for each execution path and calculates it's conjunction with the secondary constraint predicate. Crystal checks whether the ultimate predicate is satisfiable. If not, the plant is marked as verified. Otherwise, the global requirements could be violated, because there would be a concrete value set that satisfies the negated primary constraint (violates the primary constraint) and satisfies the secondary constraint. This would be equivalent to the primary constraint being a subset of the second constraint; therefore, the cyber-physical flight control unit is not verified (is either buggy or malicious) and its execution can lead to unsafe drone states such as a physical ground crash.

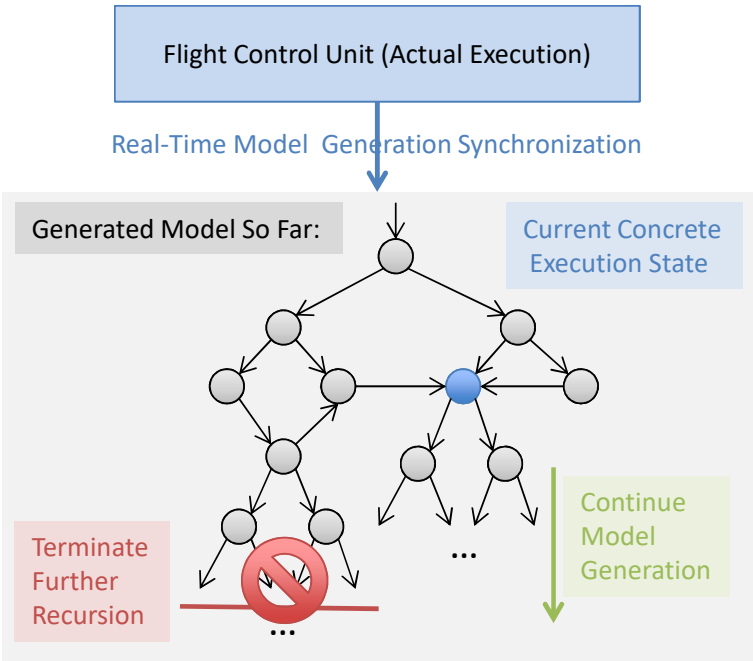


Figure 3.5: Model Generation, Refinement, and Checking

6 JAT Verification and Recovery

In the previous section, we described how Crystal develops a hybrid verification of the drone operation. However, the above-mentioned analysis does not consider the drone operation across subsequent sense-process-actuate scan cycles (Section 2.2). The flight control code symbolic execution (Section 5: step 2) goes through individual execution paths of the control logic just once, whereas the flight controller's actual execution immediately starts the next scan cycle based on the updated sensor inputs right after the current one finishes. The control logic often leverages stateful variables such as timers and counters that retain their values across successive cycles; this causes inter-cycle output value dependencies.

Just-ahead-of-time analysis. To address subsequent cycles, Crystal explores possible symbolic hybrid states of the drone and creates the corresponding state-based finite state automaton. Each symbolic hybrid state captures the symbolic values of the controller code as well as the flight dynamics parameters. Each state transition in the automaton represents a new cycle. The first state represents the drone right after it is turned on. The increasing depths within the automaton represent subsequent scan cycles, and the number of outgoing transitions from each

state (branching factor) represents the number of feasible execution paths in the flight controller code. As the result, Crystal considers individual control logic executions and the corresponding flight dynamical evolution and creating the corresponding automaton state. Crystal performs the formal verification steps (Section 5: step 4) for that state specifically to ensure the drone safety in that state.

The main problem with the algorithm above is scalability due to the exponential growth of the automaton size on each depth. The traditional alternative to offline formal analysis is the runtime methods to detect the unsafe states as the system enters those states dynamically. In cyber-physical real-time drone operations, runtime solutions are not sufficient generally, because they often report safety violations too late for a timely response and recovery countermeasure. For instance, a runtime monitor for a drone may report "the drone is about to hit a pedestrian". That is too late for the operator or automated decision maker to spend time for picking a response and carrying it out while the drone (physical world) has inertia and hence will keep moving. Crystal implements just-ahead-of-time (JAT) verification.

Crystal starts symbolic exploration and safety verification of the automaton states as discussed above. After a predefined time margin t_m, Crystal launches the flight control unit's actual execution in parallel, while the depth exploration is maintained at (at least) the same speed as the concrete execution, and hence it stays t_m time units ahead of the actual execution permanently. Figure 3.5 shows JAT in operation, where the automaton has been explored and created two depths ahead of the current system state. Crystal avoids exploration of the states that are not reachable from the system's current concrete state.

Human-assisted drone safety recovery. Crystal enables human-assisted system recovery through its fast enough JAT verification to maintain the time margin t_m between JAT and the flight controller native execution. If JAT encounters a potentially unsafe future state, Crystal asks for the operator's recommendation. A potentially unsafe state represents the existence of a feasible execution path from the flight controller's current native execution state to a future point where the drone safety requirements are violated. The violation would be caused by the flight controller actuation outputs that change the critical flight parameters.

The violation may be because of a malicious or buggy flight controller program. Crystal

facilitates such a hybrid cyber-physical reasoning about the drone operation through its model piping (Figure 3.4). Due to the model's symbolic state variable values, Crystal uses a formal SMT solver to determine the possibility of safety requirement violation in every encountered future state through one of its concretizations. Crystal expects the operator's feedback within t_m, otherwise, the native execution catches up and may enter the same unsafe state, while Crystal is not yet given with the optimal response action. In those cases, the controller takes a default non-optimal safe response action that is set initially before the controller program execution launch time.

The operator's recommendation within t_m could be any sequence of the following: *i)* to modify or read any control logic variable value on the flight control unit dynamically; *ii)* to inject an instruction (e.g., call a recovery subroutine) on the running control logic; *iii)* to upload a new control logic (possibly a safe controller following the Simplex paradigm [168]); and *iv)* to halt the flight controller's dynamic maneuvers so that the drone enters the hover mode. Crystal implements the operator's recommendation immediately if the native execution enters the target symbolic state *and* the state's current *concrete* variable values actually violates the safety requirements. Regardless, Crystal stores the operator's recommendation to avoid inquiring the operator again later for similar scenarios, where JAT encounters the same or a symbolically equivalent state.

Optimization for practical feasibility. To ensure that JAT will keep up with the native execution speed, we leverage various drone architectural features and implement several optimization techniques: *i)* symbolic execution. Before the runtime setup, Crystal implements an offline symbolic execution of the flight controller code to avoid exploring individual concrete traces with similar outcomes in terms of the compliance with the safety requirements. Additionally, the offline symbolic execution enables Crystal runtime model exploration to consider only feasible control logic execution paths and not waste its online analysis time on unreachable (infeasible) states. The use of symbolic execution significantly reduces JAT's search space and improves its runtime performance. Crystal then pipes the cyber- and flight physics-side analysis results (input-to-output mappings) to use a full cyber-physical system model for JAT verification (Figure 3.4). *ii)* runtime model pruning. Any time the native execution takes (or

does not take) a branch at time t, a subset of the future states at time $t + t_m$ that JAT is about to explore may become unreachable. To maximize its time utilization, Crystal periodically investigates the native execution state and prunes the unreachable model states accordingly. Crystal's runtime model pruning gives an exponential speedup and eliminates the exponential recursive branching of the model states during the exploration. Such exponential branching leads to the impracticality of the previous formal method solutions [128]. It leads to a finite state sub-space within JAT's forward-sliding t_m window over the complete model's state space (Figure 3.1). *iii)* parallel JAT. Given the exponential speed up using the runtime model pruning, Crystal's exploration is accelerated further through its multi-threaded implementation theoretically up to the point that JAT can complete verification of every model depth within an execution-cycle interval. In our implementations, Crystal achieves this objective for complex controller programs using a quad-core machine. *iv)* physics-aware flight dynamics prediction. The neural network created is trained offline and the model is used for the prediction purposes to improve the analysis performed. Crystal predicts the drone's physical state and hence limits the upcoming possible sensor input values to the control logic. The introduced input constraints mark many originally-feasible control logic execution paths as infeasible saving analysis time.

7 Evaluations

We have implemented and evaluated Crystal on two commercial products *i)* 3DR Solo Quadcopter [1] and *ii)* advanced Siemens S7-300 PLC controller [72]. We designed the experiments to evaluate the following questions: *i)* How accurately and fast can Crystal intercept and reverse engineer the flight controller-bound machine codes between the ground base station and the drone's flight control unit processor? *ii)* How efficiently do the proposed symbolic hybrid cyber-physical system analysis techniques calculate the formal sensing/actuation constraints? and *iii)* Can Crystal perform JAT verification efficiently for real controller programs?

We have taken three steps to optimize our implementations to ensure Crystal maintains its exploration ahead of the native execution: *i)* symbolic execution enables Crystal to consider similar multiple concrete states through a single symbolic state analysis; *ii)* periodic acquisition of native execution state and the predictive modeling of the drone's physical dynamics using

a hybrid approach of convolutional neural network and EKF enables Crystal to avoid wasting verification time on unreachable future states; *iii)* multi-threaded future state exploration and verification gives Crystal's performance a linear boost. Our experimental results are promising and show that Crystal can detect violation of drone safety properties and recover the safe operation successfully.

Evaluation on 3DR Solo Quadcopter

Implementation We implemented Crystal on a Raspberry Pi 3 embedded computer running Linux kernel 4.4. We used Keras with TensorFlow as a back end to implement the predictive neural network. The extended Kalman filter was implemented in C/C++ code. The Z3 theorem prover is used by Crystal for checking the feasibility of paths and simplifying the symbolic outputs obtained during symbolic execution. The sensor data consists of flight data collected in stabilize, acro, drift, altitude hold, position hold and loiter modes in various scenario's including a couple of crash scenarios.

We implemented Crystal on a 3DR Solo quadcopter. The 3DR Solo has a flight control unit which is isolated from the external world for programming and an i.MX6 solo processor from Freescale which runs Linux based operating system 3DR Poky (based on Yocto project reference Distro) and has connections with the external world as well as the flight control unit. The Raspberry Pi can communicate to flight control unit through UDP. The Raspberry Pi sends MAVLink [130] packets to request the sensor data from the quadcopter. This sensor data is feed to the predictive model to obtain the predicted sensor data. This predicted sensor data is used to prune the TEG and determine if there is any unsafe state in the near future and inform the operator about it. The flight control unit has a relatively less powerful processor (168 MHz / 252 MIPS Cortex-M4F) compared to i.MX6 Solo (ARM Cortex A9 1Ghz, 1 CPU core with VPU and GPU) and Raspberry Pi 3 (1.2GHz 64-bit quad-core ARMv8 CPU), Hence the implementation for JAT verification will not lag behind the flight control unit's actual execution.

Case Studies: 3DR Solo Quadcopter Control Attack

- **Control attack on Motor and servo control**

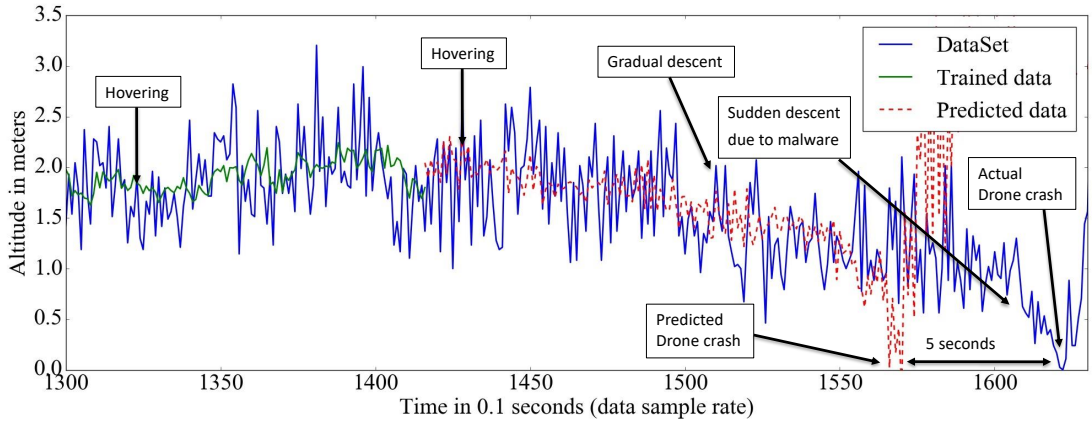


Figure 3.6: Crystal predicting the crash before the actual crash occurred

`Safety requirements`: The angular velocity of motors on the quadcopter should be within safety limits. $\{P\}C\{Q < \gamma\}$ where P is the current angular velocity, C is the next command and Q is the resulting angular velocity which must be less than γ boundary condition for safe operation. This limit changes with respect to the mode of operation. eg.., the rate will be a bit relaxed when the quadcopter is in acro mode compared to stabilize mode.

`Safety violation`: We implemented a malware (Maldrone) to crash the quadcopter while landing. The Maldrone we developed will decrease the thrust on the quadcopter significantly while landing leading to crashing the quadcopter. These kind of attacks are difficult to detect by humans if the quadcopter is flying at a distance away from the operator.

`Violation detection`: The predictive model predicts the significant reduction of the thrust on the quadcopter ahead of time, based on the current and past reduction rate of the thrust. Crystal estimates the possible unsafe state in future and informs the operator about the unsafe state. We were able to receive the information about the unsafe state well in advance by using Crystal. Figure 3.6 shows the predicted value before the crash which could not be detected until the crash had occurred without Crystal.

- **Control attack on AHRS**

`Safety requirements`: The prediction value on the flight control should not deviate from the actual sensor value by more than γ (The safe operation range).

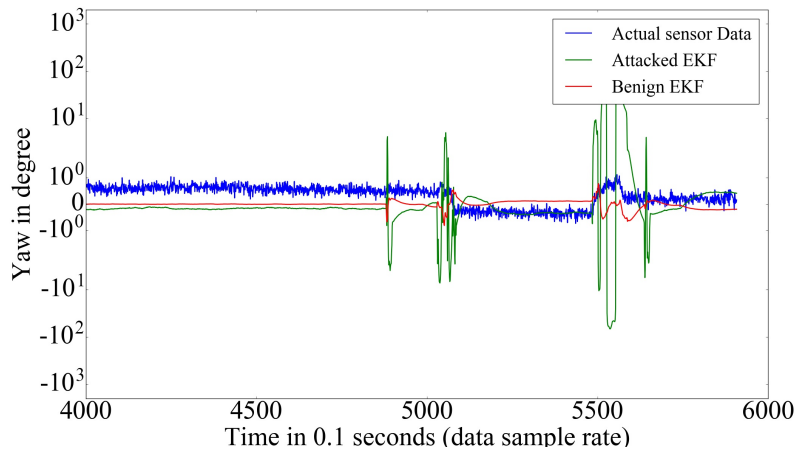


Figure 3.7: Crystal predicting the attack on attitude and heading reference system (AHRS)

`Safety violation`: The synthetic malware which disrupts the estimation of the EKF was introduced into the control algorithm. This malware modifies the EKF algorithm so that the estimation values do not follow the safety requirements.

`Violation detection`: Crystal predicts the upcoming unsafe conditions and informs the operator about the situation. Figure 3.7 shows that the Crystal could predict and inform the operator.

- **Drifting due control attack on PID control**

  `Safety requirements`: The quadcopter should not drift more than certain tolerance which depends on the accuracy of the sensors and the external environmental factors. The drift considered over a cycle as well as over an accumulated period of time.

  `Safety violation`: The PID parameters of the motor are changed due to which the adaption of the change in the motor speed was not accurate. Hence a lot of drift was introduced into the system.

  `Violation detection`: The predictive model detects this drift caused due to change in PID parameters and informs the operator about the violation.

- **Control Attack on altitude control**

  `Safety requirements`: The quadcopter should not change its altitude by more than the tolerance range. The tolerance is due to external environment factors like wind and/or

Table 3.1: Average mean absolute error (MAE) for extended Kalman filter (EKF) and neural network (NN) model during minimal and heavy transitions

Sensor Data	**NN MAE**	**EKF MAE**
Roll during minimal transition	1.3242	0.1136
Roll during heavy transition	0.1713	0.8268
Yaw during minimal transition	1.9644	1.6359
Yaw during heavy transition	3.5643	18.5647

due to the accuracy of the sensors.

`Safety violation`: The synthetic malware was introduced into the system which changes the speed of rotation of the motors (angular velocity) and changes the altitude of the quadcopter. The malware injects false data into the control algorithm.

`Violation detection`: The predictive model detects this change in altitude by more than the tolerance value in altitude hold mode and informs the operator about the unsafe conditions. Since the firmware is trusted Crystal gets its sensor values from the firmware, hence false data injection on the control algorithm could be easily detectable by the Crystal.

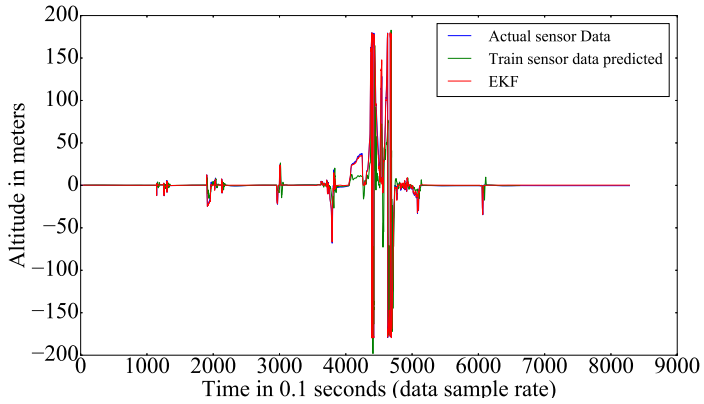
- **Control Attack on position control**

  `Safety requirements`: The quadcopter should hold its position and not change the position by more than tolerance. Again the tolerance might be due to external environmental factors such as wind.

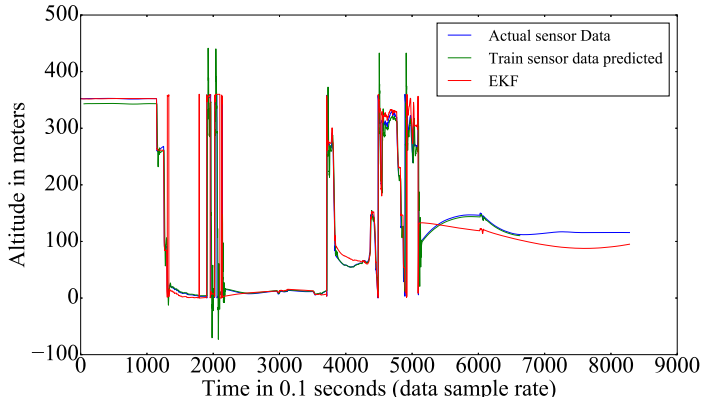
  `Safety violation`: The synthetic malware was introduced to change the position of the drone by more than the tolerance range. The malware can inject false data of either GPS and /or IMU of the sensor data to the control algorithm.

  `Violation detection`: The predictive model detects the change in the position greater than the tolerance range and informs the operator about the unsafe condition.

Accuracy In Figure 3.8a and Figure 3.8b the estimation of sensor data is better by using EKF for gradual and smooth transitions of the sensor data since EKF is aware of the physical dynamics much more accurately than neural network. However, in case of heavy and violent transitions, the neural network is better than EKF as it just takes one pass whereas EKF takes



(a) Roll sensor data



(b) Yaw sensor data

Figure 3.8: Two figures showing that extended Kalman filter (EKF) is better in estimating with smoother transitions than neural network and during violent transitions, neural network is better than EKF

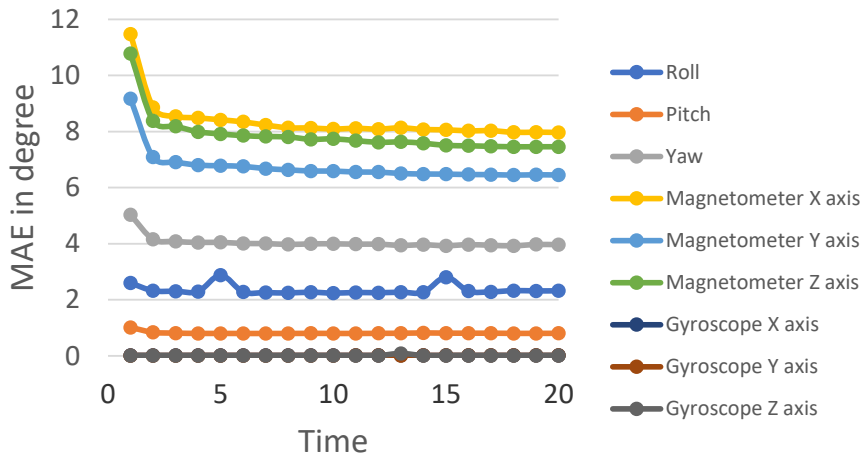


Figure 3.9: Mean Absolute Error vs time

few iterations to converge to the actual data based on the noise parameters used. Table 3.1 shows the results for the accuracy during smooth transitions and violent transitions. The error is represented using mean absolute error. Mean absolute error (MAE) is the average error between the predicted value and the absolute value.

$$Mean\ Absolute\ Error(MAE) = \frac{1}{n}\sum_{t=1}^{n}|e_t| \tag{3.2}$$

Figure 3.9 shows the MAE for different iterations (epochs) of learning the sensor data. The learning data converges between 10 to 15 iterations (epochs). Hence Crystal's neural network uses 12 epochs for the experiments. The false positive rates due to the predicted sensor data are shown in Figure 3.10.

Performance The performance of the EKF and predictive neural network running on a Raspberry Pi 3 is shown in Table 3.2. The analysis is performed for predicting the sensor data for five seconds and ten seconds ahead of time. The time taken for EKF prediction is almost equal to the scan rate of the sensor data, while the time for NN is around four times of it. The performance of the neural network can be further improved by using minimal lightweight tools written in native code compared to heavy tools like Keras and TensorFlow.

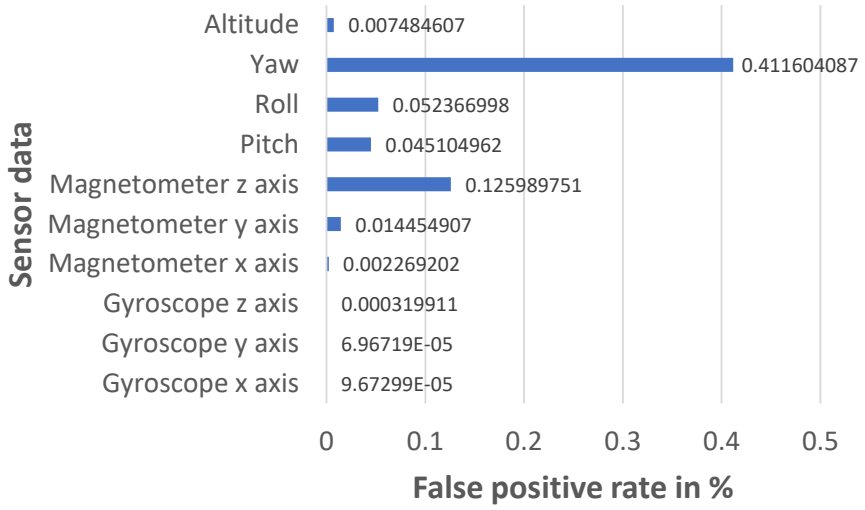


Figure 3.10: False positive rate due to sensor prediction

Table 3.2: Latency in milliseconds for predicting sensor data with data points accumulating to 5 and 10 seconds

Estimation methods	**Min**	**Avg**	**Max**	**Mdev**
EKF (5 seconds)	10	56	120	26.64
EKF (10 seconds)	60	104.5	140	23.5
NN (5 seconds)	290	389.5	480	54.72
NN (10 seconds)	330	417.5	530	59.9

8 Related Work

Control system security. The related work to protect the control systems' trusted computing base (TCB) are insufficient as software patches are often applied only months after release [156], and new vulnerabilities are discovered on a regular basis [30, 153]. The traditional perimeter-security tries to keep adversaries out of the protected control system entirely. Attempts include regulatory compliance approaches such as the NERC CIP requirements [145] and access control [74]. Despite the promise of information-security approaches, thirty years of precedence have shown the near impossibility of keeping adversaries out of critical systems [101] and less than promising results for the prospect of addressing the security problem from the perimeter [117, 114, 135]. Embedded controllers from most major vendors [114, 204] and popular HMIs [135] have been shown to have fundamental security flaws.

Controller program analysis. Basic static program analysis approaches use SAT-based model checking through Boolean logic [152, 90, 126] that could analyze sequence-based control systems with timers, but those are only narrowly applicable. Unlike [43], the two theorem-proving based approaches [99, 147] handle numerical instructions but do not implement rules for overflow checks or mixed bit vector and integer arithmetic. Almost all static analysis techniques [128] fall short in either checking for all program details or scaling up to large-scale critical infrastructures. To improve dynamic SCADA infrastructure monitoring techniques [188], PLC-based approaches have been suggested [31, 104] for dynamic physical plant monitoring. Dynamic plant behavior safety monitors [133] and mathematical intrusion detectors [49] are also related. In addition to being intrusive and causing performance overhead, dynamic monitoring solutions such as WeaselBoard [146] focus mainly on accidental failures, ignoring malicious actions, and/or leave an insufficient time buffer for an effective response and recovery in case of an attack or failure.

Drone security and safety. There have been several recent efforts on offline and runtime formal verification of drone platforms [162]. Javaid et al. [107] investigates potential threats against UAV platforms and how existing cybersecurity techniques fall short in defense due to the lack of consideration of the physical dynamics. Chan et al. [46] present an overview of

formalizing stability properties of cyber-physical systems and drone platforms using the Coq proof assistant. The proof procedures introduced require fairly tedious human involvement. R2U2 [166] proposes a runtime formal verification for monitoring of security properties and diagnosing of security incidents. R2U2 continuously monitors inputs from various sources such as the GPS and the ground control station and identifies anomalous behaviors once they occur. R2U2 relies on the models of the controller code that are assumed given by the operators. Additionally, as discussed earlier, R2U2's runtime verification failure alerts often result in too-late notifications for a timely recovery strategy selection and deployment.

9 Conclusion

We presented Crystal, a just-ahead-of-time formal verification and controller recovery solution for cyber-physical system and evaluated our solution over 3DR Solo quadcopter and commercial Siemens logic controllers. Crystal's just-ahead-of-time analysis eliminates the state explosion problem and gives the operators a time gap to choose recovery actions. Additionally, unlike traditional online monitoring solutions, Crystal leaves the operators with an arbitrarily-adjustable time gap to decide upon how to recover the system normal operation mode in case of an unsafe state. Our experimental results show that Crystal can proactively detect unsafe states, and recover the system with a negligible performance overhead.

Chapter 4

Contactless Control Flow Monitoring via Electromagnetic Emanations

1 Introduction

Industrial control systems (ICS) are fundamental parts of modern society as they control and monitor critical infrastructures such as electricity grids, health-care, chemical production, oil and gas refinery, transportation and water treatment. Due to their importance and large attack surfaces, they are becoming attractive targets for malicious penetrations leading to catastrophic failures with substantive impact [176, 118] including the recent BlackEnergy worm against Ukranian electricity grid [71]. Recently, the Stuxnet malware uploaded malicious code to programmable logic controllers (PLCs), and physically damaged 20% of Iranian PLC-controlled centrifuges [72]. The discovery of Duqu [50] and Havex [163] show that such attacks are not isolated cases as they infected ICS in more than eight countries. Some of these vulnerable controllers are Internet-connected [70] and exposed by computer search engines like Shodan [34]. There have been an increasing number of reports on malicious attempts of PLC port scanning, automated PLC malware generation, modifying control system-specific protocols and access to system diagnostics [40, 126]. Nevertheless, the ICS market is expected to grow to $10.33 billion by 2018 [194].

There has been an increasing number of past works on embedded systems and PLC protection. Offline formal control logic analysis have been investigated by solutions such as TSV [128], through symbolic execution and model checking mechanisms. Solutions such as WeaselBoard [137] and CPAC [69] perform runtime PLC execution monitoring using control logic and firmware-level reference monitor implementations. Most related to our chapter, there

have been attack and defense solutions that employ side-channel analyses to either disclose secret information (e.g., cryptographic keys [81]), or detect anomalous misbehavior (e.g., execution tracking [121]). Side channel-based attacks require selective monitoring of only execution points of interest, such as the encryption subroutines. On the other hand, side channel-based defenses have to monitor throughout the execution looking for anomalies.

In this chapter, we present ZEUS, a contactless PLC control flow integrity monitor that monitors the program execution by analyzing the PLC's runtime electromagnetic (EM) emanation side channel. Given a PLC controller program, ZEUS profiles the PLC's electromagnetic emanation during the execution of feasible paths of the legitimate program. ZEUS pre-processes the signal traces and uses them offline to train a neural network model. The model is later used during the runtime operation to either determine the fine-grained control flow of the execution based on the real-time EM emanations or declare unknown (malicious) code execution.

Contactless monitoring enables ZEUS to ensure security of crucial controllers in mission-critical applications with tight real-time constraints. The operators are often very reluctant to instrument those controllers' software stack with additional security probes that cause performance overhead and hurt the underlying real-time guarantees. Additionally, from security viewpoint, contactless monitoring keeps ZEUS away from the attack vectors that target the controllers because of the introduced air gap between the monitor and the victim controller. Other side channel-based techniques such as power signal analyses draw and monitor current from controllers' circuitry. In contrast, ZEUS is completely non-intrusive and passive; it does not require any instrumentation of the controller and does not affect its electronics.

ZEUS monitors all the network links bound to the PLC, and captures the control logic uploads by the human-machine interface (HMI) servers that are sent for execution on the PLC. ZEUS exercises various code segments of the binary while capturing the electromagnetic emanations. ZEUS profiles the control logic on the PLC with deactivated output modules to ensure the underlying physical process (actuators) are not affected during the training phase. The training is implemented in two stages. First, ZEUS executes control logic symbolically[1] and removes infeasible paths. Through counterexample guided inductive synthesis [178], ZEUS

[1]Complete symbolic execution of embedded PLC control logic programs is often feasible as they are mostly not branch-heavy in practice.

generates different test inputs for each execution path, and trains a neural network based on the corresponding electromagnetic emanation. The trained neural network allows ZEUS to detect the execution of an illegitimate control flow and/or malicious code injection.

The contributions of this chapter are as follows:

- We present a new execution control flow tracking solution for embedded PLC controllers that enables security monitoring with air-gapped electromagnetic sensors.
- We develop an online signal processing framework to analyze the electromagnetic signals and extract minimal feature set necessary for execution integrity monitoring.
- We evaluated ZEUS using an inexpensive sensor against widely-used control programs, e.g., proportional-integral-derivative (PID) controllers, on commercial Allen Bradley PLC devices (most popular in North America) with ARM Cortex-M3 processors. ZEUS detects malicious code injections with 98.9% accuracy in real-world settings.

Overview and Organization. In Section 2, we explain our assumptions about the adversaries and their capabilities. In Section 3, we provide a brief background on programmable logic controllers and their typical configurations as well as neural networks that ZEUS employs for program behavioral modeling. In Section 4, we discuss about the electromagnetic signals emitted by the PLCs and how they characterize the program execution. We discuss how ZEUS generates training data points for program behavioral modeling and transforms the signal traces into spectrum sequences. In Section 5, we present our fine-grained emanation analysis model, where a neural network model of the legitimate program control flows is constructed and trained using electromagnetic emanation signals. In Section 6, we present our empirical evaluations of ZEUS's various components on ten real-world PLC programs and attack scenarios similar to Stuxnet [72]. In Section 7, we review the recent most related work in the literature, and finally, we conclude the chapter in Section 8.

2 Threat model

One of the most prominent security failure causes in control systems using PLCs is the failure to guard PLCs against remote programming [216]. PLC programmer machines are most often

based on commodity operating systems, and often lag security update releases by months [196]. In the following, we state the assumptions made on the security measures that must be successfully in place for ZEUS to function correctly.

We assume there is some trusted path from ZEUS to system operators to alert them of any malicious executions. Unlike software-based solutions, ZEUS's contactless monitoring enables secure monitoring even if the software stack below the PLC's control logic (e.g., firmware or operating system) is compromised. The PLC-bound network link used to transfer the control logic programs for execution is assumed to be trusted. This allows ZEUS to obtain a legitimate copy of the control logic to compare with the runtime PLC executions for control flow integrity checks. ZEUS does not assume source code availability and works with binaries.

ZEUS defends against *control channel* attacks (e.g., Stuxnet [72]), where the adversaries upload arbitrary and potentially malicious control logic on the PLC for execution. More specifically, the types of control logic attacks that ZEUS can protect against are *i)* modified control logic such as injection, removal, and replacement of code segments in the legitimate control logic program; and *ii)* hijacked control flow of the legitimate control logic execution through network exploits (e.g., code reuse attacks[2] such as return-oriented programming). ZEUS does not defend against *sensor channel* attacks, where sensor data is forged by compromised sensors. In such a case, the control logic may behave exactly as intended, but on false sensor data [122]. Additionally, Zeus itself may be attacked by external signal jammers leading to false positives. However, this would not affect the integrity of the control logic execution on the PLC.

3 Background

Programmable Logic Controllers. Programmable logic controllers are multiple-input-multiple-output computers. They have input and output modules to interact with the physical world (plant) to monitor and control critical infrastructures such as manufacturing, robotics, and avionics. The PLC's input modules are connected to sensors within the plant and receive measurements about the plant's status continuously. The PLC's output modules are connected to

[2]Protection against control flow and code reuse attacks are simpler in PLCs compared to conventional computers, because the PLCs' restrictive and more primitive programming languages (e.g., type safe and no indirect call sites) allows deterministic modeling of the legitimate control flows.

plant actuators and convey the commands to control it. The PLC converts 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. The control logic programs are developed by the control system operators on human-machine-interface (HMI) servers that are connected to the PLCs through network links. Once developed, the control logic is compiled and sent to the PLC for execution. The program is executed repeatedly in fixed intervals, called scan cycles. During each scan cycle, the control logic program reads input values from memory and stores the output values to memory. The PLC firmware is responsible for the interchange of these updated values to and from the PLC's input/output ports to interface the physical world. The firmware also implements the reporting mechanisms such as the LED display on the device and real-time data transfers to the HMI about the plant's current status.

Deep Neural Networks. Neural network is a class of supervised learning models that tries to learn the complex nonlinear mapping between input data and their targets (e.g. class labels). A basic neural network unit architecture consists of a linear mapping followed by an activation:

$$y = \sigma(Wx + b), \tag{4.1}$$

where x is the input feature vector, and W represents the weight matrix. σ is the activation function. It is a nonlinear function that models the complex relation between input x and output y. Common activation functions include sigmoid [102], rectified linear unit (ReLU) [224], tanh, etc. Figure 4.1 shows a graphical illustration of Equation 4.1. The edges between the Input layer and the hidden layer represents the weights W, b. Since each node in the input layer is fully connected with all nodes in the hidden layer, such unit is also called a dense layer.

A neural network can go large, which is increasing number of nodes in the hidden layer, or go deep, which is stacking multiple network units together (increasing number of hidden layers), such that more complex nonlinear mappings between data and targets can be learned. All forms of artificial neural networks essentially follow the aforementioned basic architecture. ZEUS utilizes recurrent neural network (RNN) [159] to model the execution behavior of PLC programs (Section 5).

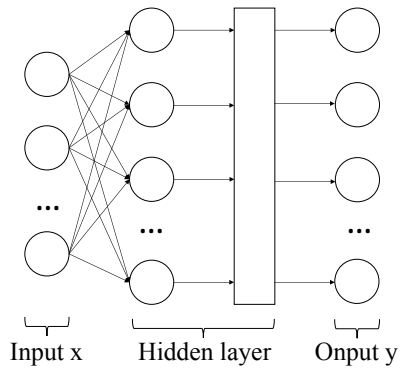


Figure 4.1: A basic neural network unit architecture.

For training, the data samples, each with a corresponding label, are fed to the network's input layer. The network is trained to learn discriminative features from samples by itself. This completely data driven approach, compared to traditional hand crafted feature extraction methods, leads to much simpler-to-use and more reliable outcomes in practice. In our experiments (Section 6), we empirically show that RNNs overcome traditional techniques such as hidden Markov models (HMMs) in terms of PLC execution monitoring accuracy and performance.

Neural networks can be trained in an iterative manner using the gradient descent algorithm [220]. At each iteration, all input data are passed through the network. The output are compared with their corresponding targets t. A loss function l is defined between the network output y and the expected target on each data sample:

$$l_i = loss(y_i, t_i), \tag{4.2}$$

The loss function measures the difference between the current and target outputs. ZEUS uses mean square error (MSE) as the loss function. The total loss is the sum over individual losses of all the data samples:

$$l = \sum_{i=1}^{N} l_i, \tag{4.3}$$

where N represents the total number of data samples. Computing the total loss is called the forward pass. To update weights or parameters of the network, partial derivatives of the total loss with respect to all weights are calculated to identify their maximal descending direction using back propagation. All weights are updated accordingly (the backward pass). Forward

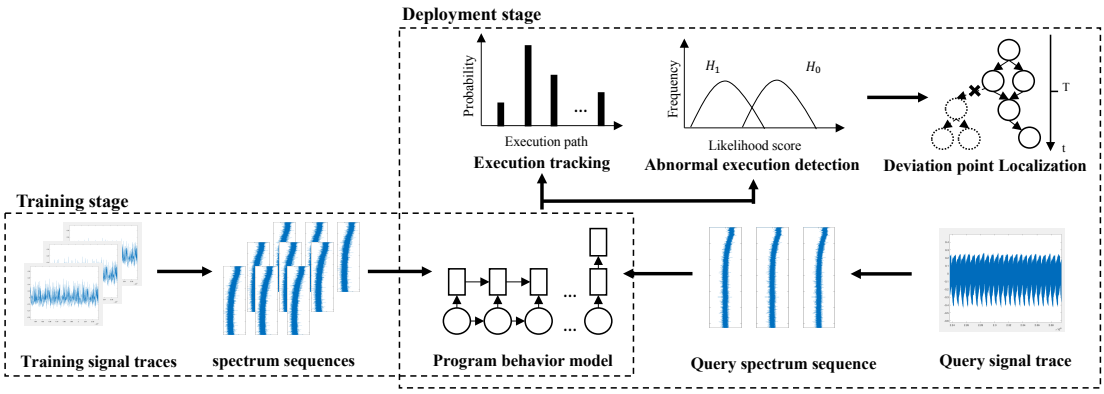


Figure 4.2: ZEUS's control flow integrity monitoring.

and backward passes are repeated iteratively until the values converge. The resulting network is able to produce outputs close to the expected targets.

4 PLC Program Emanation Analysis

During the PLC code execution, the processor clock frequency and switching of the underlying CMOS devices along with the power regulation board result in change of electric current in the PLC circuitry. The current produces time-varying magnetic field that interacts with the electric field leading to an electromagnetic (EM) wave. The EM wave propagates perpendicular to electric and magnetic fields [19]. In order to radiate this EM emanation, an antenna is required. The components on the PLC's printed circuit board (PCB) act as antennas. The transmission range of these waves increases with the increase in the surface area of the antenna. These emanations from the PLC boards can be captured by an external electromagnetic sensor placed near the emanation source.

ZEUS uses these near-field EM waves as the PLC side channel, because they leak information about the program running on the device [80]. Different instructions usually expose different emanation patterns. Thus, the collected electromagnetic signal traces during program executions have unique local characteristics depending on the runtime control flows. This observation is utilized by ZEUS to fingerprint the side-channels of legitimate program executions and identify unknown (malicious) code injections and/or control flow hijacking attempts.

Recent attempts have been made to monitor micro-controllers such as STC89C52 [121] and PIC16F687 [66] based on power signal analysis [121, 66] that require physical manipulation of

the circuits for sensor placement. The data acquisition draws current from the controller boards potentially affecting its functionalities that triggers a red flag for practical deployment in controllers for mission-critical real-time operations. In comparison, ZEUS's contactless, passive and non-intrusive monitoring of commercial PLC ARM processors using an inexpensive EM sensor for control flow integrity is a more challenging endeavor.

Due to the PLC architecture, the execution times of individual instructions are not fixed to the processor's clock cycle. A list of estimated completion times for instructions is provided in the user manual. However, in practice based on our observations, even the execution time of a single instruction varies across different execution runs of the same PLC code. This makes the signal analysis using time-based truncation infeasible. Being contactless, ZEUS has to deal with a large amount of signal noise. Our collected electromagnetic signal traces have very low signal to noise ratio (SNR) such that repeatable local patterns along the execution paths (leveraged by [121]) cannot be observed in the time domain.

To deal with these problems, ZEUS borrows ideas from speech recognition research [157]. ZEUS looks at frequency representations of signal sections within a local sliding window. ZEUS extracts signal segments via a sliding window on the collected signal. Each segment consist of several consecutive instructions. ZEUS then computes the power spectral density of each segment.

Unlike time domain signals, we observed that the patterns in the frequency representations are much more stable and robust to noise. This is because the local spectra (spectrum sequence) are computed through weighted summation of all time signal points within the window. Hence, the white noise is not cumulated, while the underlying desired deterministic signal is. Therefore, a sequence of local spectra extracted from the PLC code execution EM signal trace includes repeatable patterns to characterize individual execution paths. ZEUS deploys the aforementioned analysis to model the execution behavior of target PLC programs.

For completeness of the results, signal traces of all feasible control flows of the program are collected. ZEUS monitors the network link between the HMI servers and the PLC controllers, and intercepts the control logic uploads to the PLC. Through symbolic execution [128], the execution path predicates are aggregated and checked by an SMT solver for satisfiability. Consequently, infeasible executable paths are eliminated.

For each remaining feasible path, ZEUS calculates several concrete test cases through counterexample-guided inductive synthesis [178]. More specifically, to calculate the first test case for a path, its aggregated path condition expressed as a conjunctive logical expression $\varphi_p = (\varphi_1 \wedge \varphi_2 \wedge \cdots \wedge \varphi_n))$ is fed to the SMT solver. The solvers produces a concrete input value set (e.g., $i = 20$). Calculating the second concrete input for the same path involves feeding $\varphi_p \wedge \neg\varphi_i$ to the SMT solver, where $\varphi_i := (I == 20)$ and $\neg$ represents logical negation. The next concrete inputs are calculated similarly.

ZEUS runs the program on the PLC using the generated test cases for each execution path, and collects the electromagnetic emanations using an external sensor. The collected signal traces along with their labels (corresponding control flows) are fed to a sequence neural network classifier for training. All these steps are performed offline. During the runtime, ZEUS's external sensor collects the PLC's emanations and employs the classifier to determine whether the signal trace belongs to the feasible legitimate execution paths. A modified execution path (e.g., a maliciously injected PLC program) will lead to a change in electromagnetic emanations away from the samples observed by the classifier during the training phase. The deviation triggers a red flag by the classifier, and the execution is marked as malicious.

5 EM-Based Control Flow Monitoring

There[3] have been works utilizing electromagnetic side channel signals to detect abnormal executions [185]. They follow a template matching scheme, where the query signal is compared with all constructed templates of the execution paths. Based on our experiments, such time domain-based signal matching techniques cannot distinguish fine-grained characteristics of complex program control flows accurately. To address this, ZEUS constructs a sequential neural network classification model of pre-processed frequency domain data samples. The model therefore learns from control flow transitions from the training frequency data and encodes them in its network weights. This model describes the behavior of the program.

ZEUS maps the control flow transitions of any new legitimate data sequence with the learned

[3]This work was completely done by Yi Han from Rutgers University and added in my thesis to understand the complete picture of the solution.

transitions, modeled by the neural network, and determines a specific control flow that the observations correspond to. A data sequence with abnormal components such as unseen segments (due to code injection attacks) or invalid transitions between segments (due to control flow hijacking) will cause mismatches. Such mismatches accumulate along the sequence, cause the neural network states and thus the output of the model to deviate from expected values.

Figure 4.2 shows ZEUS's work flow. During the offline training stage, each collected signal trace is transformed into a spectrum sequence (Section 4). The sequence neural network model is trained using spectrum sequences of all classes (legitimate control flows). After deployment, ZEUS feeds the online spectrum sequences of collected query signal traces into the trained model. This results in a probability distribution computed by the model over all the classes. The class with the highest score is compared to a predefined threshold. If the score exceeds the threshold, ZEUS assigns it to the query signal trace as the execution path that the program is taking currently. If the threshold check fails, it indicates a mismatch between the query signal trace and the trained model. Consequently, ZEUS triggers an alert about an illegitimate control flow. ZEUS provides more fine-grained reports about the mismatch regarding the execution point that the real-time control flow deviated from the legitimate expected flows. This information can be used later for detailed vulnerability discovery, e.g., how the control flow was hijacked. We consider the vulnerability analysis phase outside the scope of this chapter.

5.1 Offline Model Construction and Training

To construct the classification model with sequential inputs, we use a long term short memory (LSTM) network layer [82]. LSTM is a variation of the recurrent neural network (RNN) used for modeling sequential data. LSTM takes a sequence of inputs and maintains a hidden state vector along the sequence. This fits ZEUS's use-case, where the observables are EM signals, while the hidden states represent the underlying unobserved code segments and basic blocks.

At each time step of the input sequence, current hidden state vector is computed based on both the previous hidden state vector and the current input. The hidden state carries long term dependency between time steps. This enables ZEUS to capture contextual information in the sequential control flow transitions.

Let $[x_1, x_2, ..., x_N]$ be a spectrum sequence computed for a collected EM signal trace. x_t

indicates the input in the sequence at time t. The vector size N depends on the execution time of the control flow. The current hidden state h_t can be computed as:

$$h_t = o_t * \tanh(c_t), \tag{4.4}$$

where tanh is the hyperbolic tangent activation function; $*$ denotes the entry-wise product; o_t is the output gate vector; and c_t is the cell state vector. The two vectors can be computed as:

$$\begin{aligned} o_t &= \text{sigmoid}(W_o x_t + U_o h_{t-1} + b_o) \\ c_t &= f_t * c_{t-1} + i_t * \tanh(W_c x_t + U_c h_{t-1} + b_c). \end{aligned} \tag{4.5}$$

In Equation 4.5, $W_o, U_o, b_o, W_c, U_c, b_c$ are the weights of the neural network units. Note that the design extends the basic network architecture described in Section 3. c_{t-1} and h_{t-1} are state vectors passed from the previous time step. f_t and i_t represent the forget and input gate vectors, respectively. These vectors are designed to keep only useful contextual information and acquire new information:

$$\begin{aligned} f_t &= \text{sigmoid}(W_f x_t + U_f h_{t-1} + b_f) \\ i_t &= \text{sigmoid}(W_i x_t + U_i x_t + b_i), \end{aligned} \tag{4.6}$$

where $W_f, U_f, b_f, W_i, U_i, b_i$ are the unit weights. We add a dense layer followed by a softmax function after the output of the LSTM layer at the last time step h_N. This maps the network to a probability distribution over all legitimate control flows in the PLC code.

$$p = \text{softmax}(W h_N + b), \tag{4.7}$$

where the height the weight matrix W is the same as number of legitimate control flows.

Intuitively, the neural network model output is a one-hot [202] vector q. It has a 1 on its entry that corresponds to the identified control flow and 0s on all its other entries. q can also be viewed as a probability distribution. We define the loss function of our model as the cross

entropy between the model's actual output p and the target vector q. The resulting cross entropy measures the difference between two probability distributions:

$$l = -\sum_i p_i \log q_i, \tag{4.8}$$

where i is the index of the legitimate control flows. The total loss of our model is the sum of losses over all the training EM spectrum sequences and their corresponding class labels (control flows). During the training, weights of the model are tuned iteratively as described in Section 2. A well-trained model will have its output probability distribution very close to its corresponding one-hot vector, and the output distribution will bias remarkably towards the corresponding control flow.

The overall architecture of ZEUS's network model is shown in Figure 4.3. The size of the hidden state vector h_t can be increased to carry more information along the sequence. Moreover, multiple LSTM can be stacked to be more capable of characterizing the EM spectrum sequences for PLC execution classification. However both adjustments increase the computational complexity. To ensure efficient online classification, we kept the network model size minimal as long as it did not affect ZEUS's accuracy of malicious execution detection.

The collected electromagnetic signals suffer from random perturbations caused by EM interference of other components on the PLC device. To reduce the noise effect, ZEUS provides the neural network model training with many EM signal traces for each control flow. This is possible through ZEUS's PLC code analysis and generation of several test-cases for each feasible execution path. Consequently, the neural network training algorithm receive many traces with the same label (control flow) each incorporating random noise. This enables the neural network's data-driven feature extractions procedures to train its unit weights based on the main signal ignoring the noise margins.

5.2 Online PLC Execution Monitoring

ZEUS uses the trained model at runtime to detect anomalous executions. EM spectrum sequences from legitimate executions will have their network outputs distribution heavily biased towards the corresponding class label. The class with the highest probability will be reported

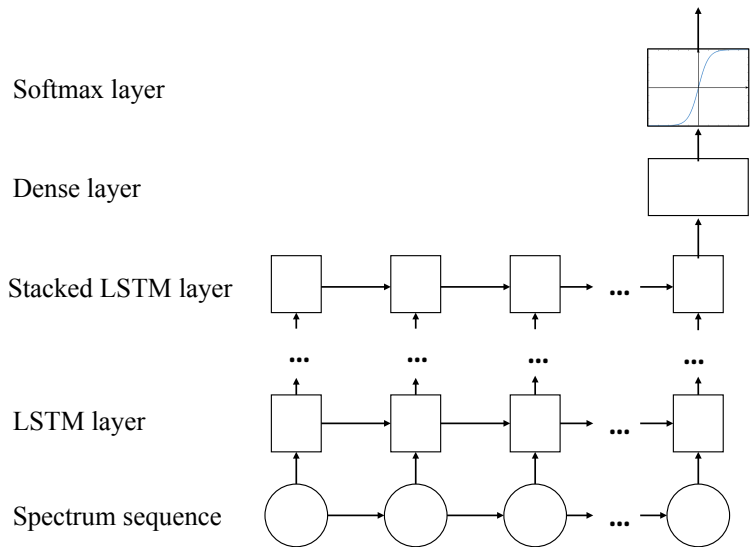


Figure 4.3: Network architecture of proposed model.

as the identified control flow I:

$$I = \arg\max_i p_i, \tag{4.9}$$

and its likelihood score can be calculated as $L = \max_i p_i$. L represents how likely this signal trace belongs to the legitimate PLC code. In the case of correct classification outcome, the network's input transitions match with the corresponding control flow transitions of the PLC control logic and the desired network state is maintained as the most likely state along the input EM trace.

Malicious control flows constitute either execution of a maliciously injected new code or code reuse attacks that execute the available instructions while deviating from legitimate control flows at some point. The introduced new instructions or the control flow deviation cause a mismatch between the observed EM signals and the neural network's learned transitions. This reduces the bias in the neural network model's calculated probability distribution increasing its entropy. Therefore, by setting a threshold on the likelihood score L, abnormal executions can be identified as they match none of the known legitimate control flows.

Let $H_0(H_1)$ indicate the legitimate (malicious) execution, the detection problem can be expressed as:

$$L \overset{[}{H_0}]H_1 \lessgtr \varepsilon, \tag{4.10}$$

where ε is the preset threshold.

For malicious executions, ZEUS can also locate the point, where PLC execution deviated from the legitimate flows. Let $h = [h_{n1}, h_{n2}, \cdots, h_{nN}]$ be the hidden state sequence of the n-th LSTM layer of our model for a query EM spectrum sequence input for a malicious execution. Let's also assume ZEUS identifies the hidden state sequence $h^I = [h^I_{n1}, h^I_{n2}, \cdots, h^I_{nM}]$ as the most likely legitimate control flow that corresponds to a given query EM trace. The deviation point can be located by computing the distance d_t between the two sequences at each time step t:

$$d_t = \sqrt{(h_{nt} - h^I_{nt})^2}. \tag{4.11}$$

The deviation of the PLC execution from the legitimate control flows is reflected in the sudden change of signal traces and thus the neural network inputs. A changed spectrum input will cause its corresponding hidden state vector to move away from its expected vector in the state space. Therefore, a sudden step increase in the distance sequence $[d_1, d_2, \cdots]$ indicates the point, where the deviation happens.

6 Implementation and Evaluation

We evaluated ZEUS on real-world settings with commercial PLC devices and using legitimate and malicious control logics. We first describe our experimental setup including the signal acquisition system and the target PLC model. We will discuss the results on the electromagnetic emanations of the target PLC and their discriminative spectrum characteristics. We measure ZEUS's accuracy in classifying legitimate control flows, and detecting malicious executions. We compare ZEUS's data-driven approach with traditional model-based solutions using hidden Markov models [121] We finally test the performance of ZEUS on several real applications. An Intel i7-6800K CPU was used to compute frequency representations, and HMM training and testing. Our LSTM neural network model was trained on an NVIDIA GTX1080 GPU.

6.1 Experimental Setup

Figure 4.4 shows our signal recording setup that consists of a recording probe and an amplifier. The corresponding test-bed configuration and component interconnection is shown in

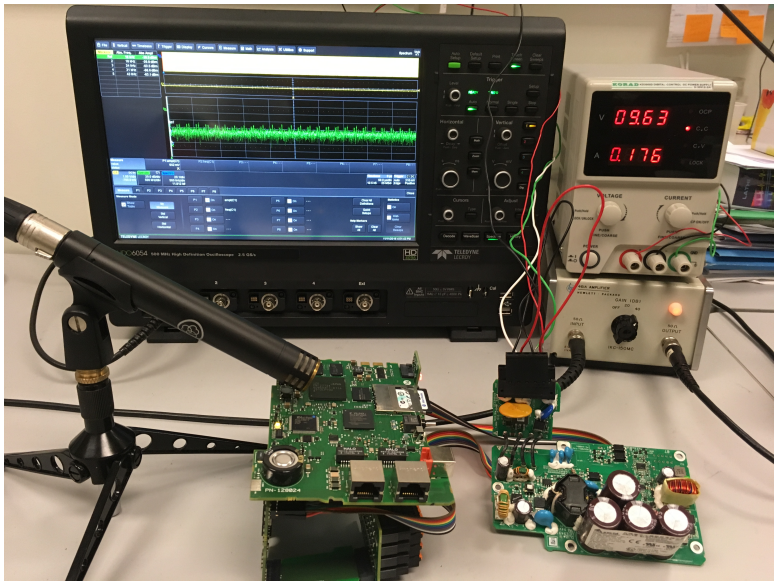


Figure 4.4: Experimental setup including the PLC, external sensing probe, the amplifier, and the sampling oscilloscope.

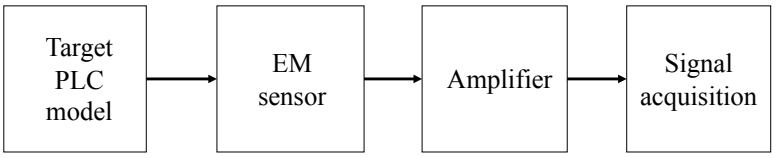


Figure 4.5: Experimentation test-bed configuration for electromagnetic (EM) side channel analysis.

Figure 4.6: AKG P170 condenser microphone without transducer serving as an electromagnetic probe.

Figure 4.5. In our experiments, we used the Allen Bradley 1769-L18ER-BB1B CompactLogix PLC (with ARM Cortex-M3 processor). Allen Bradley PLCs are the most popular and widely used controllers in many industrial control systems in North America. We used an AKG-P170 condenser microphone without the acoustic capsule or transducer (Figure 4.6) as an antenna to receive the electromagnetic emanation[4]. We also tried a Langer LF-R 400 passive antenna. However, we achieved the best signal sensitivity from the AKG-P170 microphone. This is because the AKG-P170 microphone together with the phantom power supply can be viewed as an active antenna. Active antennas have better sensitivities than passive antennas, since signals are pre-amplified.

We used an HP-461A amplifier to increase the signal strength. We observed that most of the informative frequency variations appear below 5 MHz. Accordingly, we set our sampling rate to preserve most frequency information while maintaining a moderate computation complexity for online malicious code detection performance.

[4]When the acoustic capsule is detached from the microphone, the remaining part of the microphone serves as an antenna since the coil in the microphone is sufficiently long to receive the signals emitted from the board.

6.2 PLC Electromagnetic Emanations

We performed numerous tests and inspected various regions of the three PLC PCB boards to identify the point that emits the most distinguishable EM signals. Once that point was identified, we adjusted our directional EM probe to focus on the point while collecting the EM emanations for our experiments.

Figure 4.7a shows the components that we mainly investigated. The main sources of emanation were the proprietary Allen Bradley chip, the field-programmable gate array (FPGA) and the surface mount device (SMD) capacitors on PLC's communication PCB board. The SMD capacitors are involved in the voltage regulation for the chips. Figure 4.7b shows the strength of the corresponding emanation from each point. Since the surface area of the SMD capacitors is very small, the corresponding emission was rather weak. The surface area of the Allan Bradley chip is relatively larger, and hence the corresponding emission was stronger. We proceeded by focusing on that chip for our following experiments. Figure 4.7c shows how the captured signal appears with as the probe-chip distance increases. The EM signals were collected by the probe located 0.1 cm away from the proprietary chip.

We investigated the differences among the EM emanations from the PLC execution of different instruction types. Different PLC instructions have different execution times and computation complexities thus different power consumptions that is reflected in the emanation signals as discriminative spectral patterns. Figure 4.8 shows the results. These spectral pattern types are the core basis for ZEUS's design.

Figure 4.8 visualizes the discriminative spectral patterns of different PLC instructions. PLC's (ARM) ISA include 22 different types of instructions. We show the results for only the 16 types that are commonly used in PLC programs[5]. Modules involving complicated computations, such as PID[6] were also tested. Figure 4.8 shows that different instructions give rise to EM signals with different intensities at different frequencies. ZEUS exploits these fingerprints to estimate the PLC's internal runtime execution state and dynamic control flow using

[5]The instructions include arithmetic instructions (ADD, MUL, DIV, DEG), advanced math instructions (LN, SIN, XPY, STD), comparing instructions (XOR, GRT), array manipulation instructions (BSL, AVE, FLL) and control instructions (TON, JMP).

[6]This PLC programming module implements the proportional-integral-derivative (PID) control algorithm [23].

Table 4.1: Confusion matrix for the classification.

	ADD	SIN	XOR	BSL	JMP	PID
ADD	**78.63% (747)**	5.26% (50)	8.21% (78)	1.58% (15)	4.95% (47)	1.37% (13)
SIN	5.36% (56)	**83.54% (873)**	5.65% (59)	1.05% (11)	2.39% (25)	2.01% (21)
XOR	8.13% (75)	7.48% (69)	**69.31% (639)**	0.11% (1)	12.47% (115)	2.49% (23)
BSL	1.24% (12)	1.65% (16)	0.10% (1)	**95.24% (921)**	0.21% (2)	1.55% (15)
JMP	5.44% (53)	3.29% (32)	12.32% (120)	0.21% (2)	**76.49% (745)**	2.26% (22)
PID	0.32% (3)	2.11% (20)	2.64% (25)	0.21% (2)	1.27% (12)	**93.46% (886)**

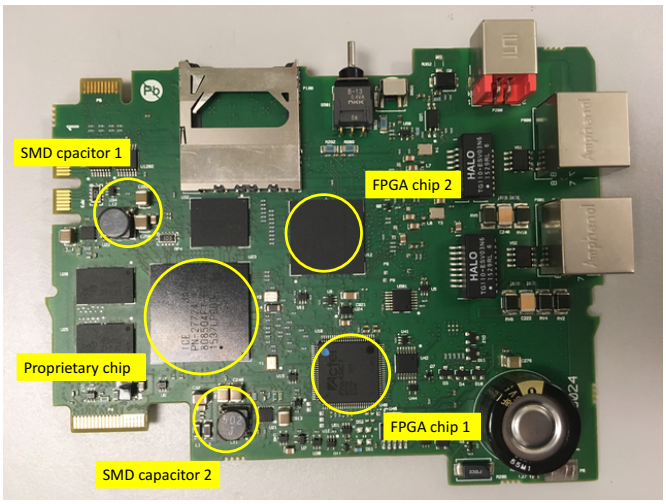
the collected EM emanations.

We further verify our visual observations by performing a classification validation on spectra of emanation signals of these instructions using the random decision forests algorithm [94]. We used Weka [92] to implement the classifier. One instruction of each instruction type (Figure 4.8) was tested. An emanation signal of 200 μs was collected under sampling rate of 50 MHz and transformed into spectrum representation. 1000 such signal traces were collected for each instruction type to train the classifier, and the same number of traces were collected for validation testing.

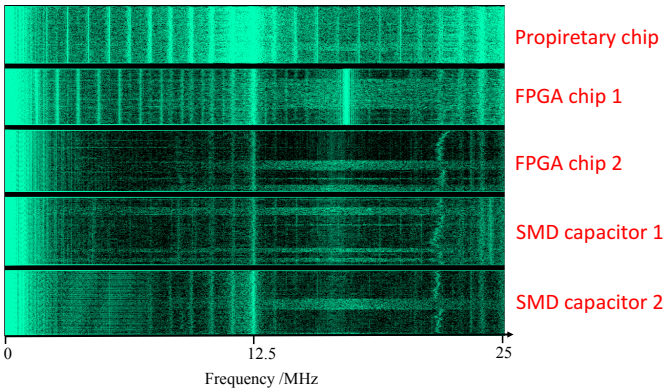
Table 4.1 shows the classification confusion matrix. Most signals are correctly classified correctly as their actual instruction type. This shows that the spectral patterns of different types of instructions are indeed discriminative and can be used for control flow integrity monitoring.

ZEUS applies a sliding window to the collected emanation signals. Because of various instruction execution times, sliding windows of the same length at different points of the signal cover different combinations and number of instructions. This helps for spectral patterns of the signal segments within different windows across the signal trace to be distinguishable. Therefore, the spectra of these local signal segments characterize the emanation signals, the execution paths. ZEUS utilizes this to construct the program behavior model.

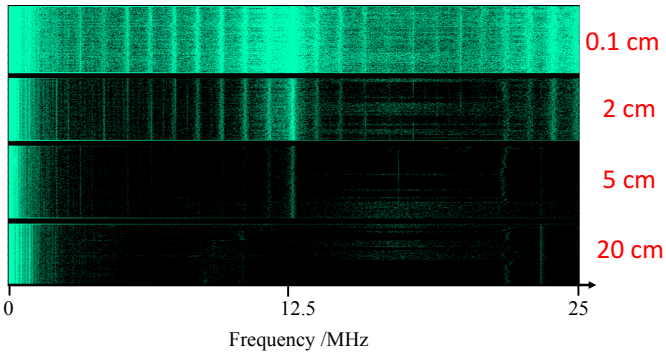
Figure 4.9a shows the EM emanation (between seconds 6 and 12) while a control logic program is installed for execution on the PLC. The visible EM emanation pattern can be used to detect runtime (malicious) control logic uploads similar to Stuxnet [72]. Figure 4.9b shows the electromagnetic emanation patterns during the PLC's boot-up process. These patterns can be used to detect when the PLC is remotely rebooted by an adversary.



(a) One of the PLC's three PCB boards: surface area of the propitiatory chip is larger compared to other chips.

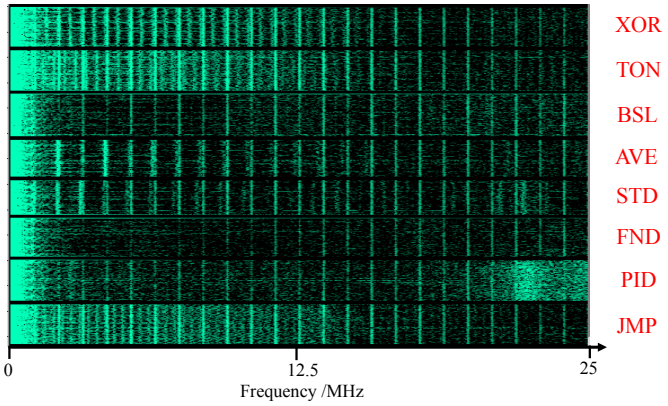


(b) Spectrogram for different locations.

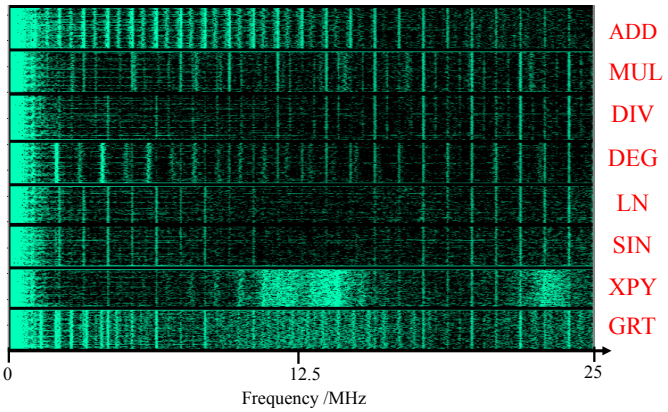


(c) Spectrogram for different distances.

Figure 4.7: EM emanation by the PLC's communication board.



(a) Pattern for PLC instructions.

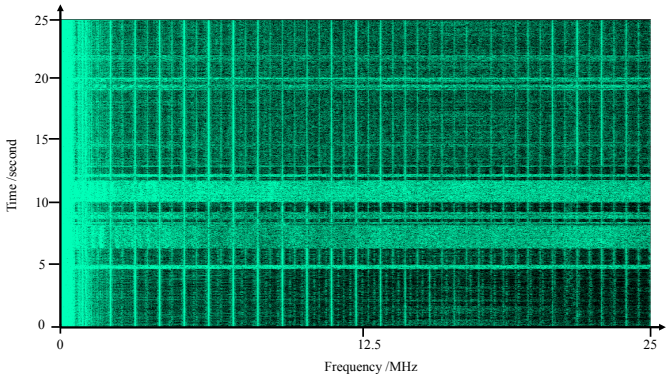


(b) Pattern for PLC instructions.

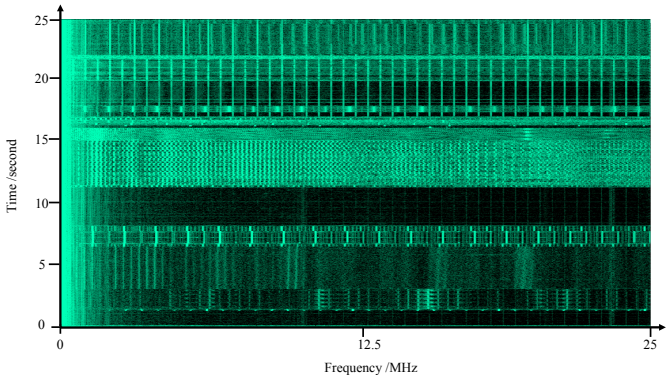
Figure 4.8: Spectrogram patterns of PLC instructions.

Table 4.2: Evaluation programs and descriptions.

Class	**Name**	**Description**	**Example applications**	**Average length (msec)**
Vector arithmetic	Matrix	Matrix multiplication	Sensor array data processing	3.3
	Q-sort	Quick sort	Value searching, element uniqueness	2.1
Numerical methods	GD	Gradient descent	Power flow optimization	5
	Newton	Newton's method	Vehicle trajectory estimation	3.5
Signal processing	Conv	Convolution	Signal filtering	9.2
	DCT	Discrete cosine transform	Audio lossy compression	17.3
Communications	Dijkstra	Dijkstra's algorithm	Routing optimization in smart grid	11.3
Cryptography	AES	AES-128 encryption	Data protection, access control	18.1
Control systems	PID	PID control	Vehicle cruise control	6.5
	Patfilt	particle filter	Object Tracking, localization estimation	2.5



(a) Spectrogram during uploading of PLC program.



(b) Spectrogram during the booting phase of the PLC.

Figure 4.9: Spectral patterns of PLC instructions.

6.3 Accuracy

We evaluated ZEUS for PLC execution monitoring, control flow classification of ten real applications, and detection of malicious code executions. We computed spectrum sequences and estimated the power spectral density of signal segments. The segments were extracted using sliding windows of size 200μs, with 90% overlap between successive windows.

A stacked two-layer LSTM network with 100 nodes on both layers was employed to fingerprint the execution behavior of each program. We trained the network using stochastic gradient descent (SGD). An average of 50 epochs (iterations) were required for the network to converge on the tested programs. We obtained 100 traces for every feasible control flow of each program for training the model. For each program, a 2-fold cross validation was performed 10 times to stabilize the result.

We chose ten real PLC programs from different application domains for evaluation purposes. Table 4.2 lists the control logics along with their functionalities. These programs fall in the classes of vector arithmetic, numerical methods, control algorithms, cryptography, signal processing and communications.

Comparison with HMM solutions. We compared our LSTM network model with a traditional hidden Markov model (HMM) based program behavior modeling approach [121]. For the HMM, the observations were defined as the signal segment or its frequency representation. HMM state was defined as unique samples in the observation set. The number of HMM states was defined as a adjustable parameter. We set the HMM number of symbols to be 100. We fit the observations of each state with a multivariate Gaussian distribution. The parameter set (HMM's transition probabilities, observation models and initial probabilities) was estimated using Baum-Welch algorithm [134]. We used the forward algorithm [157] to calculate the observation sequence likelihoods. We will present the accuracy results for both ZEUS and HMM solutions below.

We evaluated ZEUS accuracy from two aspects: *i)* execution monitoring - to determine the control flow of a running legitimate PLC code, and *ii)* malicious execution detection - to detect the control flows that are not a part of the legitimate program. Table 4.3 shows the execution monitoring accuracy results. We evaluated ZEUS (LSTM) with both pre-processed spectrum

traces (Freq) and raw time domain signals (Time) and compared the results with HMM-based solutions. LSTM using the frequency representation (Freq-LSTM) achieves almost perfect results on all the evaluated programs.

LSTM's better results in comparison with HMM-based solutions can be explained by the following two observations. First, the ZEUS's LSTM network architecture is able to capture long-term dependency in the input sequence. This contextual information corresponds to the control flows of the program, and hence is essential in distinguishing different execution paths. HMM models, on the other hand, assume only 1st order data dependency in the sequence, and hence miss a lot of useful information.

Second, ZEUS's model is able to extract discriminative features from the input due to its stacked multi layer architecture. This contributes to the classification performance. For HMM, however, the input data is directly used for parameter estimation without any feature extraction. When raw signal segments are used as inputs, both LSTM and HMM are not able to achieve good performance (Table 4.3). This is because raw time signals contain lots of noise, so the underlying signal characteristics cannot be recognized and hence learned by the two models. The frequency representation, however, reveals the signal characteristics as the noise (low frequency) stays far from the main signal (high frequency).

For detection of malicious executions, Figure 4.10 shows the likelihood scores L (Section 5.2) of positive (abnormal) and negative (normal) samples for two example applications, namely Newton-Raphson numerical method and AES encryption algorithm. Note the negative samples tend to concentrate within a small range, while the positive samples are more spread out. This is because the number of control flows with each program is finite, and each control flow is well recognized by our network through training. Thus, the signal traces of the legitimate control flows match closely with the LSTM model. The malicious programs can take any arbitrary control flow especially in the case of malicious code injection attacks; therefore, their matching degree vary a lot.

Figure 4.11 shows the ROC curve for the frequency and time domain data using ZEUS's LSTM and HMM solutions. The numbers are average over all the ten applications. ZEUS (LSTM) using the frequency traces achieves almost perfect detection performance. Steeper

Table 4.3: Classification accuracy of all evaluation programs over four evaluation settings.

Program	Time_HMM	Time_LSTM	Freq_HMM	**Freq_LSTM**
Matrix	55%	52%	60%	**100%**
Q-sort	49%	60%	41%	**100%**
GD	40%	64%	40%	**98%**
Newton	48%	51%	63%	**100%**
Conv	57%	69%	56%	**100%**
DCT	53%	45%	51%	**94%**
Dijkstra	62%	72%	65%	**100%**
AES	50%	50%	67%	**98%**
PID	40%	62%	71%	**99%**
Patfilt	51%	45%	67%	**100%**

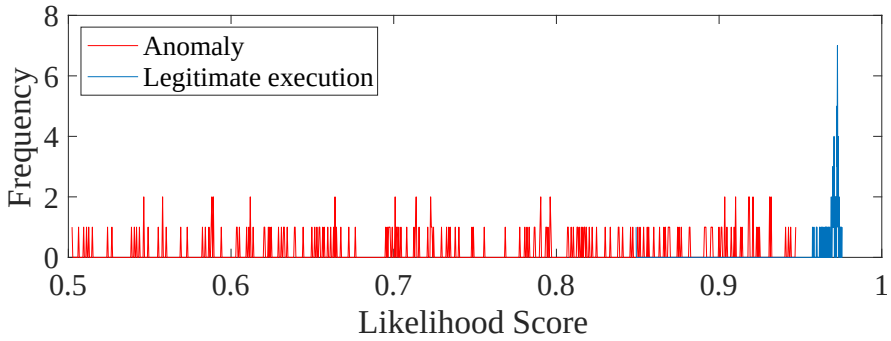
ROC curve indicates better separation of positive and negative samples, and thus better performance. This is usually measured by the area under the curve (AUC). AUC is usually between 0.5 (random guess) and 1 (perfect separation). Table 4.4 shows the AUC for each target PLC program and each evaluation setup. The stacked multi-layer architecture of ZEUS's network model captures important information both from the hidden states and the inputs, and carries it along the sequence. This results in better learning of the program behavior from the signal traces.

Note the HMM using the raw time domain signal performs worse than random guessing (diagonal line on ROC - Figure 4.11). This is because HMM, due to its limited first order dependency assumption, is not able to characterize the temporal behavior of the signal traces well. Additionally, noisy raw time domain signal traces further contribute to its randomness and poor accuracy.

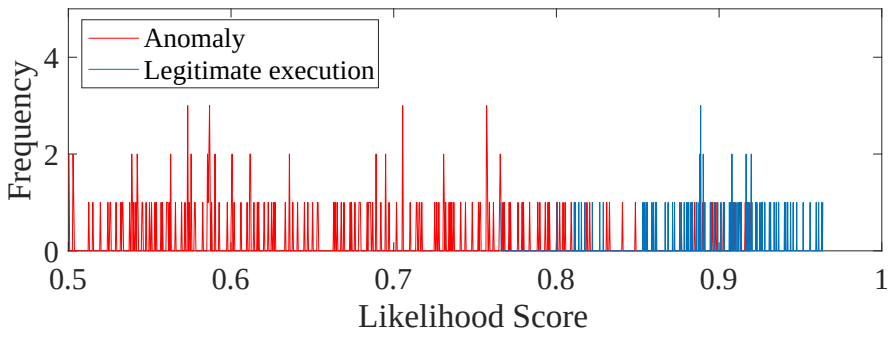
We intentionally reduced the convergence threshold for the neural network's training that led to larger number of training iterations. The main reason is ZEUS's goal to detect malicious executions and not only to classify (previously seen) legitimate control flows. The increased number of iterations resulted in more discriminatory classification outcomes, i.e., more biased probability distribution over the classes (legitimate control flows) and larger likelihood score L. Hence, we were able to increase the classification threshold ε as well (Equation 4.10). Consequently, in the presence of malicious control flows, ZEUS's likelihood score L did not exceed ε. Hence, the flows were classified as malicious correctly. This reduced ZEUS's false negative and positive rates.

Table 4.4: Area under curve (AUC) of all evaluated programs over all four evaluation settings.

Program	Time_HMM	Time_LSTM	Freq_HMM	**Freq_LSTM**
Matrix	0.34	0.52	0.90	**0.99**
Q-sort	0.52	0.48	0.76	**1.00**
GD	0.24	0.55	0.86	**0.98**
Newton	0.25	0.62	0.86	**0.99**
Conv	0.62	0.65	0.81	**0.99**
DCT	0.14	0.61	0.81	**0.99**
Dijkstra	0.44	0.51	0.85	**1.00**
AES	0.56	0.57	0.82	**0.96**
PID	0.34	0.66	0.79	**1.00**
Patfilt	0.22	0.73	0.87	**0.99**



(a) Newton's method



(b) AES-128 algorithm

Figure 4.10: Example likelihood score distributions of the evaluated programs produced by the Freq+LSTM setting.

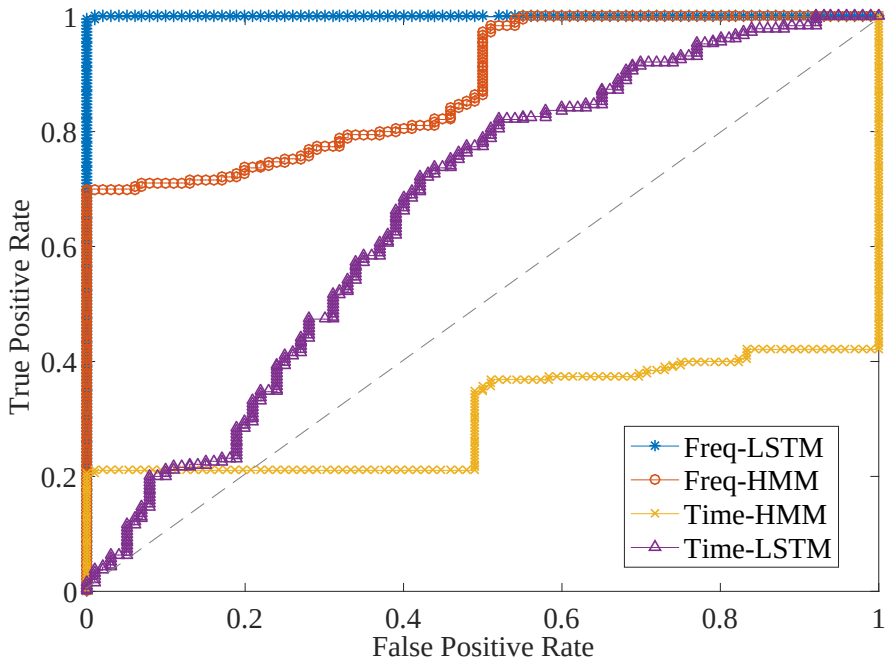
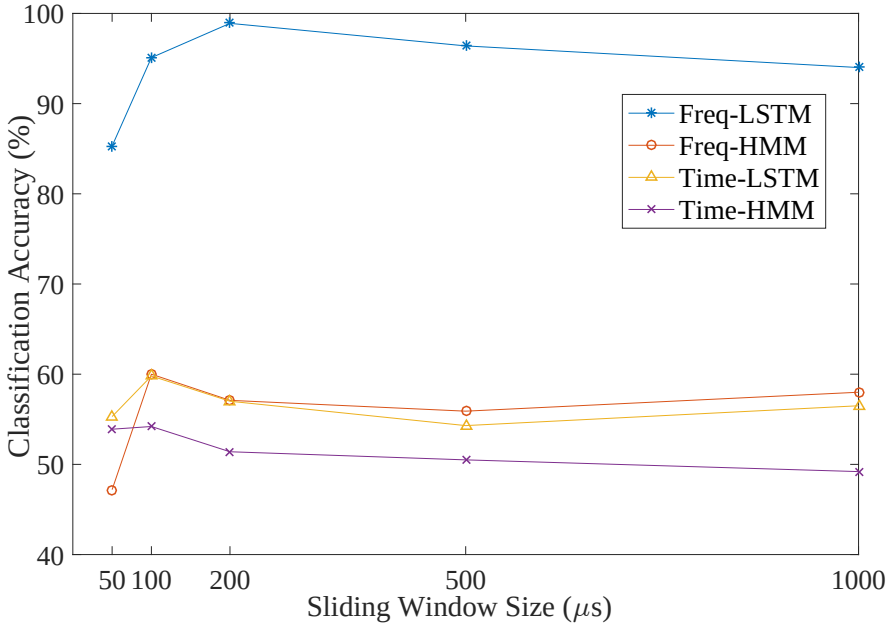


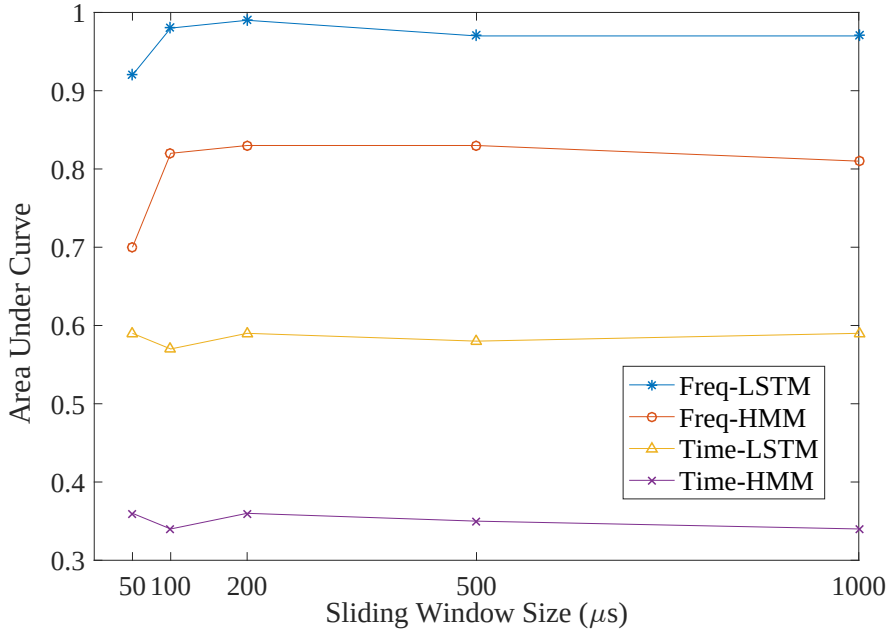
Figure 4.11: ROC curves of all evaluated programs. AUC of the four settings: Freq-LSTM is 0.99, Freq-HMM is 0.83, Time-LSTM is 0.59, Time-HMM is 0.36.

Sliding window size. We investigated the influence of various sliding window sizes on ZEUS accuracy. By using window of different sizes, ZEUS essentially looks into the program execution at different granularities. A smaller window can capture finer grained transitions in the signal trace, but the frequency resolution of the spectra will be lower. This results in a less discriminative representation of the signal segments. Smaller windows also result in longer data sequence, therefore more recurrences of ZEUS's neural network model. This makes the model less robust to random perturbations, since biases on the network states accumulate through the recurrences. A larger window size, on the other hand, will have spectra of better frequency resolution and better robustness, but some small transitions in the signal trace will be ignored.

Figure 4.12 shows how sliding window size affects ZEUS accuracy (Figure 4.12a) and AUC (Figure 4.12b) both averaged on all ten applications. Using frequency data with a LSTM classifier outperforms all the other settings for all the window sizes. When the size of sliding window increases, both the classification and detection accuracy degrade because of the ignored useful transient information (as discussed above). Too small windows also cause accuracy degradation because of the resulting lower frequency resolution and less discriminative spectra.



(a) Classification accuracy vs. sliding window size.



(b) Runtime vs. query trace length.

Figure 4.12: Area under curve vs. sliding window size.

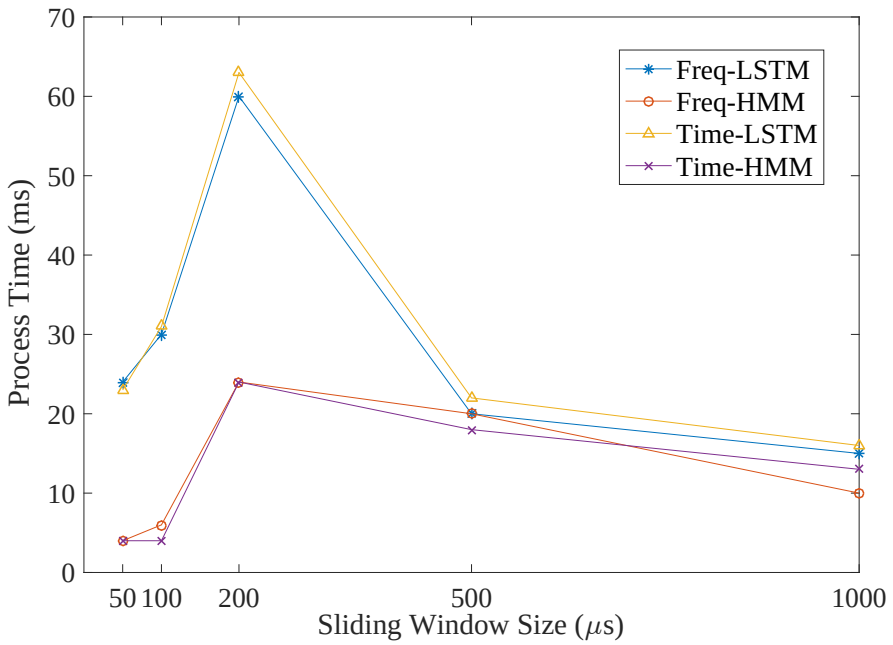


Figure 4.13: Average process time of all the programs for the four evaluation settings.

6.4 Performance

We measured the time requirements to complete ZEUS's classification of the collected EM traces. The required time includes the time of computing the spectrum sequence from the raw time domain EM signal trace if frequency representation is employed, and the time of passing the data sequence through the trained neural network model to get the prediction.

Figure 4.13 shows the average processing time for one input signal trace and various sliding window sizes. The numbers are averaged over all the ten applications. All the four evaluation settings are able to process the query signal within tens of milliseconds. LSTM-based approaches are overall slower than HMM-based solutions, because passing the input sequences through the network involves more time-consuming array computations. HMM's faster speed comes at the cost of its remarkably lower classification and detection accuracy.

The figure also shows that the larger sliding window sizes lead to reduced time requirements. This is expected as the larger sliding window produce shorter data sequences for a given EM signal trace. Consequently, there are fewer recurrences in the neural network.

7 Related Work

We discuss related work on controller and critical infrastructure security in terms of defense mechanisms and possible attacks.

Side channel analysis. There have been several recent works on analyzing side channels of various modalities for the purpose of inspecting runtime execution. On contactless monitoring, Eisenbarth et al. [66] determine the instruction types (not instances) by modeling individual instructions as HMM states. Msgna et al. [136] perform similar analyses by modeling CFGs as HMMs. The authors assume equal-length basic blocks that is not often the case in practical settings. Other similar HMM-based program behavior modeling have also been studied [219, 211, 78, 217]. On monitoring with contact requirements, cross correlation between the side channels (power [87] and RF traces [185, 184, 186]) and the program's single golden execution have been investigated for anomaly detection. However, obtaining a single golden execution is not feasible in practice. A complex PLC program may go through different execution paths depending on the inputs (sensor measurements). Vermon et al. [208] uses power signal side channels to reverse engineer the bytecode running on a Java smart card. Attacks to disclose substitution tables of the A3/A8 algorithm execution [144, 52] have been proposed. These methods focus on recovering the lookup table only. ZEUS increases the accuracy of passive side-channel analysis of complete execution profiles using inexpensive contactless EM sensors.

The most related work [121] provides code execution tracking based on the power signal, which requires connections to an 11 MHz 8-bit AVR microcontroller. The microprocessor is directly connected to the power supply using a single resistor. The sensor is a Tektronix MDO3034 oscilloscope with sampling rate of 1.25 GHz. ZEUS provides *contactless* execution monitoring of *commercial* PLC processors (120 MHz ARM Cortex M3 with three separate PCB boards for I/O, logic processing, and power supply) through a different *modality* (electromagnetic emanations) and using *lower-frequency* sensing sampling rates (10 MHz) with more than two orders of magnitude saving on the sensor cost.

Another related work [141] also performs execution monitoring and anomaly detection on IoT devices via the electromagnetic side channel. They looks at the prominent frequency peaks in the spectra of the signal segments as feature representations and models program executions

with statistical distributions. ZEUS uses the sequential neural network model to both extract discriminative features from signal segments and model the control flow transitions in a end to end manner. Moreover, they puts instrumentations at all the loop nests and examines them separately while ZEUS looks at full executions without any instrumentation.

Controller program analysis. Although a few processors contain a dedicated hardware unit for execution monitoring, e.g., embedded trace macrocell [41], many embedded controllers lack such hardware support. To analyze the software, offline control command verification solutions [128, 152, 90] implement formal methods to verify the safety of the control code immediately before it is executed on the PLC. They face scalability problem, caused by state-space explosion [128, 126, 99, 147]. Consequently, every control logic upload request by the operators, including the emergency cases, should wait for possibly minutes, hours, or more before the code is fully verified for PLC execution. Such delays are often unacceptable for real-time safety-critical control system operations.

Information security approaches. The related work to protect the control networks' trusted computing base (TCB) are insufficient as software patches are often applied only months after their release [156], while new vulnerabilities are discovered on a regular basis [30, 153]. The traditional perimeter-security tries to keep adversaries out of the protected control system entirely. Attempts include regulatory compliance approaches such as the NERC CIP requirements [145] and access control [74]. Despite the promise of information-security approaches, thirty years of precedence have shown the near impossibility of keeping adversaries out of critical systems [101] and less than promising results for the prospect of addressing the security problem from the perimeter [117, 114, 135]. Embedded controllers from most major vendors [114, 204] and popular Human Machine Interfaces (HMIs) [135] have been shown to have fundamental security flaws.

8 Conclusions

We presented ZEUS, a contactless, passive, and non-intrusive control flow integrity monitoring solution for PLCs. ZEUS identifies malicious code execution through side channel analyses of

the controller's electromagnetic emanation signals. ZEUS's data acquisition is done by an electromagnetic sensor, which provides an air gap between the trusted computing bases of the target PLC and ZEUS's monitoring engine. Our empirical studies with commercial PLC controllers and several real application binaries show ZEUS can monitor high frequency commercial processors with low frequency sensor sampling. ZEUS can detect malicious code executions on popular Allen Bradley PLCs with %98.9 accuracy and with zero runtime overhead by its design.

Chapter 5

Securing Critical Infrastructure with Cyberphysical Access Control

1 Introduction

Critical national infrastructure has become increasingly complex. For decades, systems such as the power grid were comprised solely of physical, mechanical components that could be reasoned about using classical physics. However, as computing has become increasingly miniaturized and ubiquitous, adding computational resources into these environments becomes not just feasible, but practical and beneficial. In the case of the power grid, adding computing elements allows for essential capabilities such as *state estimation* (i.e., understanding where the power in a grid is flowing at any given time) and *contingency analysis* (i.e., determining whether the grid is resilient to the failure of components within it). The grid exemplifies a *cyber-physical* infrastructure, with data collected from its physical components and processed by algorithms running on computers to provide for accurate and safe monitoring and control. To realize this, modern smart grids make heavy use of programmable logic controllers (PLCs) which act as dedicated embedded systems that change actuators based off of sensor values in a continuous feedback loop.

Malware-based attacks against these infrastructures, such as Stuxnet [72], Havex [163], and Dragonfly [5], have been well studied, and different solutions have been proposed [51, 142]. However, erroneous activity by human operators, whether intentionally or by mistake can have even more due consequences than existing malware attacks.

The lack of protections against system misconfigurations can lead to severe consequences. In 2011, a lack of real-time situational awareness and limit protections on transmission lines resulted in a cascading series of power outages, affecting large portions of Arizona, southern California, and northern Mexico, causing 1.5 million customers in these areas to lose power for

up to 12 hours [4]. Even worse, malicious activities can also seem to be operation mistakes, such as the coordinated attack on the Ukranian power grid [22]. Moreover, an operator once logged in, usually has a complete view of the whole system, even if the operator is only in charge of a sub area of the system. This unlimited access to system variables and the simple static policy controls for operators demonstrate that cyber-physical infrastructures are unprepared to maintain their safe and secure operation in the face of human mistakes, leaving alone malicious adversaries.

The key takeaway from these episodes is that *insufficient access control* coupled with an *insufficient understanding of the relationship between the control infrastructure and the underlying physical system* leads to vulnerabilities, which can be turned into attacks either by careless operators or malicious adversaries. While past approaches attempt to use information flow analysis for system modeling, have tended to ignore the physical world and miss important inter-dependencies. Moreover, traditional discretionary and mandatory access control mechanisms are often based on manually-generated policy rule sets that do not consider the underlying physics of the grid, and its complexity precludes attempts at formal analysis.

In this chapter, we present CPAC, a *cyber-physical access control* framework that enables fine-grained enforcement of context-aware policies in a real-time control system environment. CPAC takes a comprehensive view of both the computing and physical elements comprising the control system, and simultaneously incorporates both continuous physical dynamics i.e mathematical models and discrete computing i.e administrator specified policies into its security monitoring and control calculations. In doing so, we can accept high-level requirements such as "Alice should not [directly or indirectly] manipulate the [power output] for the generator G_i" or "Bob should not know about power transformer T_j's failure", and have them enforced as low-level policies that ensure control system constraints are maintained. To generate secure policies for access requests, CPAC implements a layered ensemble of lightweight information flow analysis mechanisms. On the device side, we mark variables within PLC devices to determine data flow, and we infer information flows through the grid using physics-based, inter-component dependencies. Information is visible to operators whose access to read and modify variables is tailored to their particular roles (static polices) and depending on the information flow analysis (dynamic polices). Combining the physics model, information flow analysis on

PLCs, and logic-based policy control, we are able to provide finer-grained access control and better situational awareness of the power grid than previous solutions, securing the grid from human mistakes (or insider attacks), maintaining the operation privacy, and supporting N-x contingencies.

Our contributions can be summarized as follows:

- **Physics-based engine:** We demonstrate that by leveraging the underlying mathematical model within a power system, we can analyze information flow by the physics equations and restrict operations that would violate system safety.
- **Information flow analysis:** We introduce a lightweight taint-tracking mechanism into PLCs. The lightweight code instrumentation reports the dynamic control flow used in conjunction with symbolic execution of the PLC code to determine variable taints. This symbolic execution is performed offline ensure minimal performance overhead during PLC code execution.
- **Logic-based policy control:** We introduce a new context-aware policy control using Prolog, where policies are written in logic statements and the querying the permissibility of an operation in the Prolog engine. Combined with the physics engine and information flow analysis, a context-aware policy is able to guarantee the safety and privacy of an operation.
- **Scalability and performance in real-world scenarios:** We model the Polish power grid, consisting of over 2,700 buses, and model three past blackout events within this real-world system setting, demonstrating that CPAC would detect and mitigate all of these problems. CPAC's analysis and policy evaluation can be performed in under 150 *ms*, fast enough that large-scale outages can be prevented. Because CPAC maintains system context, it can manage not only $N-2$ contingency analysis (simultaneous failure of two nodes), but $N-x$ analysis, which is infeasible with existing energy management system (EMS) solutions. CPAC thus provides an effective new means of maintaining robust operation in the face of coordinated cyber attacks.

Section 2 reviews existing EMS solutions and how they fail to withstand operation mistakes or even attacks. Section 3 overviews CPAC's high-level architecture and components,

describing its operation within a simple control system. Section 4 explains the physical side information flow analysis. Section 5 describes policy enforcement and Section 6 describes device-level information flow tracking in CPAC. Section 7 describes CPAC's real-world implementations and extensive experimental results. Section 8 reviews related work and Section 9 concludes.

2 Energy Management Systems

An EMS[1] 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. As shown in Figure 5.1, an EMS contains supervisory control and data acquisition (SCADA) functionality, comprising a suite of applications. These include:

1. A power system topology processor [16] that continuously retains and updates electrical system topology such as branch impedance, loading, connectivity, and circuit breaker status information, with topology details used as input to the state estimation process (detailed below);
2. A data historian (database) [76] that stores sensor measurements and system asset configuration information for later grid analysis and billing;
3. A state estimation system [16] that receives plant sensor measurements and the power system's current topology, and dynamically calculates accurate state of the power system, i.e., voltage, magnitude, and phase angle on each power system bus;
4. Contingency analysis software [88] that performs what-if risk analysis of potential component failures given the power system's current state;
5. Optimal power flow control analysis [88] to calculate optimal feasible power system configuration and actuation parameters for load generation balance (i.e., the generated power should equal the end-users' electricity consumption); and
6. A human-machine interface (HMI) that includes visualization of system parameters for the operators to monitor and modify.

[1] We discuss the configuration of existing *energy management systems* (EMS) used to control the power grid infrastructure. We also discuss their corresponding limitations and vulnerabilities (Section 2.1). Our discussion is necessarily abbreviated; a comprehensive overview of these issues is presented by Sridhar et al. [182].

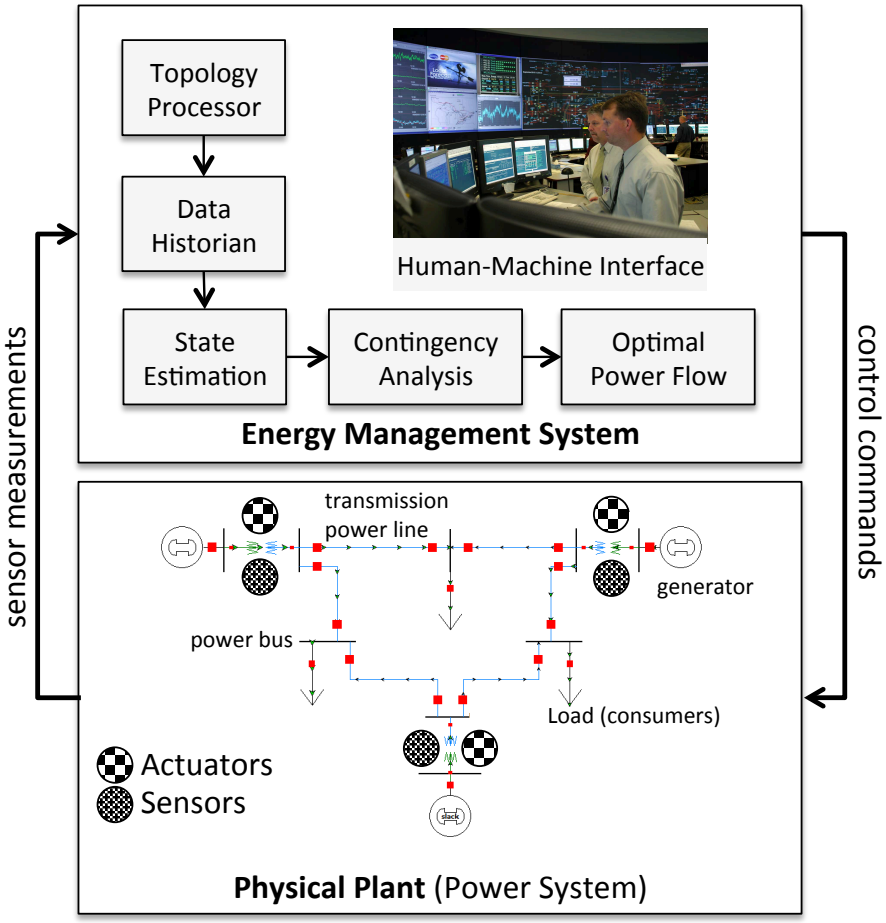


Figure 5.1: Existing Energy Management Systems

2.1 Existing EMS Solutions

Current EMS solutions [198, 21, 32] are designed to protect smart power grids against accidental component failures, but are limited in the protections they offer. For example, data historians enable local data storage and coarse-grained sharing of bulk system information and sensor measurements, but lack the ability to determine where data entries originate and the understanding of plant physics necessary to capture inter-data entry correlation. As a result, simple mistakes from operators can bring down the whole grid, causing millions of dollars of damage [4, 54]. Similarly, while state estimation modules provide a global view of the power system's state and parameters such as line current and bus voltages, they cannot restrict unprivileged operators from observing sensitive system information, which compromises operation privacy [119, 127]. Furthermore, while power flow solutions [85] have functionality to drive

the system away from unsafe states, they do not distinguish among operators with different privilege levels. In general, current EMS solutions solely count on correct actions from operators, who usually only need a password to log in the system and are governed by simple policies (if even these exist), and ignore the risk of operational errors or insiders attacks.

Another significant shortcoming in existing EMS solutions is the limited ability to perform contingency analysis. Within North America, power utilities must implement $N-1$ contingency analyses to comply with the North American Electric Reliability Corporation Critical Infrastructure Protection (NERC-CIP) requirements [115]. An $N-1$ analysis determines whether a power system with N components (e.g., generators) can maintain its operation despite any single component failure. However, a coordinated attack against more than one element within the grid or multiple involuntary components failures renders $N-1$ analysis ineffective as occurred in the southwest blackout incident [4]. The state space explosion associated with performing $N-2$ analysis and for larger numbers of component failures makes these analyses infeasible as shown in Section 1.

2.2 Security Threats

The threats we consider are mistakes from careless operators and intentional system manipulation from malicious adversaries, who could be operators or anyone having access to the EMS. Instead of focusing on the authentication of EMS operators, CPAC tries to authorize each operation request from legal users. Note that the whole EMS is trusted, and we assume operators do not have physical access to these machines except through the GUI/CLI terminal provided by the EMS.

Why don't existing access control mechanisms suffice? Existing host and network-based mechanisms that rely strictly on access control have proven to be insufficient in ICS environments, where cyber and physical components interact as a part of the system operation. A shortcoming of existing access control solutions, such as host-based policy enforcement (e.g.,

SELinux) and network firewalls, are that they ignore the underlying physics of the control systems that they protect[2]. Consequently, implementation of privilege separation and least privilege principles in highly dynamic control system environments become infeasible as access control policies for individual subjects and roles depend on the dynamically changing physical state of the plant. The state of the system may change due to actions by other subjects, e.g., a legitimate power operator on a remote substation computer increases the amount of power generation, or external malicious adversaries, e.g., malware on a remote substation computer opens a power transmission line leaving it out of service. Such incidents change the state of the underlying power system and affect access control policy rules for operators. This increases the risk that subsequent operations (either mistakes or attacks), permitted by a static policy, could compromise system dynamics thus cause damages.

Why do control system safety mechanisms fail to stop operational errors or even attacks? Traditional control system safety mechanisms have been designed to maintain safety only for physical system operations. For example, safety mechanisms in power systems include protection relays and circuit breakers to isolate transmission lines with over-capacity high current flow. Moreover, these mechanisms only consider the physical component involved in the operation rather than a complete system impact of the operation. They are designed to provide reliability and robustness in the case of accidents or harsh environments. They do not, however, take into account a careless operator's mistake, which may crash the whole system, let alone a malicious insider who analyzes the operational changes in a system as it responds to problems, and exploits this behavior to further force the system into an unsafe state [44]. While research into secure control estimation [73, 150, 132] can aid in developing more robust control algorithms, these approaches are largely theoretical and do not consider mistakes from operators or attacks from insiders.

3 CPAC Architecture Overview

We provide a high-level overview of CPAC and describe how it addresses the issues raised above. We further detail in Section 3.3 the factors resulting in the 2011 California outage

[2]E.g., SELinux is not able to limit the CPU temperature.

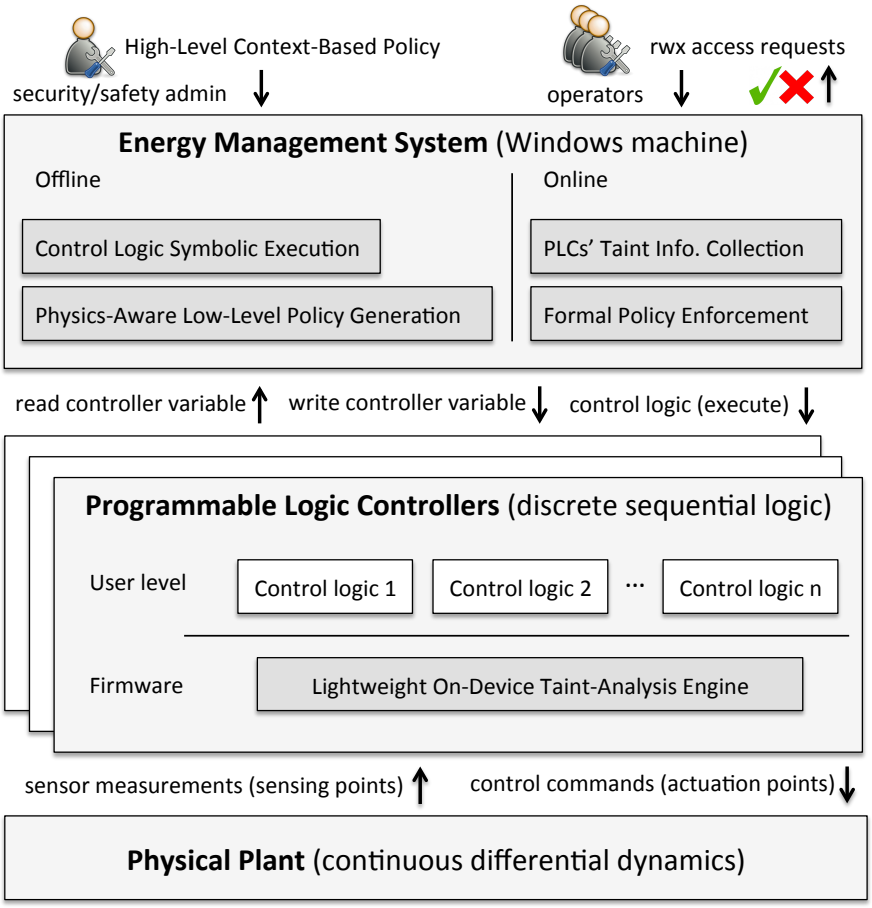


Figure 5.2: CPAC's High-Level Architecture

discussed in the introduction [4], and how CPAC could have prevented this failure.

While the guarantees that CPAC provides could be applicable to any cyber-physical infrastructure, we focus on its use as a security protection and access control solution for the smart power grid, with multiple PLCs receiving information and sending data back to an EMS. This setup is illustrated in Figure 5.2. In practice, each PLC often ships with proprietary engineering software running within the EMS. This is used both offline, for control logic development and execution on the PLC, and online within the EMS, for run-time monitoring and modification of a deployed PLC's variables. The PLC is also connected to the physical plant through lines from sensors within the plant that serve as input, and outgoing wires to actuators within the plant for process control.

3.1 Information tracking

CPAC facilitates security access control in cyber-physical power grid infrastructures and consists of two major components, one residing within the EMS and the other within the PLCs. As the PLC has limited computational resources and hard real-time requirements for processing data, any security solution must minimize performance overhead. To meet these requirements, CPAC offloads most computation from the PLC to a server at the EMS, which communicates with individual PLCs to obtain fine-grained information about device execution. We use offline pre-processing techniques to minimize run-time requirements. Given a new PLC control logic, CPAC symbolically executes the code and determines the source of incoming data for every output variable over all feasible execution paths. This information is stored in a lookup table. Consequently, rather than typical heavyweight run-time taint analysis, CPAC calculates the taint information through lightweight execution path profiling to minimize run-time overhead. CPAC's PLC-based dynamic analysis engine only tracks the execution path of the running control logic (Figure 5.2). Dynamic tracking of the execution path merely requires run-time monitoring for branch instructions on the PLC, a significant computational reduction compared to dynamic on-device byte-level taint analysis. CPAC uses lightweight control logic instrumentation before every control logic download on the PLC. The PLC-based agent sends collected execution path information to CPAC's EMS-side agent, which consults the symbolic execution lookup table for taint information regarding the affected sensing points.

Apart from the EMS, the device side must also be controlled. For example, Tom (Figure 5.3) could violate policy by downloading malicious or buggy control logic onto a PLC, or modifying its internal variables through the EMS interface. In either case, enforcing the policy requires analysis at the granularity of individual PLCs to calculate how Tom's actions would affect sensors and actuators throughout the plant. Therefore, before every control logic download to the PLC, CPAC performs an offline symbolic execution of the control logic (Figure 5.2) and fills out a lookup table where each entry represents an execution path of the control logic, and includes the corresponding path condition along with the symbolic values of the control logic variables at the end of the execution (scan cycle). Upon Tom's variable write or control logic download request, CPAC consults the lookup table for taint information, to determine

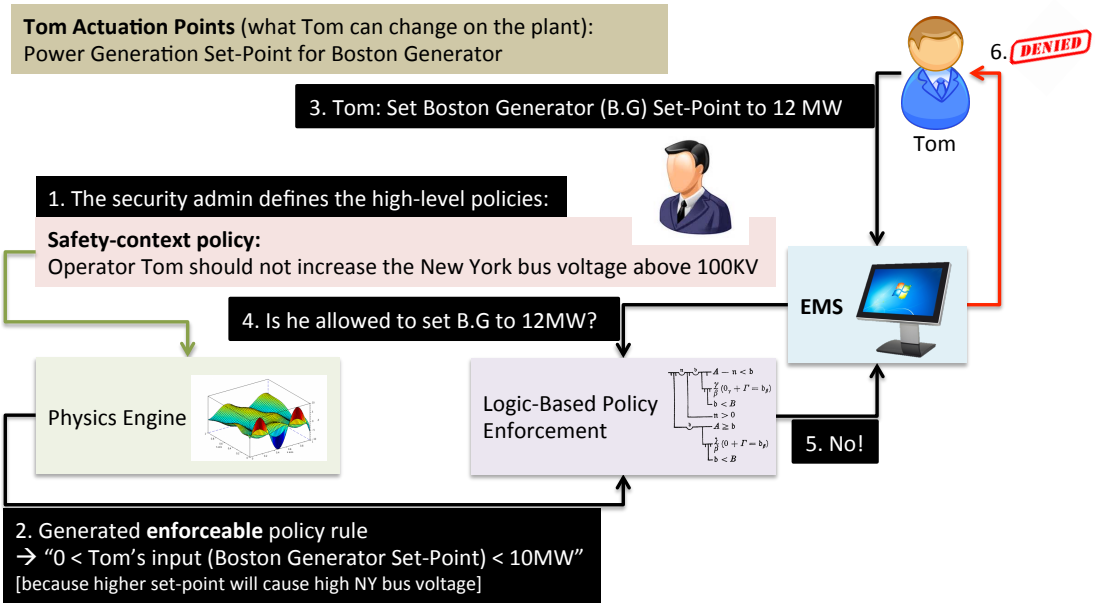


Figure 5.3: Physics-Aware Access Control

which actuation points may be affected by Tom's request. Such analysis considers changes throughout the entire power system, relying on information generated based on an information flow analysis performed after every topology server update.

3.2 Defining policies

Consider the workflow shown in Figure 5.3. An administrator defines a high-level safety-context plant policy, e.g., "Tom [a power operator] should not be able to cause the bus voltage on the New York power transmission line past its capacity 100kV." The policy is defined based on the transmission line's physical limitations, and exceeding the line capacity could potentially cause a line outage, redistributing the downed line's power through its adjacent lines [175] followed by a catastrophic blackout.[3] Intentionally or otherwise, Tom sets the Boston Generator set-point to 12 MW. In doing so, the physical model calculates that the New York transmission line would exceed 110 kV. The model is based on fundamental circuit laws that are dependent on the power system's topology, dynamically updated by the EMS topology processing server. Now the policy enforcement engine evaluates the new set point request and upon determining

[3]This situation is exactly what occurred during the Aug. 2003 Northeast blackout, which caused $6 billion in damage [54].

that granting this request would cause an unsafe state, denies the request, an result that is returned to Tom.

To be practical, CPAC must automatically enforce policies without requiring the administrator to redefine them on every system topology update. CPAC eliminates the need for this involvement through differential equation-based analysis of the EMS plant model, such that the safety policy described above is automatically enforced based on the current system topology.

CPAC's architecture enables policy enforcement to satisfy privacy, safety and regulatory requirements. For instance, a privacy policy may require that some system parameters or sensor measurements about a particular power system incident not be visible to certain operators. Privacy is not only important for preventing the data leaks from certain operators but also to prevent external attacker from knowing additional information which can lead to more effective attack;as an example, the web attack against a Ukraine power plant was caused by the attackers sending commands to open circuit breakers, creating power outage [2]. A safety policy may forbid increasing a line's current beyond capacity. By considering interdependencies between policies from different contexts, CPAC evaluates the whole system to determine the allowed actions.

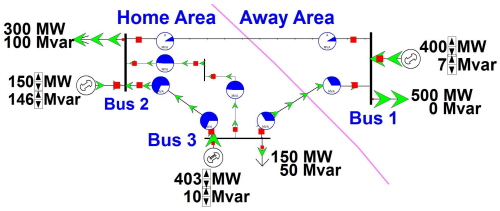
3.3 Case Study: California 2011 Blackout Emulation

As a demonstration of how the multi-layered design of CPAC allows it to maintain a secure environment, we demonstrate how CPAC could protect against a simplified emulation of the California 2011 blackout. For simplicity, we consider an EMS with an underlying four-bus power system (Figure 5.4a). We assume that the high-level safety and regulation-context policy rules for CPAC's enforcement are defined as follows:

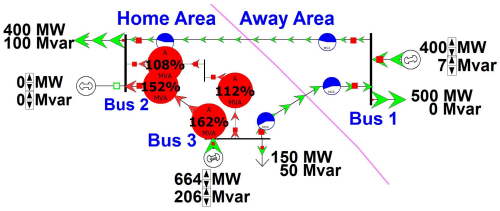
Safety policy: $I_l \leq 0.9 \cdot C(l) \quad \forall l \in L$

Regulation policy: $59Hz \leq f_b \leq 60.5Hz \quad \forall b \in B$

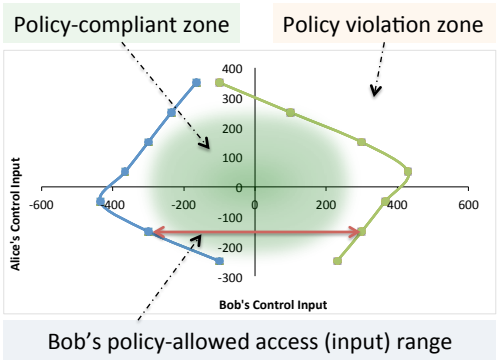
which requires current I on every transmission line $l \in L$ to be below 90% of the line's physical capacity C, and the AC power frequency f on each bus $b \in B$ to be within the government's mandatory NERC-CIP margins. A security administrator defining high-level policy does not need to define low-level technical details of allowable actions for individual operators, e.g.,



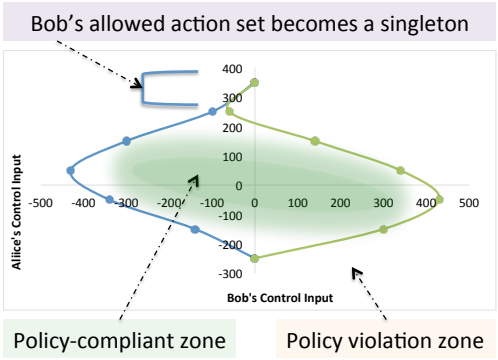
(a) Four-Bus Power System: Normal Operation



(b) California 2011 Incident Emulation



(c) Policy Region for when Relay is Closed



(d) Policy Region for when Relay is Open

Figure 5.4: Case Study Four-bus Power System and the Operator's Policy-Compliant Control Input Subspaces

whether an operator should be allowed to open a particular circuit breaker given the above policy, which is also dependent on the power system's topology and current state. CPAC extracts EMS-enforceable low-level policy rules automatically given the defined high-level policies and the plant topology. Were CPAC deployed, the California incident would not have occurred. Importantly, CPAC denies the operator's mistaken circuit breaker opening, which sparked the blackout. By preventing this action, we prevent a large power system frequency drop in the grid, which would violate the regulation policy. Additionally, opening the circuit breaker would cause line current overflows (Figure 5.4b), violating the safety policy.

To further clarify CPAC's range-based EMS-enforceable policy generation, consider the safety policy assuming two operators, Alice and Bob, who are in charge of controlling the power generation set-points on buses 2 and 3 respectively in Figure 5.4a. To apply their control, the operators could either directly change the variables on EMS screens or upload controller programs on the corresponding PLCs. CPAC receives the policy regarding the line currents, and calculates the allowed generation set-point ranges for Alice and Bob's access requests using Kirchhoff's laws shown in

Figure 5.4c, create a calculated policy-compliant region in Alice's and Bob's control input sub-space (the policy compliance zone extends to the edges of the left and right lines). The horizontal and vertical axes represent Bob's and Alice's one-dimensional action space, respectively. Note that the policy-compliant control input range for each operator depends on the system state caused by the other operator's control input value. For instance, if Alice requests a -150 write access to her control input variable on bus 2, Bob's allowed range will be limited to approximately $[-300, 300]$ illustrated by the bidirectional horizontal arrow in Figure 5.4c. CPAC calculates the plant's policy-compliant region every time the system's state changes since it changes for different plant states. Figure 5.4d shows the region for a different plant topology when the circuit breaker between buses 2 and 3 is open. In Figure 5.4d, Bob's allowed input value is constrained to a single value rather than a range if Alice's control input falls between $[280, 350]$, i.e., Bob's actions are constrained by Alice's inputs. It takes approximately 150 *ms* to calculate the region for large-scale plants (e.g., the Polish power grid; which we evaluate in Section 7).

4 Physics-Based Information Flow Analysis

CPAC leverages the underlying power system plant's mathematical model to perform physical-side information flow analysis. The power system is a nonlinear electric circuit, where system parameters are correlated according to corresponding equations that represent the physics model. Any perturbation of a particular system parameter causes updates across other parameters such that all values will comply with the equations. We define the physical-side information flow based on such inter-parameter value dependencies. For instance, changing the voltage difference on the two ends of a line with fixed resistance will cause its current update to satisfy the $V = I \cdot R$ relation. CPAC considers this to be an information flow between V and I, because measurement of the line current reveals information about the changes in voltage difference of the two ends.

An n-bus power system's dynamic behavior can be represented by parameterized differential equations [89]:

$$\dot{x} = f(x, u, \lambda) \tag{5.1}$$

where f is a continuously differentiable function representing the physical plant's dynamic behavior; $x \in \mathcal{R}^{2n-1}$ represents the system state vector that includes the voltage magnitude and phase angles for each bus; $u \in \mathcal{R}^{m}$ represents the plant's control input vector that could be manipulated by the operators, such as generator set points; λ represents a vector of discrete events that change the plant's topology, and hence its continuous differential equations.

The sensor measurements are correlated with the plant state and the operator's control inputs through

$$w = h(x, u) \tag{5.2}$$

where w is the sensor measurement vector, and h is called the measurement function. CPAC's physical-side information flow analysis leverages the sensitivity investigation of the plant's differential equations given any stable point x_0 and calculates the margin by which each system parameter changes due to physical dependencies if a particular control input is applied to the

system. CPAC marks control input actuation points as sources, and every sensing point (measured system parameter) with change margins larger than a predefined threshold ε as the corresponding information flow sinks, ignoring negligible change margins that cannot be practically recognized due to sensor noise. CPAC uses the calculated information about sink parameters to later enforce access control policies. For instance, an operator may be denied applying a particular control input value because she should not be allowed to impact a remote sink parameter beyond a limit or at all based on the safety or confidentiality/privacy context policies.

CPAC determines the allowed value ranges for individual actuation points of the plant that do not violate physics-based policy rules or sensitive parameter changes, e.g., an overloaded transmission line (safety-context policy violation) or a confidential load disclosure (privacy-context policy violation [127]). We call the control input values, beyond which the system enters the policy-violating states, the *boundary points*. The policy boundary margin M is defined as

$$M = |u_* - u_0| \tag{5.3}$$

where $u_* \in \mathbb{R}^m$ represents a policy boundary point (vector) and $u_0 \in \mathbb{R}^m$ is plant's input at equilibrium or stable state. CPAC uses the difference M to either allow or deny an operator's request for an actuation point change, i.e., requests that exceed the calculated range are denied. CPAC performs this analysis for individual operators separately to calculate their corresponding allowed actuation point value ranges.

CPAC implements the physical information flow analysis through dynamic behavior inspection and sensitivity analysis of the plant around Equation 5.1's equilibrium state:

$$\begin{aligned} f(x_0 + \Delta x, u_0 + \Delta u, \lambda_0 + \Delta\lambda) \approx f(x_0, u_0, \lambda_0) + \\ + f_x \Delta x + f_u \Delta u + f_\lambda \Delta\lambda \end{aligned} \tag{5.4}$$

First-order Taylor series expansion of Equation 5.1 around its equilibrium state is given by Equation 5.4 which uses the power plant's vector-valued function partial derivatives $f_x = \frac{\partial f}{\partial x}(x_0, u_0, \lambda_0)$, $f_u = \frac{\partial f}{\partial u}(x_0, u_0, \lambda_0)$, and $f_\lambda = \frac{\partial f}{\partial \lambda}(x_0, u_0, \lambda_0)$ which are nonlinear Jacobian matrices given in Figure 4 of Section 1. x_0, u_0 and λ_0 are values at stable or equilibrium state. Assuming

that f_x is non-singular, we can reorder Equation 5.4 as follows

$$\Delta x = -f_x^{-1} f_u \Delta u - f_x^{-1} f_\lambda \Delta\lambda \tag{5.5}$$

which formulates how the power plant's state changes every time an operator modifies an actuation point. Equation 5.5 shows the physical-side information flow between the actuation points and the state variables. This is useful for an operator's `write` access control, where the operator request to apply a control input and the policies are defined to prevent the system from entering unauthorized (e.g., unsafe) states. However, actuation point-to-state vector information flow analysis is not sufficient for read access requests, where the operator requests to see a particular sensor measurement, e.g., transmission line current, that is often not the same as a state variable, i.e., power bus voltage magnitude and phase angles. To support read access requests, CPAC implements actuation point-to-sensor measurement information flow analysis to determine how each sensor measurement is affected as the result of a control input application anywhere in the system. Following Equation 5.2's first-order Taylor expansion around its equlibrium (x_0, u_0), gives us

$$\Delta w = [w_u - w_x f_x^{-1} f_u]\Delta u \tag{5.6}$$

where the changes in measurements Δw that the operators could have read access request for are calculated as the result of any change in the system Δu. The Jacobian matrices w_u, w_x and f_x are in Figure 4 of Section 1. For more accuracy, second-order Taylor expansion is given in Section 1. These Taylor series expansions around the equilibrium points are used by the system to determine boundary points after perturbing around its equilibrium points and the physics equations are used to determine the information flow between different objects or parameters.

5 Logical Policy Enforcement

A[4] key of the EMS is the HMI used by operators to facilitate checking process states, system variables, and control system settings within the physical plant devices. Most software still relies on user name and password input as the sole method of authentication and authorization.

[4]This work was completely done by Dave (Jing) Tian from University of Florida and added in my thesis to understand the complete picture of the solution.

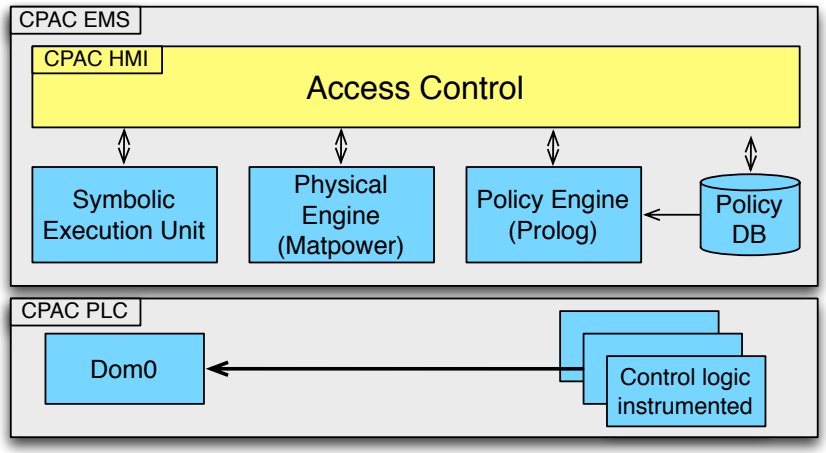


Figure 5.5: The CPAC EMS/PLC architecture.

Some systems contain elements of role based access control (RBAC) [164], where certain roles are limited to certain operations through the EMS. However, RBAC requires administrators to examine all available operations provided by the EMS, assuming a static policy. Consider a trivial case where Alice cannot view the voltage or current value of a generator G_i, based on a policy that Alice should not know the working status of that generator. Alice can still learn this information by checking the temperature of G_i. These policies become more complicated when the interaction of different operations cannot be detected until run-time. Additionally, support for storing detailed provenance [38] of applications is lacking. Most EMS software provides some logging abilities to record user activity, however, these logs are mainly designed for postmortem analysis rather than policy enforcement, where incorporating provenance could allow additional fine-grained policy controls. For instance, we may want to add restrictions dynamically to operators who tried and failed certain operations over a time window.

5.1 Context-Aware Policy Control

Policy-based access control has been well studied and solutions including MAC, RBAC and capabilities have been applied into commodity computer systems [214, 172, 12]. There are also policy specification languages, such as SPL (Security Policy Language) [160] and RDL (Role Definition Language) [125]. Unfortunately, as we have mentioned above, none of these fits perfectly into the requirements and setting of cyber-physical systems (CPS), which requires,

we argue:

- *Information flow control:* Unlike normal policy control systems, whose target are processes, CPS also need to control the information flow of a task/process, guaranteeing no sensitive information leakage[5].
- *Context awareness:* Not only user names, but also time epochs, locations (e.g., IP addresses) and detection of events (e.g., voltage outages) are needed to make policies more useful and practical.
- *Provenance-awareness:* All operations should be logged to allow the use of provenance data to support policies based on user historical behaviors.

To support finer-grained policy, the CPAC EMS consists of a general Modbus [9] transport layer from pvbrowser [11], a HMI access control terminal, the physics engine, the symbolic execution unit, and the policy engine to enforce the policy control and provide provenance support as shown in Figure 5.5. The transport layer (not shown in the figure) communicates with the PLC via the Modbus protocol (widely supported by most PLCs) over TCP, since we aim for the EMS to be independent of the PLC hardware, while the PLC is running instrumented control logic. The HMI within the EMS provides basic user authentication and accepts operation requests. Both the physical engine and symbolic execution unit provide input for the policy engine, which attempts to authorize operational requests based on policies and adds provenance meta data to these requests for future decision making.

5.2 Policy layers of CPAC

We define three further requirements for a policy control implementation: simplicity of writing policies, correct and potentially formally verifiable policy control logic, and low operational overhead. Under such considerations, we implemented our policy engine using Prolog[6], transforming a policy enforcement query into a logic reasoning process. There are four layers in the

[5]Thanks to the symbolic execution unit on the EMS side, and the nature of PLC control logic (less branches comparing to normal x86 binary), taint tracking per task is possible. However, this does not mean the adversary could not learn anything from the running program, e.g., via timing side channels.

[6]Prolog code can be compiled with the native C/C++ code to generate the binary executable, which runs much faster than its interpretive mode.

policy engine, each in charge of a different policy enforcement task. Each layer is evaluated in order until a layer results in a check failure or there are no more layers to check.

1. **Physical Layer** When CPAC EMS receives an operation permission request from an EMS, it determines whether an operation is physically possible using the analysis described in the previous section. For example, the temperature of CPU should be only readable but not writable.

2. **MAC Layer** This acts as a capability system enforcing which users can do what operations on which variables on a PLC. For instance, Alice is able to read the voltage, the current and the temperature of generator 1, but only to write/change the voltage value. This and the physical layer implement the security features most EMS software share. However, unlike traditional implementations, CPAC counts on logic rules as policies and reasoning as permission checking.

3. **Taint Analysis Layer** This layer uses taint tracking information from the PLC to find information leakage missed by the two previous layers. A trivial example may be that since Bob is not allowed to read the voltage value, he should not be allowed to read the current or temperature either. The taint analysis layer supports both the predefined static taint information (which can be derived from physical modeling), and dynamic tainting provided by the symbolic execution unit and the taint tracking enabled PLC (Section 6 gives more details).

4. **Context/Provenance-Aware Layer** This layer leverages the time, locations, events and provenance to check for permissions (e.g., operations are only allowed during the day from certain IP address for Alice; Bob is not allowed to access variables if a generator fails). As with the above layers, all policies are written in logic rules and facts, and the permissions check is a matter of querying or reasoning.

Both the physical and MAC layers generate static policies, which check for the legitimacy of operations. Passing these two layers proves the validity of an operation request from the traditional access control point of view. The next two layers then try to refute the request using the dynamic tainting information and current running context. Note that CPAC does not try

to blacklist all possible illegal operations, number of which may be infinite. Instead, CPAC enables system administrators to retrospect a legitimate operation request in a rich context.

5.3 Formal description of CPAC

To grant permission for an operation, the EMS submits a query to the logic rule *cpac_granted*, which is defined with seven arguments $\{\mathcal{T}, \mathcal{L}, \mathcal{U}, I, \mathcal{N}, \mathcal{W}, \mathcal{V}\}$, representing timestamps, locations, users, operations, PLC variable names, new values (if written) and the current value of all the PLC variables respectively, as shown below. Note that $\mathcal{V}$ could be viewed as a global variable, whose value is visible to all rules in the Prolog engine, even though it may not appear in each logic rule. It represents values of all variables available on the PLC when the query is submitted.

$$cpac\_granted(\mathcal{T}, \mathcal{L}, \mathcal{U}, I, \mathcal{N}, \mathcal{W}, \mathcal{V}) \leftarrow physical\_granted(\mathcal{N}, I, \mathcal{W}) \wedge$$
$$mac\_granted(\mathcal{U}, I, \mathcal{N}) \wedge taint\_granted(\mathcal{U}, I, \mathcal{N}) \wedge context\_granted(\mathcal{T}, \mathcal{L}, \mathcal{U}, I, \mathcal{N}, \mathcal{W}).$$

physical_granted grants the permission if the I/O operation is read. Otherwise, it checks if the variable in PLC is writable and if the new value to be written is in the legal range. Note that this layer tries to check the permission from the point of PLC's constrains without considering any other policies. The system administrator is responsible for providing legal ranges for all variables based on specifications, and writing them in the format of Prolog facts (e.g., $svi(voltage, 0, 10, rw)$ shows the name of the variable, the minimum value, the maximum value, and the possible I/O operations), which can be used by the *in_range* rule (and other rules) directly.

$$physical\_granted(\mathcal{N}, I, \mathcal{W}) \leftarrow read(I) \vee (write(I) \wedge writable(\mathcal{N}) \wedge in\_range(\mathcal{W})).$$

mac_granted grants the I/O operation based on the user's capabilities. For all the variables exported by the PLC, the system administrator should assign different permissions to different users. This layer implements the general access control applied by most EMS systems. Within the layered access control structure in CPAC, *physical_granted* comes first. This means even if the operation would be allowed by *mac_granted* through the user access control policy, it may be denied based on rules defined by the physics layer. The layered approach in CPAC

thus provides more modular access policies. To add a new user or modify existing policies, the system administrator only needs to create or modify the corresponding Prolog facts, such as *cap_read(bob,[current])* (giving bob the permission to read variable *current* (only)).

$$mac\_granted(\mathcal{U}, I, \mathcal{N}) \leftarrow (read(I) \wedge cap\_read(\mathcal{U}, \mathcal{N})) \vee (write(I) \wedge cap\_write(\mathcal{U}, \mathcal{N})).$$

taint_granted determines whether the target variable could be tainted by some other variables not visible to this user, and rejects the operation accordingly to avoid data leakage. CPAC supports both taint analysis by writing the static taint rules directly and the dynamic taint tracking provided by the symbolic execution unit and PLC during the run time. This layer uncovers missing policies not easily found in the traditional access control implementations. $\mathcal{Z}$ stands for all the variables visible to the EMS side (same as the $\mathcal{N}$ used in *cpac_granted*, such as the temperature and current). Both the static and dynamic rules share similar Prolog construction, *taint_X(z1, z2)*, meaning variable *z1* tainted by variable *z2*. As shown below, if variable *z2* cannot be accessed by this user, the request for accessing variable *z1* would be rejected.

$$\begin{gathered} taint\_granted(\mathcal{U}, I, \mathcal{N}) \leftarrow \forall z \in \mathcal{Z}: \\ ((\neg taint\_static(\mathcal{N}, I, z)) \vee (taint\_static(\mathcal{N}, I, z) \wedge cap\_read(\mathcal{U}, z))) \wedge \\ ((\neg taint\_dynamic(\mathcal{N}, I, z)) \vee (taint\_dynamic(\mathcal{N}, I, z) \wedge cap\_read(\mathcal{U}, z))). \end{gathered}$$

context_granted leverages contextual information to help system administrators write polices fitting into their specific domains, (e.g., the power grid). To simplify the rule/policy writings, we introduce an event-driven reasoning framework and fix the default action of policies to be operation blocking. The final permission granting is then the conjunction of negations of all the *blocking* rules, which are *context_denied_X*, where *X* is an integer used to differentiate all these Prolog rules.

$$\begin{gathered} context\_granted(\mathcal{T}, \mathcal{L}, \mathcal{U}, I, \mathcal{N}, \mathcal{W}) \leftarrow \\ (\neg context\_denied\_0(\mathcal{T}, \mathcal{L}, \mathcal{U}, I, \mathcal{N}, \mathcal{W})) \wedge (\neg context\_denied\_1(\mathcal{T}, \mathcal{L}, \mathcal{U}, I, \mathcal{N}, \mathcal{W})) \wedge (...). \end{gathered}$$

All *blocking* rules are event-driven and follow the same construction. Note that all events should be predefined by the system administrator based on domain knowledge. One simple

example is *event_g0_failure(V) :- g0_power=<0.*[7]. With event $\mathcal{E}$ defined, the *blocking* rule is defined as below. Given user U's access request on variable N, if event E happens, and the corresponding rule *context_policy_block* contains N in its blocking list B, the context layer will deny the request.

$$context\_denied\_X(\mathcal{T},\mathcal{L},\mathcal{U},I,\mathcal{N},\mathcal{W}) \leftarrow$$
$$event\_E \wedge contex\_policy\_block(\mathcal{T},\mathcal{L},\mathcal{E},\mathcal{U},\mathcal{B}) \wedge member(N,B).$$

Below we demonstrate a real code snippet within CPAC. We choose a complicated policy to demonstrate the ease of policy writing once the corresponding event is predefined by the system administrator. This rule states that for any condition, once generator 0 (`g0`) fails, the temperature value of that generator should not be visible to the operator 'dave'[8].

```
context_policy_block(_,_,g0_failure,dave,[temp0]).
context_denied_0(T,L,U,I,N,W) :-
event_g0_failure(_),
context_policy_block(_,_,g0_failure,U,B),
member(N,B).
```

To support provenance both for forensic analysis and run-time provenance-based policy enforcement (e.g., an event related with user's previous operation history), CPAC records each operation request from the EMS side, either granted or denied, both in a standalone provenance logging file and the Prolog engine as a 'fact', using the unified format:

$$provenance(\mathcal{T},\mathcal{L},\mathcal{U},I,\mathcal{N},\mathcal{W},\mathcal{R},\mathcal{V}).$$

Here $\mathcal{R}$ stands for the final result for this operation request (granted (g) or denied (d)) and $\mathcal{W}$ is reused to hold the return value for read operations, as well as the new value for write operations. Other variables are the same as the ones in the rule *cpac_granted*. A concrete example is shown below, where user `dave`'s request to read variable `temp0` from IP address `10.10.10.10` at time `2015071411550` was granted, with all other variable values at that time dumped in the list.

[7]Note that in this Prolog rule, argument V is not used at all, since this event call be determined solely by checking the power of the generator. Complex events can have multiple arguments and take full usage of them.

[8]In Prolog, '_' is wildcard, meaning that the value of that variable does not matter.

```
provenance(20150714115507, 10-10-10-10, dave, r,
temp0, 3000, g, [3000,4000,5,40,38,17,15]).
```

With more provenance added into the CPAC EMS Prolog engine, making provenance-aware polices is possible. For example, users with more than 10 denials within an hour could be blocked, as the user account may have been compromised. Also any unseen IP address used by a certain user could be blocked, which actually implements a naive intrusion detection mechanism. Since all provenance is also saved into a standalone logging file, this file can be loaded into the Prolog engine every time the EMS is restarted. With the help of the Prolog interpreter, one could submit queries, such as "who read the variable `temp0` in the past but was denied" (`provenance(_,_,X,r,temp0,_,d,_).`), and Prolog would find all users satisfying the query.

5.4 Trade-offs

Besides all the desired requirements of implementing a policy enforcement component mentioned before, one of the biggest concerns using a logic programming language to write policies is how easy it would be for system administrators to use. As shown before, the logic reasoning framework is already provided, as well as some sample constructions. Other than the definition of events, which gets complicated when the event itself is complex, CPAC EMS expects only simple inputs from users, such as the range of certain variable, and permissions for certain user. In general, we believe the advantages of using Prolog to implement the policy control outweigh the impediment of writing simple Prolog facts.

Another concern comes from the scalability issue when writing policies for real-world complex systems, e.g., a nation-wide power grid system. As we will show in Section 7, the final Prolog policy file used to simulate the Polish power grid containing more than 2700 buses is almost 1 MB, with 25K lines, among which, more than 24K lines are simple Prolog facts mentioned before and generated by a Python script. While most policies can be generated automatically given the system specification, the definitions of events used by the context-aware policies need human intervention with specific domain knowledge. As shown in Figure 2, writing in Prolog is straightforward and does not provide extra obstacles comparing to writing in

other policy languages[9].

6 Device Level Information Flow

CPAC deploys its dynamic information flow analysis through a lightweight instrumentation of the VxWorks real-time operating system, which is widely used within industrial control systems (40% of the market share [14]) and mission-critical settings, such as the Mars Curiosity rover [116]. To support such an environment, CPAC must meet two requirements: *i)* very low run-time performance overhead to prevent missed real-time deadlines for PLC-level workflows; and *ii)* very high taint analysis accuracy to prevent possibly fatal safety hazards.

Traditional information flow techniques for x86 architectures using byte-by-byte data flow tracking solutions cannot be applied due to their unacceptable run-time execution slowdown (e.g., 6X by BitBlaze [181]). Several proposals considered how to speed up dynamic taint analysis on resource-constrained devices [67] and how to extract semantic information [8]. While useful for desktop and smartphone applications, these solutions will not meet real-time deadlines (e.g., 15% overhead by [67]). As an alternative to dynamic taint analysis, static techniques remove the run-time performance problem. However, the strict accuracy requirements for control system applications limit their practicality significantly due to their well-known high false positive rates. A false positive taint analysis outcome in CPAC could potentially lead to denying an operator's legitimate access request to take care of an emergency situation. Consequently, neither dynamic nor static techniques by themselves can address both the above-mentioned control system requirements completely.

We use a hybrid approach with CPAC, leveraging specific features of the PLC execution logic and VxWorks architecture to ensure high taint analysis accuracy and minimize operational intrusiveness. In practice, PLC controller programs include far fewer branch instructions than x86 binaries. This facilitates comprehensive offline analysis of the controller before its launch time due to less path explosion. CPAC implements the PLC code symbolic execution as discussed in [128] on its EMS-side modules. CPAC uses the symbolic values to obtain the

[9] In reality, such a large-scale system is usually divided into multiple sub- areas, which are maintained by different system administrators. A global policy can be defined using predefined local events from sub areas rather than dealing with thousands of variables directly.

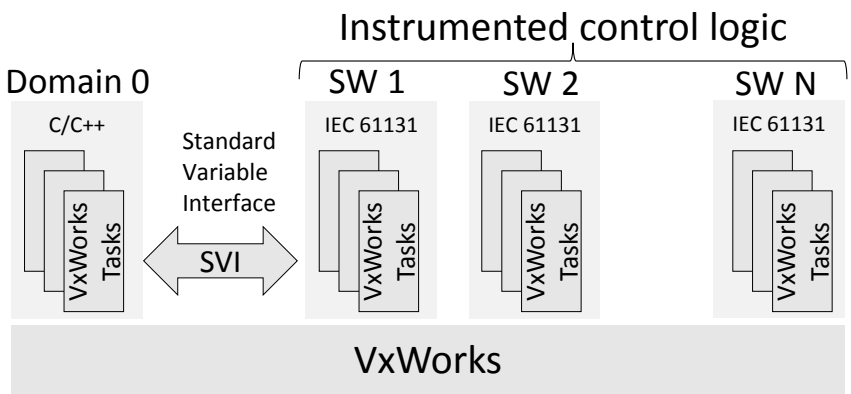


Figure 5.6: Domain 0 along with instrumented control logic

taint information for all program variables including the outputs, depending on which input values every program variable is tainted by. It creates a taint look-up table and uses it to speed up its run-time performance remarkably. Moreover, CPAC's run-time modules on the PLC do not have to implement full dynamic taint analysis, but instead just profile only the execution paths taken by the control logic. The small number of branch instructions in typical control logic that need profiling further motivates CPAC's approach. The EMS receives the execution path profiles and consults the look-up table to determine the taint information needed for the access policy enforcement.

Figure 5.6 shows CPAC's device-level module on a Bachmann MX231 controller running VxWorks. The main module (so-called *Domain 0*) is written in C/C++ and inserted into the PLC's kernel as a *.m* binary file. VxWorks allows several control logic modules to run simultaneously on the PLC. Before every control logic is uploaded into the PLC by the EMS, CPAC instruments the program with a lightweight inline reference monitor using the PLC instructions (IEC61131 [109]) to profile the control flow. Domain 0 dynamically collects control flow information from the running controller programs through the VxWorks standard variable and module interfaces that enable on-device remote procedure calls [109]. It then transfers the information to EMS modules over Modbus to perform taint analysis and policy enforcement. To minimize overhead, we only collect control flow information on demand, i.e., only if there is a corresponding access request from the EMS.

7 Evaluations

We evaluated CPAC on two power plants, the four-bus system (Figure 5.4a) and Poland's publicly available power grid with over 2,700 buses (>2,800 transmission lines) since other networks are not publicly available. We extended the open source pvbrowser v4.7.9 [11] EMS (2-core Intel 2.40GHz; 4GB memory) with our logic-based access control engine. CPAC's power system analysis module uses MatPower [225]. The PLC-based taint analysis and the symbolic execution implementations in CPAC are specifically deployed for the Wind River VxWorks operating system v5.5 running on a Bachmann MX231-Controller PLC with a GIO 212 IO modules. We then designed a set of experiments to verify whether CPAC can be useful and practical in real-world scenarios by answering the following questions empirically:

1. How accurately would CPAC prevent past real-world control system and power grid severe incidents?
2. How efficiently does CPAC perform the PLC-based taint-analysis, physical-side information flow analysis, and EMS-side logic-based policy enforcement?
3. How well does CPAC scale up for large-scale real-world control systems and power grid infrastructures?

7.1 Case Studies

To answer these questions, we validated CPAC's functionality and performance across six use cases. The first three scenarios are based on the four-bus power system; they are derived based on our practical experience and interactions with power utilities to highlight CPAC's capabilities in a typical power grid infrastructure. We assume there are two operators: Bob, a control operator on Bus 2, and Alice, a maintenance operator for the home area on the grid. Figure 5.4a shows the power system and its two areas separated by a line (the left area is the home area). The following scenarios list the policies for Alice and Bob in different contexts. Case A is below for intuition and the other two scenarios on four-bus power systems (case B and case C) are described in Section 1, while we directly discuss the real world scenarios which occurred in the past.

Case A: Read access control for crucial plant values. Alice, as the maintenance operator,

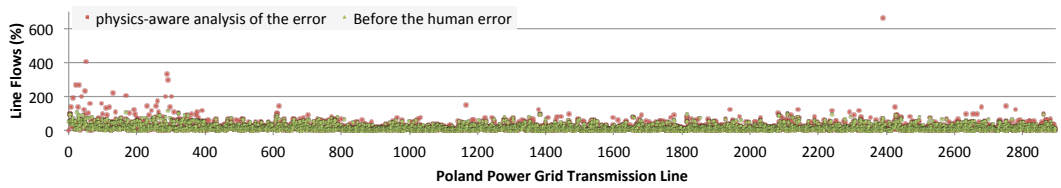


Figure 5.7: The system capacity overload state in case E (Section 7.1). Note that one line has been overloaded to 661% of it allowable current, a situation that CPAC would prevent from reaching.

requests to see the real-time transient power output of the generator on Bus 2. The value represents a PLC variable within a droop control logic [57] that controls the generator's power output through its governor.

Source of incident: A lack of enforcing confidentiality over sensitive control data.

Required access control policy: Only control operators are allowed to see sensitive plant control values (defined as generators' real-time frequencies).

Effects of CPAC *deployment:* CPAC denies Alice's request due to the potential for sensitive data disclosure. CPAC's PLC-based information flow analysis marks the target variable tainted by the incoming frequency measurements, which is not readable by Alice. The droop control correlates generator frequency and output power such that knowledge of one value could be used to infer the other.

The next three scenarios are based on real-world power grid incidents that had large-scale effects on millions of power grid customers, some of which were international in scope. Though N-1 contingency was enforced, due to cascading failures these events occurred. We evaluate how CPAC could have prevented these incidents by simulating their effects on the real-world model of Poland's entire power grid interconnect, consisting of over 2,700 buses. This will demonstrate CPAC's scalability to national-scale grid environments.

Case D: Southwest 2011 blackout. The Southwest (California) blackout affected 7M people in California, Arizona, and Mexico, which we describe in detail in Section 3.3. The reports from Federal energy regulatory commission (FERC) showed that "the system was not in an NERC-CIP N-1 compliant state. Utilities are required to operate the system so that the malfunction of one component can not cause instability, separation, or cascading" [4].

Source of incident: Human error from an operator violating compliance with NERC-CIP N-1

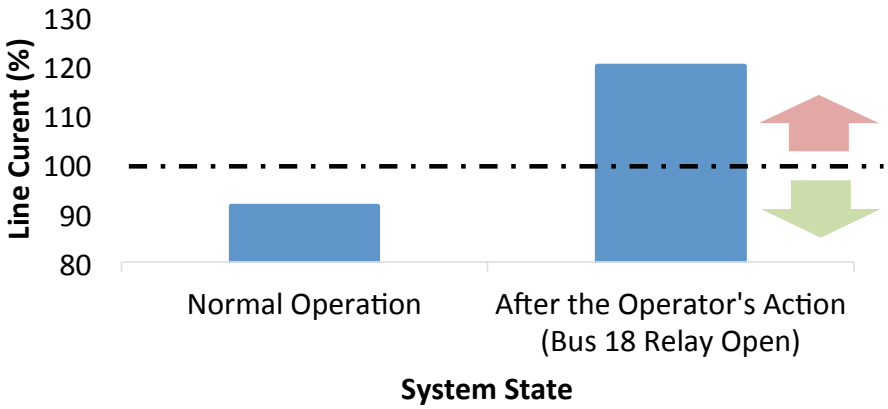


Figure 5.8: Southwest Blackout Prevention using CPAC. On evaluating the effects of line current on bus 18 after opening the relay. CPAC determines line would be overloaded and prevents the action.

contingency regulations.

Required access control policy: No operator may issue a control command that puts the grid in a state that violates the CIP N-1 requirements.

Effects of CPAC *deployment:* Figure 5.8 shows the line current on bus 18 before and after the operator opened a relay on a different line. CPAC speculatively calculates the potential global impact of the operator's action and denies the action, as it would lead to an unacceptable current flow on the line that violates the NERC-CIP N-1 requirements.

Case E: Florida 2008 grid blackout. The Florida Power and Light (FPL) Company reported a widespread grid blackout occurring at the Flagami substation in west Miami as a field engineer was diagnosing a switch that had malfunctioned. Contrary to standard procedures, the engineer disabled two levels of relay protection. Because both levels of protection had been removed, the arc that resulted from the fault caused an outage that spread through the grid as power plants and transmission lines tripped off-line to protect themselves. "Standard procedures [unenforced] do not permit the simultaneous removal of both levels of protection," the utility wrote [7].

Source of incident: Human error from an operator disabling internal redundancy protections, causing a maintenance operation to disrupt the grid's real-time operation.

Required access control policy: Operators must not remove both levels of relay protection simultaneously.

Effects of CPAC *deployment:* CPAC denied the operator's second protection removal request

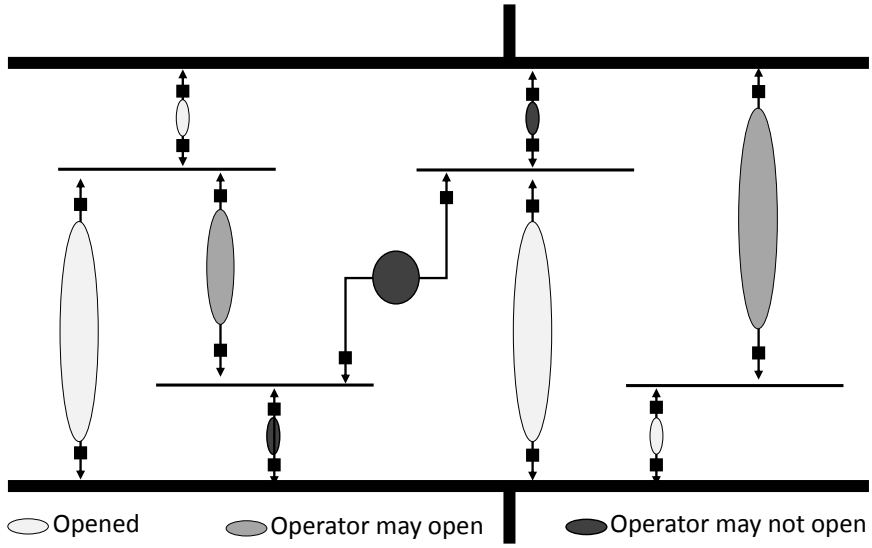


Figure 5.9: Columbian Blackout Prevention via CPAC. On evaluating the effects of opening the critical relays after few relays are opened. CPAC determines line would be overloaded and prevents the action.

due to the policy. Additionally, it overloads the most of the power components (Figure 5.7). Bus #2,392 is overloaded to 661%, which would quickly cause a line outage and damage neighboring assets due to the extremely high current.

Case F: Colombia 2008 total blackout. Colombia suffered a total blackout affecting 25 million people due to human error at the 230 KV Torca substation. An operator at the substation did not follow the correct (but unenforced) sequence of maneuvers when transferring circuits from one busbar to another within a substation before a scheduled maintenance task. The wrong maneuver overloaded the inter-bus breaker, and the breaker malfunction de-energized the whole Torca substation, igniting a cascade of events that brought down the entire Colombian electric power system.

Source of incident: Human error from an operator not correctly following substation interlocking procedures, and lack of enforcement to ensure these procedures are followed.

Required access control policy: The maintenance operator's sequence of busbar interlocking actions must not overload any inter-bus breaker.

Effects of CPAC *deployment:* CPAC denied the operator's action request as it violates the interlocking requirements. Figure 5.9 shows the substation-level busbar configuration where

the top and bottom busbars are connected to neighboring substations, and should never be disconnected from each other. The figures shows the set of already opened breakers, the ones that the operators could open, and the breakers that are prevented from opening, resulting in an operator deny action by CPAC as they separate the two main busbars affecting the electricity grid globally.

7.2 Performance

We measured CPAC's performance for all six scenarios. Table 5.1 shows the Prolog engine's execution time for scenarios *a-c* averaged over 20 runs. CPAC takes under 0.3 *ms* to process the access request and render a policy decision. This quick processing time is due in large part to our optimized implementation, where we compiled the logic into assembly using the `gplc` compiler. Table 5.2 shows the corresponding overhead for the domain 0 to launch taint tracking within the PLC. Most taint information collection could be done within 100 *ms*. As domain 0 is implemented as a standalone kernel module with the lowest priority, we have minimized the impact of domain 0 on other PLC tasks. On the EMS side, there are 30 power system variables in scenarios *a-c* that an operator may be able to see based on policy. As Table 5.3 shows, CPAC's EMS modules completes all these scenarios within 40 *ms*. Given the general EMS OS overhead and transmission delays (e.g., the 5 minute time requirements by NERC for EMS-side contingency analyses [143]), CPAC's overhead will be minimal to operators. Note that CPAC's EMS modules include the physics engine, the Modbus transport library and the Prolog policy engine.

We measured scenarios *d-f* using the topology of the entire Polish power system, comprising over 2,700 buses. Table 5.4 shows the general overhead of CPAC's physics engine with these real-world cases. The physics engine is able to finish the forward analysis within approximately 100 *ms*. The result was computed using MATLAB and will likely be even faster if the engine is developed in C/C++. Table 5.5 shows the overhead of the Prolog policy engine, reasoning about 1,000 simultaneous variables. For the three cases, the Prolog engine completed policy analysis in approximately 15 *ms*, due to our compilation of logic into native assembly. The overhead of the full analysis (without the overhead of user operations and network transmission delay) is within 150 *ms* (100 *ms* from physics engine using MATLAB, 15 *ms* from

Table 5.1: Prolog Micro-Benchmark (*us*).

Scenario	Min	Avg	Max	Mdev
(a)	125.0	154.9	205.0	21.1
(b)	147.0	186.5	235.0	21.9
(c)	176.0	214.2	280.0	29.8

Table 5.2: Domain 0 and instrumented taint (*ms*).

Scenario	Min	Avg	Max	Mdev
(a)	90.909	96.871	100.200	3.974
(b)	94.787	97.711	99.338	1.949
(c)	90.909	96.693	99.668	3.856

Table 5.3: EMS Macro-Benchmark (*ms*).

Scenario	Min	Avg	Max	Mdev
(a)	30.961	31.376	33.991	0.600
(b)	30.933	31.571	32.976	0.601
(c)	29.979	30.442	32.994	0.601

Prolog policy engine, 30 *ms* from EMS).

7.3 Scalability: NERC-CIP N-x Compliance

The state-of-the-art NERC-CIP v5 standards[10] protect the power grids against single component malfunctions. However, extensive research [47] has shown the insufficiency of single failure consideration because of increasing complexity of existing smart grids, and more importantly, the possibility of cyber attacks with (automated) subsequent component exploitations. Up to now, guaranteed $N-x$ compliance has not been scalable or feasible in practice. The main reason is that, to fully support $N-x$ contingencies, existing systems must analyze

$$\sum_{i=1}^{x} \binom{N}{i} = N + \frac{N(N-1)}{2} + \cdots + \frac{N!}{x!(N-x)!} \tag{5.7}$$

different contingencies that each require independent full solution of the power system. Continuing along these lines, one could show that for k simultaneous outages, $O(N^{x+1})$ power flow

[10] Available at http://www.nerc.com/pa/CI/Pages/Transition-Program.aspx

Table 5.4: Physics engine Macro-Benchmark (*ms*).

Scenario	Min	Avg	Max	Mdev
(d)	102.048	102.945	104.413	0.653
(e)	100.982	101.571	102.4825	0.644
(f)	97.626	98.116	98.886	0.285

Table 5.5: Prolog Macro-Benchmark (*ms*).

Scenario	Min	Avg	Max	Mdev
(d)	8.000	14.750	19.000	2.175
(e)	8.000	14.600	17.000	2.080
(f)	8.000	15.250	20.000	1.600

solutions[11] are required to process the contingency list. For practical power systems, the number of lines tends to scale linearly with the number of buses B in the system ($N \in [B, 1.5 \cdot B]$). $N-x$ compliance thus requires $O(B^{x+1})$ power flow solutions. In the Polish system, where $B = 2,746$, $N-2$ and $N-3$ compliance require $> 3.7M$ and $> 3.4B$ contingency considerations, respectively. Figure 5.10 shows the results for different number of contingencies. Each contingency takes approximately 2.4 seconds to complete, and power utilities mostly run contingency analysis procedures every 5 minutes. Consequently, traditional methods do not scale up to existing strict requirements and complex grid infrastructures. Several recent efforts attempt to provide $N-2$ contingency analysis support [56, 226]; however, they are not exhaustive, and instead selectively choose and analyze particular contingencies. Consequently, the previous work may miss a contingency that may occur in practice, resulting in incorrect NERC-CIP compliance assurance. Additionally, they do not consider multiple (more than two) subsets of contingencies, i.e., they miss a combination of small contingencies that collectively contribute to a large-scale power grid blackout. *None of the traditional solutions can handle this intractable search space.* CPAC takes an alternative approach that enables $N-x$ contingency analysis even in large-scale systems. Traditional contingency analysis techniques are offline, and need to complete their analysis before any incident occurs or is about to happen. CPAC's policy enforcement framework instead takes a run-time approach analyzing any sequence of incidents before it determines whether they violate requirements. In case of a violation, CPAC

[11] Intuitively, the time complexity of $\binom{N}{i}$ is $O(N^i)$, and the geometric series as the result of Equation 5.7 grows with the order to $O(N^{x+1})$.

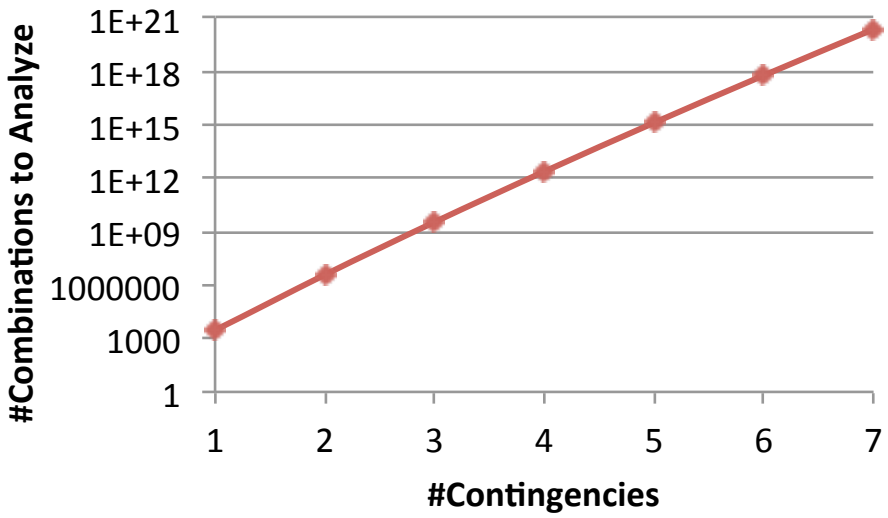


Figure 5.10: N-x Contingency Analysis Complexity

denies the request and prevents the system from entering an unsafe state.

8 Related Work

Control system safety. Stouffer et al. [187] present a series of NIST guideline security architectures for the industrial control systems that cover supervisory control and data acquisition systems, distributed control systems, and PLCs. Such guidelines are also used in the energy industry [203, 140]. It has, however, been argued that compliance with these standards can lead to a false sense of security [212, 154]. There have also been efforts to build novel security mechanisms for control systems. Mohan et al. [133] introduced a monitor that dynamically checks plant behavior safety. A similar approach using model based intrusion detection was proposed in [49]. Goble [86] introduce mathematical analysis techniques to quantitatively evaluate aspects of a control system such as safety and reliability, including PLC devices. However, the proposed solution focuses mainly on accidental failures and does not investigate malicious actions.

Access control. Most of the control systems, nowadays, rely on network access control [13], and host-based user authentication to protect against unauthorized plant monitoring and control activities. Additionally, PLC and HMI vendors themselves have included some rudimentary security measures into their solutions. Based on market data by Schwartz et al.[167], we studied the security measures used by PLCs accounting for 74% of market share. This included

PLCs from Siemens (31%), Rockwell (22%), Mitsubishi Electric (13%), and Schneider Electric (8%). We found that all four vendors use only password authorization, typically with a single privilege level. Furthermore, password authentication measure can be disabled in all four systems. Recently, more access control capabilities have been added to HMI engineering software. For instance, certain Siemens systems, e.g., SIMATIC STEP 7 TIA Portal [31] use client-side authentication for individual IDE projects. Additionally, recent device fingerprinting mechanisms (e.g., [74]) facilitate deployment of higher level access control functionalities such as CPAC in control systems. Almost none of the existing control system access control solutions take into consideration the physical dynamics of the plant while defining or enforcing the policies. This allows the attacker to completely bypass authentication by exploiting the physical system's dynamics and inter-component interdependencies to disclose sensitive measurements and manipulate critical plant actuation points.

Information flow analysis. Many existing solutions have proposed information flow control [222, 205] and dynamic taint analyzers [37] for general-purpose computing systems, smartphones [67] and embedded devices [221]. However, they have almost never been used in real-world control systems, because of *i)* their high run-time performance overheads limiting their deployability for safety-critical real-time settings and *ii)* insufficient accuracy due to fully ignoring the physical-side information flows. Existing control system data historians [76] within energy management systems [10] provide a bulk databases-level offline information flow control between large power system areas (control centers). Such coarse-grained solutions *i)* do not support dynamic and/or fine-grained information flow control; *ii)* often result in inflexible architectures, i.e., too permissive (allow data exchange between two control centers) or restrictive (no database exchange allowed); and *iv)* completely miss the physical dependencies between various database entries within and across the control centers.

9 Conclusions

We present CPAC, a cyber-physical access control solution to protect industrial control systems against operation mistakes and insider attacks. CPAC implements lightweight on-device and mathematically sound physical-side information flow analyses to maintain a complete system

view. It uses physical system model, information flow tracking, and logic-based context-aware policies to stop operations which could harm the whole system or leak sensitive information to malicious insiders. Our experimental results with CPAC's working prototype on Bachmann PLCs and EMS servers show that CPAC can terminate several past real control system incidents and perform $N-x$ contingency analysis with run-time performance overhead of only 150 ms.

Chapter 6

TRUSTED INTEGRITY VERIFIER FOR ADDITIVE MANUFACTURING

1 Introduction

Additive Manufacturing (AM) is gaining popularity in critical manufacturing such as aerospace and medical industries. General Electrical (GE), SpaceX, Airbus, and Naval Air Systems Command (NAVAIR) are all manufacturing some of the critical components used in aircraft by AM [53, 75, 138, 93]. Oak Ridge National Laboratory's Manufacturing Demonstration Facility created submersible hull manufactured using AM for defense sector [106]. Along with aerospace and submarines, medical industries is also using AM for manufacturing human tissue, organs, implants, customized prosthetics, and anatomical models [206].

STL files describe the geometry of objects to be printed and are the standard geometry exchange files in AM. As shown in Figure 6.1, the geometry of an object to be printed can be designed by a variety of 3D modeling software packages and the object can be printed by a variety of 3D printing techniques, such as Fused Deposition Modeling (FDM), Selective Laser Sintering (SLS), PolyJet, StereoLithography (SLA), Digital Light Processing (DLP), and Direct Ink Write (DIW). STL files are the common link between 3D modeling software packages and AM systems, and hence are the most critical elements in AM. Since STL files are one of the most critical elements in AM pipeline, we focus on defending against attacks on structural integrity by detecting the malicious features or defects in the STL files before they are printed.

As with any cyber-physical system, AM systems are subject to cyber-attacks [95, 180, 218, 151]. The most significant attacks attempt to compromise the structural integrity of functional objects. The compromise of structural integrity refers to the inability of functional objects to maintain their structural attributes during their normal operation whereas appearing normal during inspection or quality control. *The attacks compromising the structural integrity could*

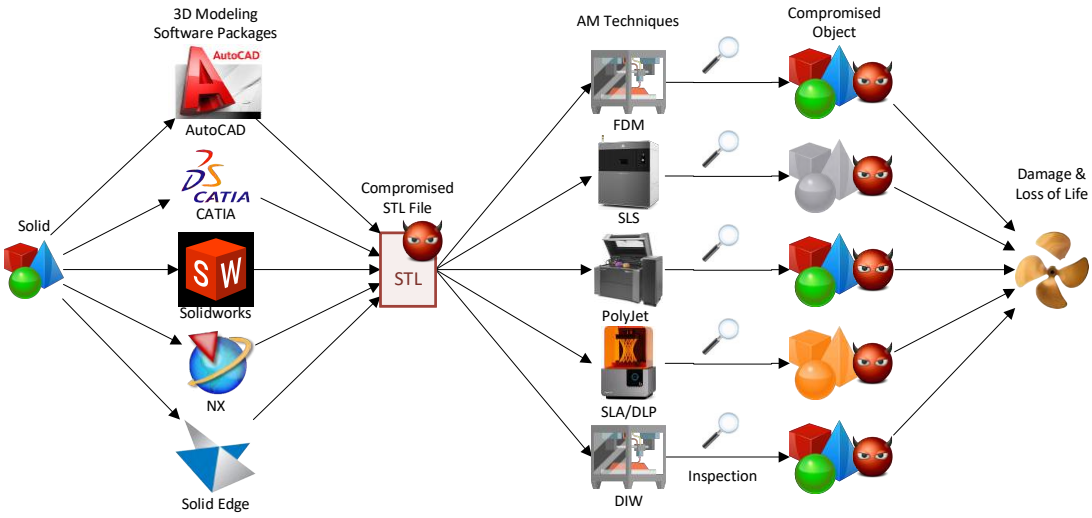


Figure 6.1: Additive manufacturing process and location of STL files in the process. Compromised STL file leads to failure of printed object.

damage the systems that uses these printed objects physically and be catastrophic. Due to the serious consequences and potential risks of such attacks, it is necessary to develop techniques to defend against such attacks.

Strum *et al.* showed that inserted voids could evade the inspection of unsuspicious AM operators [190], and Zeltmann *et al.* showed that sub-millimeter scale defects were undetectable by ultrasonic inspection [223]. Belikovetsky *et al.* demonstrated attacks could be accomplished by just inserting defective voids into STL files [29]. By adding voids to the critical locations of a propeller, they were able to crash a drone flying mid-air.

There are recently proposed mechanisms to protect 3D printers and their controllers [27]. Most of the 3D printing service providers such as Shapeways [171], 3D Systems [15], and *i.materialise* [103] have analysis process to determine the printability of the objects. The traditional software and network security mechanisms focuses on protecting against any cyber attacks on the printer's operating system, firmware and inputs of the printer (STL design) that lead to exploits in the printer.

However, these kind of security mechanisms fail to detect new types of structural integrity attacks[190, 223, 29] on the design files that impact the system where the printed objects are being used. Also, there has been no comprehensive automated solution to detect such attacks other than educating AM operators on potential risks of cyber-attacks in AM systems.

To address the problem, we present a framework called Trusted Integrity Verifier (TIV) in this paper. First, TIV verifies the design's shape. Second, TIV searches the design for malicious features and finally verifies if the design printed when being used in critical systems will violate the safety operating conditions. In the final step, the numerical method can verify if the size of the object matches the desired system and if the printed object can withstand the required loading conditions to operate normally during its functional state.

Our contributions in this paper are as follows:

- We developed a classifier to verify the STL file's geometry for object printed to be the intended object.
- We leverage a computer vision technique (flood filling) to localize and detect any malicious features present that compromise the structural integrity of the STL files.
- We used numerical analysis methods (Finite Element Analysis) to distinguish between the malicious features and the legitimate design features through calculating the mechanical stress acting on the object during its normal operation.
- We performed an extensive empirical study to validate TIV's performance and scalability on a large real-world STL design file dataset (16,000 designs).

Organization. In Section 2, we briefly explain background information that is necessary to understand technical details of TIV, structural integrity, and attacks on STL files. In Section 4, we present an overview of the TIV framework and the threat model. We present the threat model in Section 3 and overview of the system in Section 4. We present the object classifier to verify the geometrical shape of the object in Section 5, details of the suspicious feature detection module in Section 6 and malicious feature verification in Section 7. In Section 8, we provide verification results on 16,000 STL files. In Section 9, we review related work in the literature. Finally, we conclude the paper in Section 10.

2 Background

In this section, we briefly present background information to help understand the structural integrity and attacks on STL files. In this paper, we refer to the geometry of an object as a solid

solid *name*
  facet normal $n_{1,x}$ $n_{1,y}$ $n_{1,z}$
    outer loop
      vertex $v_{1,1x}$ $v_{1,1y}$ $v_{1,1z}$
      vertex $v_{1,2x}$ $v_{1,2y}$ $v_{1,2z}$
      vertex $v_{1,3x}$ $v_{1,3y}$ $v_{1,3z}$
    endloop
  endfacet
  ...
  facet normal $n_{N,x}$ $n_{N,y}$ $n_{N,z}$
    outer loop
      vertex $v_{N,1x}$ $v_{N,1y}$ $v_{N,1z}$
      vertex $v_{N,2x}$ $v_{N,2y}$ $v_{N,2z}$
      vertex $v_{N,3x}$ $v_{N,3y}$ $v_{N,3z}$
    endloop
  endfacet
endsolid *name*

Figure 6.2: The format of an STL file with N triangles.

body (or simply a solid) or a design model (or simply a design). When a solid is manufactured by an AM system, the result is an object (or a part).

2.1 Format of STL Files

Many attacks on STL files leverage the format of STL files because STL files lack the connectivity of geometry primitives, and as a result, it is relatively easy to modify features in STL files.

In STL files, a solid is represented by its boundary surfaces. The surfaces are represented by triangular facets. Each triangular facet is described by an outward normal vector and three ordered points. An STL file is obtained by stacking the components of the normal vectors and the coordinates of the points in a hierarchical manner. Figure 6.2 shows the structure of an STL file with N facets, where $n_{i,x}$ $n_{i,y}$ $n_{i,z}$ are the x, y, and z components of the normal vector for facet i, whereas $v_{i,jx}$, $v_{i,jy}$, and $v_{i,jz}$ are the coordinates of vertex j in facet i. In STL files, facets are always triangular and we can simply refer a triangular facet as a facet. Due to this structure of the STL file, it is relatively easy to weaken the design by just addition of facets. Negative volume shell injection attack can be performed by addition of just four facets to STL files.

2.2 Attributes of STL Files

This section describes the attributes of the STL files. These attributes can be leveraged to attack STL files.

Mesh Complexity The complexity of a mesh refers to the number of facets in the mesh. With limited computational resources, a very complex STL file cannot be sliced or analyzed.

Mesh Quality The quality of a mesh is determined by the number of spikes (or sharps) and facets with high aspect ratios. An STL file with lots of sharps and facets with high aspect ratios are hard to be sliced or analyzed.

STL Format Integrity The files those obey the STL file format are said to have STL format integrity and those that do not obey are deemed to be corrupted.

Geometric Integrity A collection of facets forms a polygonal mesh (or simply a mesh in the following discussion) if the facets are connected together without holes (or gaps), intersections (or overlaps), and over-connection. If the mesh encloses a solid, the mesh defines the solid. An STL file has geometric integrity if the file can describe a solid.

Structural Integrity The structural integrity refers to the ability of functional objects to maintain their structural attributes such as physical shape and mechanical strength during there normal operation. The attacks such as unintentional voids, cracks or holes at critical locations of the objects to compromise there structural attributes leads to the violation of structural integrity of the STL files. Depending on the type of loads, structural integrity can be classified into *i)* static structural integrity and *ii)* operational structural integrity. Static structural integrity refers to the ability of an object to maintain its structural attributes under its own weight. Operational structural integrity refers to the ability of a functional object to maintain its structural attributes under operational loads eg., the operational structural integrity of a propeller refers to its ability to maintain its structural attributes when the propeller is in operation and subject to centrifugal forces and thrust.

2.3 Attacks on STL Files

This section summarizes attacks on STL files which are found in the literature [213, 29, 223, 190].

Scaling An attacker can multiply coordinates of all points (excluding components of normal vectors) in an STL file by a positive number to scale the solid described by the STL file. For a scaling factor that the solid remains printable, the structural integrity is usually affected as a result. It is hard for scaling attack to succeed for functional objects because these objects are subject to inspections. When the size of a functional object is changed, it is almost impossible to pass an inspection on dimensions.

Orientation Orientation with respect to the building bed affects structural integrity because 3D printed objects are anisotropic [17]. An STL file comes with a coordinate system, and the building bed is located on the XY plane by default. An attacker can carefully select a default orientation in an STL file such that the strength of the printed object is weakest.

Vertex Movement One or more vertices in an STL file can be moved to locally change the geometry and the structural integrity can be affected as a result. The vertex movement must remain hidden under inspection. One way to do this is to move vertices located at areas that are hard to be inspected.

Void Injection Because an STL file is simply a collection of facets, extra facets can be directly inserted into the STL file to create unwanted voids in the solid described by the STL file. Research has shown that if the size and location of voids are properly chosen, the voids can significantly reduce the structural strength and remain hidden from AM operators [29, 190]. Research has also shown that if the size of voids is reduced below sub-millimeters, ultrasonic inspection cannot detect the presence of the voids in PolyJet 3D printing [223].

Complex Modification Complex modification results in multiple vertices being moved, deleted, and/or created. An example of complex modification to compromise structural integrity is to

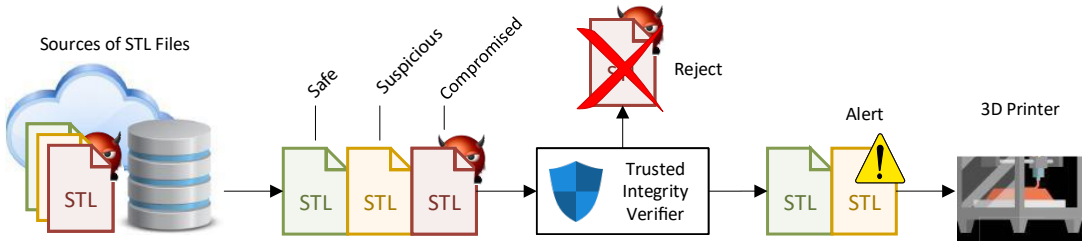


Figure 6.3: Threat model and application of the TIV framework.

change the gage width of a tensile test specimen, which reduces the strength of the specimen and remains unnoticeable by operators [213].

Some attacks are easy to detect and are currently detected by tools such as tweaker and mesh labs, but existing techniques fail to detect more stealthy attacks such as vertex movement, void insertion, and complex modification during inspection. We focus on the aforementioned stealthy attacks that cannot be detected by current techniques.

3 Threat Model

We assume that AM operators are different from designers. This assumption is valid when printing tasks are outsourced to 3D printing service providers, such as Shapeways [171], 3D Systems [15], and *i.materialise* [103] or when the designs are outsourced to 3D modeling service providers, such as Cad Crowd[42].

In the scenario where the design is outsourced, most of these workstations are internet facing. Hence, workstations become an easy target for attackers compared to printers which is located behind demilitarized zone (DMZ) and have limited access to internet. Phishing attacks are one of the common ways of cyber attack and studies shows that 91% of cyberattacks are due to phishing [227]. The first step in the attack of Ukraine power plant was through sending Blackenergy malware through phishing emails[45].

Design toolchains used by designers on workstations could be compromised by such attacks even though the designers do not have any malicious intent. Acad.vlx virus [193] and ACAD/Medre.A worm [68] are examples of malware that have attacked design toolchains. Although there are traditional software mechanisms and techniques to protect against attacks on toolchain, they fail to be effective in protecting against zero day vulnerabilities.

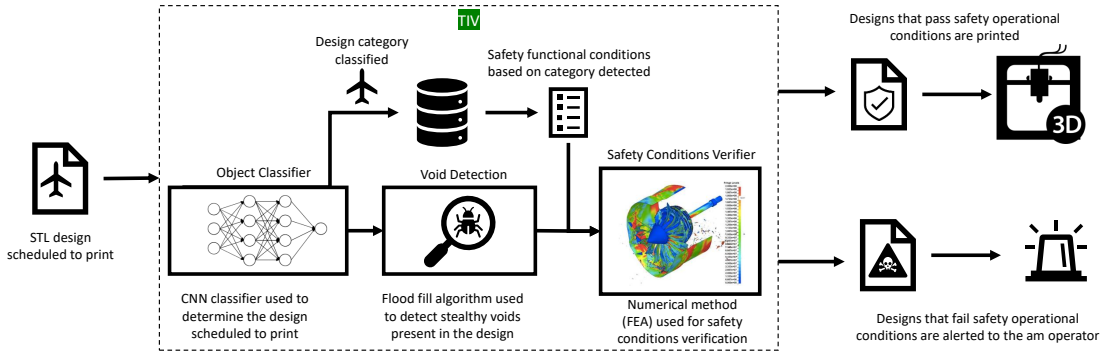


Figure 6.4: Structure of the TIV framework.

Existing security methods, such as secure hash, end to end encryption, securing important supply chain data using secure channels [100], and using blockchains to protect the integrity of supply chain data [58], do not protect against such attacks on design toolchains as the workstation is not trusted. The design file is one of the inputs to printers apart from raw materials for manufacturing. The design files generated by such compromised workstations could be malicious and cannot be trusted for using in critical applications.

We assume that the value chain after TIV are trusted such as AM operators, 3D printers and their controllers (firmware and slicers), and material supplies.

4 TIV Overview

We take inspiration from trusted safety verifier (TSV) [128] which is a bump in the wire solution that sits between workstations and programmable logic controllers to verify the safety properties of any program being uploaded to the controller. Analogous to TSV, we developed TIV which sits before printers to verify the structural integrity of objects to be printed. TIV does not require any software or hardware modifications on the existing systems.

TIV is composed of a neural network based object classifier, computer vision based void detection module and numerical based malicious void verification module to determine if a void is from the design or malicious. The modules are used in tandem, as shown in Figure 6.4. TIV answers questions such as *i)* can we automatically detect attacks such as any major changes to the design, *ii)* can we detect any detect any minor stealthy attacks such as voids that are not visible after the object is printed, and *iii)* can we verify if the object printed

Object Classifier The object classifier module is used in TIV to detect any geometrical shape changes in the STL file. Since the workstation is compromised the attacker can replace the intended design with an alternative design. Object classification module will detect such attacks that change the entire shape of the design file. This classified category is feed to a data base consisting of safety functional conditions defined by category.

Object classifier uses convolutional neural networks (CNN) to classify the objects based on their shapes. CNN's are popular for 2D images but the STL files are 3D objects. We convert the 3D objects into octree data structure and feed the octree data structure as the feature for CNN.

Void Detection Module Any changes to geometrical shape is detected by the object classifier, but any malicious features added that will weaken the structure of the printed object will not be detected by the object classifier. The void detection module detects such suspicious features in the design described by the STL file. The suspicious features are voids in critical parts of the design to weaken the structure of the object. The void detection module takes an STL file as input and localizes the voids in the object using the flood fill algorithm.

The flood fill algorithm scans the voxels in depth first search to find the voids present in the 3D design files. If the void detection module detects any suspicious features, they are fed to malicious void verification module to determine if the feature is due to design or due to an attack compromising the structural integrity of the printed object.

Voids are a special type of suspicious features in AM systems. Most attacks on STL files present in the literature are malicious insertion of voids [29, 190]. Voids can take various shapes, such as cubic, rectangular, pyramid, cone, etc. Void detection module detects the voids irrespective of there size and shape.

Safety Conditions Verification Module If the void detection module detects any potentially malicious voids, they have to be verified to determine if the voids are legitimately a part of the design or due to an attack against the structural integrity of the object.

The strength to weight ratio is a popular matrix to consider while designing. The best design will have higher strength with less material used. Hence the designers tend to remove

material in the places that do not affect the strength of the printed object. Legitimate voids created in the design process are inserted by the designer to reduce the material and weight of the object *without* compromising the mechanical strength of the object. Voids of this nature are considered benign.

On the other hand, malicious voids tend to reduce the strength of the printed object significantly. Hence the benign design features can be easily differentiated from the malicious features. We use a numerical methods based finite element analysis to calculate the strength of the printed object. The safety functional conditions are feed from a database to the safety condition verifier. The verification is performed based on this safety functional conditions. Section 8 shows the designs with malicious features and without malicious features to show the effectiveness of our verification module.

5 Object Classification

The object classification module is used to detect any changes to the intended object to an alternative shape. The object classification module uses CNN to detect the object to be printed and verifies if the shape of the object is an intended object shape. Since the printing operator knows the object to be printed, they can give the command to the object classifier to verify if the object is the intended object to be printed. This section describes the methodology used for classification of the 3D objects using CNN.

Database The database was constructed from the combination of the modelnet database[215] with well annotated labels and the design files downloaded from popular website *thingiverse* [195] with no annotation labels. The database consists of 298,056 design files from 44 different categories. The categories range from critical parts from aerospace, medical, power tools, sports equipment's. 237,864 files were used for training and 60,192 were used for testing. The design files from modelnet are in object file format. Hence, for uniformity of the design file format, the STL files from *thingiverse* were converted into the object file format while building the database.

Octree In the field of computer vision, convolutional neural networks (CNN) are often used to classify 2D images. However this method cannot be directly used for design files because of their 3D shape and irregular triangle meshes. Hence the 3D design files have to be converted into an appropriate format to use traditional convolutional neural networks.

We convert the 3D design files to octree [129] and use the octree as the inputs to the CNN. An octree is a tree data structure most often used to partition a three-dimensional space by recursively subdividing it into eight octants. Octrees are popularly used in 3D graphics and 3D game engines for rendering, modeling and collision detection. We generate a sparse octree occupied by the boundary surface of the 3D shapes.

Since the design files have irregular triangular meshes such as flipped normals, non-manifolds and overlapped triangles, we convert them into point cloud that consists of just points. We use the ray shooting algorithm [18] to sample dense points from the 3D shapes. We then divide this dense point cloud into a unit 3D bounding cube and recursively subdivide the bounding cube of 3D shape until the required octree depth is reached. TIV uses octree depth of six. We traverse all the non-empty octants occupied by the 3D shape bounding cube in the current depth and subdivide them into eight child octants for the next depth. We use this octree data structure as input for the CNN to classify the 3D design into appropriate category.

Convolutional Neural Network We used the octree data structure as input for the CNN (O-CNN) [210]. The CNN has a basic layer block which consists of a convolutional layer, batch normalization, ReLu output activation function and pooling layer. Batch normalization is used to reduce the internal-covariate-shift [105]. We used 3 basic layer blocks in sequence followed by two fully connected layers, a softmax layer and two dropout layers. Dropout layers are used to avoid overfitting and softmax is used to categorize.

Safety Functional Conditions The safety conditions are determined by the finite element analysis experts. The experts determine the safety conditions based on the category of the design. The conventional analysis on the design end is intensive and have to be defined for every design. We just categorize the objects and create a template of safety conditions for the entire category from the experts and maintain a database. Once the category of object is determined the safety conditions are extracted from the database and used of verification in Section 7.

6 Void Detection Module

The void detection module [1] is designed to detect and locate malicious voids in an STL file. A malicious void is potentially a malicious geometric structure that can lead to the structural failure of a printed object if the structure is present in the design of the printed object.

TIV's void detection module leverages ideas from the flood fill algorithm that is widely used in image processing to change the color in a continuous region of an image to another color. The conventional flood fill algorithm is mainly used for 2D pixels in images. We extended the flood fill algorithm for 3D voxels in 3D print design files.

Since the STL files just describe the geometry, it is difficult to directly apply flood fill algorithm. Hence, we first convert the STL files into voxels. The process of converting STL files to voxels is analogous to rasterizing a vector image to an array of pixels. STL files are converted to voxels by the following steps *i)* TIV slices the STL files to get a series of perimeter paths that are vector images. *ii)* TIV then transform these vector images to pixels using standard rasterizing techniques. *iii)* TIV stacks these pixels up to form an array with different layers to form voxels.

TIV implements the steps above to scan the 3D design file for voids. Specifically, TIV fills in the object using the same material of the object. The spaces corresponding to the unfilled voxels are the voids. TIV leverages a depth first search to scan and find the voids in the object. The number of unfilled voxels corresponds to the number of voids in the object. The information of the unfilled voxels is collected for later analysis to verify if the identified void(s) is/are malicious.

Automated Attack on Design Since it is hard to find STL files with voids in the wild, We developed a methodology to insert voids with random shapes, sizes and random locations in the design. The insertion of voids will make sure that the attack is stealthy and not visible after it is being printed. We took the inspiration from past works on attacking STL files [190, 223, 29] and improvised them to automatically attack STL files.

Our algorithm is inspired by the ray-casting algorithm [84]. The ray-casting algorithm is a

[1] This work was completely done by Sizhuang Liang from Georgia Tech and added in my thesis to understand the complete picture of the solution.

computer graphics algorithm used to determine the intersections of ray-surfaces. We first find the maximum and minimum coordinates of an object in all three directions. These are achieved by using ray casting algorithm. The results are denoted by X_{min}, X_{max}, Y_{min}, Y_{max}, Z_{min}, Z_{max}. These coordinates define the geometrical boundaries of the object.

We use a random number generator to pick the number of voids, shapes of voids and locations in the object where the voids should be inserted. We then apply the ray-casting algorithm around the randomly picked location $(X_{\text{stress}}, Y_{\text{stress}}, Z_{\text{stress}})$ in the object.

Using the following matrices, we determine the boundaries of the void that can be inserted without being detected visually after printing. This determines the size of the object. The inputs for the ray casting algorithm are *i)* first row of ϕ_{max} and ϕ_{min} for x-axis *ii)* second row of ϕ_{max} and ϕ_{min} for y-axis, and *iii)* third row of ϕ_{max} and ϕ_{min} for z-axis, where ε is a small parameter that leads the coordinates slightly go outside the geometrical boundaries.

$$\phi_{\text{max}} = \begin{bmatrix} X_{\text{max}} + \varepsilon & Y_{\text{stress}} & Z_{\text{stress}} \\ X_{\text{stress}} & Y_{\text{max}} + \varepsilon & Z_{\text{stress}} \\ X_{\text{stress}} & Y_{\text{stress}} & Z_{\text{max}} + \varepsilon \end{bmatrix} \tag{6.1}$$

$$\phi_{\text{min}} = \begin{bmatrix} X_{\text{min}} - \varepsilon & Y_{\text{stress}} & Z_{\text{stress}} \\ X_{\text{stress}} & Y_{\text{min}} - \varepsilon & Z_{\text{stress}} \\ X_{\text{stress}} & Y_{\text{stress}} & Z_{\text{min}} - \varepsilon \end{bmatrix} \tag{6.2}$$

Figure 6.5 shows the coordinates used by automated attack to determine the boundaries along all the axes at the randomly selected location. Some designs might be so irregular that the voids inserted by previous procedures might be visible on the outer surface. When this happens, we use boolean operations to subtract that part of the void that is outside of the outer surface (visible). The placement of void by this method will go unnoticed leading to a stealthy attack on STL files.

Accuracy of Flood Fill Algorithm Figure 6.6 shows an example of an attacked STL file with a rectangular void and a spherical void. The locations of voids are highlighted using red rectangles. The results of the flood fill algorithm to detect voids are shown in Table 6.1. As we can see from the result, the flood fill algorithm can accurately detect the number of voids in all the

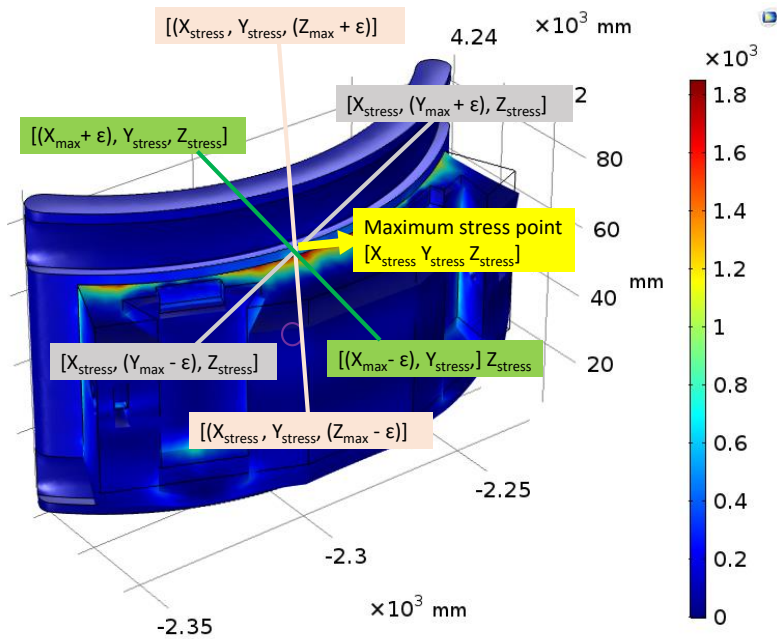
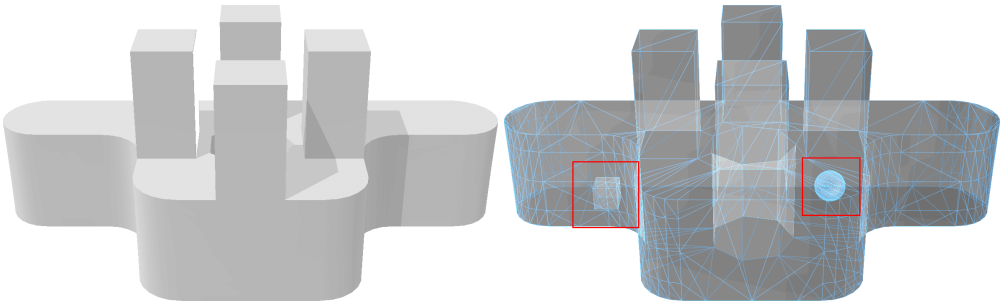


Figure 6.5: Ray casting algorithm being used to determine the geometrical boundaries to determine the size of the void that should be inserted.



(a) Normal View. (b) X-Ray View.

Figure 6.6: Example of a manually attacked STL file.

File	Detected voids	Actual voids	Detection Rate
Aerospace	6	6	100%
Automotive	3	3	100%
Engineering	2	2	100%
Handtools	4	4	100%

Table 6.1: Evaluation of malicious feature detection using flood fill on a manually attacked STL file dataset.

Algorithm 3: Geometric analysis

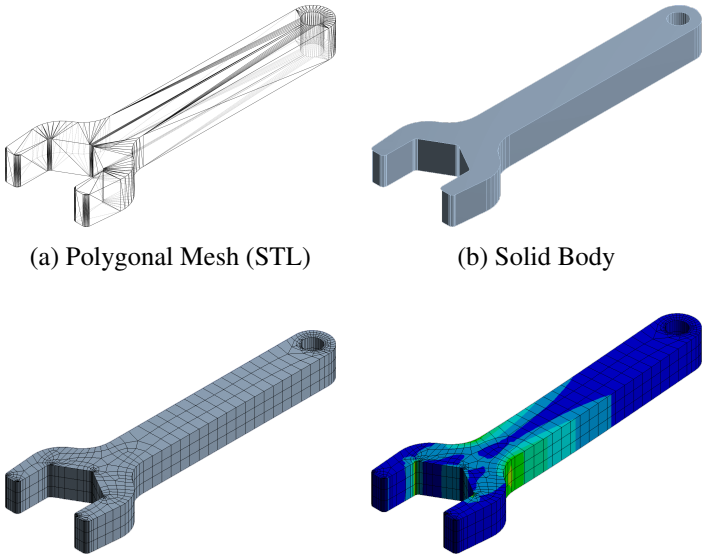
```
input STL file;
output Solid Model;
α ← read(STLfile);
if corrupt(α) then
    return (Failed)
else
    α ← FixGeometricIrregularities(α);
    if FixSucessful then
        [β, γ, δ, ε, ζ] ← GeometricStatistics(α);
        if ζ == 1 then
            η ← ConvertToSolid(α);
    else
        return (Failed)
return (η, Success)
```

manually attacked STL files. The detailed large scale evaluations is performed by automated attack and is shown in Section 8.

7 Safety Conditions Verification

All the voids present in the object are detected by the void detection module in the previous section. However, we do not know if these all these detected voids are malicious or just used by the designer to reduce the weight without compromising the structural integrity. If the object to be printed satisfies the safety operational conditions, then they are deemed to be innocuous to print. If the object to be printed violates the safety operational conditions, then it is deemed to be malicious and dangerous to be used in critical systems.

The safety operational conditions for the objects printed are the mechanical loads or mechanical stresses that the object can bare without failing or breaking. In this section we use FEA to determine the mechanical stresses acting on the printed object. The process of FEA is explained in this section in detail with geometrical analysis, structural analysis and calculating the stresses acting on the object.



(a) Polygonal Mesh (STL) (b) Solid Body

(c) Volumetric Mesh (d) Structural Property

Figure 6.7: Polygonal mesh, solid body, volumetric mesh, and structural property of a spanner.

7.1 Geometric Analysis

The geometric analysis routine takes as input an STL file and outputs a solid model, as shown in algorithm 3. An STL file describes a polygonal mesh, as shown in Figure 6.7a. The polygonal mesh only represents the surface of the solid, whereas FEA requires a volumetric mesh, as shown in Figure 6.7c. Apart from this, the pattern of a polygonal mesh in an STL file is very different the pattern of a volumetric mesh that is suitable for FEA, because a polygonal mesh in an STL file tends to be dense at areas where the geometry is complex whereas a volumetric mesh for FEA tends to be dense at areas where stresses concentrate [120]. In order to convert the polygonal mesh to a volumetric mesh, we first of all convert the facets into a solid, as shown in Figure 6.7b.

Check File Integrity The first step is to check the validity of the STL file. line 4 of algorithm 3 shows this step. We proceed if the STL is an STL file and is not corrupted. We also limit the size of the file to prevent over consumption of computational resources.

Fix Geometric Integrity We attempt to fix holes, intersections, and over-connected edges, if there are any. If the geometric integrity does not exist and cannot be restored, the analysis is aborted. line 7 of algorithm 3 shows this step.

Algorithm 4: Structural analysis

```
1  input Solid Model;
2  output Structural Properties;
3  α ← read(SolidModel);
4  VolumetricMesh ← Mesh(α);
5  if MeshSuccessful then
6  |   AssignMaterial(VolumetricMesh);
7  |   AssignBoundaryConditions(VolumetricMesh);
8  |   ApplyLoads(VolumetricMesh);
9  |   StructuralProperties ← Solve();
10 else
11 |   return (Failed)
12 return (StructuralProperties, Success)
```

Get Geometric Statistics We extract the number of vertices, faces, edges, sharps, and pieces after the STL is fixed. line 9 of algorithm 3 shows this step. The number of vertices, faces, edges, sharps, and pieces are represented by β, γ, δ, ε, and ζ. If there are more than one piece, we either abort the analysis or separate the pieces, put them into different STL files, and do analysis for each separated STL file.

Convert to Solids We attempt to convert the polygonal mesh to a solid. Conversion may still fail due to the quality of the polygonal mesh being low. line 11 of algorithm 3 shows this step and η is the solid model as the output.

7.2 Structural Analysis

The structural analysis routine takes as input a solid body from the geometric analysis routine and outputs the stresses of the solid body under loads in operation, as shown in algorithm 4.

A printed object can have various infill patterns and densities depending on the AM technique. However, we assume that the infill pattern of a printed object is solid because many functional objects are printed using SLS, and the infill pattern is usually solid.

Generate Meshes When the solid to be analyzed has a solid infill pattern, the volumetric mesh can be directly obtained. line 4 of algorithm 4 shows this step.

Properties	PLA	ABS	Unit
Density	1.24	1.05	$g{\cdot}cm^{-3}$
Young's Modulus	3.5	2.6	GPa
Poisson's Ratio	0.36	0.35	-
Yield Strength	60	45	MPa

Table 6.2: Material properties of PLA and ABS

Assign Material The material used for printing must be specified before a solid is analyzed. Since the printing operator is trusted and know the material used for manufacturing, he can input those information to the TIV. The relevant properties of material for our analysis are density, Young's modulus, Poisson's ratio, and Yield Strength. Table 6.2 shows properties of Polylactic acid (PLA) and Acrylonitrile butadiene styrene (ABS) [197]. line 6 of algorithm 4 shows this step.

The 3D printed objects have anisotropic properties due to the techniques of AM [17]. For fused deposition modeling (FDM), the strength of a 3D printed object is less along the building direction because the bonding between layers is weaker than the bonding within a layer. We consider this property in our system by using anisotropic material properties.

Assign Safety Operational Conditions Safety operational conditions refer to motion constraints of objects to be analyzed. The motions constraints are the mechanical loads and stresses that the printed object can bare during the normal operation without breaking or failure.

For static structural integrity analysis, the bottom surface of the object to be analyzed is fixed. These resembles that the object is placed on a flat surface and the forces acting on them are gravity and the forces due to its own weight. For operational structural integrity analysis, safety operational conditions depend on the object and operation conditions. line 7 and line 8 of algorithm 4 show this step.

The safety operational conditions depends on the type of object and it application. We used different safety operational conditions for different object categories depending on the applications where we intend to use. The object category is obtained from the classifier and corresponding safety operational conditions are picked from the database for verification.

The safety operational conditions includes the property of material used for printing, amount of infill, infill pattern, gravity acting on the object, and the kind of mechanical loads acting on

the object during there normal operation. Objects such as propellers, gears, wheels have rotational actions resulting in centripetal forces acting on them. Objects such as a wrench, and a hammer have shear forces acting on them. Objects such as suspensions have compressive forces acting on them. Objects such as hooks have tensile forces acting on them. TIV uses object classifier to classify the type of category.

Export Stresses The Cauchy's stress tensor can be used to represent stresses at any point inside an object. The Cauchy's tensor matrix is as follow

$$\sigma = \begin{bmatrix} \sigma_{xx} & \tau_{xy} & \tau_{xz} \\ \tau_{yx} & \sigma_{yy} & \tau_{yz} \\ \tau_{zx} & \tau_{zy} & \sigma_{zz} \end{bmatrix} = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \sigma_{13} \\ \sigma_{21} & \sigma_{22} & \sigma_{23} \\ \sigma_{31} & \sigma_{32} & \sigma_{33} \end{bmatrix} \tag{6.3}$$

where σ_{xx}, σ_{yy}, and σ_{zz} are normal stresses and τ_{xy}, τ_{xz}, τ_{yx}, τ_{yz}, τ_{zx}, and τ_{zy} are shear stresses. Normal stresses are stresses acting perpendicular to the surface of the object and shear stresses are stresses acting parallel to the surface of the object. At equilibrium, the matrix is a symmetric matrix. Hence the stress tensor can be represented in the Voigt notation as

$$\sigma = (\sigma_{xx}, \sigma_{yy}, \sigma_{zz}, \tau_{xy}, \tau_{xz}, \tau_{yz}) \tag{6.4}$$

FEA exports σ when analysis successfully completes. Line line 9 of algorithm 4 shows this step.

7.3 Malicious Void Verification

The void detection module determines the shape, size and position of the void present in the design. To verify that a void is malicious or benign, we use FEA to calculate the stress acting on the objects.

Microlattice Structures Recent development is pushing the designers to design more efficient objects with higher strength to weight ratios. The material and time to print can be reduced significantly by designing effectively. Hence, the popularity of the micro lattice structure [158] based designs are increasing.

The micro lattice structure have a repeated lattice structure. The microlattice structures can provide similar strength to a solid infill but with less material and print time. Due to the nature of the design the microlattice structure based designs have air voids in there designs. The microlattice structure are used as both structural and functional objects. Most of the structural objects are solids, whereas the functional objects could be with the microlattice structure.

The void detection module predicts these voids as malicious although they are due to design. We use FEA to determine the mechanical stresses acting on the objects that has to be printed. The FEA is performed on the object which was flagged to have voids by void detection module. The mechanical stresses from the safety operational conditions are used for FEA. If the printed object withstands the mechanical loads applied during FEA, then we deem it to be innocuous and signal for printing. If the object cannot withstand, then they are deemed to be malicious and dangerous to be used on critical systems.

8 Evaluation

To show the generality and scalability of our solution, we picked 16,000 files randomly from wild for evaluation. We demonstrate a detailed step by step analysis of a vertebra from the medical category as a case study to demonstrate the application of each module in TIV and later provide empirical results for 16,000 STL files from eight different categories.

Experimental Setup The conversion of the STL file format to OBJ file format was done using stl2obj tool written in C++. A virtual scanner written in C++ was used to scan the triangular meshes and convert them into dense point cloud. The virtual scanner uses ray shooting algorithms to calculate intersecting points and oriented normals. These point cloud was converted to octree data structure and the lmdb database of octree data structure was constructed. These lmdb databases were feed into the Caffe tool, which was used for the convolutional neural network.

We used two popular FEA tools for the computation of stresses on designs, ANSYS which is popular in industry and COMSOL which is popular in academia. We used IronPython for automating ANSYS and MATLAB for automating COMSOL.

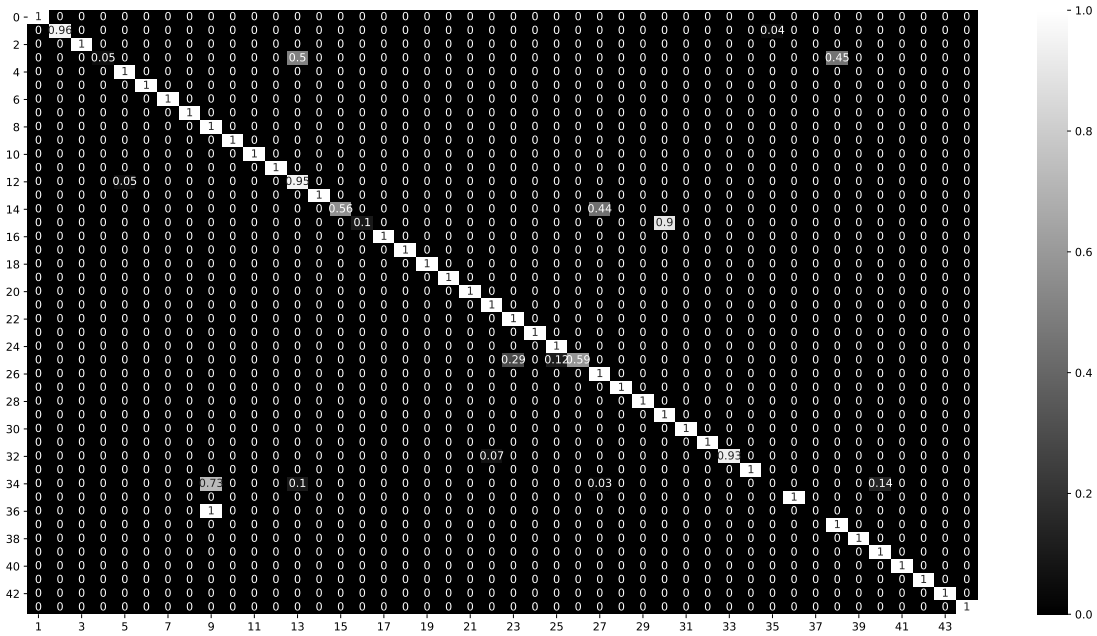


Figure 6.8: Confusion matrix for classification of the objects into 44 different categories. 1 indicates all of the objects are classified correctly and 0 indicates none of them are classified correctly. Diagonal 1's indicates that most of the objects are classified correctly.

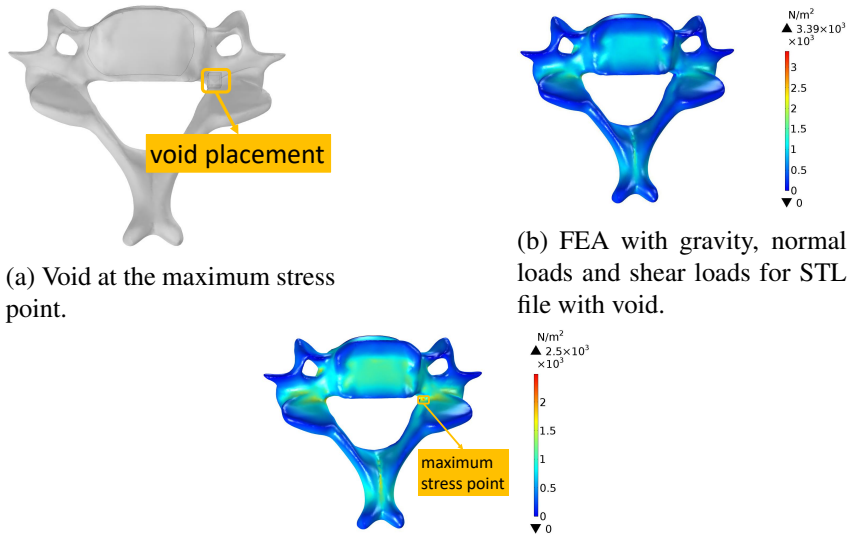
Object Classification The database created was used for training and classification of the objects. The depth of the octree fed into the neural network was six. The kernel size for convolutional neural network was three with stride length of one. The base learning rate, momentum and weight decay of the neural network training were 0.1, 0.9, 0.0005 respectively. The database was trained for 40 epochs.

The confusion matrix of the classifier is shown in Figure 6.8. The confusion matrix shows that the classifier can accurately predict most of the category of the objects. The classifier accuracy was 92% in classifying the objects into its categories.

8.1 Case Study: Analysis of a Vertebra

Object classification The object classification module accurately classified the object as vertebra.

Void Detection The void present in Figure 6.9a was successfully detected by the flood fill algorithm. The algorithm not only allows for the detection of attacks that are difficult to detect with the naked eye, but also provides a less expensive method compared to other traditional



(a) Void at the maximum stress point.

(b) FEA with gravity, normal loads and shear loads for STL file with void.

(c) FEA with gravity, normal loads and shear loads for STL file after removing the void.

Figure 6.9: Malicious void verification on cervical vertebra (C6) from medical category. The STL file with a void increases the stress from 2.53 kN to 3.39 kN ($\sigma_{Differential}$ = 34%) for the same loads acting on the object.

methods such as destructive and non-destructive testing techniques.

Malicious Void Verification Figure 6.9 shows the effect of malicious void detection on the vertebra. Figure 6.9a shows voids detected by void detection module. This detected void is examined by the malicious void detection module to determine if the void detected is malicious. The malicious void detection module calculates the mechanical strength of the object as shown in Figure 6.9b.

To determine if the voids are malicious, stress analysis is performed on the vertebra with static structural analysis and operational structural analysis. The safety operational conditions for the operational structural analysis are dynamic loads on the vertebra due to physical activities such as jumping and squatting. The acceptable limits of the spine to shear load is 1 kN for 90% of the working age population for infrequent loading [77]. But this increase for active population who are into sports and weight training. So we estimate the safety operational condition to be 3 kN.

Figure 6.9c shows the result of stresses acting on the vertebra. The vertebra with voids has stress more than the safety operational conditions (3.39 kN). Hence, we determine that

Category	Pass Rate	False Positive Rate	False Negative Rate
Aerospace	96.6	3.4%	0%
Automotive	96.4%	3.6%	0%
Engineering	97.7%	2.3%	0%
Hand tools	97.4%	2.6%	0%
Machine tools	98.1%	1.9%	0%
Medical	91.3%	8.7%	0%
Robotics	96.6%	3.4%	0%
Sports	97.7%	2.3%	0%
Average	**96.5%**	**3.5%**	**0%**

Table 6.3: Evaluation of Malicious void detection using flood fill on a wild STL file dataset.

this feature is a malicious feature and not a feature in the design to reduce the material used or to decrease the weight of the object. To better understand the verification and show that the verification is efficiency and accurate in detecting the malicious designs, we remove the voids and evaluate the mechanical stresses acting on the objects. The maximum von Mises stress (σ_{vonMises}) from the analysis is 2.5 kN for vertebra without voids and the maximum stress location is at (15.69 67.78 13.29).

8.2 Large Scale Analysis on STL Files

Void Detection Flood fill algorithm was used to detect voids in the STL files. If the algorithm does not detect any void in an STL file, the STL file passes the test. For each test, the pass rate is listed in Table 6.3. On average, 96.3% STL files passed the flood fill algorithm test. The medical category has the lowest pass rate. This is due to the fact that objects in the medical category are complex. The flood fill algorithm reports the number of voids detected in each STL file. For STL files with detected voids, most STL files only have one void. Some STL files do have multiple voids. Figure 6.10 shows the histogram of the number of voids for all categories. The horizontal axis at 10^2+ actually indicates 100 voids or more.

To test the accuracy of the flood fill algorithm for false positive rates, we manually inspected the STL files that were determined to contain voids by the flood fill algorithm. We loaded each STL file and displayed the STL file in an X-Ray view, as shown in Figure 6.6. If an STL file contains voids, the voids should be visible in the X-Ray view. We found that most STL files that were determined to contain voids actually did not contain voids at all. These STL files are

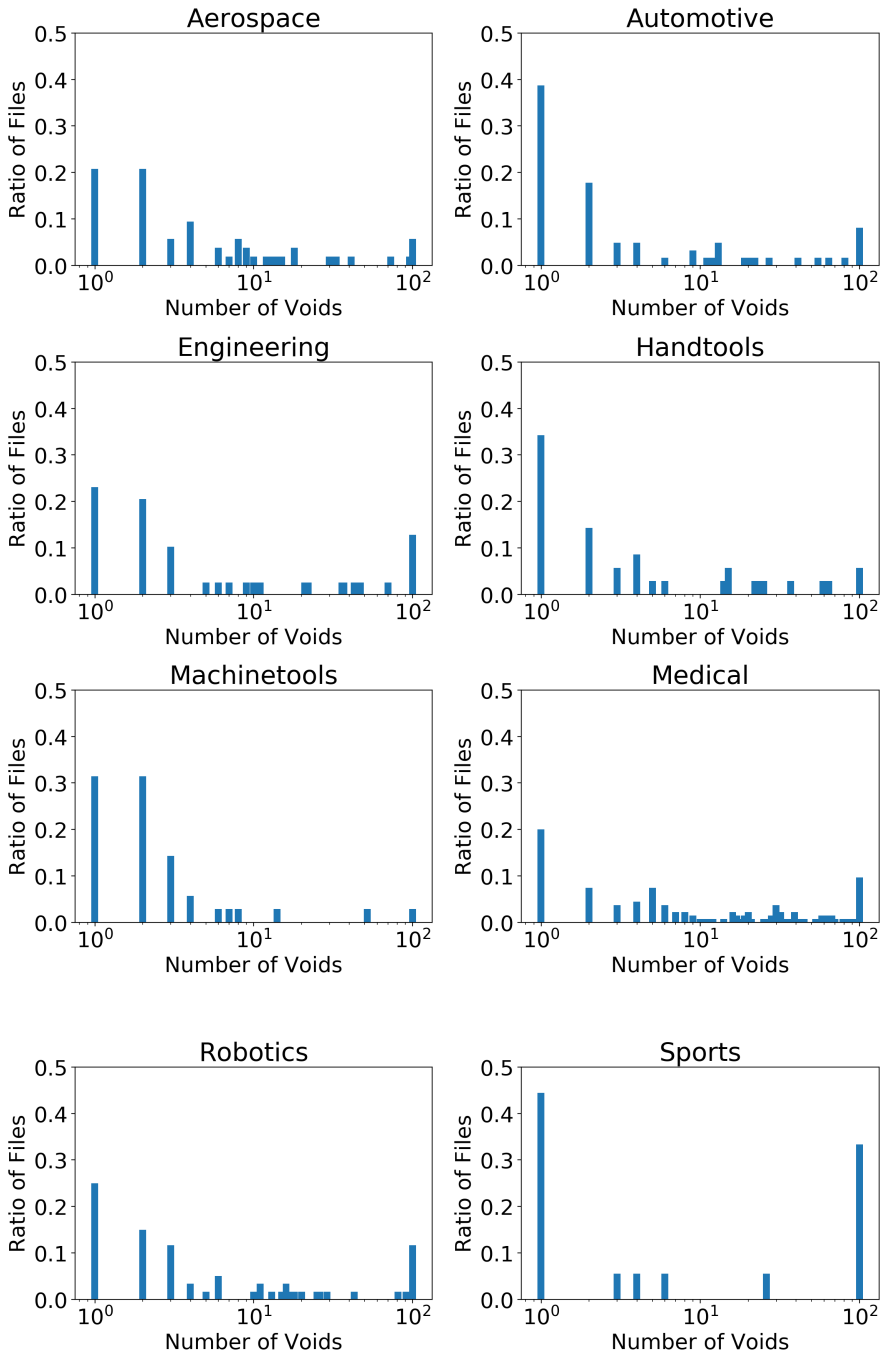


Figure 6.10: Distribution of number of voids across STL files in each category.

(a) Layer 1 (b) Layer 2 (c) Layer 3

Figure 6.11: Analysis of false positives by the flood fill algorithm.

false positives by the flood fill algorithm. The false positve rates were calculated and recorded in Table 6.3.

To analyze the accuracy of the flood fill algorithm in terms of false negative rates, we manually analyzed the STL files that passed the flood fill algorithm. The inspection showed that no STL file has any void inside and the false negative rates for all categories are zero.

We analyzed the false positives and determined that the false positives were due to a slicing problem. Three consecutive layers of an STL file with detected voids are shown in Figure 6.11. The red rectangle highlights the problematic area. When the slicer slices the STL file, it has to rasterize each layer. However, for curves, the results are not very consistent across layers. For layer 2, one pixel can be filled in or not since it is located at the tip of the curve. In this situation, the pixel is filled in. Also notice that this pixel is surrounded by filled pixels to the left, right, top, and bottom. For layer 1 and 3, the pixels at the same locations are filled in. As a result, the pixel in layer 2 mentioned above is considered a void. To mitigate this problem, we can apply a variant of the flood fill algorithm where pixels can propagate diagonally as well. Another possible solution is to require each void to be large enough to be considered a valid void.

Static Structural Analysis The material used in this analysis is ABS and the tensile yield strength is 45 MPa. Stresses above this value leads to failure. Our solution successfully detected stresses on the objects leading to failure. Table 6.4 shows the time to analyze objects in failure stage and the number of files that were successfully analyzed from the database. Pre-analysis success rate is the success rate to import an STL file, get its geometrical features (such as number of vertices, number of holes or defects), and convert it into a volumetric mesh. The stress analysis rate is the success rate of conducting stress analysis.

Category	Pre-analysis success rate	Stress analysis success rate	Average time for analysis (seconds)	Objects that break
Aerospace	83.43%	84.38%	207.38	2.42%
Automotive	86.7%	85.67%	98.73	3.03%
Engineering	89.52%	88.03%	213.48	1.65%
Handtools	88.05%	89.52%	90.91	1.45%
Machinetools	91.3%	87.83%	188.38	1.25%
Medical	78.5%	73.99%	194.37	6.12%
Robotics	88.84%	88.94%	128.8	3.36%
Sports	89.44%	82.2%	139.86	2.04%

Table 6.4: Static structural analysis in eight different categories.

Category	$\lambda = 0.1$	$\lambda = 0.25$	$\lambda = 0.5$
Aerospace	79.79%	62.77%	27.66%
Automotive	81.33%	68%	30.67%
Engineering	75.61%	59.76%	14.63%
Handtools	71.05%	53.95%	19.74%
Machinetools	75%	60.53%	30.26%
Medical	71.91%	56.18%	15.73%
Robotics	78.2%	56.41%	17.95%
Sports	71.05%	51.32%	13.16%

Table 6.5: Percentage of design files containing voids are verified to be malicious based on different mechanical stress thresholds.

Malicious Voids Verification The detected voids have to be verified if they are malicious. The FEA is used to determine the stresses on the objects.

To evaluate the mechanical stress of the STL file, the voids were removed from the files. The histogram of stresses with and without voids in eight categories are shown in Figure 6.12. As seen from Figure 6.12 not all the voids lead to failure. Only certain voids with increases the stress on the objects significantly break. Table 6.5 shows the number of files that had voids were detected as malicious based on different thresholds of λ for 0.1,0.25, and 0.5.

Table 6.6 shows the mechanical stresses for design files with and without 1mm^3, 5mm^3, and 10mm^3 voids. The mechanical stress increases with the increase in the size of the void. The assessment was performed for different sizes of voids. The mechanical stresses are higher in critical applications such as robotics, aerospace, and automotive, and lower in engineering, medical, and sport. To evaluate the mechanical stresses of the STL files, the files were analyzed

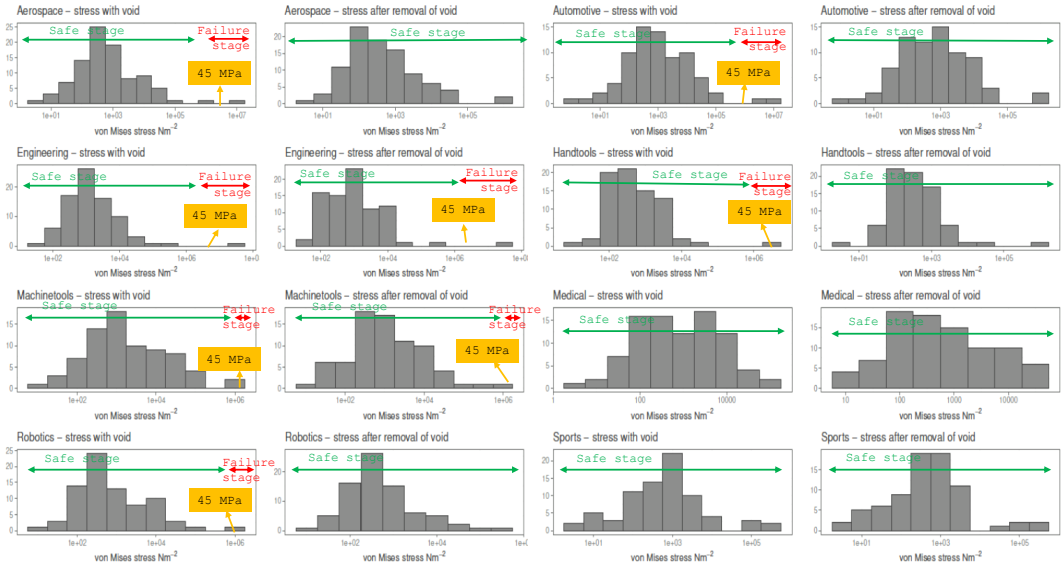


Figure 6.12: Histogram of stresses of flagged with voids and after removal of voids. The Histogram after removal of voids is moved to the left indicating the decrease in the stress levels for the same loads acting on them

with and without suspicious features to check the increase in stress levels.

9 Related Work

Attacks on Designs Turner *et al.* conducted a study on security of designs in additive manufacturing and mentioned that there are no physical or common cyber-security mechanisms that are used on manufacturing machines. They also mentioned that most of the transfers of design files are through insecure e-mails or USB drives. Furthermore, they noted that there are no design integrity checks and the quality control process is expensive yet not tailored to detect cyber attacks [200].

Yampolskiy *et al.* proposed using 3D printers as weapons, or attacks on structural integrity of printed objects. They pointed out that by attacking on additive manufacturing systems, printed objects can be structurally compromised and fail during operation to cause physical damage and loss of lives [218]. Pan *et al.* presented a detailed taxonomy of attacks on AM systems, including attacks on structural integrity by maliciously modifying geometry, and they also presented a detailed taxonomy of defenses against attacks on AM [151]. Wells *et al.* demonstrated that in a subtractive manufacturing process, a change of design model can reduce

Category	Differential stresses for 10mm^3 void	Differential stresses for 5mm^3 void	Differential stresses for 1mm^3 void
Aerospace	98.05%	98.08%	26.07%
Automotive	80.085%	80.91%	26.78%
Engineering	56.45%	56.45%	8.43%
Handtools	52.5%	52.16%	20.39%
Machinetools	82.87%	82.40%	22.36%
Medical	58.98%	59.1%	15.51%
Robotics	79.05%	77.73%	26.03%
Sports	45.28%	45.71%	12.54%

Table 6.6: Percentage change in mechanical stresses with and without voids. Warning is determined by the threshold and the proximity to the yield stress (failure).

the strength of manufactured object without being noticed by machine operators [213]. The same principle can be applied to additive manufacturing.

Strum *et al.* developed an algorithm to automatically insert voids into STL files to reduce the strength of printed objects[190]. They carried out an experiment with human subjects and demonstrated that the insertion of voids was not noticed by additive manufacturing operators. Belikovetsky *et al.* demonstrated that it is possible to inject voids into an STL file to compromise the printed object, and the printed object fails in operation, causing damage to the whole system [29].

Defensive Solutions Strum *et al.* proposed a system with a piezoelectrical sensor to determine the defects in the printed object [189]. Bayens *et al.* developed a three layer framework of acoustic, spectroscopic, and gyroscopic to verify the printing in additive manufacturing systems [27]. However, these techniques are during the printing or after printing, where as our framework verifies before printing saving the time and resources used for printing. Zeltmann *et al.* studied effects of sub-millimeter defects and orientation on printed objects by PolyJet 3D Printing [223]. They found that ultrasonic inspection could not detect sub-millimeter defects in a printed object, but the defects did not decrease the strength of the object. Nevertheless, they pointed out that the work could be extended to SLS 3D printing.

Smykla wrote an algorithm to automatically carry out finite element analysis on STL files of femurs [177]. Stave *et al.* presented a system to automatically analyze and improve the

static structural integrity of 3D printed objects [183]. The method they used was standard FEA. Umetani *et al.* proposed an cross-sectional structural analysis method to analyze structural properties of 3D printed objects based on the fact that printed objects fracture more easily along the building direction [201].

Liu *et al.* combined an octree based polyhedral mesh generation method with the scaled boundary finite element method to perform automatic stress analyses of STL models [120]. The method accepts any valid STL file and can recover sharp features. It can be expected that Liu's method can be employed in TIV to improve the success rate and analysis speed of stress analysis. Past studies have shown using FEA [199] to detect the structural attacks on the design. We perform an automated analysis on a large number of designs and also localize the structural attacks in the design. We also consider the residual stress build up in the printed object due to layer by layer fabrication.

Pure Cyber Solutions Past studies have shown securing important supply chain data using secure channels [100] and others have shown using the block chain to protect the integrity of the supply chain [58]. However, their solutions are based on the assumption that all tools and machines used to create designs are trusted. In other words, the solutions do not consider attacks on design tool chains. In addition, these solutions are targeted towards industrial manufacturing and will be very expensive for people who want to print on their desktop printers at home. We provide a simple to use and less expensive solution for industry as well as home users.

10 Conclusion

STL is a popular design file format for additive manufacturing. Since most of the designs are outsourced and the workstations used by designers are internet facing, they are easy targets for attacks on the additive manufacturing process. Although the traditional software security mechanisms present to detect any malicious activities on the printer, they fail to detect any errors in the STL design file which is one of the inputs to the printer.

Any malicious STL files used for design could lead to devastating effects when used in critical systems. We provided a comprehensive automated solution to detect any malicious designs before they are printed to save the resources used for printing such as raw materials,

energy and time.

We successfully verified 16,000 STL files from wild for structural integrity and was successful in detecting the malicious features present in the STL files. The complete framework is available for verification as a service so that users can submit STL files and get reports of the STL files.

Chapter 7

Conclusion

This dissertation provided security assessment and security solutions to protect against attacks on safety-critical systems. The security solutions protect against attacks on different levels of abstractions in the systems such as physical system, control logic, firmware, modification of actuator values by authenticated operators at vulnerable timings. This dissertation leverages the physical and control invariants properties of the cyber-physical systems in safety-critical systems to provide control flow monitoring, verifying inputs and outputs against safety requirements of the critical systems. The physical and control invariant properties used in this dissertation are EM emanation and system dynamics such as flight dynamics, power flow equations, and swing equations. This dissertation also provided cyber and physical combined access control to protect against changes to the actuators by authenticated users either by remote spoofing attacks or disgruntle employees.

First, we introduce a novel approach to vulnerability assessment in safety-critical systems by cyber-physical interdependency. We provided an attack synthesis method for power grids that is analogous to the penetration testing in cybersecurity. We showed that an attacks can synthesize attack algorithmically using hybrid dynamics of the power grid. We also showed that, the attacker can sequentially modify the output to the actuators to drive the system into an unsafe state without being detected by traditional protection systems such as circuit breakers. In the best case scenario, the attacker can bring down the complete power system leading to blackouts.

Second, we provided a runtime control logic verification solution for the attack against control logic on a cyber-physical system. We provided *just ahead of time* verification solution by predicting the future control paths of the control logic. Just ahead of time verification will provide a time buffer for the solution or controlling operator to react ahead of time and prevent

the system entering into an unsafe state. Since the solution only verifies the control logic till a certain time horizon (just a few steps ahead of time), it reduces the risk of state space explosion of the control logic program's execution paths. By leveraging the physical invariant properties of the systems, we proposed verification techniques that can be used without state space explosion when used on cyber-physical systems.

Third, we provided a contactless side channel control flow monitoring technique by receiving the electromagnetic emanations from the PLC. The EM waves generated when the controller is running is received by the EM probe and was compared with the legitimate control flow of the program that was fingerprinted through a secure medium before the program was deployed on the system. This provides control flow integrity of the program running on the embedded controller. Since this solution does not require any changes to the current running systems, it can be applied to legacy systems as well.

Fourth, we introduce cyber-physical access control to restrict the capabilities of the operators considering the cyber and physical interdependencies for safety-critical systems. By the solution provided we can protect the system from any disgruntle employee, less knowledgable employee and external attacker trying to cause physical damage to the cyber physical system.

Finally, to provide such safety guarantees at a much earlier stage in the pipeline at the time of manufacturing of such safety-critical systems, we provided a safety verification system. Attacks on these design files will lead to a malicious manufactured object which could cause serious damage to property and human life. We introduce TIV framework to detect voids in the STL file before the manufacturing to reduce wastage to time and resources. TIV focuses on security solution to detect attacks against the critical inputs to the controllers such as designs used for describing the 3D printed objects. This security solution will check the structural integrity of the designs before they are printed. Hence, only permitting designs that are benign and will not fail during there normal operational conditions.

With the security solutions for control logic, firmware, inputs to the controllers, and the raw materials, this dissertation is intended to provide a step forward towards a more comprehensive understanding of cyber-physical systems security.

Bibliography

[1] "3dr solo quadcopter, available at https://3dr.com/solo-drone/," 2017.

[2] "Analysis of the Cyber Attack on the Ukrainian Power Grid http://www.nerc.com/pa/CI/ESISAC/Documents/E-ISAC_SANS_Ukraine_DUC_18Mar2016.pdf," 2016.

[3] "Nasa drone hack revealed," *http://www.uasvision.com/2016/02/02/nasa-drone-hack-revealed/*.

[4] "Arizona-Southern California Outages; available at http://www.ferc.gov/legal/staff-reports/04-27-2012-ferc-nerc-report.pdf," 2012.

[5] "Dragonfly: Cyberespionage Attacks Against Energy Suppliers https://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/Dragonfly_Threat_Against_Western_Energy_Suppliers.pdf," July 2014.

[6] "The industrial control systems cyber emergency response team (ics-cert); available at https://ics-cert.us-cert.gov/," 2015.

[7] "Human error cited as cause of Florida blackout; available at http://appanet.files.cms-plus.com/PDFs/March10PPW.pdf," 2008.

[8] "Dynamic taint propagation for Java, author=Haldar, Vivek and Chandra, Deepak and Franz, Michael," in *ACSAC*, 2005.

[9] "Modbus" ; available at http://www.modbus.org/.

[10] "OSIsoft" Real-Time Intelligence; available at https://www.osisoft.com/, 2015.

[11] "PV-Browser: Process Visualization Browser; available at http://pvbrowser.de," 2015.

[12] "NIST Role-Based Access Controls" ; available at http://csrc.nist.gov/rbac/ferraiolo-kuhn-92.pdf.

[13] "ViaSat" Critical Infrastructure Protection; available at https://www.viasat.com/services/critical-infrastructure-protection, 2015.

[14] "Wind River Recognized as Global Embedded Leader: Process Visualization Browser; available at http://www.windriver.com/news/press/pr.html?ID=10681#sthash.PPtTnAIX.dpuf," 2012.

[15] "3d systems," 2018. [Online]. Available: https://www.3dsystems.com/

[16] A. Abur and A. G. Exposito, *Power system state estimation: theory and implementation*. CRC Press, 2004.

[17] B. Adhikari *et al.*, "Strength and failure mechanisms in 3d printed parts," Master's thesis, Aalto University, School of Engineering, 2016.

[18] P. K. Agarwal and J. Matoušek, "Ray shooting and parametric search," *SIAM Journal on Computing*, vol. 22, no. 4, pp. 794–806, 1993.

[19] D. Agrawal, B. Archambeault, J. R. Rao, and P. Rohatgi, *The EM Side—Channel(s)*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 29–45. [Online]. Available: http://dx.doi.org/10.1007/3-540-36400-5_4

[20] P. M. Anderson and A. A. Fouad, *Power system control and stability*. John Wiley & Sons, 2008.

[21] D. T. Askounis and E. Kalfaoglou, "The Greek EMS-SCADA: from the contractor to the user," *Power Systems, IEEE Transactions on*, vol. 15, no. 4, pp. 1423–1427, 2000.

[22] M. J. Assante, "Confirmation of a Coordinated Attack on the Ukrainian Power Grid," SANS Industrial Control Systems Security Blog; available at https://ics.sans.org/blog/2016/01/09/confirmation-of-a-coordinated-attack-on-the-ukrainian-power-grid, Jan. 2015.

[23] K. J. Åström and T. Hägglund, *Advanced PID control*. ISA-The Instrumentation, Systems and Automation Society, 2006.

[24] S. Bak, S. Bogomolov, and T. T. Johnson, "Hyst: A source transformation and translation tool for hybrid automaton models," in *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control*. ACM, 2015, pp. 128–133.

[25] D. Bamburry, "Drones: Designed for product delivery," *Design Management Review*, vol. 26, no. 1, pp. 40–48, 2015.

[26] B. Bamieh and D. F. Gayme, "The price of synchrony: Resistive losses due to phase synchronization in power networks," in *American Control Conference (ACC), 2013*. IEEE, 2013, pp. 5815–5820.

[27] C. Bayens, T. Le, L. Garcia, R. Beyah, M. Javanmard, and S. Zonouz, "See no evil, hear no evil, feel no evil, print no evil? malicious fill patterns detection in additive manufacturing," in *26th USENIX Security Symposium (USENIX Security 17)*. Vancouver, BC: USENIX Association, 2017, pp. 1181–1198. [Online]. Available: https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/bayens

[28] S. Belikovetsky, M. Yampolskiy, J. Toh, and Y. Elovici, "dr0wned-cyber-physical attack with additive manufacturing," *ArXiv e-prints*, 2016.

[29] S. Belikovetsky, M. Yampolskiy, J. Toh, J. Gatlin, and Y. Elovici, "dr0wned – cyber-physical attack with additive manufacturing," in *11th USENIX Workshop on Offensive Technologies (WOOT 17)*. Vancouver, BC: USENIX Association, 2017. [Online]. Available: https://www.usenix.org/conference/woot17/workshop-program/presentation/belikovetsky

[30] D. Beresford, "Exploiting Siemens Simatic S7 PLCs," in *Black Hat USA*, 2011.

[31] H. Berger, "Automating with simatic s7-1200: Configuring, programming and testing with step 7 basic," vol. 11. John Wiley Sons, 2013.

[32] S. Bi and Y. J. Zhang, "Defending mechanisms against false-data injection attacks in the power system state estimation," in *GLOBECOM Workshops (GC Wkshps), 2011 IEEE*. IEEE, 2011, pp. 1162–1167.

[33] A. Bitcraze, "Crazyflie 2.0," 2016.

[34] R. Bodenheim, J. Butts, S. Dunlap, and B. Mullins, "Evaluation of the ability of the shodan search engine to identify internet-facing industrial control devices," *International Journal of Critical Infrastructure Protection*, vol. 7, no. 2, pp. 114–123, 2014.

[35] W. Bolton, *Programmable logic controllers*. Newnes, 2015.

[36] S. A. Boyer, *SCADA: supervisory control and data acquisition*. International Society of Automation, 2009.

[37] D. Brumley, I. Jager, T. Avgerinos, and E. J. Schwartz, "BAP: a binary analysis platform," in *CAV*, 2011.

[38] P. Buneman, S. Khanna, and W.-C. Tan, "Data provenance: Some basic issues," in *FST TCS 2000: Foundations of software technology and theoretical computer science*. Springer, 2000, pp. 87–93.

[39] A. G. Butkovskii, *Distributed control systems*. Elsevier Publishing Company, 1969.

[40] D. I. Buza, F. Juhász, G. Miru, M. Félegyházi, and T. Holczer, "Cryplh: Protecting smart energy systems from targeted attacks with a plc honeypot," in *Smart Grid Security*. Springer, 2014, pp. 181–192.

[41] D. A. Byrne and J. J. Holm, "Shared embedded trace macrocell," Feb. 28 2006, uS Patent 7,007,201.

[42] "Cad crowd," 2018. [Online]. Available: https://www.cadcrowd.com/

[43] G. Canet, S. Couffin, J.-J. Lesage, A. Petit, and P. Schnoebelen, "Towards the Automatic Verification of PLC Programs Written in Instruction List," in *IEEE International Conference on Systems, Man, and Cybernetics*, vol. 4, 2000, pp. 2449–2454.

[44] A. A. Cárdenas, S. Amin, B. Sinopoli, A. Giani, A. Perrig, and others, "Challenges for Securing Cyber Physical Systems," in *DHS Workshop on Future Directions in Cyber-physical Systems Security*, 2009.

[45] D. U. Case, "Analysis of the cyber attack on the ukrainian power grid," *Electricity Information Sharing and Analysis Center (E-ISAC)*, 2016.

[46] M. Chan, D. Ricketts, S. Lerner, and G. Malecha, "Formal verification of stability properties of cyber-physical systems," 2016.

[47] D. Chatterjee, J. Webb, Q. Gao, M. Vaiman, M. Vaiman, and M. Povolotskiy, "N-1-1 AC contingency analysis as a part of NERC compliance studies at midwest ISO," in *Transmission and Distribution Conference and Exposition, 2010 IEEE PES*. IEEE, 2010, pp. 1–7.

[48] X. Chen, E. Ábrahám, and S. Sankaranarayanan, "Flow*: An analyzer for non-linear hybrid systems," in *Computer Aided Verification*. Springer, 2013, pp. 258–263.

[49] S. Cheung, B. Dutertre, M. Fong, U. Lindqvist, K. Skinner, and A. Valdes, "Using Model-based Intrusion Detection for SCADA Networks," in *Proc. SCADA Security Scientific Symposium*, 2007.

[50] E. Chien, L. OMurchu, and N. Falliere, "W32.Duqu - The precursor to the next Stuxnet," Symantic Security Response, Tech. Rep., nov 2011.

[51] A. Clark, Q. Zhu, R. Poovendran, and T. Basar, "An impact-aware defense against Stuxnet," in *American Control Conference (ACC), 2013*. IEEE, 2013, pp. 4140–4147.

[52] C. Clavier, "Side channel analysis for reverse engineering (scare)-an improved attack against a secret a3/a8 gsm algorithm," 2004.

[53] B. P. Conner, G. P. Manogharan, A. N. Martof, L. M. Rodomsky, C. M. Rodomsky, D. C. Jordan, and J. W. Limperos, "Making sense of 3-d printing: Creating a map of additive manufacturing products and services," *Additive Manufacturing*, vol. 1, pp. 64–76, 2014.

[54] E. C. R. Council, "The economic impacts of the August 2003 blackout," *Washington, DC*, 2004.

[55] J. Daafouz, P. Riedinger, and C. Iung, "Stability analysis and control synthesis for switched systems: a switched lyapunov function approach," *Automatic Control, IEEE Transactions on*, vol. 47, no. 11, pp. 1883–1887, 2002.

[56] C. M. Davis and T. J. Overbye, "Multiple element contingency screening," *Power Systems, IEEE Transactions on*, vol. 26, no. 3, pp. 1294–1301, 2011.

[57] K. De Brabandere, B. Bolsens, J. Van den Keybus, A. Woyte, J. Driesen, and R. Belmans, "A voltage and frequency droop control method for parallel inverters," *IEEE Trans. Power Elec.*, vol. 22, no. 4, pp. 1107–1115, 2007.

[58] deloitte, "Using blockchain to drive supply chain innovation," 2017. [Online]. Available: https://www2.deloitte.com/content/dam/Deloitte/us/Documents/process-and-operations/us-blockchain-to-drive-supply-chain-innovation.pdf

[59] J. Ding, E. Li, H. Huang, and C. Tomlin, "Reachability-based synthesis of feedback policies for motion planning under bounded disturbances," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, May 2011, pp. 2160–2165.

[60] P. P. Directive, "Presidential policy directive—critical infrastructure security and resilience," 2013.

[61] A. Donzé, "Breach, a toolbox for verification and parameter synthesis of hybrid systems," in *Computer Aided Verification*. Springer, 2010, pp. 167–170.

[62] A. Donzé, B. Krogh, and A. Rajhans, "Parameter synthesis for hybrid systems with an application to simulink models," in *Hybrid Systems: Computation and Control*. Springer, 2009, pp. 165–179.

[63] P. S. Duggirala, S. Mitra, and M. Viswanathan, "Verification of annotated models from executions," in *Proceedings of the Eleventh ACM International Conference on Embedded Software*. IEEE Press, 2013, p. 26.

[64] P. S. Duggirala, S. Mitra, M. Viswanathan, and M. Potok, "C2e2: A verification tool for stateflow models," in *Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2015, pp. 68–82.

[65] M. Egerstedt and B. Mishra, Eds., *Hybrid Systems: Computation and Control, 11th International Workshop, HSCC 2008, St. Louis, MO, USA, April 22-24, 2008. Proceedings*, ser. Lecture Notes in Computer Science, vol. 4981. Springer, 2008.

[66] T. Eisenbarth, C. Paar, and B. Weghenkel, "Building a side channel based disassembler," in *Transactions on computational science X*. Springer, 2010, pp. 78–99.

[67] W. Enck, P. Gilbert, S. Han *et al.*, "TaintDroid: an information-flow tracking system for realtime privacy monitoring on smartphones," *ACM Trans. Comp. Sys.*, vol. 32, no. 2, p. 5, 2014.

[68] ESET, "Acad/medre.a 10000's of autocad designs leaked in suspected industrial espionage," 2014. [Online]. Available: https://www.welivesecurity.com/media_files/white-papers/ESET_ACAD_Medre_A_whitepaper.pdf

[69] s. Etigowni, D. Tian, G. Hernandez, K. Butler, and S. Zonouz, "Cpac: Mitigating attacks against critical infrastructure with cyber-physical access control," in *to appear in Annual Computer Security Applications Conference (ACSAC); available at http://dx.doi.org/10.1145/2991079.2991126*, 2016.

[70] European Network and Information Security Agency (ENISA). (2011) Protecting industrial control systems - recommendations for Europe and Member States. https://www.enisa.europa.eu/.

[71] F-Secure Labs, "BLACKENERGY and QUEDAGH: The convergence of crimeware and APT attacks," 2016.

[72] N. Falliere, L. O. Murchu, and E. Chien, "W32.Stuxnet Dossier," Symantic Security Response, Tech. Rep., Oct. 2010.

[73] H. Fawzi, P. Tabuada, and S. Diggavi, "Secure Estimation and Control for Cyber-Physical Systems Under Adversarial Attacks," *IEEE Trans. Automat. Contr.*, vol. 59, no. 6, pp. 1454–1467, Jun. 2014.

[74] 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 *NDSS*, 2016.

[75] J. Foust, "Spacex unveils its 21st century spaceship," *NewSpace Journal*, 2014. [Online]. Available: http://www.newspacejournal.com/2014/05/30/spacex-unveils-its-21st-century-spaceship/

[76] A. Fras and T. Dang, "Improving industrial application's performances with an Historian," in *IEEE Intl. Conf. on Industrl Tech.*, 2004.

[77] S. Gallagher and W. S. Marras, "Tolerance of the lumbar spine to shear: a review and recommended exposure limits," *Clinical Biomechanics*, vol. 27, no. 10, pp. 973–978, 2012.

[78] D. Gao, M. K. Reiter, and D. Song, "Beyond output voting: Detecting compromised replicas using hmm-based behavioral distance," *IEEE Transactions on Dependable and Secure Computing*, vol. 6, no. 2, pp. 96–110, 2009.

[79] 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)*, 2017.

[80] D. Genkin, L. Pachmanov, I. Pipman, A. Shamir, and E. Tromer, "Physical key extraction attacks on pcs," *Communications of the ACM*, vol. 59, no. 6, pp. 70–79, 2016.

[81] D. Genkin, L. Pachmanov, I. Pipman, E. Tromer, and Y. Yarom, "Ecdsa key extraction from mobile devices via nonintrusive physical side channels," Tech. Rep., 2016.

[82] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with lstm," *Neural computation*, vol. 12, no. 10, pp. 2451–2471, 2000.

[83] R. Gerth, D. Peled, M. Y. Vardi, and P. Wolper, "Simple on-the-fly automatic verification of linear temporal logic," in *International Symposium on Protocol Specification, Testing and Verification*. IFIP, 1995.

[84] A. S. Glassner, *An introduction to ray tracing*. Elsevier, 1989.

[85] J. Glover, M. Sarma, and T. Overbye, *Power System Analysis and Design*. Cengage Learning, 2011.

[86] W. M. Goble, *Control Systems Safety Evaluation and Reliability*. International Society of Automation, 2010.

[87] C. R. A. Gonzalez and J. H. Reed, "Power fingerprinting in sdr & cr integrity assessment," in *MILCOM 2009-2009 IEEE Military Communications Conference*. IEEE, 2009, pp. 1–7.

[88] J. J. Grainger and W. D. Stevenson, *Power system analysis*. McGraw-Hill New York, 1994, vol. 31.

[89] S. Greene, "Margin and sensitivity methods for security analysis of electric power systems," Ph.D. dissertation, University of Wisconsin–Madison, 1998.

[90] J. Groote, S. van Vlijmen, and J. Koorn, "The Safety Guaranteeing System at Station Hoorn-Kersenboogerd," in *Tenth Annual Conference on Systems Integrity, Software Safety and Process Security*, June 1995, pp. 57–68.

[91] L. Habets, P. J. Collins, and J. H. van Schuppen, "Reachability and control synthesis for piecewise-affine hybrid systems on simplices," *Automatic Control, IEEE Transactions on*, vol. 51, no. 6, pp. 938–948, 2006.

[92] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: an update," *ACM SIGKDD explorations newsletter*, vol. 11, no. 1, pp. 10–18, 2009.

[93] N. Headquarters, "Navair marks first flight with 3-d printed, safety-critical parts," *NAVAIR News*, 2016. [Online]. Available: http://www.navair.navy.mil/index.cfm?fuseaction=home.NAVAIRNewsStory&id=6323

[94] T. K. Ho, "Random decision forests," in *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*, vol. 1. IEEE, 1995, pp. 278–282.

[95] 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*, ser. CCS '16. New York, NY, USA: ACM, 2016, pp. 883–894. [Online]. Available: http://doi.acm.org/10.1145/2976749.2978323

[96] J. Hoskins, *IBM AS/400: a business perspective*. Wiley, 1991.

[97] Z. Huang and S. Mitra, "Proofs from simulations and modular annotations," in *Proceedings of the 17th international conference on Hybrid systems: computation and control*. ACM, 2014, pp. 183–192.

[98] Z. Huang, Y. Wang, S. Mitra, G. E. Dullerud, and S. Chaudhuri, "Controller synthesis with inductive proofs for piecewise linear systems: an smt-based algorithm," *arXiv preprint arXiv:1509.04623*, 2015.

[99] R. Huuck, "Semantics and Analysis of Instruction List Programs," *Electronic Notes in Theoretical Computer Science*, vol. 115, pp. 3–18, 2005.

[100] "Identify3d," 2018. [Online]. Available: https://identify3d.com/

[101] V. M. Igure, S. A. Laughter, and R. D. Williams, "Security issues in SCADA networks," *Computers & Security*, vol. 25, no. 7, pp. 498–506, 2006.

[102] A. Iliev, N. Kyurkchiev, and S. Markov, "On the approximation of the step function by some sigmoid functions," *Mathematics and Computers in Simulation*, vol. 133, pp. 223–234, 2017.

[103] "i.materialise," 2018. [Online]. Available: https://i.materialise.com

[104] M. G. Ioannides, "Design and implementation of plc-based monitoring control system for induction motor," *Energy Conversion, IEEE Transactions on*, vol. 19, no. 3, pp. 469–476, 2004.

[105] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[106] T. Jackson, "Navy partnership goes to new depths with first 3d-printed submersible," *Office of Energy Efficiency & Renewable Energy*, 2017. [Online]. Available: https://energy.gov/eere/articles/navy-partnership-goes-new-depths-first-3d-printed-submersible

[107] A. Y. Javaid, W. Sun, V. K. Devabhaktuni, and M. Alam, "Cyber security threat analysis and modeling of an unmanned aerial vehicle system," in *Homeland Security (HST), 2012 IEEE Conference on Technologies for*. IEEE, 2012, pp. 585–590.

[108] S. Jha and S. A. Seshia, "A Theory of Formal Synthesis via Inductive Learning," *ArXiv e-prints*, May 2015.

[109] K.-H. John and M. Tiegelkamp, *IEC 61131-3: programming industrial automation systems*. Springer Science & Business Media, 2010.

[110] S. Kamkar, "Skyjack: autonomous drone hacking," *Online*, 2013. [Online]. Available: http://samy.pl/skyjack

[111] D. K. Kaynar, N. Lynch, R. Segala, and F. Vaandrager, *The Theory of Timed I/O Automata*, ser. Synthesis Lectures on Computer Science. Morgan Claypool, November 2005, also available as Technical Report MIT-LCS-TR-917.

[112] G. Klein, K. Elphinstone, G. Heiser, J. Andronick, D. Cock, P. Derrin, D. Elkaduwe, K. Engelhardt, R. Kolanski, M. Norrish *et al.*, "sel4: Formal verification of an os kernel," in *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*. ACM, 2009, pp. 207–220.

[113] P. Kundur, N. J. Balu, and M. G. Lauby, *Power system stability and control*. McGraw-hill New York, 1994, vol. 7.

[114] E. V. Kuz'min and V. A. Sokolov, "On construction and verification of PLC-programs," *Modelirovanie i Analiz Informatsionnykh Sistem [Modeling and Analysis of Information Systems]*, vol. 19, no. 4, pp. 25–36, 2012.

[115] R. Lepofsky, "North american energy council security standard for critical infrastructure protection (nerc cip)," in *The Manager's Guide to Web Application Security:*. Springer, 2014, pp. 165–176.

[116] L. Leshin, P. Mahaffy, C. Webster, M. Cabane, P. Coll, P. Conrad, P. Archer, S. Atreya, A. Brunner, A. Buch *et al.*, "Volatile, isotope, and organic analysis of martian fines with the Mars Curiosity rover," *Science*, vol. 341, no. 6153, p. 1238937, 2013.

[117] T. G. Lewis, *Critical infrastructure protection in homeland security: defending a networked nation*. John Wiley & Sons, 2006.

[118] J. Leyden, "Polish Teen Derails Tram after Hacking Train Network," http://www.theregister.co.uk/2008/01/11/tram_hack/, 2008.

[119] X. Liao, D. Formby, C. Day *et al.*, "Towards secure metering data analysis via distributed differential privacy," in *IEEE DSN*, 2014.

[120] Y. Liu, A. A. Saputra, J. Wang, F. Tin-Loi, and C. Song, "Automatic polyhedral mesh generation and scaled boundary finite element analysis of stl models," *Computer Methods in Applied Mechanics and Engineering*, vol. 313, pp. 106 – 132, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0045782516306326

[121] Y. Liu, L. Wei, Z. Zhou, K. Zhang, W. Xu, and Q. Xu, "On code execution tracking via power side-channel," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 1019–1031.

[122] 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)*, vol. 14, no. 1, p. 13, 2011.

[123] Z. Lu and Z. Zhang, "Bad data identification based on measurement replace and standard residual detection," *Automation of Electric Power Systems*, vol. 13, p. 011, 2007.

[124] K. Manamcheri, S. Mitra, S. Bak, and M. Caccamo, "A step towards verification and synthesis from simulink/stateflow models," in *Proceedings of the 14th international conference on Hybrid systems: computation and control*. ACM, 2011, pp. 317–318.

[125] C. Masone *et al.*, "Role Definition Language (RDL): A language to describe context-aware roles," 2002.

[126] S. McLaughlin and P. McDaniel, "SABOT: specification-based payload generation for programmable logic controllers," in *ACM CCS*, 2012.

[127] S. McLaughlin, P. McDaniel, and W. Aiello, "Protecting consumer privacy from electric load monitoring," in *ACM CCS*, 2011.

[128] S. McLaughlin, S. Zonouz, D. Pohly, and P. McDaniel, "A trusted safety verifier for process controller code," in *NDSS*, 2014.

[129] D. J. Meagher, *Octree encoding: A new technique for the representation, manipulation and display of arbitrary 3-d objects by computer*. Electrical and Systems Engineering Department Rensseiaer Polytechnic Institute Image Processing Laboratory, 1980.

[130] L. Meier, J. Camacho, B. Godbolt, J. Goppert, L. Heng, M. Lizarraga *et al.*, "Mavlink: Micro air vehicle communication protocol," *Online]. Tillgänglig: http://qgroundcontrol.org/mavlink/start.[Hämtad 2014-05-22]*, 2013.

[131] S. Mitra, "A verification framework for hybrid systems," Ph.D. dissertation, Massachusetts Institute of Technology, 2007.

[132] Y. Mo and R. M. Murray, "Multi-dimensional state estimation in adversarial environment," in *34th Chinese Control Conference (CCC)*. IEEE, 2015, pp. 4761–4766.

[133] S. Mohan, S. Bak, E. Betti, H. Yun, L. Sha, and M. Caccamo, "S3A: Secure System Simplex Architecture for Enhanced Security of Cyber-Physical Systems," http://arxiv.org, 2012.

[134] T. K. Moon, "The expectation-maximization algorithm," *IEEE Signal processing magazine*, vol. 13, no. 6, pp. 47–60, 1996.

[135] T. H. Morris, A. K. Srivastava, B. Reaves, K. Pavurapu, S. Abdelwahed, R. Vaughn, W. McGrew, and Y. Dandass, "Engineering future cyber-physical energy systems: Challenges, research needs, and roadmap," in *IEEE North American Power Symposium (NAPS), 2009*, 2009, pp. 1–6.

[136] M. Msgna, K. Markantonakis, and K. Mayes, *The B-Side of Side Channel Leakage: Control Flow Security in Embedded Systems*. Cham: Springer International Publishing, 2013, pp. 288–304. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-04283-1_18

[137] 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," tech. report SAND2013-8274, Sandia National Laboratories, Tech. Rep., 2013.

[138] A. Muller and S. Karevska, "How will 3d printing make your company the strongest link in the value chain," *EY's Global 3D Printing Report*, 2016.

[139] D. Muniraj and M. Farhood, "A framework for detection of sensor attacks on small unmanned aircraft systems," in *Unmanned Aircraft Systems (ICUAS), 2017 International Conference on*. IEEE, 2017, pp. 1189–1198.

[140] National Energy Regulatory Comission, "NERC CIP 002 1 - Critical Cyber Asset Identification," 2006.

[141] A. Nazari, N. Sehatbakhsh, M. Alam, A. Zajic, and M. Prvulovic, "Eddie: Em-based detection of deviations in program execution," in *Proceedings of the 44th Annual International Symposium on Computer Architecture*. ACM, 2017, pp. 333–346.

[142] Nell Nelson,Rob VandenBrink, "The Impact of Dragonfly Malware on Industrial Control Systems https://www.sans.org/reading-room/whitepapers/ICS/impact-dragonfly-malware-industrial-control-systems-36672," 2016.

[143] North American Electric Reliability Corporation, 2011, stuxnet attackers used 4 Windows zero-day exploits, available at http://www.nerc.com/pa/Stand/Project%20200902%20Realtime%20Reliability%20Monitoring%20and/Project_2009-02_rmacsdt_white_paper_021611.pdf.

[144] R. Novak, "Side-channel attack on substitution blocks," in *International Conference on Applied Cryptography and Network Security*. Springer, 2003, pp. 307–318.

[145] D. of Energy Office of Electricity Delivery and E. Reliability, "North american electric reliability corporation critical infrastructure protection (nerc-cip) standards; available at http://www.nerc.com/pa/Stand/Pages/CIPStandards.aspx," 2015.

[146] D. of Homeland Security, "Cyber security division transition to practice technology guide," 2014.

[147] S. Ould Biha, "A Formal Semantics of PLC Programs in Coq," in *IEEE COMPSAC*, 2011.

[148] T. J. Overbye and J. D. Weber, "Visualization of power system data," in *System Sciences, 2000. Proceedings of the 33rd Annual Hawaii International Conference on*. IEEE, 2000, pp. 7–pp.

[149] P. Paganini, "Hack-proof drones possible with hacms technology," *http://resources. infosecinstitute. com/hack-proof-drones-possiblehacms-technology*, 2014.

[150] M. Pajic, J. Weimer, N. Bezzo, P. Tabuada, O. Sokolsky, I. Lee, and G. J. Pappas, "Robustness of attack-resilient state estimators," in *ICCPS'14: ACM/IEEE 5th International Conference on Cyber-Physical Systems (with CPS Week 2014)*. IEEE Computer Society, 2014, pp. 163–174.

[151] Y. Pan, J. White, D. C. Schmidt, A. Elhabashy, L. Sturm, J. A. Camelio, and C. Williams, "Taxonomies for reasoning about cyber-physical attacks in iot-based manufacturing systems," *International Journal of Interactive Multimedia & Artificial Intelligence*, vol. 4, pp. 45–54, 2017.

[152] T. Park and P. I. Barton, "Formal Verification of Sequence Controllers," *Computers & Chemical Engineering*, vol. 23, no. 11, pp. 1783–1793, 2000.

[153] D. G. Peterson, "Project Basecamp at S4," http://www.digitalbond.com/2012/01/19/project-basecamp-at-s4/, January 2012.

[154] L. Pietre-Cambacédes, M. Tritschler, and G. N. Ericsson, "Cybersecurity myths on power control systems: 21 misconceptions and false beliefs," *IEEE Transactions on Power Delivery*, vol. 26, no. 1, pp. 161–172, 2011.

[155] A. Platzer, "Logic and compositional verification of hybrid systems," in *Computer Aided Verification*. Springer, 2011, pp. 28–43.

[156] J. Pollet, "Electricity for Free? The Dirty Underbelly of SCADA and Smart Meters," in *Proceedings of Black Hat USA 2010*, July 2010.

[157] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[158] M. Rashed, M. Ashraf, R. Mines, and P. J. Hazell, "Metallic microlattice materials: A current state of the art on manufacturing, mechanical properties and applications," *Materials & Design*, vol. 95, pp. 518–533, 2016.

[159] S. Ravuri and A. Stolcke, "A comparative study of recurrent neural network models for lexical domain classification," in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 6075–6079.

[160] C. Ribeiro, A. Zuquete, P. Ferreira, and P. Guedes, "SPL: An Access Control Language for Security Policies and Complex Constraints." in *NDSS*, vol. 1, 2001.

[161] M. Robinson, "Knocking my neighbor's kid's cruddy drone offline," in *Defcon*, 2015.

[162] A. Rogers and J. Hill, *Unmanned: Drone warfare and global security*. Between the Lines, 2014.

[163] J. Rrushi, H. Farhangi, C. Howey, K. Carmichael, and J. Dabell, "A Quantitative Evaluation of the Target Selection of Havex ICS Malware Plugin," *Industrial Control System Security (ICSS) Workshop*, 2015.

[164] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, "Role-based access control models," *Computer*, vol. 29, no. 2, pp. 38–47, 1996.

[165] R. Santamarta, "Here be backdoors: A journey into the secrets of industrial firmware," *Black Hat USA*, 2012.

[166] J. Schumann, P. Moosbrugger, and K. Y. Rozier, "R2u2: monitoring and diagnosis of security threats for unmanned aerial systems," in *Runtime Verification*. Springer, 2015, pp. 233–249.

[167] M. D. Schwartz, J. Mulder, J. Trent, and W. D. Atkins, "Control system devices: Architectures and supply channels overview," *Sandia Report SAND2010-5183, Sandia National Laboratories, Albuquerque, New Mexico*, 2010.

[168] L. Sha, "Using simplicity to control complexity," *IEEE Software*, no. 4, pp. 20–28, 2001.

[169] N. Shachtman, "Computer virus hits us drone fleet," *CNN. com*, 2011.

[170] S. Shane and D. E. Sanger, "Drone crash in iran reveals secret us surveillance effort," *The New York Times*, vol. 7, 2011.

[171] "Shapeways," 2018. [Online]. Available: https://www.shapeways.com/

[172] J. S. Shapiro, J. M. Smith, and D. J. Farber, "EROS: a fast capability system," in *ACM SOSP*, 1999.

[173] A. Shareef, Y. Zhu, M. Musavi, and B. Shen, "Comparison of mlp neural network and kalman filter for localization in wireless sensor networks," in *Proceedings of the 19th IASTED International conference on parallel and distributed computing and systems*. ACTA Press, 2007, pp. 323–330.

[174] S. J. Sheather, "Weighted least squares," in *A Modern Approach to Regression with R*, ser. Springer Texts in Statistics. Springer New York, 2009, pp. 115–123.

[175] S. Singh and S. Srivastava, "Improved voltage and reactive power distribution factors for outage studies," *IEEE Trans. Power Systems*, vol. 12, no. 3, pp. 1085–1093, 1997.

[176] J. Slay and M. Miller, "Lessons Learned from the Maroochy Water Breach," in *Critical Infrastructure Protection*. Springer, 2007, pp. 73–82.

[177] A. SMYKLA, "Developing fem models for difficult to describe 3d structures by using stl files - the case study of the human femur modelling," *Archives of Civil and Mechanical Engineering*, vol. 6, no. 2, pp. 5 – 20, 2006. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1644966512602490

[178] A. Solar-Lezama, *Program synthesis by sketching*. ProQuest, 2008.

[179] Y. Son, H. Shin, D. Kim, Y.-S. Park, J. Noh, K. Choi, J. Choi, Y. Kim *et al.*, "Rocking drones with intentional sound noise on gyroscopic sensors." in *USENIX Security*, 2015, pp. 881–896.

[180] 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*, ser. CCS '16. New York, NY, USA: ACM, 2016, pp. 895–907. [Online]. Available: http://doi.acm.org/10.1145/2976749.2978300

[181] D. Song, D. Brumley, H. Yin, J. Caballero, I. Jager, M. G. Kang, Z. Liang, J. Newsome, P. Poosankam, and P. Saxena, "BitBlaze: A new approach to computer security via binary analysis," in *International Conference on Information Systems Security*. Springer, 2008, pp. 1–25.

[182] S. Sridhar, A. Hahn, and M. Govindarasu, "Cyber-Physical System Security for the Electric Power Grid," *Proc. IEEE*, vol. 100, no. 1, pp. 210–224, Jan. 2012.

[183] O. Stava, J. Vanek, B. Benes, N. Carr, and R. Měch, "Stress relief: Improving structural strength of 3d printable objects," *ACM Transactions on Graphics*, vol. 31, no. 4, pp. 48:1–48:11, Jul. 2012. [Online]. Available: http://doi.acm.org/10.1145/2185520.2185544

[184] S. Stone and M. Temple, "Radio-frequency-based anomaly detection for programmable logic controllers in the critical infrastructure," *International Journal of Critical Infrastructure Protection*, vol. 5, no. 2, pp. 66–73, 2012.

[185] S. J. Stone, "Radio frequency based programmable logic controller anomaly detection," DTIC Document, Tech. Rep., 2013.

[186] S. J. Stone, M. A. Temple, and R. O. Baldwin, "Detecting anomalous programmable logic controller behavior using rf-based hilbert transform features and a correlation-based verification process," *International Journal of Critical Infrastructure Protection*, vol. 9, pp. 41–51, 2015.

[187] K. Stouffer, J. Falco, and K. Scarfone, "Guide to industrial control systems (ICS) security," *NIST special publication*, pp. 800–82, 2011.

[188] K. A. Stouffer, J. A. Falco, and K. A. Scarfone, "Sp 800-82. guide to industrial control systems (ics) security: Supervisory control and data acquisition (scada) systems, distributed control systems (dcs), and other control system configurations such as programmable logic controllers (plc)," 2011.

[189] L. Sturm, M. Albakri, C. B. Williams, and P. Tarazaga, "In-situ detection of build defects in additive manufacturing via impedance-based monitoring," in *International Solid Freeform Fabrication Symposium*, 2016.

[190] L. D. Sturm, C. B. Williams, J. A. Camelio, J. White, and R. Parker, "Cyber-physical vulnerabilities in additive manufacturing systems: A case study attack on the .stl file with human subjects," *Journal of Manufacturing Systems*, vol. 44, no. Part 1, pp. 154 – 164, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0278612517300961

[191] J. E. Sullivan and D. Kamensky, "How cyber-attacks in ukraine show the vulnerability of the us power grid," *The Electricity Journal*, vol. 30, no. 3, pp. 30–35, 2017.

[192] P. Sun, X. J. Tang, H. H. Wang, W. Z. Zhong, J. Wang, and H. M. Luo, "Review of AGC and Primary Frequency Regulation," vol. 986, pp. 1263–1267, 2014.

[193] A. Support, "How to detect and remove the acad.vlx virus," 2015. [Online]. Available: https://knowledge.autodesk.com/support/autocad/learn-explore/caas/sfdcarticles/sfdcarticles/How-to-detect-and-remove-the-Acad-vlx-virus-s.html

[194] TechNavio. (2014) Global Industrial Control Systems (ICS) Security Market 2014-2018. http://www.technavio.com/report/global-industrial-control-systems-ics-security-market%C2%A02014-2018.

[195] "Thingiverse," 2018. [Online]. Available: https://www.thingiverse.com/

[196] S. Tom, D. Christiansen, and D. Berrett, "Recommended pratice for patch management of control systems," *DHS control system security program (CSSP) Recommended Practice*, 2008.

[197] J. Torres, J. Cotelo, J. Karl, and A. P. Gordon, "Mechanical property optimization of fdm pla in shear with multiple objectives," *Jom*, vol. 67, no. 5, pp. 1183–1193, 2015.

[198] N. Toshida, M. Uesugi, Y. Nakata, M. Nomoto, and T. Uchida, "Open distributed EMS/SCADA system," *Hitachi Review*, vol. 47, no. 5, pp. 208–213, 1998.

[199] N. G. Tsoutsos, H. Gamil, and M. Maniatakos, "Secure 3d printing: reconstructing and validating solid geometries using toolpath reverse engineering," in *Proceedings of the 3rd ACM Workshop on Cyber-Physical System Security*. ACM, 2017, pp. 15–20.

[200] H. Turner, J. White, J. A. Camelio, C. Williams, B. Amos, and R. Parker, "Bad parts: Are our manufacturing systems at risk of silent cyberattacks?" *IEEE Security & Privacy*, vol. 13, no. 3, pp. 40–47, 2015.

[201] N. Umetani and R. Schmidt, "Cross-sectional structural analysis for 3d printing optimization," in *SIGGRAPH Asia 2013 Technical Briefs*, ser. SA '13. New York, NY, USA: ACM, 2013, pp. 5:1–5:4. [Online]. Available: http://doi.acm.org/10.1145/2542355.2542361

[202] A. V. Uriarte-Arcia, I. López-Yáñez, and C. Yáñez-Márquez, "One-hot vector hybrid associative classifier for medical data classification," *PloS one*, vol. 9, no. 4, p. e95715, 2014.

[203] U.S. Department of Energy Office of Electricity Delivery and Energy Reliability, "A Summary of Control System Security Standards Activities in the Energy Sector," October 2005.

[204] S. E. Valentine, "PLC code vulnerabilities through SCADA systems," Ph.D. dissertation, University of South Carolina, 2013.

[205] S. VanDeBogart, P. Efstathopoulos, E. Kohler *et al.*, "Labels and Event Processes in the Asbestos Operating System," *ACM Trans. Comput. Sys.*, vol. 25, no. 4, 2007.

[206] C. L. Ventola, "Medical applications for 3d printing: current and projected uses," *Pharmacy and Therapeutics*, vol. 39, no. 10, p. 704, 2014.

[207] Verizon, "Data breach digest. scenarios from the field," 2016. [Online]. Available: http://www.verizonenterprise.com/resources/reports/rp_data-breach-digest_xg_en.pdf

[208] D. Vermoen, M. Witteman, and G. N. Gaydadjiev, "Reverse engineering java card applets using power analysis," in *IFIP International Workshop on Information Security Theory and Practices*. Springer, 2007, pp. 138–149.

[209] M. Vutsinas, "Contingency analysis using synchrophasor measurements," Ph.D. dissertation, Clemson University, 2008.

[210] P.-S. Wang, Y. Liu, Y.-X. Guo, C.-Y. Sun, and X. Tong, "O-cnn: Octree-based convolutional neural networks for 3d shape analysis," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 4, p. 72, 2017.

[211] C. Warrender, S. Forrest, and B. Pearlmutter, "Detecting intrusions using system calls: Alternative data models," in *Security and Privacy, 1999. Proceedings of the 1999 IEEE Symposium on*. IEEE, 1999, pp. 133–145.

[212] J. Weiss, "Are the NERC CIPS making the grid less reliable," *Control Global*, 2009.

[213] L. J. Wells, J. A. Camelio, C. B. Williams, and J. White, "Cyber-physical security challenges in manufacturing systems," *Manufacturing Letters*, vol. 2, no. 2, pp. 74 – 77, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2213846314000066

[214] C. Wright, C. Cowan, J. Morris, S. Smalley, and G. Kroah-Hartman, "Linux security module framework," in *Ottawa Linux Symposium*, vol. 8032, 2002, pp. 6–16.

[215] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3d shapenets: A deep representation for volumetric shapes," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1912–1920.

[216] C. Wueest, "Targeted attacks against the energy sector," *Symantec Security Response, Mountain View, CA*, 2014.

[217] K. Xu, D. D. Yao, B. G. Ryder, and K. Tian, "Probabilistic program modeling for high-precision anomaly classification," in *2015 IEEE 28th Computer Security Foundations Symposium*. IEEE, 2015, pp. 497–511.

[218] M. Yampolskiy, A. Skjellum, M. Kretzschmar, R. A. Overfelt, K. R. Sloan, and A. Yasinsac, "Using 3d printers as weapons," *International Journal of Critical Infrastructure Protection*, vol. 14, pp. 58 – 71, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1874548215300330

[219] D.-Y. Yeung and Y. Ding, "Host-based intrusion detection using dynamic and static behavioral models," *Pattern recognition*, vol. 36, no. 1, pp. 229–243, 2003.

[220] K. Yuan, Q. Ling, and W. Yin, "On the convergence of decentralized gradient descent," *SIAM Journal on Optimization*, vol. 26, no. 3, pp. 1835–1854, 2016.

[221] J. Zaddach, L. Bruno, A. Francillon, and D. Balzarotti, "AVATAR: A framework to support dynamic security analysis of embedded systems firmwares," in *NDSS*, 2014.

[222] N. Zeldovich, S. Boyd-Wickizer, E. Kohler, and D. Mazières, "Making information flow explicit in HiStar," in *OSDI*, 2006.

[223] S. E. Zeltmann, N. Gupta, N. G. Tsoutsos, M. Maniatakos, J. Rajendran, and R. Karri, "Manufacturing and security challenges in 3d printing," *Jom*, vol. 68, no. 7, pp. 1872–1881, 2016.

[224] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising," *IEEE Transactions on Image Processing*, 2017.

[225] R. D. Zimmerman, C. E. Murillo-Sánchez, and R. J. Thomas, "MATPOWER: Steady-state operations, planning, and analysis tools for power systems research and education," *Power Systems, IEEE Transactions on*, vol. 26, no. 1, pp. 12–19, 2011.

[226] 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*, vol. 5, no. 1, pp. 3–13, 2014.

[227] S. Zurier, "91% of cyberattacks start with a phishing email," *Dark Reading*, 2016.

Appendices

1 Second-order Taylor expansion

CPAC implements the physical information flow analysis through dynamic behavior inspection of the plant around Equation 5.1's (Section 4) equilibrium state using the plant's Taylor approximate equivalent Equation 5.5 which uses the first-order partial derivatives (Jacobian matrix) of the power plant's vector-valued function $f_x = \frac{\partial f}{\partial x}(x_0, u_0, \lambda_0)$, $f_u = \frac{\partial f}{\partial u}(x_0, u_0, \lambda_0)$, and $f_\lambda = \frac{\partial f}{\partial \lambda}(x_0, u_0, \lambda_0)$. Assuming that f_x is non-singular, we can reorder Equation 5.4 as

$$\begin{aligned} \Delta w = (w_u - w_x f_x^{-1} f_u)\Delta u + \\ + \Delta u^T (\frac{1}{2}(f_x^{-1} f_u)^T w_{xx} f_x^{-1} f_u - w_{ux} f_x^{-1} f_u + \frac{1}{2} w_{uu})\Delta u, \end{aligned} \tag{1}$$

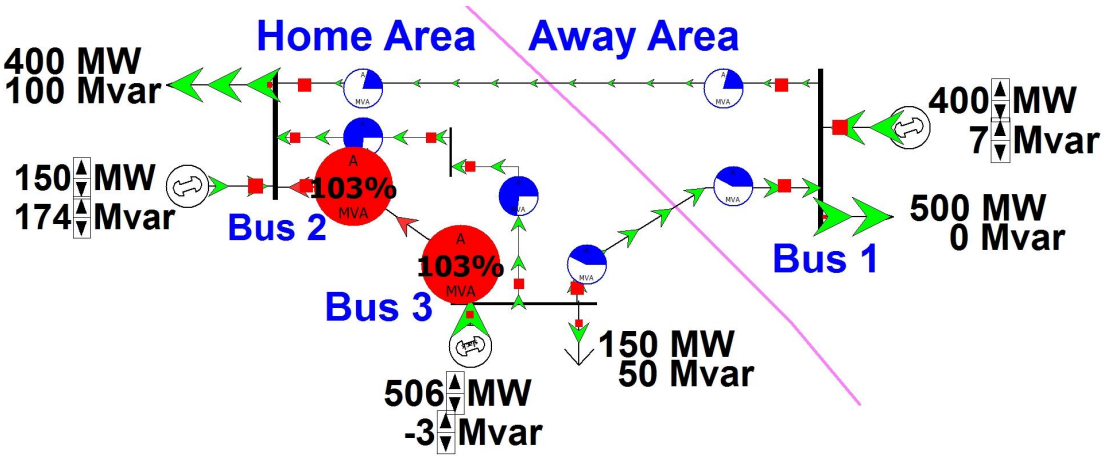
The sensor measurements are correlated with the plant state and the operator's control inputs through (5.2) at nominal operating point

The first order taylor series expansion of (5.2) is given in (5.6). For line flow analysis, w is line flow vector and u is the vector of series capacitor reactances. fx is the jacobian matrix know from Newton-Raphson and the jacobin matrices w_u, w_x and f_u are shown in Figure 4.

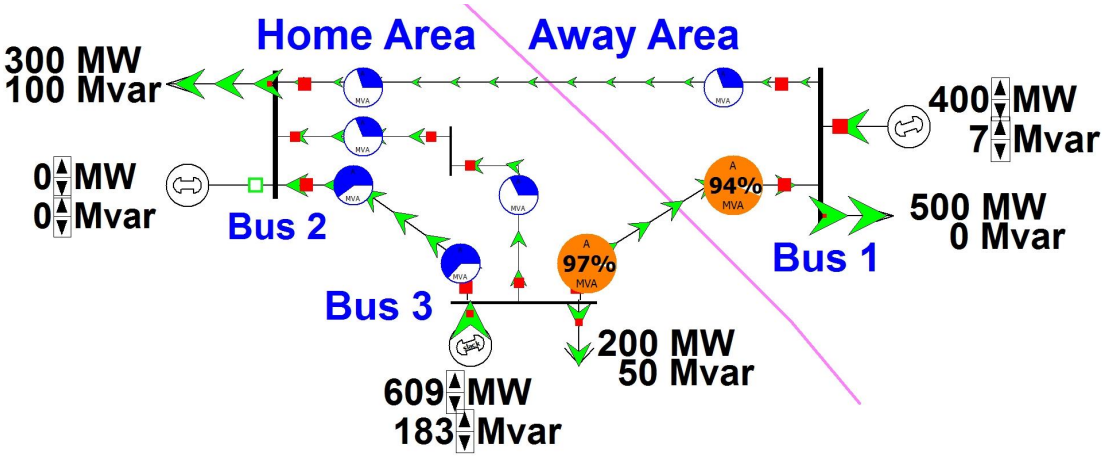
For higher policy enforcement accuracy via considering higher order dynamics of the plant, CPAC makes use of second order approximation as in (1). $w_x x, w_u x$ and $w_u u$ are second order jacobian matrices for line flow analysis are shown in Figure 4.

2 Four bus power system case study

Case B: Ensuring safe power grid control. The power system is already in an unsafe state (shown in Figure 1a), where two transmission lines experience high currents. Bob, as the control operator, asks to increase the generation set-point on power bus 2. Based on the power system flow equations, this would increase line flows across the system.
Source of incident: No enforcement of control system integrity.
Required access control policy: In the case of unsafe states, control operators must not take actions that further worsens the situation (i.e., increases the overflows).
Effects of CPAC *deployment:* CPAC denies Bob's request since his action would violate the policy (Figure 2), because the action's execution drives the system further into less safe states while the system is already not safe.
Case C: Inter-area power transfer regulation. Alice requests to open the generator on Bus 2 from the rest of the grid so that she can perform follow-up maintenance tasks on the generator. In real-world practice, the inter-area power transfers should be maintained based on the scheduled values Sun et al. (2014). *Source of incident:* No regulation of actions that can affect remote power systems. *Required access control policy:* Maintenance operators' action impact should be limited to the home area; their action must not affect the away area's operation. *Effects of* CPAC *deployment:* CPAC denies Alice's action request once it completes its physics-based analysis. Figure 1b shows the state that the system would enter following Alice's action. The line on inter-area tie-line (the line that connects home and away areas together) is indirectly affected if Alice's action occurs, and hence her action is denied.



(a) Bob's Action Request in Case B.



(b) Alice's Action Request in Case C.

Figure 1: The four-bus system presented in Figure 1a after operator modification requests.

```
curr_low(0).
curr_high(100).
event_curr_inrange_alice(U,I,N,W) :-
U==Alice,I==w,N==currReq,
curr_low(L),curr_high(H),W>=L,W=<H.
```

Figure 2: Prolog Policy Rule for Case B.

3 Global safety conditions

The class of attacks and the relevant global safety requirements for the attacks on the quadcopter are shown in Table 2

4 Normal Operation Mode Physical Modeling

The nonlinear discrete-time system obtained from the flight dynamics equations can be written by the state equations as

$$x(n+1) = f(x(n), u(n)) + w(n) \tag{2}$$

$$y(n) = h(x(n)) + v(n) \tag{3}$$

where x is the (i * 1) state vector of sensor data, y is the (j * 1) observation vector of actuator data, f is the state transition model, h is the observation model. Both f and h are nonlinear functions. w and v are the processes and observation noises which are zero-mean Gaussian noises with known covariance q and r respectively. u is the control vector. n is the current sample of the sensor data and n + 1 is the future sample of the sensor data. For the system equations in Equation 2 and Equation 3 the EKF solution for state estimation of one step ahead of time are given by Equations 3-9

$$\hat{x}(n+1|n) = f(\hat{x}(n|n)), u(n)) \tag{4}$$

Equation 4 is the predicted sensor data values for one step ahead of time.

$$\hat{x}(n|n) = \hat{x}(n|n-1) + K(n)[y(n) - h(\hat{x}(n|n-1))] \tag{5}$$

Equation 5 is the sensor values estimation update.

$$K(n) = P(n|n-1)H(n)^T[H(n)P(n|n-1)H(n)^T + R(n)]^{-1} \tag{6}$$

K(n) is (i * j) Kalman gain matrix

$$P(n+1|n) = F(n)P(n|n)F(t)^T + Q(n) \tag{7}$$

Equation 7 is the predicted covariance estimate.

$$P(n|n) = P(n|n-1) - K(n)H(n)P(n|n-1) \tag{8}$$

Equation 8 is the update of the covariance estimate

Jacobian matrix $w_u = \begin{bmatrix} w_{u_c} \end{bmatrix}$

$$\frac{\partial w_l}{\partial Z_{cl'}} = \begin{cases} \gamma_l(V_i^2 - V_iV_j\cos\theta_{ij}) - \beta_l V_iV_j sin\theta_{ij} & l = l' \\ 0 & l \neq l' \end{cases}$$

Jacobian matrix $F_u = \begin{bmatrix} P_{z_c} \\ Q_{z_c} \end{bmatrix}$

$$\frac{\partial P_i}{\partial Z_{Cl}} = \begin{cases} \gamma_l(V_i^2 - V_iV_j\cos\theta_{ij}) - \beta_l V_iV_j sin\theta_{ij} & i \neq SLK, l \in S^{L(i)} \\ 0 & \text{otherwise} \end{cases}$$

$$\frac{\partial Q_i}{\partial Z_{Cl}} = \begin{cases} \beta_l(-V_i^2 + V_iV_j\cos\theta_{ij}) - \gamma_l V_iV_j sin\theta_{ij} & i \in S^{PQ}, l \in S^{L(i)} \\ 0 & \text{otherwise} \end{cases}$$

Jacobian matrix $w_{ux} = \begin{bmatrix} w_{Z_c\theta} & w_{Z_cV} \end{bmatrix}$

$$\frac{\partial w_{Z_c}}{\partial \theta_i} = \begin{cases} \gamma V_iV_j sin\theta_{ij} - \beta_l V_iV_j\cos\theta_{ij} & i \neq SLK, i, j \in l \\ 0 & \text{otherwise} \end{cases}$$

$$\frac{\partial w_{Z_c}}{\partial \theta_j} = \begin{cases} -\gamma_l V_iV_j\sin\theta_{ij} + \beta_l V_iV_j\cos\theta_{ij} & i \neq SLK, i, j \in l \\ 0 & \text{otherwise} \end{cases}$$

$$\frac{\partial w_{Z_c}}{\partial V_i} = \begin{cases} 2\gamma_l V_i - \gamma_l V_j\cos\theta_{ij} - \beta_l V_j\sin\theta_{ij} & i \in S^{PQ}, i, j \in l \\ 0 & \text{otherwise} \end{cases}$$

$$\frac{\partial w_{Z_c}}{\partial V_j} = \begin{cases} -\gamma_l V_i\cos\theta_{ij} - \beta_l V_i\sin\theta_{ij} & i \in S^{PQ}, i, j \in l \\ 0 & \text{otherwise} \end{cases}$$

Jacobian matrix $w_{xx} = \begin{bmatrix} w_{\theta\theta} & w_{vv} \end{bmatrix}$

$$\frac{\partial^2 w_l}{\partial \theta_i^2} = \begin{cases} g_l V_iV_j\cos\theta_{ij} + b_l V_iV_j\sin\theta_{ij} & i \neq SLK, i, j \in l \\ 0 & \text{otherwise} \end{cases}$$

$$\frac{\partial^2 w_l}{\partial \theta_j^2} = \begin{cases} g_l V_iV_j\cos\theta_{ij} + b_l V_iV_j\sin\theta_{ij} & i \neq SLK, i, j \in l \\ 0 & \text{otherwise} \end{cases}$$

$$\frac{\partial^2 w_l}{\partial V_i^2} = \begin{cases} 2g_l \\ 0 \quad \text{otherwise} \end{cases}$$

$$\frac{\partial^2 w_l}{\partial V_j^2} = \begin{cases} 0 & i \in S^{PQ}, i, j \in l \\ 0 & \text{otherwise} \end{cases}$$

Jacobian matrix $w_{uu} = \begin{bmatrix} w'_{z_c} \end{bmatrix}$

$$\frac{\partial w_l}{\partial Z_{Cl'}} = \begin{cases} \gamma'_l(V_i^2 - V_iV_j\cos\theta_{ij}) - \beta'_l V_iV_j sin\theta_{ij} & l = l' \\ 0 & l \neq l' \end{cases}$$

Figure 3: Physical-Side Sensitivity-Based Information Flow Analysis

Jacobian matrix $w_x = \begin{bmatrix} w_\theta & w_v \end{bmatrix}$

$$\frac{\partial w_l}{\partial \theta_i} = \begin{cases} g_l V_i V_j \sin\theta_{ij} - b_l V_i V_j \cos\theta_{ij} & i \neq SLK, i, j \in l \\ o & \text{otherwise} \end{cases}$$

$$\frac{\partial w_l}{\partial \theta_j} = \begin{cases} -g_l V_i V_j \sin\theta_{ij} + b_l V_i V_j \cos\theta_{ij} & i \neq SLK, i, j \in l \\ o & \text{otherwise} \end{cases}$$

$$\frac{\partial w_l}{\partial V_i} = \begin{cases} 2g_l V_i - g_l V_j \cos\theta_{ij} - b_l V_j \sin\theta_{ij} & i \in S^{PQ}, i, j \in l \\ o & \text{otherwise} \end{cases}$$

$$\frac{\partial w_l}{\partial V_j} = \begin{cases} -g_l V_i \cos\theta_{ij} - b_l V_i \sin\theta_{ij} & i \in S^{PQ}, i, j \in l \\ o & \text{otherwise} \end{cases}$$

Notations:

n_l	number of transmission lines
l, l'	indices for transmission lines
z_l	series impedance of line l
y_l	series admittance of line l
Z_c	series capacitive resistances
P	active power injections at all nodes except slack node
Q	reactive power injections at PQ-nodes
w	active power line flows
$S^{L(i)}$	set of lines connected to bus i
S^{PV}	set of PV-nodes
S^{PQ}	set of PQ-nodes
SLK	slack node

Definitions:

z_l	$r_l + j(x_l - x_{cl})$
y_l	$g_l + jb_l$
Z_c	${x_{cl}}^{nl}$
θ_{ij}	$\theta_i - \theta_j$
γ_l	$\frac{\partial g_l}{\partial x_{cl}} = \frac{2r_l(x_l - x_{cl})}{(r_l^2 + (x_l - x_{cl})^2)^2}$
β_l	$\frac{\partial b_l}{\partial x_{cl}} = \frac{-(x_l - x_{cl})^2 + r_l^2}{(r_l^2 + (x_l - x_{cl})^2)^2}$
γ_l'	$\frac{-2r_l(r_l^2 + (x_l - x_{cl})^2) + 6r_l(x_l - x_{cl})^2}{(r_l^2 + (x_l - x_{cl})^2)^3}$
β_l'	$\frac{2(x_l - x_{cl})(r_l^2 + (x_l - x_{cl})^2)^2 + 4(r_l^4 - (x_l - x_{cl})^4)(x_l - x_{cl})}{(r_l^2 + (x_l - x_{cl})^2)^4}$

Figure 4: Physical-Side Sensitivity-Based Information Flow Analysis (continued)

Table 1: Evaluation attacks and descriptions

Class	**Description**	**Example attack**	**Attack consequence**	**Example safety requirements**
Motor and server control	Motors or the servo control, controls the movement of the quadcopter	Switching off the motor	Crash to the ground due to the lack of thrust	$Thrust \geq \gamma, \forall altitude > 0$
		Increasing the speed of a motor	Fly high due to increase in thrust	$\frac{Thrust}{dt} < \lvert\gamma\rvert, \forall altitude > 0$
		Increasing the adjacent motors	Move towards the direction away from the motors	$\frac{Thrust\_on\_adjacent\_Motors}{dt} < \lvert\gamma\rvert, \forall altitude > 0$
		Increasing the opposite motors	Rotates along its axis	$\frac{Thrust\_on\_opposite\_Motors}{dt} < \lvert\gamma\rvert, \forall altitude > 0$
AHRD	Attitude and heading reference systems (AHRS) provides heading and attitude information based on magnetometer, accelerometer and gyroscope	IMU sensor data modification on control algorithm	Heading towards undesired directions	$IMU\_Data\_on\_Drone - IMU\_data\_predicted < \lvert\gamma\rvert$
		Timing attack by delaying the sensor data	Delays the control response leading to undesired motion of the quadcopter	$Time\_on\_Drone = Time\_on\_Crystal$
		Modified control algorithm	Inaccurate control commands given to quadcopter	$State\_on\_Drone = State\_on\_Crystal$

Table 2: Evaluation attacks and descriptions (continued)

Class	**Description**	**Example attack**	**Attack consequence**	**Example safety requirements**
Position hold	Holds the quadcopter in a position by using GPS data	IMU sensor data modification on control algorithm	Movement of the quadcopter	$IMU\_Data\_on\_Drone - IMU\_data\_predicted < \lvert\gamma\rvert$
		GPS sensor data modification on control algorithm	Change in position of quadcopter	$GPS\_Data\_on\_Drone - GPS\_data\_on\_Crystal < \lvert\gamma\rvert$
		Modified control algorithm	Inaccurate control commands given to quadcopter	$State\_on\_Drone = State\_on\_Crystal$
Altitude hold	Holds the altitude of the quadcopter by using barometer	Barometer sensor data modification on control algorithm	Change of altitude	$Barometer\_Data\_on\_Drone - Barometer\_data\_predicted < \lvert\gamma\rvert$
		Modified control algorithm	Inaccurate control commands given to quadcopter	$State\_on\_Drone = State\_on\_Crystal$
Drift	Drifts due to external environment or sensor accuracy	Sensor data modification on control algorithm	Slight drifting motions of the quadcopter	$SensorData\_on\_Drone - Sensordata\_predicted < \lvert\gamma\rvert$
		Modified control algorithm	Inaccurate control commands given to quadcopter	$State\_on\_Drone = State\_on\_Crystal$

$$F(n) = \frac{\partial f}{\partial x}_{x=\hat{x}(n|n)} \tag{9}$$

$$H(n) = \frac{\partial h}{\partial x}_{x=\hat{x}(n|n-1)} \tag{10}$$

Equation 9 and Equation 10 are Jacobian matrices.

Crystal predicts K steps ahead of time instead of estimating one step at a time. The estimation of sensor data K steps ahead of time is given by Equation 11

$$\hat{x}(n+k|n+k-1) = f(\hat{x}(n+k-1|n+k-1)), u(n+k-1)) \tag{11}$$