

Use and Misuse of the Power Side Channel in Additive Manufacturing Security

by

Jacob Gatlin

A dissertation submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Auburn, Alabama
May 6, 2023

Keywords: Additive Manufacturing Security, Side-Channels, Cyber-Physical Security

Copyright 2023 by Jacob Gatlin

Approved by

Mark Yampolskiy, Chair, Associate Professor of Computer Science and Software Engineering
Daniel Tauritz, Associate Professor of Computer Science and Software Engineering
David Umphress, COLSA Corporation Cyber Security and Information Assurance Professor
of Computer Science and Software Engineering
Anthony Skjellum, Affiliate Professor of Computer Science and Engineering, University of
Tennessee Chattanooga
Dean Hendrix, Associate Professor of Computer Science and Software Engineering

Abstract

Additive Manufacturing (AM) is growing rapidly as an industry, particularly into the production of functional, safety-critical parts. AM Security has accordingly become a key research area for the technology, as sabotage and data theft present major risks to the safety of systems using AM-produced parts and investment in the sector as a whole. One of the chief struggles in AM Security is with the cyber-physical nature of AM systems: information generated in purely digital systems is transformed into control signals driving physical actuators, which cannot be properly controlled or observed with traditional cybersecurity means. To bridge this gap, the field of AM Security has turned to side channels. These are the unintended emanations of cyber-physical processes, which can be monitored to reason about the instrumented system's behaviors.

In this dissertation, I focus on a side channel as yet unexplored by other researchers: actuator power. By instrumenting the control-signal wires of major actuators with inductive current clamps, I can non-intrusively capture that actuator's behavior in fine detail. Building on this rich source of data, I propose, develop, and test three applications: a signature-based sabotage detection system, a reconstruction-based prototype malicious data theft attack, and a visual comparison system for side channel-based sabotage forensics.

This dissertation makes a number of novel contributions to the field. The signature-based detection method represents the first such method using this side-channel, and also serves as a test of the channel's potential information content. The reconstruction attack, in addition to presenting a novel attack vector, develops a system for side channel-based reconstruction with field-leading accuracy and operating on more complex models than previously attempted. Finally, the forensic comparison system builds on my reconstruction methods to investigate attack localization and characterization, capabilities the field currently lacks that will be vital to securing AM in the future.

Acknowledgments

To my wife Hannah, for unending patience, to my parents, for encouragement, and to Mark,
for guidance.

Even long roads reach their end.

Table of Contents

Abstract	ii
Acknowledgments	iii
List of Figures	viii
List of Tables	xi
1 Introduction	1
2 Motivation and Problem Statement	4
3 Background	7
3.1 Basics of 3D Printing	7
3.2 Elements of the 3D Printing Workflow	8
3.3 Operation of FDM Printers	10
4 State of the Art	11
4.1 Side-Channels for Sabotage Attack Detection	11
4.1.1 Field Overview	11
4.1.2 Individual Papers	14
4.2 Side-Channels for Technical Data Theft	26
4.2.1 Field Overview	26
4.2.2 Individual Papers	27
5 Instrumentation and Signal Characteristics	32

5.1	Instrumentation	34
5.2	Trace Characteristics	36
5.2.1	High Frequency Noise	36
5.2.2	Unidirectional Movement	37
5.2.3	Start, Stop, and Dwell	38
5.2.4	Reversal of Direction	39
5.2.5	Trace Synchronization	40
6	Use: Detecting Sabotage Attacks	42
6.1	Master Signature Generation	44
6.2	Tested Signature Generation & Verification	47
6.3	Experimental Evaluation of Sabotage Detection	50
6.4	Results	53
6.5	Addressing Research Question #1	56
6.6	Performance Analysis of Signature Methods	58
7	Misuse: Stealing Technical Data	61
7.1	Reconstruction Method	63
7.1.1	Loading Oscilloscope Data	63
7.1.2	Peak Detection	64
7.1.3	Reversals and Error Correction	65
7.1.4	Segmentation into Good and Bad Sections	66
7.1.5	Heuristic Correction of Bad Sections	68
7.1.6	Point Cloud Generation	70
7.2	Experimental Evaluation	72
7.3	Results	74
7.4	Fundamental Limitations	76

7.5	Addressing Research Question #2	77
7.6	Performance Evaluation of Reconstruction	78
7.6.1	Loading (readCSV)	79
7.6.2	Finding Peaks (findPeaks)	81
7.6.3	Heuristic Solver (searchSolutions)	81
7.6.4	Point Cloud Reconstruction (reconstruct)	84
8	Use: Forensic Examination of Sabotage	85
8.1	A Spatial and Temporal Distance Metric for Forensic Investigation in AM	86
8.1.1	The Hausdorff Distance	87
8.1.2	Height-Partitioned Hausdorff	89
8.1.3	Toolpath-Partitioned Hausdorff	90
8.2	A Forensic Visualization System	90
8.3	Experimental Evaluation	92
8.4	Results	95
8.4.1	Prism 70% Infill HPH	95
8.4.2	Prism 20% Infill HPH	96
8.4.3	Gear 70% HPH	97
8.4.4	Gear 20% HPH	99
8.4.5	Prism 70% TPH	100
8.4.6	Prism 20% TPH	101
8.4.7	Gear 70% TPH	102
8.4.8	Gear 20% TPH	103
8.5	Addressing Research Question #3	104
8.6	Performance Evaluation of the Forensics Metrics	107
9	Discussion	111

9.1	Advantages and Drawbacks of Power Side-Channel	111
9.2	Application-Specific Results	113
9.3	Results on Error Stability	115
10	Future Work	116
10.1	Application to other AM Processes	116
10.2	Scalability Improvements	117
10.3	Purposeful Reconstruction Metrics	118
10.4	Adversarial Actors in AM	119
11	Conclusion	121
	References	122

List of Figures

3.1	A generic 3D printer workflow [76], with major elements labeled. Specific AM processes and materials can introduce deviations from this workflow. This dissertation considers a subset of the detailed workflow.	9
5.1	A Lulzbot Taz 6 3D printer.	32
5.2	A NEMA-17 Bipolar Stepper Motor.	33
5.3	Wiring diagram and excitation chart for a bipolar stepper motor. The wiring diagram shows the two phases, connected at terminal points AC and BD. The excitation diagram shows the sequence of positive and negative impulses that must be delivered to these terminals to properly actuate the motor.	34
5.4	A Lulzbot Taz 6 printer, instrumented by Picoscope 5444D oscilloscopes. The probes are Picoscope's 60A Inductive Current probes. Each motor has two clamps attached, one for each phase. The fan controller is also instrumented by a standard voltage probe. The data captured here is transmitted to a host PC running the PicoScope application.	35
5.5	An unfiltered current trace captured on the Picoscope. Note the high-frequency noise throughout, making precisely locating a peak in the center of each wave difficult.	37
5.6	Constant-speed motor trace, after applying a low-pass filter. This plot is generated in code from the processed data.	37
5.7	Beginning of motor movement after inactivity. The DC offset of the phases start at nearly 2A, then converges over time. The blue line indicates the presence of a reversal.	39
5.8	Dwell shown in the trace. Note the much lower height of the first peak at 310.25s. A change in speed is visible from the first section of activity to the second.	39
5.9	Valid reversal shown in the trace. After the central green peak, 130.4s, the direction of travel is reversed. The blue line marks the first peak after the reversal. This reversal presents no additional problems for reconstruction.	40

5.10	Reversal with multiple issues complicating reconstruction. Peaks from 258.25s to 258.4s are low-prominence, of varying width, and represent two reversals in rapid succession.	41
6.1	Normal and Sabotaged Traces, X Motor	43
6.2	Power Consumption Signature Generation	45
6.3	Power-Trace Fingerprint Generation	46
6.4	Power Consumption Signature Verification	48
6.5	Layer comparison algorithm.	51
6.6	Signature Capture Experimental Environment	52
6.7	Comparison of two benign prints	53
6.8	Comparison of a single channel/time-interval signature in the case of a single deleted G-ode command	55
6.9	Power-Trace Fingerprint Generation	58
6.10	Layer comparison algorithm.	59
7.1	DRM-Protected Outsourcing to a Malicious AM Service Provider.	62
7.2	Trace Capture in CSV Format (Excerpt).	63
7.3	A reversal captured in the trace. Peaks are annotated A, B, C, or D to indicate the firing order. Note how the order reverses at 130.41s.	65
7.4	Identifying Firing Order.	66
7.5	Identifying Good and Bad Sections.	67
7.6	Heuristic Solver Container.	70
7.7	Heuristic Recursive Solver.	71
7.8	Original STL models used in the experiment.	73
7.9	Execution time in seconds for the readCSV module, plotted against the duration of the print capture. The plotted line shows a linear regression between the two, with error of 0.177 standard deviations.	80
7.10	Execution time in seconds for the findPeaks module, plotted against the duration of the print capture. The plotted line shows a linear regression between the two, with error of 0.09 standard deviations.	82

7.11	Execution time in seconds for the heuristicSolver module, plotted against the count of bad sections. The plotted line shows a linear regression between the two. While this result is lower than with other modules, the trend is still reasonably linear in this region. The artificial bad section with length 15 is omitted from this plot and the linear regression.	83
7.12	Execution time in seconds for the heuristicSolver module, plotted against the largest bad section in each trace. The plotted line shows a regression to 4^N between the two, with error of $1.6e-07$ standard deviations. The rightmost tick mark, with length 15, indicates the results of running this module alone on an artificial bad section.	83
7.13	Execution time in seconds for the reconstruct module, plotted against the number of peaks in the trace. The plotted line shows a linear regression between the two, with an error of $5.18e-07$ standard deviations.	84
8.1	The Height-Partitioned Hausdorff Distance.	89
8.2	The Toolpath-Partitioned Hausdorff Distance.	91
8.3	Test set of models used for the forensics visualization experiments.	94
8.4	The Prism model HPH comparison, printed at 70% infill.	95
8.5	The Prism model HPH comparison, printed at 20% infill.	97
8.6	The Gear model HPH comparison, printed at 70% infill.	98
8.7	The Gear model HPH comparison, printed at 20% infill.	99
8.8	The Prism model TPH comparison, printed at 70% infill.	100
8.9	The Prism model TPH comparison, printed at 20% infill.	101
8.10	The Gear model TPH comparison, printed at 70% infill.	102
8.11	The Gear model TPH comparison, printed at 20% infill.	103
8.12	A naive approach to calculating the Hausdorff distance.	107
8.13	The Height-Partitioned Hausdorff Distance.	108
8.14	The Toolpath-Partitioned Hausdorff Distance.	110

List of Tables

4.1	Mono-side-channel signature approaches for Sabotage attack detection.	11
4.2	Multi-side-channel fusion signature approaches for Sabotage attack detection. .	12
5.1	Application-specific implementations of the general instrumentation setup. . . .	36
6.1	Detectability of <u>atomic modifications</u>	54
6.2	Detectability of known <u>sabotage attacks</u>	55
7.1	Solutions considered for badly ordered pairs and their effects on firing order. The beginning <i>ABCD</i> exemplifies a possible firing order prior to the badly ordered pair (indicated in red and underlined).	69
7.2	Point cloud renderings and metrics of the reconstructed models. Any support structure is included in the rendering.	75

Chapter 1

Introduction

Additive Manufacturing (AM) is far from secure. Major stakeholders in AM, such as the U.S. Department of Defense [61] and manufacturing consortiums [38], have repeatedly called for the development of new security measures to address potential threats. These threats are still only hypothetical; there has been no reported real-world attacks against AM. While that may reassure some, one could instead call all of AM Security a plan that has not yet come in contact with the enemy.

AM is a family of manufacturing technologies that create solid objects incrementally by depositing and fusing layers of source material, which is referred as feedstock. The ASTM International, an international engineering standards organization, maintains a Committee F42 on Additive Manufacturing Technologies; this committee develops and publishes a range of AM-specific standards. They distinguish between seven different AM processes: vat photopolymerization, powder bed fusion, binder jetting, material jetting, sheet lamination, material extrusion, and directed energy deposition [7]. Within each process are a number of specific technologies; for example, the widely-deployed Fused Deposition Modeling (FDM) is an example of a Material Extrusion process. Since 2021, the ASTM has also established Working Group WK78322 on Additive Manufacturing Security, of which I am a member.

While AM technology can be traced back to work done in the 1980s and the early patents of the field (see Wohler's History of AM, from [64]), the technology began to proliferate after the expiration of these patents in the early 2010s. It expanded from initial applications in rapid prototyping into use for functional, load-bearing, and safety critical components [64]. The first works on AM Security quickly followed. Xiao Zi Hang [72], in a public keynote

at XCon2013, presented the earliest AM Sabotage attack-modifying the printing results of a RepRap 3D printer through modifications to printer software and firmware, configuration, and object description. Subsequent years saw the rapid publication of many academic works in AM Security, both on compromising AM systems and on exploiting compromised systems to conduct AM-specific attacks.

At the time of writing, those AM-specific attacks can fall into four major threat categories: Theft of Technical Data, Sabotage, Illegal/Unauthorized Part Manufacturing, and Data In-/Exfiltration. These threat categories can be considered from the attacker and defender perspectives.

A key assertion of AM Security as a field is that pure cyber-security, while necessary, is not sufficient to secure AM [3, 12, 73, 75]. Researchers have turned to side-channel analysis, in this context the sensing and interpretation of the AM process' physical emanations, to both develop new attacks and better secure AM systems. In this dissertation, I systematically investigate the use of the actuator power side-channel for legitimate and illegitimate purposes: sabotage attack detection and technical data theft, respectively. Multiple sections of this dissertation are based on my prior publications [22, 23] in peer-reviewed journals and conference proceedings, while others represent as-yet unpublished work; this will be indicated where appropriate.

The remainder of this dissertation is structured as follows. Chapters 2, 3, and 4 cover the necessary background for this work, detailing the motivation, fundamental knowledge on the field of 3D printing, and state of the art in AM Security, respectively. Chapter 5 describes the instrumentation system used in my experiments, including both the data-collection equipment and the system under test, and also outlines the characteristics of the actuator power side-channel.

I then move into the main contributions of this work: uses and misuses of the side-channel. First, Chapter 6 develops a signature comparison method using measurements of stepper motor actuation current, establishing the information content of the side channel. Then, Chapter 7 digs further into the channel, identifying individual features and their correlation to printer behavior, using this knowledge to present a side-channel data theft attack for reconstructing a printed object. Building on this reconstruction, Chapter 8 develops and tests a similarity

measure between object reconstructions for use in forensic investigation of sabotage. Each of these chapters is structured to provide the fundamental research question being investigated, outline the relevant theory, experiments, and results, then discuss the results' bearing on the research question. In addition, as each chapter presents a body of code and in several cases novel algorithms, a performance analysis is conducted in each.

The remaining chapters consider the overall implications of my work and bring the dissertation to a close. Chapter 9 discusses larger-scale implications, either not explicitly tested by experiment or requiring consideration of elements from multiple of Chapters 6, 7, and 8. Future extensions to or developments from this work are considered in Chapter 10. Finally, Chapter 11 provides a brief summary of contributions and conclusion.

Chapter 2

Motivation and Problem Statement

AM Security has, to date, seen several cyber-physical attacks and defense measures. Side-channels are a key element of many such works. A side-channel is an intentional or unintentional emanation of a cyber-physical system, which can be measured and correlated to the behavior of the system. AM systems are complex electro-mechanical systems with multiple side-channels. For example, stepper motors in some AM systems generate acoustic emanations that can be correlated with the speed and direction of the print-head's movement [3]. Numerous others have also been investigated in the research literature, through thermal, kinetic, and visual sensing.

In this dissertation, I investigate the power side-channel in a way not yet done by other research teams: by direct current sensing of individual actuators. In particular, I use inductive current clamps, monitored via oscilloscopes, attached to the current delivery wires of each stepper motor in an AM system. I discuss the instrumentation and data capture in more detail in Chapter 5.

The investigation of side-channels for security is motivated by several unique properties of side-channel analysis, which cannot easily be achieved by other strategies. First, side-channel analysis can be performed non-invasively. In the hard real-time environment of an AM system, non-invasive security is not simply an option but an essential requirement. It is further configurable for a wide range of electro-mechanical systems, operating without any knowledge of the secured machine's software and firmware. These two properties allow me to retrofit this approach onto existing machines. Many AM machines produced to date, which represent massive

capital investments, were designed without built-in security features. Finally, a side-channel-based approach can be deployed on an air-gapped system, which can significantly increase the difficulty of simultaneous compromise on both monitored and monitoring systems.

In addition to the advantages of side-channels in general, the actuator power side-channel presents particular features for AM systems. Individual actuators can be instrumented separately, isolating their signals from one another for easy identification. The power side-channel has a high signal-to-noise ratio, which provides a clear correlation between the measurements and system behavior. The electrical system of an AM machine is a clean sampling environment, as electrical noise must be controlled for the machine to function at all.

Research Question 1: Does the power side channel provide enough information to detect sabotage attacks in AM? Answering this question requires answering several smaller questions. How are different categories of sabotage attack reflected in side-channel data? What analytical method is sufficient to detect this information in the side-channel? What are the thresholds of detection for the selected analytical approach? What factors limit the approach?

Research Question 2: Can the power side-channel be used to reconstruct a printed object as a method to bypass Digital Rights Management (DRM)? Several smaller questions follow naturally from this. How can a reconstructed object be represented? Given this representation, how can the accuracy of a reconstruction be measured from it? What accuracy of reconstruction is achievable from the power side-channel alone? What are the factors limiting the accuracy of the reconstruction?

Research Question 3: Can side channel information be used to localize and investigate a defect introduced via sabotage attack? Given the side-channel data of an original print and a suspect print, how can discrepancies between the two be identified? What metrics can be used to indicate the severity of a discrepancy? How can identified discrepancies be represented to assist in investigation?

A positive answer to research question 1 is a prerequisite to answering questions 2 and 3. If there is not enough information to detect sabotage attacks in the power side-channel, then there is likely not enough to perform a full reconstruction and forensic analysis is certainly

impossible. Having a positive answer to question 1, however, does not guarantee the same for questions 2 and 3. Each present further challenges that could render them intractable.

In this work, I exclusively focus on the Material Extrusion process, specifically the sub-category of Fused Deposition Modeling (FDM) technology. FDM printing systems are the most widespread form of 3D printer, being firmly established for hobbyist and production printing in polymers [65] and increasingly used in composite material. The operation of FDM printers is explained in detail in Chapter 3. Most importantly, the majority of attacks published in the AM Security literature to date have targeted FDM printers, paving the path for real attacks on these systems and providing diverse test cases for defensive techniques.

Chapter 3

Background

3.1 Basics of 3D Printing

3D printing, more formally known as Additive Manufacturing (AM), is defined by the ASTM International¹ as “a process of joining materials to make objects from 3D model data, usually layer upon layer, as opposed to subtractive manufacturing methodologies” [7]. The ASTM recognizes seven families of AM processes, each with distinct compatible source materials and methods for fusing material into a solid object. These are: binder jetting, directed energy deposition, material extrusion, material jetting, powder bed fusion, sheet lamination, vat photopolymerization. Each of these AM processes incorporates a number of particular implementations, referred to as technologies. For example, the Fused Deposition Modeling (FDM) technology considered in this dissertation is a representative of the Material Extrusion process family.

The 3D model data referred to by the ASTM definition is the basis for any AM process. It defines the geometry of the object to be manufactured; its shape and size. However, 3D models are not directly conducive for printing. Instead, they must first be sliced into 2D layers that are used by a 3D printer to plan the deposition and fusing of material. In some AM processes, the entire layer is deposited and selectively fused. In other processes, such as Material Extrusion, only material that will be fused is deposited. Depending on the process, the 'toolpath' that describes the actions taken to deposit and fuse a layer may have a different format and meaning.

¹ASTM International is a standardization organization, formerly known as the American Society for Testing and Materials, whose committee F42.08 is responsible for setting international standards in AM.

In any case, the toolpath describes executable commands that control the printer's actuators in an orchestrated manner.

It is common in AM to discuss the AM Digital Thread, which describes the transformation of digital data throughout the AM process. This is considered to begin with a 3D model, generated by a Computer-Aided Drafting (CAD) program. At this stage, the CAD program might be used to simulate the material properties of the object and ensure that the design meets specifications for functional properties; Solidworks, for example, provides basic Finite Element Analysis (FEA) tools to simulate the mechanical properties of printable designs. Once the design is finalized, that model is exported in an AM-compatible format such as Stereolithography (STL), Additive Manufacturing Format (AMF), or 3D Manufacturing Format (3MF). STL files contain only bare geometry, as a set of triangular facets; AMF and 3MF both extend the format to contain material information, color data, and other metadata. The legacy STL format remains the most widely used.

These AM-specific model formats must be translated into executable printing instructions: the toolpath. G-Code is a common and open-source implementation for FDM machine instructions, though other processes use different toolpath commands. Furthermore, manufacturers of industrial-grade printers often use their own proprietary formats.

Each of these necessary elements for 3D printing are typically combined into a full workflow, by which organizations bring a design from inception to physical realization.

3.2 Elements of the 3D Printing Workflow

The primary working components of a generic 3D printing workflow are a 3D printer and host PC workstation, communicating across a wired or wireless channel. These communications are comprised of, at minimum, the necessary toolpath file to carry out a print. If no feedback is required and the printer has an interface for scheduling print jobs, this file can be moved via a USB flash drive or flash memory card. However, it is more common for the host PC and printer to communicate during the print, allowing the host to submit, start, or cancel jobs, as well as to monitor status and display progress to users. In simpler printer systems this is often done over a serial USB interface. More advanced printing systems often communicate over wired

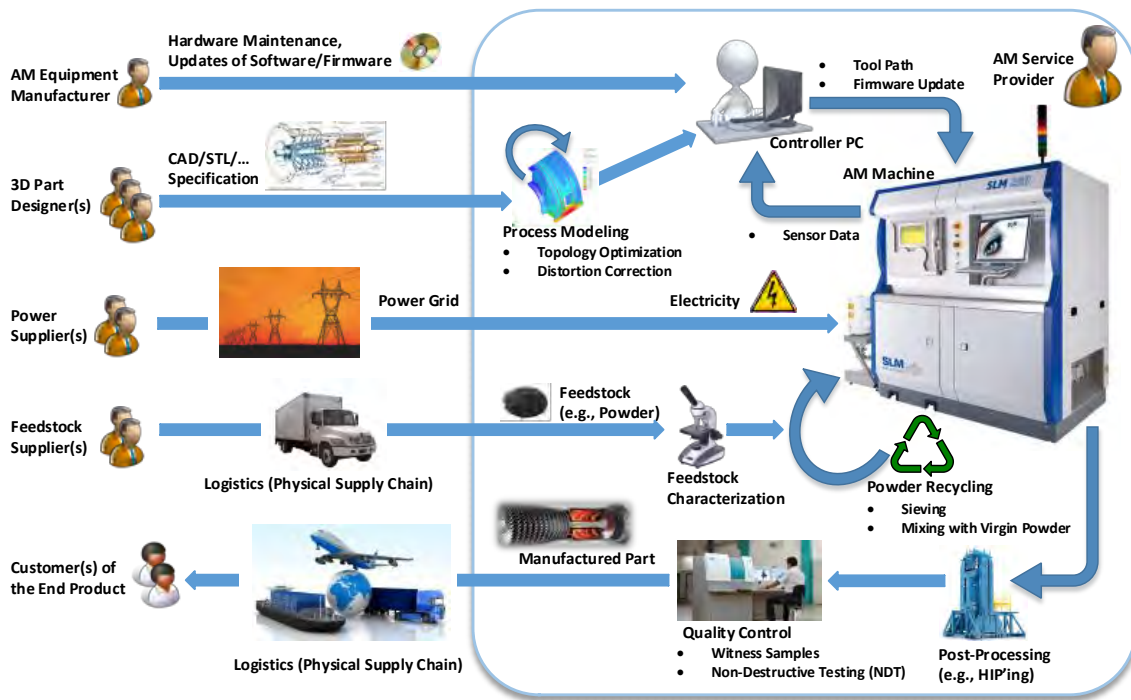


Figure 3.1: A generic 3D printer workflow [76], with major elements labeled. Specific AM processes and materials can introduce deviations from this workflow. This dissertation considers a subset of the detailed workflow.

(typically Ethernet) or wireless (typically WiFi) computer networks. Furthermore, it is likely that at least the host PC has direct access to the Internet. Increasingly, the printing system is also Internet connected. This is often used to support cloud interfaces.

Cloud-enabled printers can route several operations through a cloud service provider. This typically includes slicing, print job management, print history, and firmware management, to name a few. In some cases, these same functions are not supported from local workstations in cloud-enabled printers. Cloud service designs are growing more popular as a means to restrict access to proprietary software, such as slicers, and to enable digital rights management for both designs and printer technical data.

Within the printer itself, individual toolpath commands are interpreted to carry out the corresponding actions for the printer's actuators. In an FDM printer, the primary actuators considered in this dissertation are the stepper motors, which control positioning of the print head and extrusion of filament. There are several further actuators involved which do not

directly effect object geometry: these are the thermal elements (printing hot-end and print bed) and fans (for filament cooling).

3.3 Operation of FDM Printers

I focus on 3D printers employing the FDM process, which is widely used in both desktop and high-end polymer 3D printers. The process is based on the extrusion of heated polymer filament through a nozzle, built into a print head which can be moved along the X-, Y-, and Z-axes, and extrude or retract filament along the E-axis.

Both printhead movement and filament extrusion are driven by motors, controlled by an on-board motor controller, itself guided by the commands in the toolpath. For each layer, it moves the X and Y motors through a path which describes a space-filling curve for that cross-section. In places where material is needed for the layer, the extruder motor is actuated to push heated filament through the nozzle. The deposited filament fuses with the previous layer as it cools. Whenever a full layer is finished, the Z-motor is actuated to raise the print-head by one layer thickness, and the X and Y motors are actuated to move the print-head to the first extruding position of the next layer.

This process is repeated layer by layer until all commands in the toolpath are executed, and the physical 3D object is complete.

In the planning stage, the slicer for an FDM printer has to account for unavoidable physical properties of both the FDM technology in general and the specific printer. For example, in polymer FDM printers, printed parts will both shrink and deform slightly as they cool. This occurs due to the polymer contracting and being pulled downwards by gravity. These physical processes need to be taken into account by the slicer, so that after they occur the printed object is as close as possible to the intended geometry specified by the 3D model. The task of performing these adjustments is only partially automated, and is often instructed by parameters specified by the operator. As a result, achieving quality results in FDM printing is partly dependent on operator skill and knowledge.

Chapter 4

State of the Art

In this chapter, I cover in depth the publications directly addressing side-channels in AM Security, including both offensive and defensive applications in sabotage and technical data theft. Each section begins with a topic-oriented summary to contextualize works in the field, then reviews the publications individually. In 2018, I contributed to a comprehensive survey of AM Security [76]; an updated version of this survey and the taxonomy developed alongside it form the backbone of this review.

4.1 Side-Channels for Sabotage Attack Detection

4.1.1 Field Overview

Table 4.1: Mono-side-channel signature approaches for Sabotage attack detection.

Year	Publication	Side-Channel
2019	Belikovetsky et al. [9]	Acoustic
2019	Prakash et al. [42]	Video
2019	Gatlin et al. [23]	Power
2020	Song et al. [51, 52]	Video
2021	Shi et al. [48]	Kinetic
2021	Zhou et al. [81]	Kinetic
2021	Liang et al. [34, 35]	Multiple*
2021	Dawson et al. [17]	Power

Sabotage threats against AM can target specific manufactured parts, the AM equipment itself, and the surrounding environment of either part or equipment [73, 76]. To date, sabotage publications with practical demonstrations fall entirely in the first category-attacks against

Table 4.2: Multi-side-channel fusion signature approaches for Sabotage attack detection.

Year	Publication	K	A	V	T	P	M
2020	Song et al. [53]	✓	✓	✓			
2020	Wu et al. [68]	✓	✓	✓	✓	✓	
2020	Yu et al. [79]	✓	✓			✓	✓
2021	Rais et al. [43]	✓			✓		
2021	Yang et al. [78]		✓	✓			

Legend: Side-Channels

- K - Kinetic
- A - Acoustic
- V - Visual
- T - Temperature
- P - Power
- M - Magnetic

manufactured parts. These threats are mainly physical, effecting the three 'F's: Form, Fit, and Function [25]. Form in AM corresponds to the geometric shape of the part; Fit to the integrability of a part in a larger system; Function to the part's specified physical properties, such as strength, thermal conductivity, elasticity, etc.

Sabotage attacks in the literature target many specific aspects of the AM process and are conducted by many methods. Void attacks were the first to develop [54, 80], where an internal cavity is inserted into a part's geometry. Most commonly the void's presence reduces some critical element of Function, such as tensile strength [54] or fatigue life [10]. This type of attack is easily conducted with software manipulations ranging from direct STL modification [10] to compromised printer firmware [37], and is often presented as a unique vulnerability of AM [27]. As such it features as the test-case attack for many sabotage detection publications [8, 51–53, 59, 71].

Beyond void attacks, many novel or process-specific attacks have been proposed. Some of these, such as the manipulation of carbon-fiber piles [44], apply only within a narrow context, but layer-thickness attacks [21] and orientation attacks [80] are both more broadly applicable. Material-specific attacks, such as introducing porosity [49] or manipulating powder spreading [26] for metal AM processes, have received early practical treatments.

From the inception of the field, AM Security has approached sabotage attack detection as a problem not solvable with strictly cyber-security. Whether this is justified as appropriate

defense-in-depth or a strict necessity for a fully cyber-physical process [73], it demands the application of physical and cross-domain techniques. Some of these techniques are performed entirely after manufacturing, and might be considered security-focused Non-Destructive Testing (NDT) [5, 59, 63]. The majority, however, involve sensing and recording the physical emanations of the manufacturing process as it occurs. In these cases, I refer to a particular kind of emanation as a side-channel, and the entire system as side-channel-based sabotage detection.

Side-channel-based sabotage detection work is first distinguished based on selection of side-channel: the earliest attempts using the acoustic channel [3, 9, 12, 16] were soon joined by kinetic [21], vision [6, 21, 51, 52, 71], temperature [70], and power [21, 23, 36]. Individual authors have approached the same channel with very different instrumentation; within the power side-channel, authors have applied power-meters at the wall outlet [70, 79], and others (myself included) have sampled actuators directly [17, 23, 70]. More recently, multi-modal approaches which collect and correlate many side-channels [8, 16, 68, 79] have grown more prominent.

The other key criteria of side-channel sabotage detection is the system used to interpret the side-channel data and detect instances of sabotage. Most operate on two sets of sensed data: the known-good and the suspect. Known-good data is acquired from a single print run, under controlled conditions, the output of which may be verified by test after production. Suspect data, of which there may be many individual instances, is captured under normal production conditions and is compared against the known-good for deviations, which may indicate sabotage. As an alternative, known-good data can be generated directly from a digital representation of the object, such as the G-code [43, 79].

In both forms of known-good vs. suspect detection systems, the comparison is automated and quantitative. Since these datasets are from analog sources with significant quantities of noise and imprecision (consider acoustics in a noisy factory [50], or kinetic sensors with unrelated vibrations [21]), most authors use statistical tools and thresholds to make more reliable comparisons.

Each side-channel correlates to printer behaviors in ways specific to the physics of the channel and the real, engineered implementation of the AM system; the relevant signal features therefore vary greatly between channels and are not yet widely agreed upon by authors.

Channels with heavily periodic signals, such as the acoustic [9, 12] and actuator power [23, 70] are often reduced to time-sliced frequency data, using tools like Principal Component Analysis. Visual channels can be approached with tools from machine vision: outline isolation using convolutional filtering [52], or using a Canny filter [21]. In every case, these metrics are compared either through the authors' experimentally or theoretically derived thresholds, or by training a machine learning algorithm. For both of these, each team has developed their own unique and difficult-to-compare approach.

There is an element of these detection systems that is not yet fully characterized: what can they detect? Many publications, especially those that prepare multiple test attacks, use some variety of void attack against a simple geometric object [10, 54, 55]. These span a range of sizes, shapes, and delivery methods; there is no widely-used or replicable baseline for attack severity. In a handful of publications, layer thickness attacks [21] or orientation attacks [80] are tested, many of which have not yet been repeated. As such, comparing any two detection systems or establishing accurate thresholds for detection is nearly impossible.

4.1.2 Individual Papers

Alongside the early attack papers, lies the publication of a very early defensive technique from Albakri et al. [5]. The authors adapt the well-established practice of impedance-based Structural Health Monitoring (SHM) to detecting deviations in additively manufactured parts. Impedance-based SHM uses physically-attached piezoelectric sensors to characterize the mass, stiffness, and damping of an object in response to a wide frequency sweep of vibration. The authors reason that the introduction of defects such as voids, deformations, and porosity will alter these characteristics at a detectable level.

The general approach Albakri et al. present sets as strong a precedent in sabotage defense as Sturm et al. [54] set with attacks. It includes a known-good set of baseline physical measurements, taken using a non-destructive sensing technique. This baseline is compared against a suspect part, whose physical deviations are supposed to be detectable in measurement deviations above some arbitrary threshold. These very features are present in the majority of sabotage-defense publications to date.

In this initial foray, Albakri et al. are able to detect their deviations in dimensioning and positioning, but are unable to detect deviations in internal porosity. They argue this is because the porosity does not greatly effect the part's mass, and that better tuning of the frequency ranges might provide better results. This element of the paper (attack types and intensities that fall outside the sensitivity range of the detection technique) will also recur in future defensive works.

Fusing the concerns of Albakri et al. [5] and Turner et al. [60], Vincent et al. [63] specifically propose SHM as an additional layer of protection over inadequate (for security) manufacturing quality control systems. The key element of interest from this paper is their argument against quality control as security: quality control focuses on the measurement of key quality characteristics, which a sophisticated attacker will be aware of and avoid altering. Though this logic is straightforward within other security fields, this is its first explicit acknowledgement within AM Security. The idea of a nuanced, Advanced Persistent Threat (APT)-style of attacker has since been the premise of multiple attack papers.

2016 sees three defensive publications, each pursuing a different layer of defense. The most important for the present work is Chhetri et al. [12], who present the first side-channel defense using the acoustic channel. This technique of side-channel analysis had been used earlier in 2016 by Al Faruque et al. [3], to reconstruct and steal the design of a printed object. Chhetri et al. apply it here in concert with the same known-good vs. suspect approach as Albakri et al. [5] to detect deviations introduced by sabotage. Also noteworthy is the application of machine learning (a Support Vector Machine) to side-channel analysis—many subsequent side-channel publications have done the same.

The nature of the acoustic channel as used by the authors must be noted here. The stepper motors used in desktop 3D printers are notoriously loud, and whine at a frequency proportional to their speed. Most such printers will deploy at least 4, and more commonly 5 motors (one each for the X and Y axis control, one for filament extrusion, and one or two for the heavier Z axis). Even in isolation this produces a noisy environment; these authors and subsequent works struggle to isolate and recognize individual motors. A focus on frequency analysis and comparing mixed-source signals against equivalent mixed-source signals allows the technique to

function, as even in this initial foray the authors achieve a claimed 77.45% detection accuracy. In time, however, the acoustic channel will be just one of many valid side-channels.

Sturm et al. [55] further pursue the impedance modeling of Albakri et al. [5], extending it to operate during the manufacturing process. In a way, this redeploys SHM not as a form of non-destructive testing, but as the means to access a side-channel (local impedance) during the build process. As such, the authors adopt the known-good vs. suspect approach and go to some lengths to adapt a post-production technology to work in-situ. Readings are taken to produce impedance signatures at defined inspection layers; the same readings must be taken at the same layers during the suspect printing for comparison. Their method is successful at detecting voids at 0.8% and 6.7% of the part's volume, but the authors note that sensitivity decreases as the part's mass goes up and as the sensor is placed further from the defect. These complications, along with the other variations in impedance with respect to stiffness, material, support structure, etc., might be taken to show that some side-channels, while containing the necessary information for defensive techniques, present too complex a materials science challenge for reliability.

On the defensive front, two authors simultaneously propose and develop systems for cross-domain sensor fusion schemes: Chhetri et al. [16] and Bayens et al. [8]. Chhetri et al. first discuss the relationship between the cyber- and physical-domain components of various systems, and how they can be collected simultaneously and correlated for higher accuracy in detection. They theorize that such fusion schemes can be applied to detect information leakage, abnormal behavior (such as sabotage), for system health monitoring, and attack analysis; practical papers exploring several of these applications were published in subsequent years.

Bayens et al. [8] arrive with a practical implementation of their own system, which correlates acoustic, spectroscopic, and gyroscopic channels. The acoustic channel is composed of motor noise, as in previous applications [12], but the other two channels are novel at this stage. The spectroscopic channel is material verification using Raman Spectroscopy, and the gyroscopic is the use of direct-attached gyroscopes on the printhead as a means to track printing motions. While this suite of sensors provides coverage to detect many published manipulations, each channel has challenging weaknesses. The acoustic channel struggles with noise and

motor overlap in the same manner as earlier approaches. The Raman system deployed here has a low penetration depth of $300\mu m$, limiting its application to exposed surfaces. The gyroscopes, as the authors note, have a high enough degree of error that they must be paired with another position-tracking system such as cameras. Overall, the framework of sensor fusion is present, but the channels are individually weak and not yet correlated strongly. Subsequent publications deployed statistical tools and even broader sensor suites to improve detection rates and minimum deviation size for detection.

Belikovetsky et al. [9] present their own approach to the acoustic side channel, using Principal Component Analysis (PCA) to generate and compare audio signatures. Such an approach reduces the noisy, combined audio signal of five working motors to its essential frequencies, sacrificing distinguishability for reliable and reproducible signatures. The authors further define *atomic*, or minimal, modifications at the G-code level: command insertion, deletion, reordering, and parameter modification. Defining modifications and their sizes within the sabotage space is inconsistent, both in 2017 and today; atomic modifications are a reasonable definition within the context of G-code manipulations, but would map poorly onto the false sensor readings attacks of Slaughter et al. [49] and have complex relationships with the early void-insertion attacks [54]. This shows that definitions and measurements often apply only to narrow slices of AM Security as a whole. The cross-domain nature of most published work ensures that many concepts travel poorly.

Tsoutsous et al. [59] present a highly specialized simulationist approach. Beginning with the compiled G-code for manufacturing the object, they approximate the final model and perform Finite Element Analysis (FEA) to predict its response to various stresses. This defense should, in theory, identify any sabotage significant enough to compromise the functional properties of the part. While the system is both simple and reliant on an industry-standard FEA process, there are several downsides to note. From a practical perspective, FEA is both expensive and time-consuming, but these are excusable faults for a security system in certain cases. More importantly, the system categorically cannot detect attacks that take place after the G-code is produced, such as those conducted by Moore et al. [37], Slaughter et al. [49], or Pope et al. [41].

Wu et al. [71] submit a machine-vision approach to detecting sabotage attacks during the printing process. Grayscale images of prints-in-progress have mean, standard deviation, and intensity outliers extracted; these images are used to train and test a collection of k-Nearest Neighbor, random forest, and anomaly detection classifiers. The attack set includes voids of four shapes and poorly-fused seams. Machine-learning approaches like this generally function on a more general detection of sabotage, rather than comparison to a known-good print. In this case, however, the classifiers are trained on a set of images specifically coming from unmodified versions of the same print, in many cases pictures of the unmodified layers of the same print run. The authors here report an impressively high 96.1% detection accuracy for statically mounted cameras and the anomaly detection classifier, but given the scenario and training set this is to be expected. The applicability and usefulness of this result is questionable, and such questions persist in reported detection accuracies from essentially every author in the field.

Gao et al. [21] produce side-channel techniques for detecting an array of cyber-physical attacks against FDM printers: infill modifications, print speed, layer thickness, and cooling rate are all targeted. The sensor suite is equally broad, including accelerometers, magnetometers, microphones, and cameras. Each attack is detected using a tailored approach, drawing on the most suitable sensors and discarding the others; this contrasts with other multi-sensor fusion systems which build composite signatures using every sensor, even those with low correlation [8, 16]. Their results, particularly on the printhead path-reconstruction, represent remarkable progress on prior works and are backed up by both visualization and appropriate metrics (in this case, a Hausdorff distance between the known and reconstructed paths).

Wu et al. [69], working within the broader scope of Cyber-Manufacturing Systems (CMS), performs initial tests against their newly developed CMS testbed. The key difficulty they identify is a common one in the field: complete manufacturing workflows are complex. Field studies in working manufacturing sites are rare, as even slowing down work to conduct security tests may be prohibitively expensive. As such, developing a testbed of one's own is a prerequisite for their proposed work: a cross-domain cyber and physical Intrusion Detection System (IDS). The testbed and its physical instrumentation draw from pre-existing materials: CNC mills and 3D printers instrumented with cameras, microphones, and even an at-the-wall

power meter. Their results, at this stage, are piecemeal-detection rates for a handful of attacks. Subsequent work from this team will steadily improve and expand upon both the testbed and the IDS itself. Wu et al. [70], in what appears to be a revamped version of their earlier testbed publication [69], present a cyber-manufacturing security testbed and a collection of attack studies against it. The equipment is more diverse than in the initial publication, incorporating a conveyor belt and robot arm to automate the movement of parts between the 3D printer, CNC cutter, simulated welder, and various other manufacturing subsystems. Noteworthy for this dissertation, they have expanded their range of side-channel sensors to include several current sensors on actuators in the testbed, though none are attached to the 3D printer. Statistical properties such as mean, standard deviation, and extrema are measured within time windows and compared to known-good recordings to detect sabotage using the current channel. The attacks have also been improved, including SQL injection against saved G-code and a bash exploit targeting printer settings alongside the original void-insertion attack. Overall, the paper provides a snapshot of improvements to the Syracuse testbed as of 2020, but the bulk of its novelty remains with the other publications using the platform.

Liang et al. [34] present a short poster on side-channel-based intrusion detection in AM. The major works to that point, including Chhetri et al. [12], Belikovetsky et al. [9], myself [23], and Bayens et al. [8], are all subjected to a critical investigation. Multiple weaknesses are shown to recur in most or all prior approaches: time segmentation is weak or lacking, minor synchronization failures are either unrecoverable failures or, in the case of Bayens, handled by a windowing mechanism that obscures short-duration deviations. Liang et al. propose a framework for intrusion detection that claims to correct these issues; unfortunately, only very preliminary results on an acoustic implementation are available in the poster.

Liang et al. follow through on their promises with their 2021 publication [35] on side-channel based intrusion detection framework, which they name NSYNC. Their system aims to deal with the asynchronous nature of signals in AM systems, which they consider a major source of error for existing Intrusion Detection Systems. At the core of NSYNC is the Dynamic Window Matching (DWM) algorithm, which the authors developed to handle window-based dynamic synchronization of signals. Using this and other time-warping methods, the

NSYNC framework IDS compares known-good reference signals to the potentially warped test signal, issuing alerts when detecting a significant correlation distance between the two. The authors evaluate their proposal against a set of five sabotage attacks and using six side channels, including at-the-wall current draw, and make comparisons to each of the IDS systems they noted in their earlier proposal [34]. They find that their own system using DWM performs best, followed by my own [23], but that generally systems using time as the primary criteria for detection outperform those using signal amplitude.

Wu et al. [66, 68], in a conference paper and later expanded journal article, build out the cyber-physical alert correlation for their IDS described in [69]. Also relating to their IDS project, Wu et al. survey other IDS for cyber-manufacturing systems [67]. This involves the translation of concepts between the two domains, primarily involving timing. Cyber-compromises typically take effect immediately, whereas physical compromises might not have visible effects for hours, weeks, or months. Using other correlation methods, mainly of network IP addresses with physical equipment and user IDs associated with that equipment, the authors are able to present alerts integrating cyber and physical alerts. They test their new system against four case studies within their testbed, one of which uses a 3D printer with the void-insertion attacks implemented for their earlier image-detection work [71]. The new correlation system generates impressive results, reducing alert counts from 371 down to 3 and improving detection accuracy from 49.6% (rate of true alarms to all alarms) to 100%. However, I have the same reservations noted in my summary of [71], namely that the training and test of their machine vision detection system produces unrepresentative detection rates for general attacks against varied models.

Prakash et al. [6] identify the same issue with prior machine vision detection systems, and claim to train an image classifier on a more diverse array of attacks and printed objects. While the phase response-and-amplitude envelope features they base this classifier on may well be applicable with broader image sets, their testing does nothing to support it: the case study uses a single tensile test token (the same dogbone shape tested by Sturm et al. [54]) subjected to three types of void attacks. The images in the test set are a time-ordered sequence of video stills, captured throughout the printing process. While the statistical results of this study fall under

the same limitations as Wu et al. [71], the working theory of their data source is noteworthy. The authors consider the motion of the printhead as being the main source of usable data in the images, tuning their feature extraction accordingly.

Song et al. [53] conducts insider-threat case studies against their established CMS testbed, extended to include a simulated supply chain for the provision of raw materials and transport. Their detection system, which they term a “physical audit”, feeds the existing side-channel sensor data from the accelerometer, acoustic, and visual channels into a threshold-based statistical classifier. Each case study includes a somewhat-rough sabotage attack and a more subtle insider attack. For the AM case study, the authors deploy two standard infill manipulation attacks: one reducing the infill percentage (the insider), the other inserting a sizable void (the sabotage). Their system is able to pick up the void attack but misses the infill percentage change entirely. Going from 13% infill density to 12% infill density is, understandably, under the mean error threshold used by their classifier, but designing such an attack provides an interesting example of variety in attacker motivation: malicious insiders such as contract manufacturers have both unparalleled access and, if they’re unscrupulous, can conduct a variety of attacks well beyond destructive failure.

In a pair of simultaneous papers, Song et al. propose improvements to their machine-vision sabotage detection system from two directions: a real-time cumulative image capture approach to account for obscurement from a moving printhead [51], and semi-supervised learning on their datasets [52]. Both are presented as detecting infill defects, a broad category of their own definition including any attacks changing the infill geometry.

The first paper, on real-time layer image acquisition, assumes a fixed camera position looking down onto the printbed and attempts to acquire a complete image of each print layer, even as the printhead continuously obscures new areas of the print. The authors achieve this by averaging pixel values across sectors of the image in each frame, excluding sectors of the (mostly black) printhead and preserving sectors of the (mostly white) printed material. These partial images are stitched together within each layer to form a complete layer image. The effects of images captured across different stages of layer completion, or indeed the effect of lower contrast between printhead and material on thresholds, are not considered. To evaluate

the results, the authors apply four machine-learning approaches trained on simulated layer images of the exact printed object under test; as with Wu et al. [71], I consider this to render the 95%-and-up claimed detection accuracy meaningless in a wider sabotage-detection context.

The second paper, which tests a semi-supervised learning approach for infill defects [52], trains a multilayer perceptron network against five classes of sample image: a normal grid-infill layer of a cube, and four different positions of a large (around 30% of the cube's width in diameter) cylindrical defect. The authors are able to apply their proposed learning approach, noting that total accuracy decreases from 94.58% down to 84.28% as they allow more labels to be used by the system, essentially allowing it to distinguish between different defect types. Again, while the approach may have merit, the decision to train on only images of a single print with large, obvious defects makes the detection accuracies meaningless for sabotage detection.

In the most relevant 2020 publication for this dissertation, Yu et al. [79], a team including both Chhetri [12] and Al Faruque [4], present a multi-modal sabotage attack detection system for AM. Following through on the proposal in Chhetri et al. [16], the authors investigate and correlate the acoustic, magnetic, kinetic, and current (taken at the wall) side-channels. In the authors' model, side-channel emissions are assumed to correlate to four sets of control signals: axis selection, printhead velocity, nozzle temperature, and a fourth largely-unclear set they present as the 'distance along different axis, with extrusion', labeled as d . The confusion surrounding set d is irrelevant to the results, as the authors are only able to find substantial correlation for axis selection and printhead velocity, and so train only for those two.

Their machine-learning approach is to deploy and test a massive battery of algorithms on producing model functions to translate G-code into predictions of the corresponding, time-synchronized side-channel emanations. These predictions form the baseline known-good signatures against which suspect prints are compared, eliminating the need to produce, capture, and verify a true known-good printing process. Training data for these are the timestamped G-code instructions and features extracted from each side channel, derived from multiple prints. The results, then, should generalize better to a range of models and attacks. The authors claim specific detection accuracies of 99.17% for axis selection and 96.64% for velocity deviations, though the second applies only to velocities above 37.5 mm/s and deviations greater than 25

mm/s. They derive from this an overall detection accuracy of 98.15%, though this is specifically selected metrics of the best-tested machine learning approaches in the testing phase. Under real world test, they successfully detect attacks against two sabotaged models.

Etigowni et al. [19] reach for the much-discussed goal of an automated physical properties detection/verification system. Their approach is the Trusted Integrity Verifier (TIV), mostly-automated method for detecting voids and performing FEA to determine whether the void is compromising an important physical property. The methods are interesting; the authors detect voids by a complex system of raycasting, conversion to voxels, and flood fill on the resulting voxel cloud. FEA is what makes it semi-automated - although the authors decry FEA in their review that it requires great expertise, they offload the definition of physical constraints for object categories to expert judgement. The categorization is handled by a CNN trained on data from modelnet and thingiverse, representing such diverse categories as screw, spanner, and vertebrae. False positive rates for void detection sit at reasonable single-digit percentages, with zero false negatives, implying a slightly over-sensitive detector. The voids themselves are automatically inserted by yet another interesting technique, also relying on raycasting, that hunts for boundaries and high stress concentrations. The success rate for FEA detection of malice is all over the place, seemingly consistent within broad categories such as aerospace or sporting equipment. With results from 45% to 98% detection for the largest tested voids (10 cubic mm), and falling off with volume, it seems a little evasive to claim the system detected attacks successfully. Even so, it's a widely proposed method that is finally being put to the test, with sufficient promise for further development.

Rais et al. [43] introduce Sophos, a framework for detecting sabotage attacks on FDM printers with spatiotemporal G-code modeling. Their primary claim is that it does not rely on a known-good print, which they believe is not well suited to the mass customization they anticipate in 3D printing. While this is not exactly true - they must characterize the natural error of the printer vs. G-code in a series of test prints - the success of their approach is noteworthy. Sophos consists of a simulation slicer for the source G-code and a sampling system for the kinetic and thermal elements (printhead, extrusion, nozzle temp., bed temp.), each of which

outputs the same pixel-based data structures representing the print's layers. These contain spatial and timing data, hence the paper's title. Features within these pixels can be compared in a fairly simple way, and the delta between the "known good" G-code and the observed printer behavior used to judge if an attack has occurred. While the natural error of the printer means there is a confusion area, the authors demonstrate accuracy at much lower thresholds than previous attempts in the literature. In addition, though this is difficult to compare, the Sophos framework presents a more comprehensive and comprehensible approach. Although the claimed rate of zero error on attack detection (False Positives and Negatives both) is predictable given the experimental design (find your confusion zone, set all attacks above it), the results of Rais et al. remain praiseworthy.

Shi et al. [48] perform sabotage detection on the kinetic side-channel using an LSTM-autoencoder system, which is capable of online detection, automated feature selection, and unsupervised training. Against this system are arrayed other, more standardized machine learning systems: AdaBoost, random forest, gradient boosting, and support vector machine classifiers using a range of feature extractors. While any of these systems could be run on any side-channel dataset, only the kinetic channel using accelerometers on the bed and printhead is tested. The authors pit their detection systems against two attacks: void insertion at the STL stage and layer thickness manipulation at the slicer stage. Their accelerometers produced around 4600 sample points across 30 minute prints; a sample rate of roughly 2.5 samples/second. LSTM-autoencoding manages to outperform its opponents in the test, though only marginally in certain matchups. The experimental basis of two attacks, six trials each, all performed on a single cube model is likely not sufficient to evaluate any tested machine learning system as a general-purpose sabotage detector.

Zhou et al. [81], largely the same team as Shi et al. [48], conducts the same experiment as Shi et al. to test the effectiveness of an echo state neural network (ESN) for feature extraction. Differences in data presentation and possibly feature extraction between the two make comparing the results more difficult, but it appears that the whole battery of machine learning implementations performed better in Zhou et al., against the same attacks. The authors

conclude that ESN is both sufficient for sabotage detection and marginally improved over the tested alternatives; again, I consider the experimental basis for the second claim insufficient.

Yang et al. [78] develop online sabotage attack detection systems using the acoustic and visual side-channels. Both are signature based; if the difference between the sensor readings and a known-good recording exceeds a predetermined threshold, the system reports it as sabotage. For the acoustic channel, the authors collect time-sliced spectrogram data. For the visual channel, the authors apply a simple and effective trick: a high-contrast sticker is placed on the printhead to enable accurate machine-vision tracking using a minimum closing circle.

In both cases, the true distinction of the work is in their interpretation and use of metrics. The authors trial both the Wasserstein distance and statistical significance testing to compare acoustic spectrograms. The reconstructed printing path from the visual channel deploys the Hausdorff distance, similarly to Gao et al. [21] and my own work (see Chapter 8). Their characterization of these metrics under different conditions and comparison against other common measures is extensive, providing evidence for their performance under different levels of noise, across longer prints, and with different degrees of infill. Overall, their work here provides an excellent model for comparing across the diverse approaches in use for sabotage detection and reconstruction.

Dawson et al. [17] apply the principles of control theory to detecting sabotage via several stepper motor electrical side-channels. This is the closest work in the field to my own [22], but differs in key aspects of test system and approach. Dawson et al. construct a simulated testbed composed of a single stepper motor and Arduino controller, instrumented using inductive current sensors and a Data Acquisition (DAQ) unit. They collect data as the system is driven through forward, reverse, and idle commands across a range of rotation speeds. At this point, they explore the system response and transfer function. Their goal is to explicitly focus on frequency and phase data, excluding temporal analysis (the basis of my approach). This is reasonable, given their goal: frequency and phase are the traditional domains for control theory, as they provide much cleaner and more understandable relationships than the time domain in most cases.

From this control-theoretic approach the authors develop a detection system that hinges on phase synchrony, a momentary measure of the distance in phase between two signals. High synchrony indicates similar signals; asynchrony would here indicate sabotage. They test this against a similar approach to the atomic attack modifications of Belikovetsky et al. [9]. A sequence of random instructions is generated, then run several times to create a known-good signature. From the sequence, one instruction is selected to be incremented or decremented—this results in slightly higher or slightly lower speeds for movement, or for changes in duration for idle instructions. They are able to detect these with incredibly high reliability in their test, with no false negatives and only two false positives out of 1170 trial runs. The authors note, however, that these are not realistic conditions and that the approach must be gradually scaled up before its real-world performance is known.

4.2 Side-Channels for Technical Data Theft

4.2.1 Field Overview

While the toolpath is the last digital representation of the model, cyber-physical representations continue to exist in the control signals sent to the actuators of the 3D printer, and then also in the physical emanations of the actuators as they work. Both can be detected and recorded with specialized equipment, capturing a form of the technical data corresponding to the object. When this is done without authorization, it constitutes a form of technical data theft.

Physical emanation-based data theft has been conducted primarily through the acoustic [3, 13, 31, 50] side-channel. While many published attacks in this category have used stationary, high-quality microphones [3, 13], some have conducted more realistic surreptitious attacks using smartphones [31, 50]. In an interesting one-off attack [2], an infrared camera was deployed instead.

Side channel data, particularly the physical-emanation type, cannot be used directly for data theft. It must instead be interpreted somehow to reconstruct one of the earlier stages, such as the G-code, toolpath, or model geometry. Here approaches in the field diverge: many deploy machine learning approaches when possible [3, 31, 50], but differ in feature extraction, the class

of machine learning system deployed, and the targeted form of reconstruction. To boost the performance of such approaches, one author developed a complementary attack against slicers which increased the usable information in the acoustic side-channel [11]. For the infrared camera side-channel [3], machine learning was not viable and an established machine vision technique (Kanade-Lucas-Tomasi feature tracking) was used.

My work on power-side-channel reconstruction [22] has two major distinctions from prior works on side-channel technical data theft. First, rather than instrument to capture the physical emanations of an actuator, I directly instrument the control signals driving the actuators. Second, rather than interpret that data using general statistical analysis or machine learning, I build a functional model of the actuator's response based on engineering knowledge of its behavior. At time of publication, no other works in the field have used this approach.

A handful of defensive systems have been proposed to prevent or deter side-channel TDT attacks. An inverse application of [11] seeks to reduce the mutual information between side-channels and the original model and toolpath data [14, 15], mainly by changes to print variables such as orientation and movement speed. Deterrence has also been proposed, by placing conditionally defective features in a model [28, 29]. The proposed inclusions form fatal defects in all but a single set of print parameters, mainly orientation; the majority of side-channel TDT attacks have not been shown to capture the necessary information.

4.2.2 Individual Papers

[3] To the best of my knowledge, Al Faruque et al. [3] present the first paper describing an attack on a desktop 3D printer that leverages the sound generated by the 3D printer's motors. The printer used in the paper employs four stepper motors, three of which are used to move the printer nozzle along the X/ Y/Z axes and one to extrude the filament while printing. The authors exploit parameters like the load and speed having a distinct impact on the frequency and amplitude of the sound produced by the motor. During the training phase, a machine learning approach correlates the recorded sound with the G-code commands influencing the direction, speed, and distance of movement along all axes. During the attack phase, this information is used to reconstruct a printed object from the recorded sound. The authors experimentally

evaluate the proposed method, reporting an average 78.35% accuracy of axis prediction and an average error of 17.82% for movement length prediction.

Hojjati et al. [31] present a side-channel attack against a wide variety of manufacturing equipment, which reconstructs the form and manufacturing process of an object. The attack uses the acoustic and magnetic side channels, as recorded by a compromised cell phone. The authors propose a machine-learning-based method for automating the reconstruction, but deploy a human-led training process. The reconstruction is accurate to within one millimeter for line segment length and one degree for turn angles. As implemented, the attack uses methods similar to Al Faruque et al. [3], but achieves a substantially greater precision; this may be because of their human-led analysis, the inclusion of magnetic measurements, or better filtering of the data.

Song et al. [50] present an attack that is similar to the proposal of Hojjati et al. [31]. There are two fundamental distinctions. First, their machine learning approach extracts a different set of features and is not human-led. Second, they attempt to reconstruct not only the movement and direction information targeted by Hojjati et al., but to transform that representation into executable G-code. The authors report a mean tendency error of only 5.87% in reconstructing the printed object; as with other reconstruction approaches, it is difficult to say what this implies for the printable object or compare it to other authors' work.

In an apparent successor to [3], Al Faruque et al. focus on the thermal side channel [2]. The authors utilize an infrared camera to capture thermal images, which they analyze to identify individual actions in the 3D printing process such as nozzle movements. They develop a set of algorithms for estimating printhead motions from the video feed, and a mapping algorithm to transform the images into nozzle and base plate activity. Experimental proof of the attack fails to reproduce the movements accurately; the authors assume that this is because of the low quality of the thermal camera (50 Hz, no auto-focus, 640×480 resolution) and the single, fixed viewpoint.

Chhetri et al. [13] expand substantially on the acoustic side-channel work of Al Faruque et al. [3], grounding their understanding of the signal in the physical equations governing stepper motor vibrations. This paper contributes explicit and formal statements of some underlying

concepts in the AM acoustic side-channel: that different motors will exhibit different power frequency spectra, that intensity is correlated with movement direction when measured by a fixed-position sensor, and so on. On the practical front, the authors refine their data collection process with filtering, wavelet transformations, and tighter window sizes before feeding into a regression-based machine learning system to produce estimator functions for motor direction and speed. Against their test set (composed of multilayer square and triangular perimeters, as well as a complex key-shape), they achieve a classification accuracy of 74.43%, which is improved via post-processing to 86%. This metric indicates their performance in determining speed and direction within a time window; to characterize the reconstruction itself they report Mean Absolute Percentage Error for segment distances, achieving 20.91% and 11.11% error pre- and post-processing.

The post-processing stage, by the authors' description, removes layers that do not overlap with lower layers. For their all-perimeter test set, this allows them to discard outlier layers, but with a realistic print using infill it would not function.

Chhetri et al. [11] invert their work on information leakage to present an attack: modifying the slicer to maximize side-channel information leakage during a print. Observing again the acoustic, power, electromagnetic, and vibration side-channels, they introduce modifications in the slicer stage that generate more mutual information in the side-channels without altering the final print. Several changes are made to accommodate the new goal: modifying part of the toolchain would not be possible in their previous attack model, so they present a new one with a stronger assumed attacker. New slicer variables are used, and in different ways: fan speed is modulated to increase entropy in the channels, motor speed is varied towards the end of line segments, and power to the stepper motors is cut and restored. The exact changes of these variables are not described, instead being left to an optimization algorithm. The results of their modifications appear to be statistically significant, but whether they are meaningful within the context of reconstruction is not addressed. Their most successful trial, an increase of 21.5% on the mutual information between the power side-channel and the angle of motion, takes it from 1.05 bits to 1.28 bits of mutual information. This is interpreted as a change from 9.78% success rate to 17.19% success rate, but the mathematics used is difficult to substantiate.

Chhetri et al. [14] tackle the problem of confidentiality breach resulting from physical-to-cyber domain attacks. For an FDM desktop 3D printer, the authors develop a two-stage leakage model that describes the relationship between G-code instructions and information leakage via the acoustic side-channel. First, design-time leakage quantification can be used to correlate executed G-code commands and information leaked through various side-channels. Second, run-time leakage information is applied to adapt to changes from equipment wear. The authors then integrate the developed model in a slicer and toolpath generation algorithm. This produces design parameters that minimize information leakage through the acoustic side channel. The algorithm iterates through two design variables, orientation in the x-y plane and feedrate; across five test objects the proposed algorithm reduces acoustic information leakage by 24.76%.

Chhetri et al. [15] apply their mutual information concepts to further AM side-channels, attempting to reduce mutual information at the slicing stage by manipulating design variables. Now using acoustic, vibration, magnetic, and power side-channel monitoring, the authors experiment to reduce mutual information by manipulating part orientation and print speed. The results range from 55.65% reduction on average for vibration to 24.94% reduction for acoustic. Two radially symmetric objects in their test set (a ball and a plate) show no significant mutual information reduction, and the authors reason that a change in their base angle does not alter their G-code meaningfully. To contextualize their results in an attack scenario, they train a Random Forest estimator to guess the G-code equivalent line segments from the side-channels, and apply a "success rate" model to determine the difference between the original and information-reduced versions. Applying their approach shows a drop of just 8.74% across the tested range of line segments, but the meaningfulness of this metric is called into question in both this and subsequent papers [22]. Overall, the authors demonstrate sufficiently that there is mutual information and that the two tested design variables can change the amount for several classes of objects.

Gupta et al., in a conference publication and subsequent journal article [28, 29], present several security design features for incorporation into 3D printed objects during the modeling and slicing stages. These features, when printed with the secret knowledge of the corresponding manufacturing parameters, have no effect on the function of the part, but otherwise cause

significant and detectable failures. The design features presented are three forms of split: a spline, an ellipse, and a rectangle. In each, the feature is intended to be active when printed in a particular orientation and inactive in others. This is achieved through the complex and idiosyncratic ways that CAD programs and slicers handle intersecting geometries. The spline split uses a slightly mismatched pair of overlapping bodies to generate small gaps in some orientations; when applied to a tensile strength test specimen, the voids reduced average failure strain and toughness by 50%. The elliptical split is performed on a rectangular token with an elliptical depression; the resulting pair of adjacent bodies causes discontinuities in the opposite side from the ellipse when printed incorrectly. The rectangular split, rather than relying on inexact tessellation, tricks the slicer into generating support material in place of structural material, leaving a hollow in the part after removal. The authors present these as increasing the difficulty and reducing the reliability of stealing AM designs; the legal implications of setting traps in your design files are not discussed.

My own publication [22] on technical data theft using the actuator current side-channel is covered at length in Chapter 7. Here I will simply distinguish it from prior publications in the field by repeating its claims: a reconstruction method which taps into the much cleaner input current side-channel for stepper motors, enabling the reconstruction of full object models with realistic infill.

Chapter 5

Instrumentation and Signal Characteristics

My test printer, the Lulzbot Taz 6 (See Figure 5.1), is an FDM printer employing a Cartesian-coordinate plotting system. This means the print-head travels on linear X, Y, and Z axes of motion. The X axis is controlled by a single stepper motor, and moves the print-head via belt-drive. The Y axis is also controlled by a single motor and belt-drive, but instead moves the print bed relative to the print-head. The Z axis is driven by two synchronized stepper motors, which raise and lower the print-head armature along a pair of vertical worm screws. In addition, a single stepper motor is directly attached to the print-head to control extrusion, referred to as the



Figure 5.1: A Lulzbot Taz 6 3D printer.

E axis. This motor is geared down to control a wheel, to which the filament is clamped. The wheel then rotates to pull filament from the spool and push it into the extruder nozzle.

All of these different mechanical interconnections give each motor a different relationship between rotations of the motor drive shaft and the motion achieved on that axis. These are set in the printer's firmware as steps-per-mm, enabling the motor controller to generate the correct number of steps for a target motion. Through the user interface, the operator can modify these values to better calibrate the printer or to compensate for modifications to or replacements of the parts.

Every stepper motor on the Taz 6 is of the same type: a NEMA-17 bipolar stepper motor (see Figure 5.2). Each motor has two phases and each phase is connected by two wires, forming an electrical loop delivering current from the controller. While stepper motors can be driven using Pulse-Width Modulation (PWM) signals, which are square waves, it is common practice to drive them using a sinusoidal signal for better torque, vibration control, and efficiency [47]. Both the positive and negative peaks of the current on either



Figure 5.2: A NEMA-17 Bipolar Stepper Motor.

phase cause the motor to advance or reverse depending on the firing sequence. There are several strategies for driving the motor that are compatible with the design and allow different step sizes: the stepping pattern for this motor is shown alongside a wiring diagram in Figure 5.3.

The peaks of the current directly cause the motor to move, and the mechanical construction of the motor ensures that positional error does not accumulate [39]. A single peak on one phase will predictably move the motor from one step position to another. The direction of movement is determined by the sequence in which the two phases are fired. The speed of movement is determined by the speed of the firing sequence.

The *toolpath*, a sequence of movement and printing commands, is issued in the *G-code* format, a legacy format commonly used in desktop 3D printers. The toolpath is interpreted by the printer and translated into a series of motor movements. The motor controller generates current in the correct pattern to achieve these movements. The particular properties of this

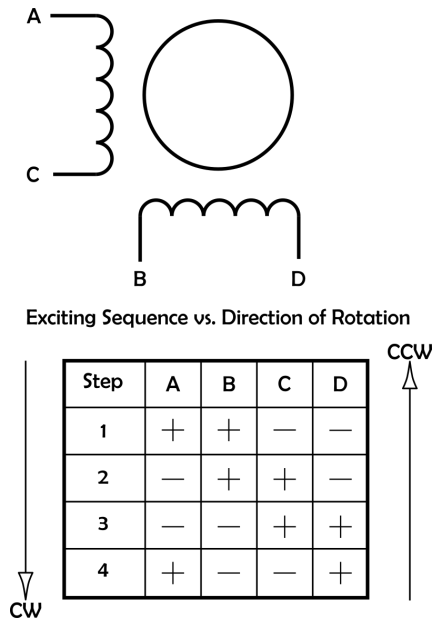


Figure 5.3: Wiring diagram and excitation chart for a bipolar stepper motor. The wiring diagram shows the two phases, connected at terminal points AC and BD. The excitation diagram shows the sequence of positive and negative impulses that must be delivered to these terminals to properly actuate the motor.

current and recognizing the patterns necessary for executing G-code commands present several challenges to the reconstruction process, described in the remainder of this section.

5.1 Instrumentation

I devised the instrumentation strategy, originally published in Gatlin et al. [23] on power side-channel signatures, using inductive current clamps to monitor motor phases. The clamps used in this work are of the Hall-effect type, and are thus capable of monitoring AC and DC currents on a target conductor with frequencies into the kilohertz range. While alternatives such as in-line ammeters or pullup resistors can also capture this data, they have notable downsides compared to inductive clamps: they can interfere with actuation signals, are difficult to attach and remove, and – when used maliciously – might be detectable by anti-tamper impedance testing.

The clamps are attached to individual wires supplying power to the stepper motors. Each motor receives power over four wires, representing the delivery and return paths of their two phases. The signals on the phases are strongly related, but not identical. When instrumenting

to capture signatures for sabotage attack detection, it is sufficient to apply one clamp to a single phase of each motor; for the other two applications of reconstruction and forensic comparison, it is necessary instrument both phases of each motor, i.e., apply two clamps to each motor.

The current clamps are only front-end instrumentation, providing momentary current readings over a cable terminated by a Bayonet Neill-Concelman (BNC) connector. To interpret and record these readings, the probes are connected to an oscilloscope with long-term datalogging capabilities. Upon capture, the data is transferred to a workstation computer running post-processing software of my own design and implementation; the specific software varies based on the cases of use or misuse considered in this dissertation, and are covered in the individual Chapters 6, 7, and 8.

The probes, oscilloscopes, and configurations used in my experimental work vary based on the specific requirements of each application. These are summarized in Table 5.1.

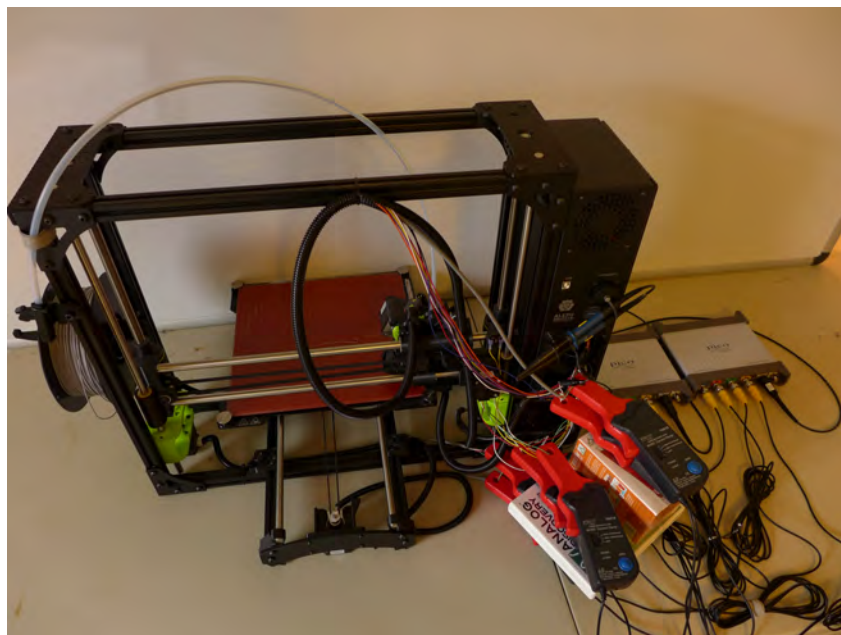


Figure 5.4: A Lulzbot Taz 6 printer, instrumented by Picoscope 5444D oscilloscopes. The probes are Picoscope's 60A Inductive Current probes. Each motor has two clamps attached, one for each phase. The fan controller is also instrumented by a standard voltage probe. The data captured here is transmitted to a host PC running the PicoScope application.

Table 5.1: Application-specific implementations of the general instrumentation setup.

APPLICATION	CLAMPS	OSCILLOSCOPE	SAMPLING	SENSITIVITY
Sabotage Detection	Tektronix A622	Teledyne LeCroy Waverunner 610	25 KS/s	10 mV/A
Design Reconstruction	Pico TAO18	PicoScope 5444D	4 KS/s	1 mV/A
Sabotage Investigation	Pico TAO18	PicoScope 5444D	4 KS/s	1 mV/A

5.2 Trace Characteristics

In this section, I will discuss particular trace features that correspond to printer behaviors. In the test printer, the toolpath is defined using the G-code command language. Each command can prompt one or more motor behaviors, based not only on the command itself but on the printer’s current physical state. Where appropriate, I mention the typical G-code commands associated with a behavior.

5.2.1 High Frequency Noise

A consistent property of the raw trace is the presence of high-frequency noise. In most cases, the signal-to-noise ratio is high, and trace properties measured across longer periods of time are unaffected. More precise properties, such as locating a peak, are affected (see Figure 5.5). Locating peaks more consistently can require pre-processing the raw trace with a low-pass filter¹, a common technique in signal processing. All trace figures to follow will be presented after filtering.

Applying a low-pass filter to these current traces creates a problem of its own. The frequency of the current and the speed of the motor are directly proportional, and the valid range of speeds overlaps in part with the frequencies of noise. Thus, I must restrict the cutoff frequency of the low-pass filter to above this range, and compensate for the remaining noise in other ways.

¹A low-pass filter is a signal processing filtering technique that dampens the power of a signal above a cutoff frequency. There are multiple ways to apply low-pass filters; images of subsequent traces have been filtered by an 8th-order Butterworth filter.

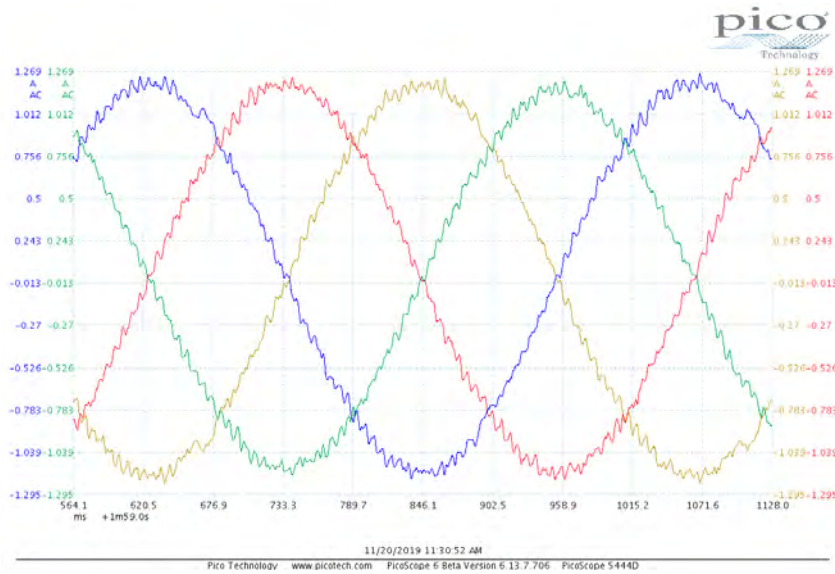


Figure 5.5: An unfiltered current trace captured on the Picoscope. Note the high-frequency noise throughout, making precisely locating a peak in the center of each wave difficult.

5.2.2 Unidirectional Movement

The simplest section of oscilloscope trace to interpret is unidirectional movement at a constant speed. This will occur in the middle of any linear movement command. Observing the current at this point will produce a trace as shown in Figure 5.6. In this figure, and in all other trace figures, I plot a positive and negative version of the current on each phase, producing a total of four. The firing sequence of the phases can then be seen just by looking at positive peaks.

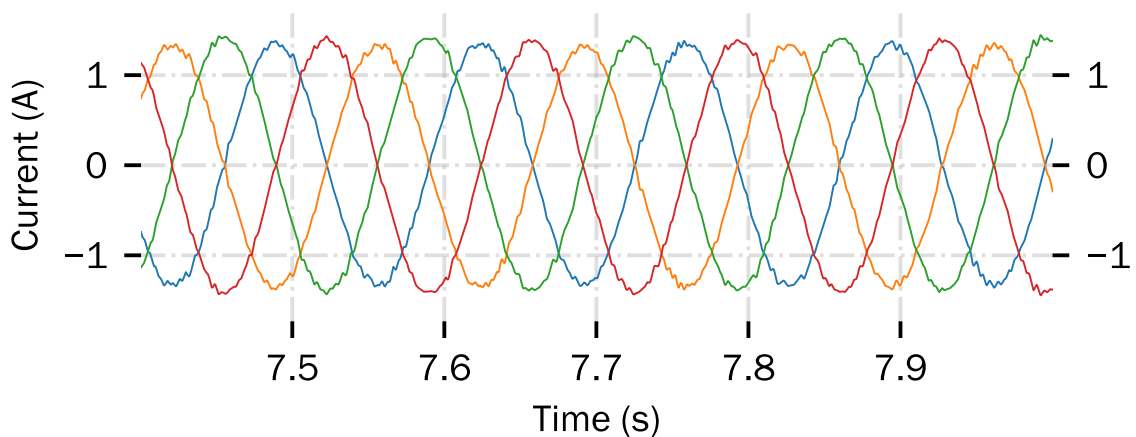


Figure 5.6: Constant-speed motor trace, after applying a low-pass filter. This plot is generated in code from the processed data.

Peaks have two independent characteristics by which they can be recognized: *height* and *prominence*. *Height* is the absolute value of the peak at its highest point. In Figure 5.6, the peaks have a consistent height of approximately 1.5 A; in other situations, the height can vary dramatically. The *prominence* of the peak can be simply described as the height of the peak over the adjacent minima. This corresponds to the amplitude of a signal in electrical terms. The prominence of the peaks in Figure 5.6 is consistently around 3 A. In other situations, the prominence of a given peak can be significantly less. Interpreting each peak as a single step, I can precisely track position. From there I can use the known property of step size, or distance per motor step, to determine both printhead and filament position at any time, and the linear speed of movements.

The phases are all operating at the same constant frequency (corresponding to a constant movement speed) and firing in the same order. It must be noted that, at the beginning and end of a linear movement, the motor controller ramps up and ramps down the frequency, respectively. This produces acceleration and deceleration in the movement, and the artifacts from it in the trace should be accounted for in reconstruction.

5.2.3 Start, Stop, and Dwell

The trace exhibits specific behavior at the beginning and end of movements. This behavior includes apparent changes in frequency and absolute value. This occurs at the beginning and end of every print, as shown in Figure 5.7. This also occurs during Dwell commands, which halt movement on every axis for a set period, as shown in Figure 5.8.

Both figures illustrate a challenge associated with these transitional periods: changing DC offsets. The height of the first peak after a period of inactivity is visibly lower, and the second peak is visibly higher, than their normal range. Over time, they gradually stabilize at the same level, as in the linear movement of Figure 5.6. In electrical terms, this difference in the absolute height of an AC signal is referred to as its Direct Current (DC) offset. Notably, the prominence of the peaks does not vary dramatically. A change in DC offset over time means that height cannot be relied upon to distinguish peaks.

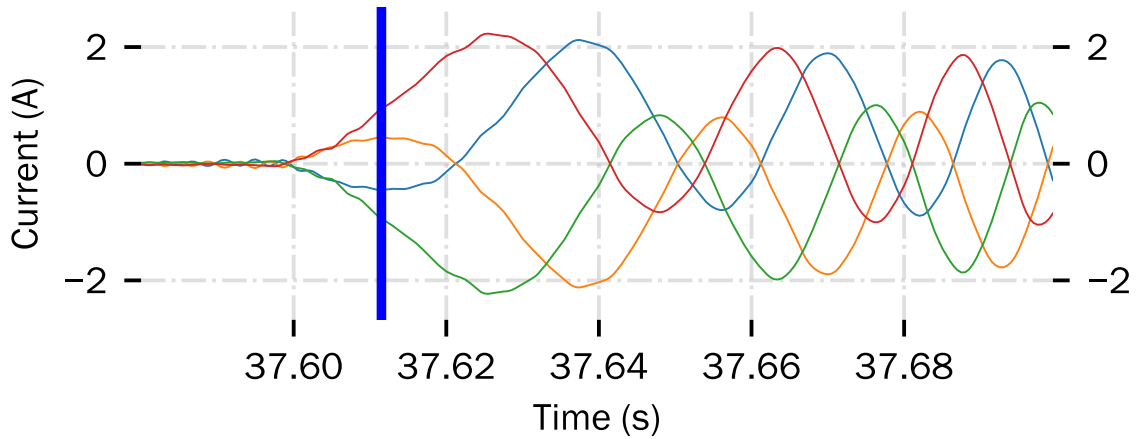


Figure 5.7: Beginning of motor movement after inactivity. The DC offset of the phases start at nearly 2A, then converges over time. The blue line indicates the presence of a reversal.

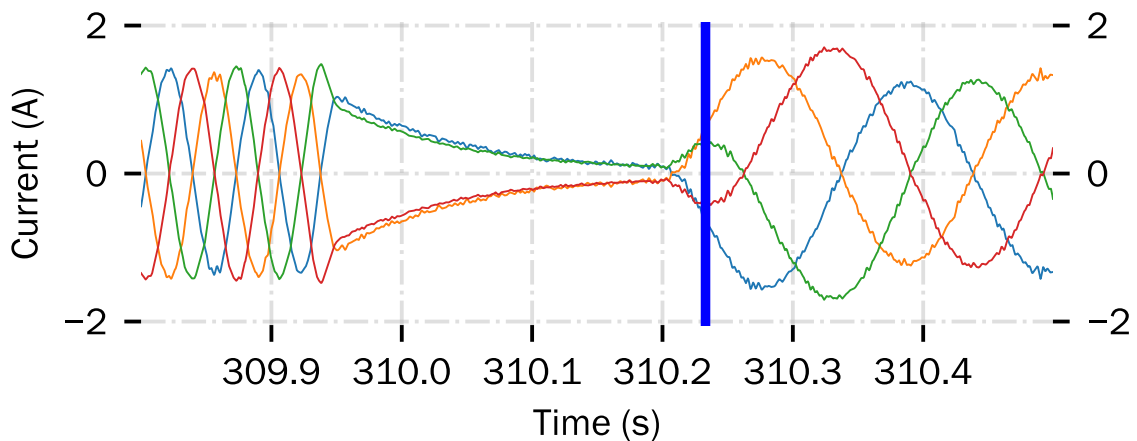


Figure 5.8: Dwell shown in the trace. Note the much lower height of the first peak at 310.25s. A change in speed is visible from the first section of activity to the second.

To aid in this filtering, I can observe the change of inter-peak intervals while transitioning into and out of inactivity. Before a period of inactivity, there is a stable inter-peak interval corresponding to constant motion. There will then be a single larger interval between the last peak before and the first peak after the inactive section. After that, the next movement command begins and, after a short period of time, shows stable inter-peak intervals once more.

5.2.4 Reversal of Direction

When a sequence of G-code commands reverses the direction of movement along an axis, I observe the behavior seen in Figure 5.9. The characteristic feature of a reversal is that it changes the firing order of the phases. They can be detected by testing for this change, but

several artifacts of the trace can complicate the process. In the figures showing reversals, I mark the first peak in the new direction with a vertical blue line.

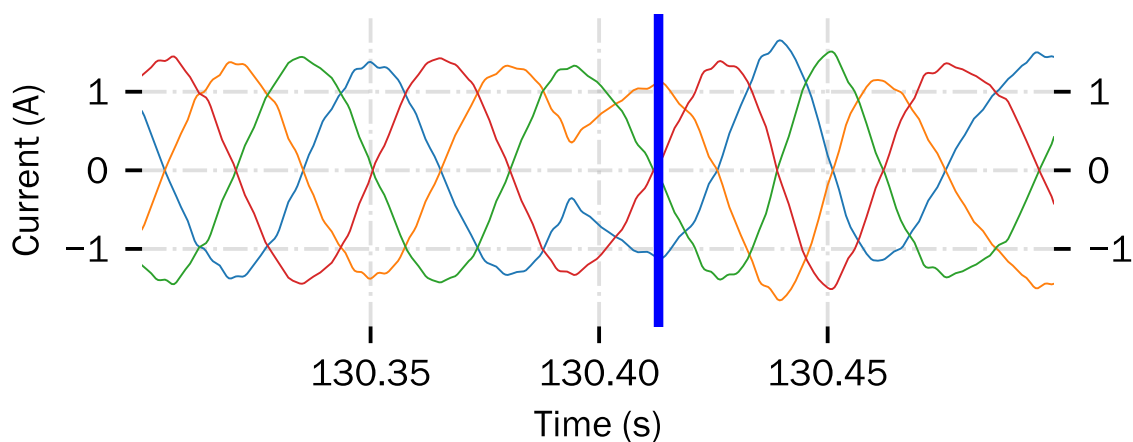


Figure 5.9: Valid reversal shown in the trace. After the central green peak, 130.4s, the direction of travel is reversed. The blue line marks the first peak after the reversal. This reversal presents no additional problems for reconstruction.

The DC offset of the trace can change during reversals to the point that it impacts peak recognition. In Figure 5.10, peaks in the vicinity of the reversals have both a low absolute value and a low prominence, but are still valid peaks (i.e., shift the motor position). This can occur not only to the reversal peak, but also to any peak before or after. Each potential missing peak represents a uniquely malformed firing order, when in reality only well-behaved reversals actually occur.

5.2.5 Trace Synchronization

The full printer system is composed of multiple motors acting together, and it is necessary to maintain synchronization when capturing their current traces. Done poorly, this can cause issues with misordered motor movements across different axes. These misalignments can distort the reconstructed shape significantly.

While it is possible to synchronize the beginning of multiple traces captured individually, several common G-code commands have variable duration, which leads to desynchronization. These include motion commands such as Home (G28), and thermal commands that use a feedback loop to reach a specific target temperature.

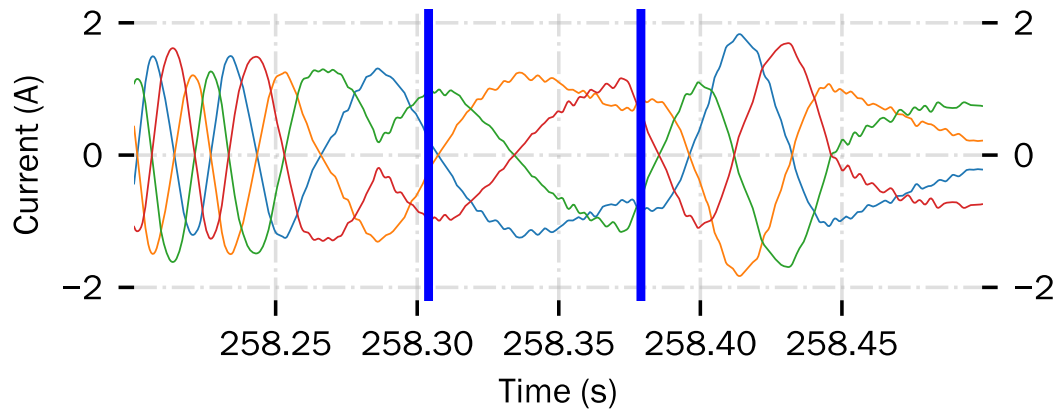


Figure 5.10: Reversal with multiple issues complicating reconstruction. Peaks from 258.25s to 258.4s are low-prominence, of varying width, and represent two reversals in rapid succession.

The most reliable synchronization by capturing all motor signals simultaneously in a single print run. In this case, capture can be started by a manual signal. Alternatively, it can be triggered on recognizing a consistent signal in the preamble, a printer-specific block of G-code that occurs before each print. The second option provides better opportunities for automation, and I have used it extensively in the experimental work of this dissertation.

Chapter 6

Use: Detecting Sabotage Attacks

Research Question 1 asks if the power side-channel provides enough information to detect sabotage attacks in AM. The **system under threat** is an FDM printing system, which is widely used in AM for manufacturing in polymers and composite materials. In order to develop an approach, I must first determine a threat model that the proposed solution should address. I consider the following assumptions about environment, adversary, defender to be realistic.

The system under threat is an FDM printer, using an open-loop control system for positioning its print head. Computerized elements of the AM workflow, such as the controller PC, the printer itself, or a connected network, can be compromised. Analog elements, such as the stepper motors and heating elements of the printer, cannot be directly compromised.

In this scenario, my detection system including probes, oscilloscope, and a monitoring PC is air-gapped from the manufacturing environment and not itself compromised. I assume that the attacker wishes to sabotage a 3D printed part produced using the instrumented printer, and is capable of delivering an undetected attack using the compromisable elements. The defender wishes to take all reasonable measures to prevent sabotage, has physical control of the instrumented printer, and can produce and destructively test at least one copy of the desired object.

My proposed solution can be outlined as follows. While a 3D object model can have various representations, from STL files to individual toolpath commands, these commands are eventually translated into electrical signals supplied to the actuators. For the Cartesian FDM printer used in this dissertation, the positional actuators are four stepper motors that control the print head's X/Y/Z movement and filament extrusion.

The function of these motors is described in Chapter 5. At this stage, it is only necessary to recall that stepper motors are driven using PWM, and that the speed and timing of the supplied waveform directly controls motor movement. This allows me to immediately make several observations. First, modifying a design (in any of its representations before printing) will produce corresponding changes in the current used to drive the actuators. I propose therefore that deviations between designs can be detected by comparison between the current traces of a monitored manufacturing process and the traces of a verified benign one. This is illustrated in Figure 6.1, showing the original (top) and an altered (bottom) current trace of the X motor captured during an experiment.

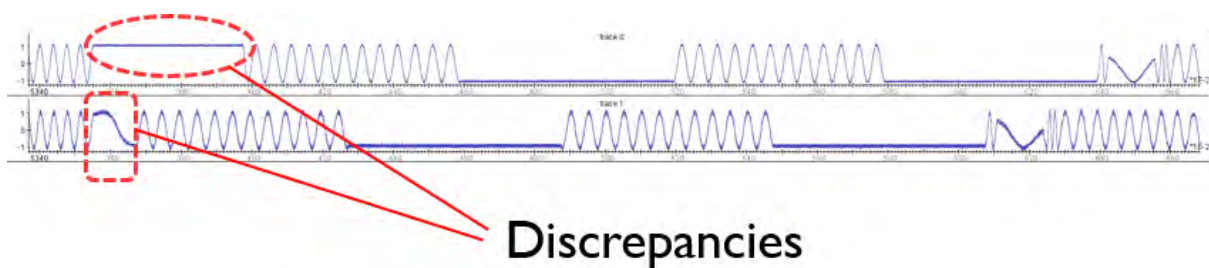


Figure 6.1: Normal and Sabotaged Traces, X Motor

However, while visual comparison provides an intuitive way to illustrate the idea, it is not practical when considering traces that are hundreds or thousands of times longer than the given example. To enable an automated comparison, I built upon an approach proposed by Belikovetsky et al. [9] for acoustic side-channel emanations. I also introduce several significant modifications to this approach. First, my instrumentation strategy (described in Chapter 5) separates the signals from each motor. I can therefore apply the approach to each motor individually, rather than to the combined signal. Second, while the original approach detects integrity violation for an entire 3D print, in this work I localize detection for each printed layer.

To distinguish between layers and layer transitions, I exploit the following observation: the Z motor does not move while a layer is being printed (represented by primarily DC current in the power trace) and moves when the layer is changed (represented by the high-amplitude oscillations in the power traces). This can be used to identify time-intervals for both layers and layer transitions. Note that some slicer settings involve Z-hops during individual layers to print smoother line breaks. Under such conditions, it would be necessary to distinguish

between hops and layer transitions; such a system is developed as a by-product of work done for Research Question 2, in Chapter 7.

It is necessary to compare printer behavior during the layer transitions. Both the X and Y motors are active during this phase to ensure correct X/Y positioning at the beginning of the next layer. Therefore, monitoring is needed to detect attempts to modify this initial position. Further, the layer transition time can be used for a burst of increased/reduced filament extrusion (an attack similar to the one presented by Moore et al. [37]). Such sabotage attempts should also be detected during this time.

6.1 Master Signature Generation

Figure 6.2 outlines the approach that I use to create a *master power consumption signature*, i.e., the signature of a verifiably unaltered object.

An object model is used to 3D print a part (in the figure, a propeller). As discussed above, the threat model assumes that this file, the computer that stores this file and controls the 3D printer, the communication network between this computer and the 3D printer, and the 3D printer itself can each be compromised and used to sabotage a printed part. The benign nature of the 3D printed part, i.e., its compliance to the required mechanical properties, can be validated by using destructive methods. This might involve a combination of scanning and layer-by-layer deconstruction and microscopy. As manufacturing and testing equipment can be air-gapped, their simultaneous compromise is unlikely.

During the 3D printing process, I monitor and capture current traces using my instrumentation strategy. One trace is captured for each individual actuator, and each can be treated as a *channel* of information. Traces from each individual channel are time-synchronized.

I can then use Z channel activity to identify transitions between layers (indicated in Figure 6.2 by two vertical dotted lines). The result of this operation is a list of $\{t_0, t_1, \dots, t_n\}$, where t_0 and t_n are absolute times of traces beginning and end. The values t_1 through t_{n-1} are times of layer transitions, i.e, the times when the Z axis changes states between moving and being steady, or vice versa. As the trace recording begins right before the very first toolpath command is sent, t_0 will always be 0. After the 3D printing is finished, the printer raises the

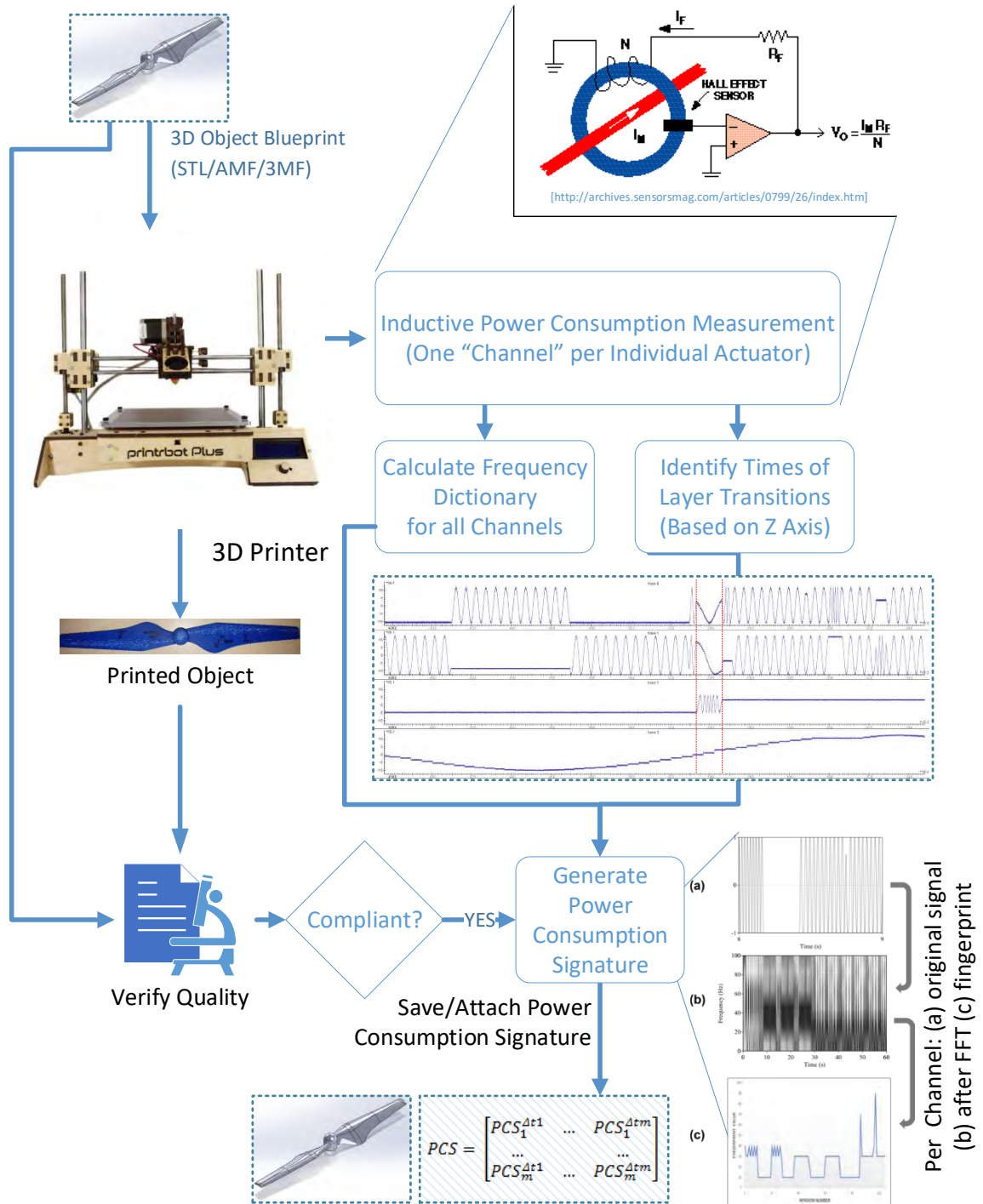


Figure 6.2: Power Consumption Signature Generation

head and ceases movement; activity on the traces will also cease at this time. The time of this event will be captured by t_{n-1} . Therefore, I can set t_n to the minimal duration of all channels. I identify t_1 through t_{n-1} as shown in Figure 6.3.

```
FINGERPRINTCREATION(signal, wnd_size, wnd_step)
    resampleRate = 400
    downsample = Resample(signal, resampleRate)
    finger_print = []
    frames = SplitToFrames(downsample, wnd_size, wnd_step)

    for each frame in frames
        fft_frame = FFT(frame)
        max_bin = Max_position(ABS(fft_frame))
        finger_print_value = (max_bin / wnd_size) * resampleRate
        finger_print.append(finger_print_value)

    return finger_print
```

Figure 6.3: Power-Trace Fingerprint Generation

The current trace information captured for all individual channels is used in the signature generation. Figure 6.3 gives pseudo code for the algorithm I use to generate the fingerprint.

In this algorithm, I first set the resample rate to 400Hz and downsample the signal. The captured signals contain the majority of their information at frequencies of 200Hz and below, so no critical data should be lost and a great deal of high-frequency noise will be removed. The signal is segmented into overlapping frames of a given size *wnd_size* at a step interval of *wnd_step*. For each frame, I calculate the dominant frequency of the signal. This is achieved by applying a Fast Fourier Transform (FFT) and identifying the frequency bin with the highest detected power. The corresponding frequency value of the bin is computed and appended to the fingerprint. The final result is a sequence of dominant frequencies for each frame in order: this is the complete fingerprint of the input signal.

First, I calculate the fingerprint signature for each channel individually, using predefined window sizes and steps. In this case, I have used a window size of 32 samples and a stepping measure of 16 samples. Then, for each channel, I segment the fingerprint with layer information from the Z-axis. The information of the Z-axis fingerprint is compared to a threshold and transformed into boolean values indicating whether the motor is moving. Please note that at this stage, I do not distinguish between possible variations of motor speed for the Z axis; accurate tracking of layer distances is achieved in answering Research Question 2, Chapter 7.

Thus, the fingerprint is segmented into time-intervals $[t_i, t_{i+1}]$. Generalizing for m monitored channels, the resulting master signature of the print can be described as a matrix of fingerprints for each individual channel and for each individual power consumption signature time interval $PCS_j^{\Delta t_i}$.

$$MasterSignature = \begin{bmatrix} PCS_1^{\Delta t_1} & \dots & PCS_1^{\Delta t_n} \\ \dots & & \dots \\ PCS_m^{\Delta t_1} & \dots & PCS_m^{\Delta t_n} \end{bmatrix}$$

As indicated above, the power consumption-based master signature is only accepted if the quality of a 3D printed object was successfully verified by destructive testing. At this point, the master signature can be saved for use in the verification algorithm.

6.2 Tested Signature Generation & Verification

To verify that the 3D printing process of a part was not altered (and thus that the part was not sabotaged), I compare a master signature with the signature generated from the current traces of the investigated process. Figure 6.4 summarizes the verification process.

For the verification process, the current delivered during the printing process is measured using a non-invasive induction method. The signal of each axis is then processed using the Power-Trace Fingerprint generation algorithm. The post-processed Z-axis fingerprint consists of 0s (for the time frames when z motor is not moving and a 3D part's layer is printed) and 1s (for the time frames when z motor is moving to start to the next layer). The transitions between 0 and 1 (or vice versa) determine the times t_i s. Both diverging amounts of printed layers and layer durations can be used as an early indicator of a sabotage attack.

I then construct a new time series that represents the changes in the Z-axis. The result is a list $\{t_0, t_1, \dots, t_n\}$ where each t_i is the beginning of a layer or the transition between layers. Then, generated signatures of the other channels are split into layer corresponding to the list of t_i s. The generated signature is a $m \times q$ matrix, where m is the number of channels and q is the number of time-frames identified for the tested power traces. The matrix elements $\overline{PCS}_i^{\Delta t_j}$ are tested signatures for each channel/time-interval in the tested power traces.

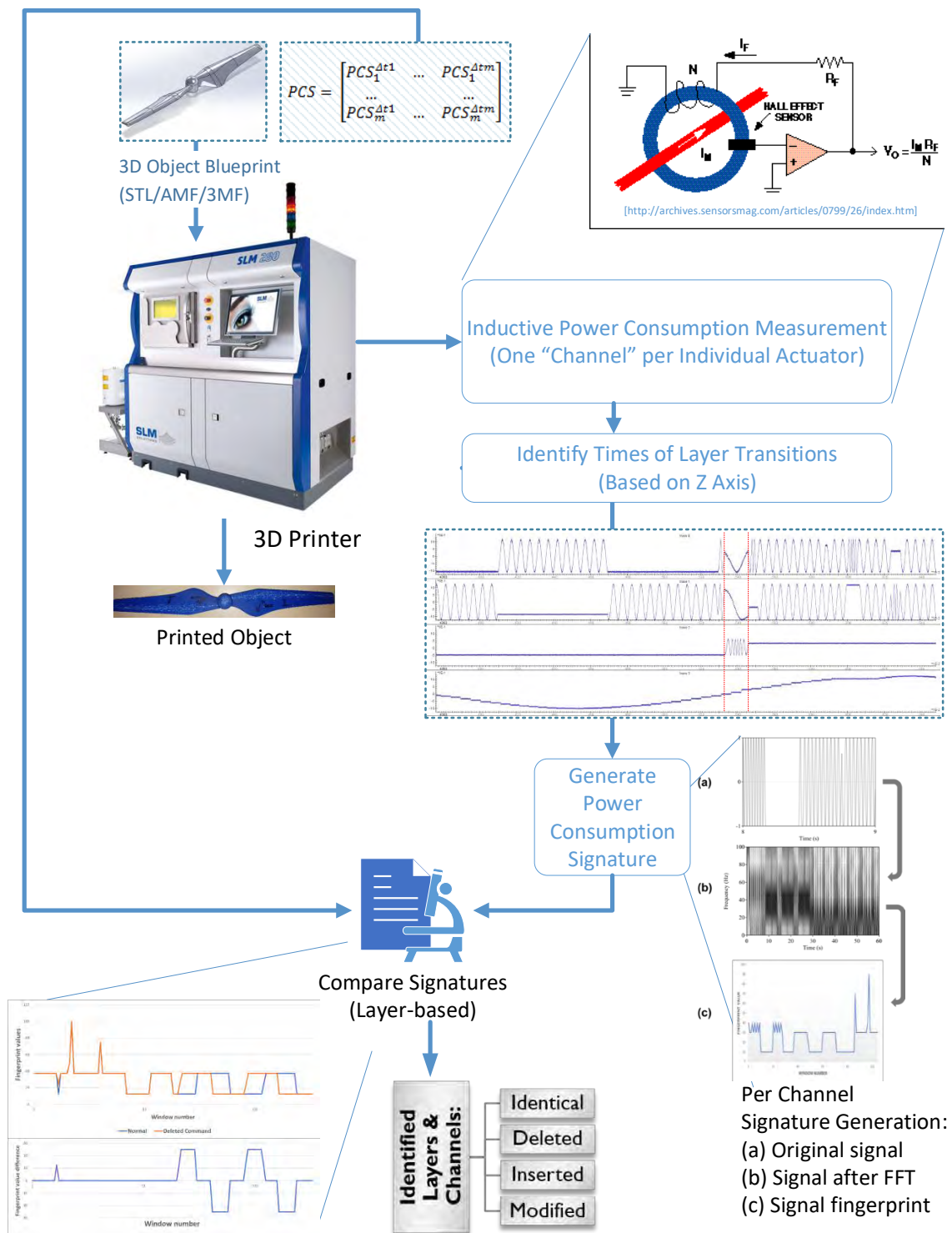


Figure 6.4: Power Consumption Signature Verification

Before I describe how master and tested signatures are compared, there are several additional remarks. First, even an unaltered 3D printing process can introduce slight deviation in the duration of identified time-frames. This can be caused by factors like the stochastic nature of the physical 3D printing process, environmental influences, or measurement errors. To accommodate for this possibility, I introduce a threshold $\Delta T_{threshold}$ under which I consider traces for two time-frames, one from master and another one from tested signature, as approximately the same length.

Second, the monitored actuators perform differently. For instance, the speed of a motor controlling filament extrusion is significantly lower than of X/Y/Z motors. The axis motors also each have different drive characteristics, to a lesser degree. As such, a different degree of similarity should be acceptable for different channels. To accommodate this, I introduce a similarity threshold vector $ST = (ST_1, \dots, ST_m)^T$; through its elements ST_i different similarity thresholds can be specified for each individual channel. Such thresholds do not necessarily have to be a single value, but can represent a complex condition.

Third, I also wish to distinguish between 3D printed layers that are either (a) identical, (b) have been deleted (c) have been inserted, or (d) have been modified (i.e., a combination of changes). This requires a similarity measure between two layers. The layer similarity can be defined based on the similarity of individual channels. In this work, I use a conservative approach, i.e., the dissimilarity even in a single channel is treated as the dissimilarity of the complete layer.

I calculate the similarity between two channel/time-intervals of master and tested signatures as follows:

$$\begin{aligned}
 & IsSimilarLayer\left(\begin{bmatrix} PCS_1^{\Delta t_k} \\ \dots \\ PCS_m^{\Delta t_k} \end{bmatrix}, \begin{bmatrix} \overline{PCS}_1^{\Delta t_l} \\ \dots \\ \overline{PCS}_m^{\Delta t_l} \end{bmatrix}, \begin{bmatrix} ST_1 \\ \dots \\ ST_m \end{bmatrix}\right) = \\
 & IsSimilar(PC S_1^{\Delta t_k}, \overline{PCS}_1^{\Delta t_l}, ST_1) \wedge \dots \wedge \\
 & IsSimilar(PC S_m^{\Delta t_k}, \overline{PCS}_m^{\Delta t_l}, ST_m)
 \end{aligned}$$

This style of comparison would allow a technician to precisely locate and evaluate an anomaly. I consider the manufactured part as validated if all of the following conditions are fulfilled:

- Amount of layers (or tested time-intervals identified based on layer transitions) is identical for both master and tested signature, i.e., $n = q$.

- Deviations of layer duration for each compared layer are below a specified threshold $\Delta T_{threshold}$, i.e., $\forall \Delta t_i, i \in [1, n]$:

$$|\Delta t_i - \overline{\Delta t_i}| < \Delta T_{threshold}$$

- For a specified threshold vector $(ST_1, \dots, ST_m)^T$, master and tested signatures are similar for each channel/time-interval, i.e., $\forall i \in [1, m], j \in [1, n]$:

$$IsSimilar(PC S_i^{\Delta t_j}, \overline{PC S_i^{\Delta t_j}}, ST_i) = TRUE^1.$$

Since even identical prints introduce a slight deviation in the values, the division of Z-axis into layers is not precise and might introduce discrepancies between layer times and durations of master and of tested signatures. Therefore, in order to compare between two individual layers I run the layer-comparison algorithm several times, each time using a different offset for the beginning of the comparison.

The algorithm in Figure 6.5 describes a comparison between the layers using a specific offset, and as such implements the above mentioned function *IsSimilar*. Each value of the two layers is compared and the mismatching values are counted. If the total number of mismatching values is below a specified threshold, the layers are considered identical. Otherwise, the comparison is done again with a different offset.

6.3 Experimental Evaluation of Sabotage Detection

In this section, I first present the experimental setup as it differs from that described in Chapter 5 and describe the experiments performed.

¹There's no need to distinguish between layers of master and tested signatures when the first condition is fulfilled.

```

ISSIMILAR(layerA , layerB , offset)
    mismatch = 0

    for(i = 0; i < len(layerA) - offset; ++i)
        if(layerA[i + offset] != layerB[i])
            mismatch++

    if(mismatch > mismatchThreshold)
        return false
    else
        return true

```

Figure 6.5: Layer comparison algorithm.

For these experiments, traces were captured from a Printrbot Plus 1404 3D printer. This is a desktop 3D printer employing polymer FDM technology and a Cartesian-Coordinate positioning system. For the X and Y axes as well as for filament extrusion, this printer utilizes single Nema 17 stepper motors, rated for 4.2V and 1.5 A per phase. The Z axis is driven by two such motors. Figure 6.6 shows the complete experimental environment.

At this stage of the experiments, several features differ from the instrumentation strategy used in later sections. I use a high value Tektronix A622 AC/DC current probe attached to a Teledyne LeCroy Waverunner 610 Zi oscilloscope to collect traces from single phases of individual motors, synchronizing them based on a trigger channel attached to the printer’s fan controller. The fan control line was pulled to an external resistor, and set to generate a falling edge immediately before printing the first layer. The oscilloscope is set to sample at 25 KS/s,

For the evaluation of the proposed solution, I follow an approach proposed by Belikovetsky et al. [9] – to test the approach on and identify the detectability threshold for the minimal possible manipulation primitives, i.e., individual G-code commands. Belikovetsky et al. refer to these as *atomic modifications*, and proposed the following categories of tests: (a) insertion of a new G-code command; (b) deletion of a G-code command present in the original STL file; (c) a reordering of two G-code commands present in the original STL file; (d) replacement of a printing move command with a non-printing move command (a “smart void”).

Real manipulations like the injection of voids in the design [10, 54], substitution of material [80], changed printing orientation [77, 80], or the amount of extruded filament [37] – will introduce deviations in one or more G-code commands, and will have a larger signature.

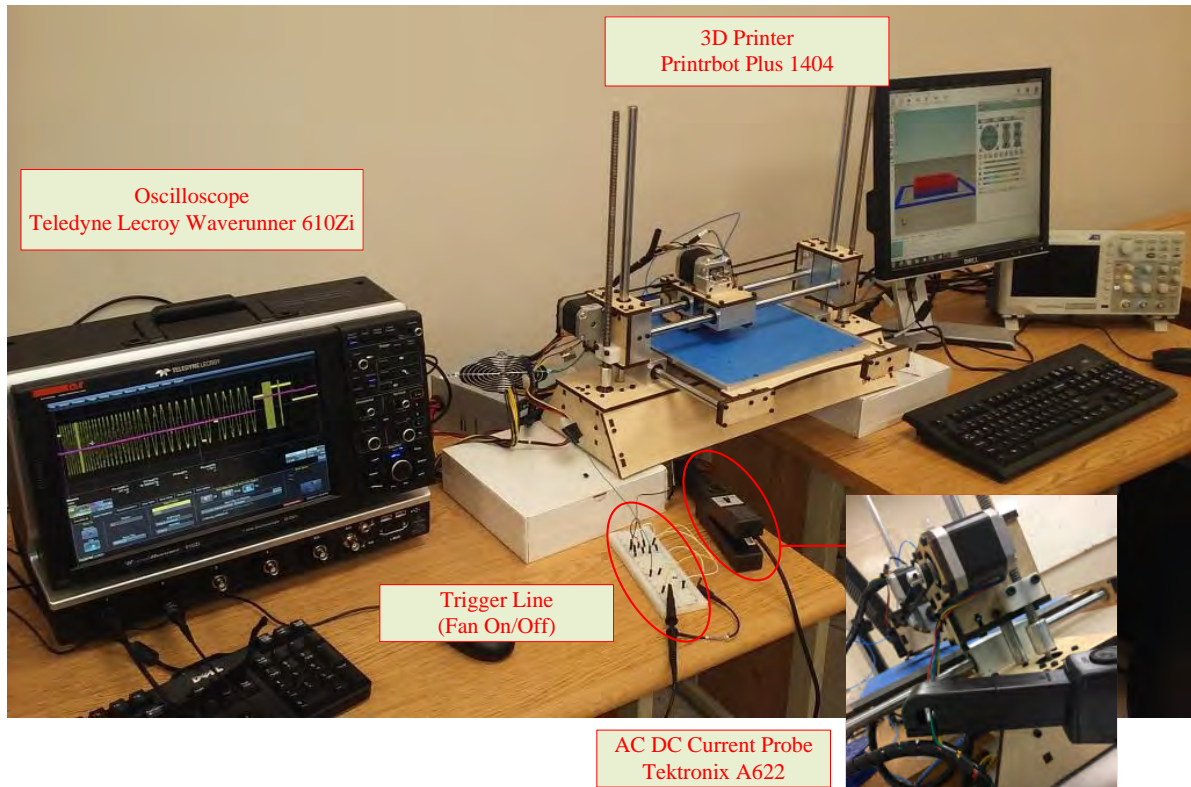


Figure 6.6: Signature Capture Experimental Environment

I have tested the proposed approach on a simple 10 layer cube with a honeycomb fill; the test cube was $1\text{cm} \times 1\text{cm} \times 0.2\text{cm}$.

My attacks were composed of the G-code commands G1, for an extruding movement, and G0, for a non-extruding movement. For the insertion attack, I inserted a G0 command for a rapid linear move in layer 7. For the deletion attack, I deleted a single G1 command in layer 7. For the reordering attack, I swapped two G1 moves in layer 7 and two more in layer 8. For the replacement attack, I replaced a G1 command with the equivalent G0 command in layer 7.

During generation of the master signature, I want to accommodate for possible variations during 3D printing. Therefore, I collected a minimum of 10 traces per channel of the object being printed. As there are four channels—X, Y, Z, and extruder motors—and only a single inductive current probe, I conducted 40 total prints of the same benign object. These measurements established a baseline or “golden” measurement for the motors. For each type of atomic modification, I collected 3 traces per channel of the modified print job.

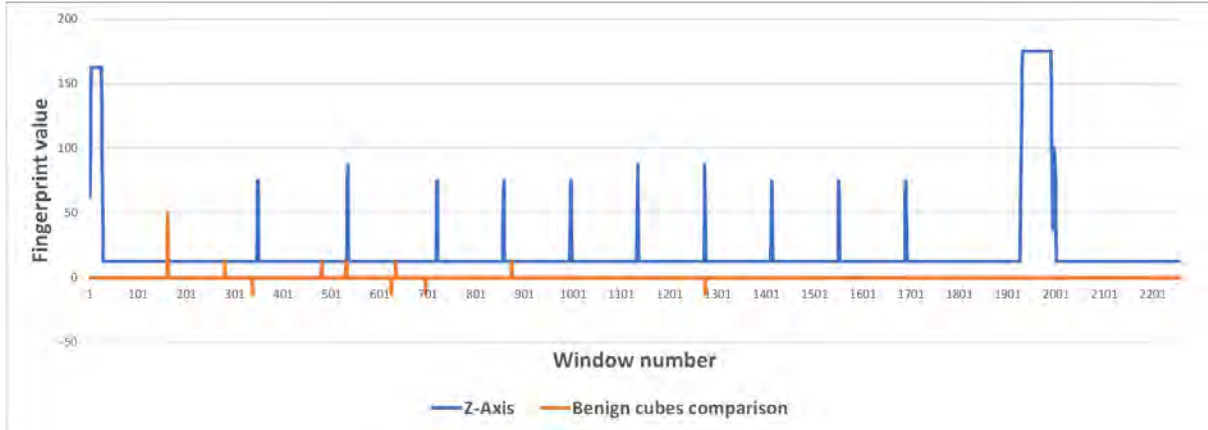


Figure 6.7: Comparison of two benign prints

6.4 Results

Figure 6.7 shows a comparison of the X axis between signature of a master signature and a signature of a benign print. The blue upper line shows the signature of the Z axis, with the spikes corresponding to the transitions between layers. The orange line represents the difference between master signature and the signature of a benign print. The figure shows that the signatures are almost identical, with exception of sporadic anomalies. These can be caused by slight stochastic delays introduced by motors, propagate in the signature generation algorithm.

My approach takes this into account. To distinguish between stochastic anomalies and attacks, I use a *mismatchThreshold*. In the experiments, this threshold is 15 mismatched windows for the entire layer, or more than three consecutive mismatched windows; these values were set by adding a 20% margin, rounded up, above the largest deviations observed between unmodified print captures. With layers of objects in our test set having at least 180 sample windows, the worst case scenario sees less than 10% of a signature mismatch for a channel/time-interval $PCS_i^{\Delta t_j}$ accepted as valid.

Table 6.1 summarizes the results of the performed tests with modified prints. It shows that, in most cases, even a single toolpath command modification could be reliably detected. When only considering channel/time-interval signatures, the detection of layer transition modification was not possible. However, if such a modification impacts the duration of the layer transition,

it is detectable when the signatures for the entire prints were compared. The introduction of a smart void, i.e., replacing an extruding movement with an identical non-extruding movement, could not be detected with the proposed approach.

Figure 6.8 presents the specific example of a comparison between the master and tested signatures, when a single toolpath command was deleted. Figure 6.8a shows two signatures for the specific channel for a time-interval corresponding to a layer in which the command was deleted. The difference between the two (shown in Figure 6.8b) stretches both over three consecutive windows and also significantly affects more than 15 windows. Both are conditions that the algorithm uses to recognize attacks.

Based on these results, I can predict the detectability of various real attacks shown in the literature, as summarized in Table 6.2.

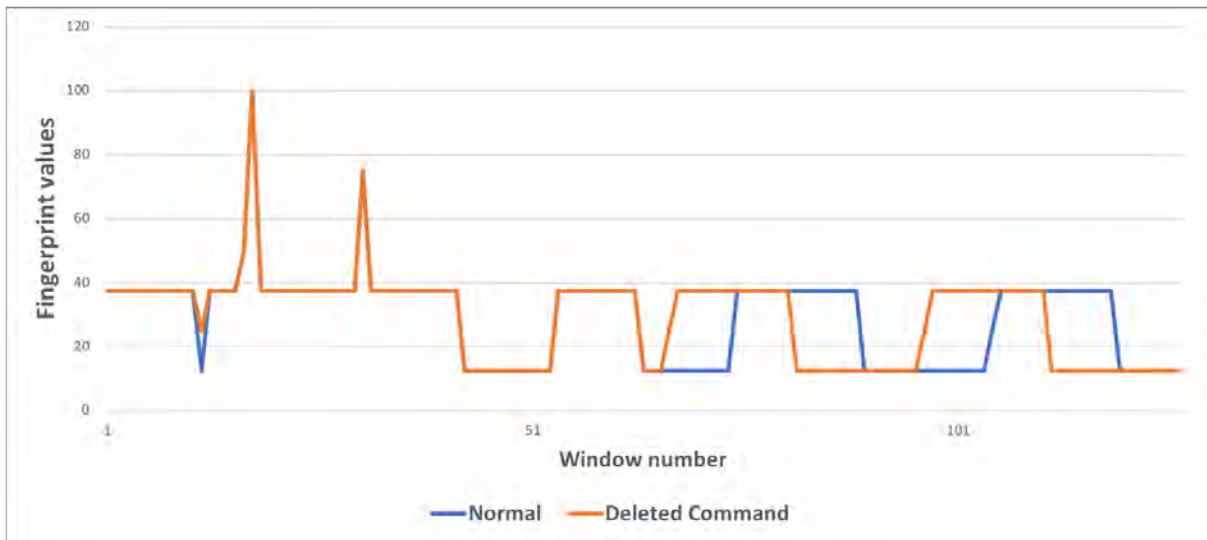
Insertion of gaps, as shown in Sturm et al. [54] and Belikovetsky et al. [10], modifies the G-code sequence and will be detected by the proposed algorithm. The same applies to sabotage attacks that scale up or down a printed object along one or more dimensions [72] since it impacts both size and the power consumption patterns.

If the layer thickness is changed for all (or many) layers [62], this will introduce cumulative delays. The resulting time offset between identical G-code commands in the master and tested signatures can be easily detected by comparing signatures over the entire print period.

The approach also reliably detects changes in the orientation of printed objects (as discussed in Yampolskiy et al. [77] and Zeltmann et al. [80]), including the orientation in the Z direction and in the X - Y plane because of significant differences in the G-code command sequences (and resulting motor movements) caused by the change in orientation.

Table 6.1: Detectability of *atomic modifications*

Level of Modification Detectability		
Modification	Entire Print	Single Layer
Insertion of Commands	✓	✓
Deletion of Commands	✓	✓
Command Reordering	✓	✓
Layer transition duration	✓	X
“Smart Voids”	X	X



(a) Master and a tested signatures



(b) Difference between master and tested signatures

Figure 6.8: Comparison of a single channel/time-interval signature in the case of a single deleted G-ode command

Table 6.2: Detectability of known *sabotage attacks*

SABOTAGE ATTACK	PROPOSED BY	DETECTABLE?
Gap/Void	[10,54]	✓
Contaminant Material	[80]	N/A
Different Layer Thickness	[62]	✓
Scale of the Printed Object	[72]	✓
Amount of Extruded Filament	[37]	✗
Z-Orientation	[77,80]	✓
Orientation in X-Y Plane	[80]	✓
Temperature of Extruded Filament	[72]	N/A
Fill Pattern modification	[8]	✓

✓ - Could Detect; ✗- Not instrumented; N/A - Not applicable

The detection method relies on synchronization and an accurate 3D printing process; thus, changes to infill patterns, such as those presented in [8], are noticeable and detected at the first occurrence.

6.5 Addressing Research Question #1

The research presented above in this section was in service of answering a particular research question, reproduced below.

Research Question 1: Does the power side channel provide enough information to detect sabotage attacks in AM?

Secondary to this question, I also asked:

- How are different categories of sabotage attack reflected in side-channel data?
- What analytical method is sufficient to detect this information in the side-channel?
- What are the thresholds of detection for the selected analytical approach?
- What factors limit the approach?

In order to answer this question, I developed a signature-based detection method for the actuator current side-channel, then trialled the method against a number of atomic-modification level sabotage attacks. The results of these trials indicated that the method could reliably detect deviations in all instrumented actuators. This supports that, for a given AM machine, a signature can be generated using the proposed method and as long as the same part has to be produced on this machine again, any deviations introduced by the sabotage attack can be detected.

These deviations, from command insertion, deletion, and modification, would allow the method to detect the great majority of published sabotage attacks (see Table 6.2). Each sabotage attack would be composed of different numbers of atomic modifications, in different sequences and proportions.

Three categories of attack were not addressed by the method, and can be explained as follows. The insertion of contaminant material, often into the feedstock, is a purely physical

attack and is not reflected in any way in the actuators. For filament extrusion and extrusion temperature, the deviations are reflected in the actuators, but in actuators that I did not instrument. The focus of this phase of research was purely on geometric deviations, which do not include these actuators.

According to the high level of detection accuracy demonstrated, I argue that Principal Component Analysis has proven a sufficient detection method for this side channel. Compared to the earlier application of PCA to sabotage detection using the acoustic side-channel [9], I have achieved higher detection accuracies and revealed more detail in the signature, allowing better localization of the attack.

For detection thresholds, I established an approach-specific threshold based on number of mismatched time windows within the signature. By my experimental results, this threshold was sufficient to detect every tested atomic modification except for single-layer Z-axis movement changes. This means that in the tested configuration, any deviation within a single layer can be detected, and changes in layer thickness can be detected across the duration of a print.

A number of limitations of the signature-based approach were exposed by the conducted experiments. The first, that the Z and E axis data had unusably low resolution, can be traced to configuration errors on those axes that are corrected by the updated instrumentation used in the follow-up research on reconstruction. This highlights the necessity of channel-specific instrumentation strategies. The lack of instrumentation on the extruder motor and thermal elements ruled out the detection of important classes of attacks. While instrumentation of the extruder motor does not differ remarkably from the other axis motors, the thermal elements operate on entirely different principals. They are driven by different types of control signals (in this case, true PWM instead of sinusoidal signals), and those control signals are integrated into a closed-loop feedback system. Addressing the instrumentation and analysis of closed-loop systems differs substantially from open-loop control systems like stepper motors, and must be addressed by future works (see Section 10).

6.6 Performance Analysis of Signature Methods

```
FINGERPRINTCREATION(signal, wnd_size, wnd_step)
    resampleRate = 400
    downsample = Resample(signal, resampleRate)
    finger_print = []
    frames = SplitToFrames(downsample, wnd_size, wnd_step)
    for each frame in frames
        fft_frame = FFT(frame)
        max_bin = Max_position(ABS(fft_frame))
        finger_print_value = (max_bin / wnd_size) * resampleRate
        finger_print.append(finger_print_value)
    return finger_print
```

Figure 6.9: Power-Trace Fingerprint Generation

For fingerprint generation, whose pseudocode is reproduced in Figure 6.9, execution time is dominated by three function calls, one of which occurs in a loop. On line 3, the signal is resampled to a lower rate. This involves selecting, and optionally linearly interpolating between, values in the original input signal array. Since we are downsampling, the number of operations is proportional to a fixed fraction of the input array, giving it linear time complexity. On line 5, the downsampled array is split into overlapping frames of a given size, with a given increment between window starting points. The worst case for number of operations would be an increment of 1, returning as many frames as there are samples in the array. Since this is bounded above by the downsampled array, which is itself bounded in length by the input array, the `SplitToFrames` call must also be bounded by a linear time complexity.

The most important call occurs on line 7, where the Fast Fourier Transform (FFT) is computed for each frame. Computing an FFT can be done in a wide variety of ways; in this case, the built-in MATLAB FFT uses the delightfully-named Fastest Fourier Transform in the West (FFTW) [20]. I will therefore use the time complexity derived by the authors, which is $O(n * \lg(n))$ for an input vector of length n , here the length of the frame. This occurs in a loop over every frame, but as the frames overlap the bound on frame length is slightly complicated. Frames are of length `wnd_size`, which is bounded by the distance of the last multiple of `wnd_step` less than the downsampled array length, and has a looser but simpler bound of the array length itself. The number of frames iterated over is the integer division of

wnd_step into the array length. Substituting this into the complexity for the FFT and adding a coefficient for the number of iterations produces

$$O\left(\frac{wnd_step * wnd_size}{n} * lg(wnd_size)\right) \quad (6.1)$$

Both parameters used can be set by the user based on their desired windowing, but in the degenerate case of one full-length window it reduces back to the original $O(n * lg(n))$. While this does depend on multiple input parameters, it is also clearly greater than linear, and will dominate the execution time of the algorithm for sufficiently large n .

```

ISSIMILAR(layerA , layerB , offset)
  mismatch = 0
  for(i = 0; i < len(layerA) - offset; ++i)
    if(layerA[i + offset] != layerB[i])
      mismatch++
  if(mismatch > mismatchThreshold)
    return false
  else
    return true

```

Figure 6.10: Layer comparison algorithm.

For fingerprint comparison, modeled by the pseudocode reproduced in Figure 6.10, the analysis is more straightforward. The generated signatures, composed of a sequence of principal frequencies segmented into printing layers, are compared to one another. The looping structure here is defined on line 3, and the comparison takes place on line 4. The comparison, and all other operations in this algorithm, take constant time—the only concern for time complexity is how many iterations occur. As the input here is a pair of outputs from the algorithm in Figure 6.9 above, the input size corresponds to the number of windows used, or $\frac{wnd_step}{n}$. While the execution time here again varies with a pair of inputs, one a user-configurable parameter, the comparison algorithm scales linearly.

Overall, the performance of the signature generation and comparison methods described in this chapter are likely not only sufficient but optimal. The dominant term of signature generation is from computing the FFT, an operation that has been optimized thoroughly by multiple generations of researchers; barring future breakthroughs, it isn't likely to be improved. Signature comparison not only scales linearly, and executes a basic comparison operation, but

has a helpful coefficient that can be manipulated by the user if enhanced performance is for some reason desired. Finally, the context of use for both algorithms is as a run-once signature method, which will most likely be executed once for the entire lifetime of a printed object. Unlike the work in subsequent chapters, there are no clear reasons why this method might need to be executed any faster than the present implementation achieves.

Chapter 7

Misuse: Stealing Technical Data

Outsourcing business models have flourished to address the need for on-demand AM; world leaders include ThyssenKrupp AG TechCenter (Germany) [57], Akhiani3D (South Africa) [1], RapidDirect (China) [45], and Treatstock (United States) [58]. While the above manufacturers are reputable, not all manufacturers will be trustworthy or safe from insider threats. A malicious AM service provider is in a unique position to steal technical data.

To address this threat, systems have been proposed offering Digital Rights Management (DRM), adapting traditional cyber-security for AM environments. In a recent patent, Oligsh-claeger et al. [40] propose to use cryptographic keys unique to a 3D printer for end-to-end encryption of designs. In a similar proposal by GE [24], technical data is decrypted by a Trusted Platform Module (TPM) (a dedicated security chip) integrated in the 3D printer, limiting access to *plaintext* (i.e., decrypted) data and cryptographic keys. Several companies, such as Identify3D, offer commercial options that implement DRM and encryption; their technology suite is claimed to provide “intellectual property protection, manufacturing repeatability, and traceability” to “unlock the potential of distributed manufacturing” [32].

Taken together, these individual developments constitute a trend towards securing outsourced manufacturing via DRM, which will further boost the already growing sector. I wish to examine a case where this technology is not only available, but has achieved its apogee; a cyber-security implementation that is comprehensive and faultless. Even in this case, I seek to demonstrate that pure cyber-security measures are not sufficient to protect IP against a malicious contract manufacturer.

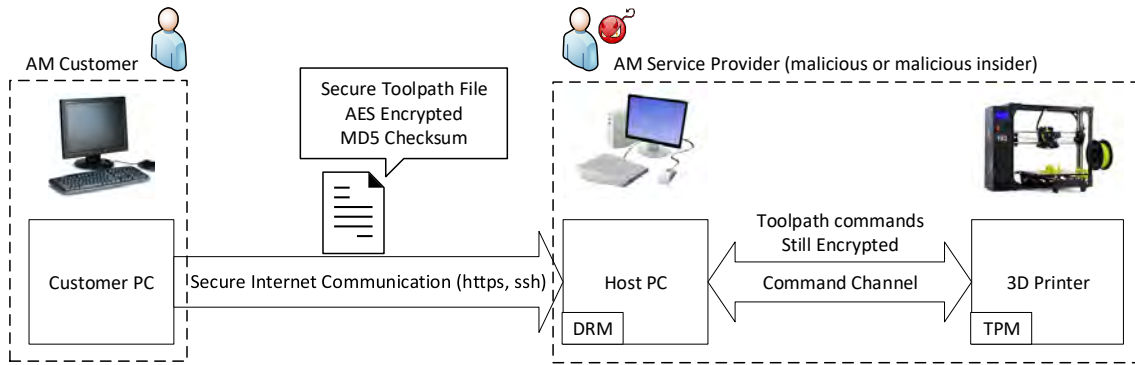


Figure 7.1: DRM-Protected Outsourcing to a Malicious AM Service Provider.

Threat Model: In this section, I assume a Man-at-the-End (MATE) threat model applied to outsourced AM. Comprehensive digital security measures are assumed to be in place to protect the customer’s data, such as end-to-end encryption of the model and a TPM module on the 3D printer (see Figure 7.1). Either the service provider itself is malicious, or has a malicious insider employee with some access to the AM equipment. The intent of the malicious party is to reconstruct the customer’s 3D printed model. Assume that they are unable to compromise the cyber-security measures, but are able to instrument the 3D printer at will. Potential implementations of the sensor suite are stealthy enough that a single insider could instrument a 3D printer and capture the side-channel data. This permits the attack against any printer an insider has even temporary physical access to. Under this threat model, I develop an approach to capture and reconstruct the 3D printed model that bypasses cyber-security measures entirely.

My novel approach to reconstruction utilizes a side-channel not yet used for this purpose, the actuator current. This side channel has the potential for perfect or near-perfect reconstruction and excellent portability to other classes of actuators. Achieving high levels of accuracy with the current side-channel will not only demonstrate the viability of this attack, but pave the way for defensive applications of reconstruction such as the forensics of Chapter 8.

My proposed method for reconstruction was developed based on the observed trace features described in Chapter 5. The first stage of development therefore involved much ad-hoc exploratory signal analysis and reference to the operational theory of stepper motors, also described in Chapter 5. As the method itself was developed, it was trialed repeatedly against the simpler models in the final model test set to identify bugs in the implementation and errors

in the approach's logic. The full reconstruction method is described below, followed by my experimental trials and results.

7.1 Reconstruction Method

At a high level, the reconstruction process consists of several stages. First, I preprocess the traces to filter noise. Then, I identify basic features of the trace such as peaks and periods of inactivity. Next, I attempt to map the features onto motor behavior, such as reversals. Because of the difficulties explained in Section 5.2.4, this stage generates a number of errors. The next stage applies a heuristic approach to correct these errors. Once a corrected sequence of features is finalized, they are used to track the position of the print head over time. During this process I distinguish between extruding and non-extruding moves to produce a point cloud corresponding to the printed figure.

I implemented this reconstruction approach in Python 3.2; I summarize the essential operation of the algorithm in pseudocode. Where the code relies on an external library for non-trivial functionality, I provide specific reference to the library and call.

7.1.1 Loading Oscilloscope Data

My reconstruction algorithm operates on the synchronized traces of the current delivered to each phase of each motor on the printer. I refer to the individual motor traces as the X, Y, Z, and E traces. The oscilloscopes capture the full waveforms corresponding to each axis; upon print completion, these readings are exported as a CSV file.

The algorithm receives CSV files of measured current values and corresponding timestamps, with the structure in Figure 7.2. The negative timestamps correspond to the values captured by the oscilloscopes before the trigger signal, at time 0s.

Time(s)	Channel A(A)	Channel B	Channel C	Channel D
-2.00050006	0.01537894	0.03644808	0.02383735	0.02752830
-2.00000006	0.01968504	0.03875492	0.02552903	0.02921998
-1.99950006	0.02399114	0.03875492	0.02614419	0.02552903
...				

Figure 7.2: Trace Capture in CSV Format (Excerpt).

Entries from the CSV files are read into `TraceEntry` data structures. The timestamp from the CSV file is read directly into the `time` field. The sample values of each motor's two phases are saved in `phase0` and `phase2`; the inverted values (negated) are saved in `phase1` and `phase3`, respectively. This duplication simplifies the peak recognition process in a later stage.

In addition to the timestamp and values, each entry is associated with an `axis` marker and a set of flags corresponding to motor behaviors. These flags are initially cleared; later stages of the algorithm will set them as appropriate. Descriptions of the flags are provided in the relevant sections.

The individual `traceEntry` elements are stored in sequential data structures, one for each axis. As the sizes of the datasets range from hundreds of thousands to hundreds of millions of these entries, I use the NumPy library's Array structure rather than standard Python datatypes.

7.1.2 Peak Detection

Later stages of the algorithm operate primarily on peaks and their timestamps, so I must first recognize the peaks from the trace.

The raw data has significant high-frequency noise. I first apply a low-pass filter to remove this noise, using the SciPy `butter` filter. The filter is applied with cutoff frequencies of 300 Hertz for the X-, Y-, and Z-axes, and 275Hz for the E-axis.

After applying the filter I use the `find_peaks` function from the SciPy Signals package [46] to identify the peaks, marking them by setting the `isPeak` flag in the corresponding `traceEntry`.

The parameters of this function, `height` and `prominence`, are applied differently per axis:

```
X: height=0.4 prominence=0.3
Y: height=0.4 prominence=0.3
Z: height=0.1 prominence=0.125
E: height=0.1 prominence=0.2
```

There are several errors that I observed in flagging peaks that must be handled before further processing. While peaks should not occur simultaneously, noise and level-shifting behavior means that there are sometimes simultaneous peaks on both phases of an axis. Even peaks that are only in very close proximity can cause issues. The first case is handled while

applying peak flags from *find_peaks* to the trace; if two phases attempt to flag the same timestamp as a peak, the phase with a higher peak is preserved. The proximity case is handled after all peaks have been flagged. If a peak occurs on a phase that is lower than another phase at the time, the algorithm checks if it is within a time threshold of the nearest peak- either within half of the average period between nearby peaks or a fixed threshold of 0.025s. If both hold true, then the low peak is removed.

7.1.3 Reversals and Error Correction

The largest issue to overcome in this algorithm is handling malformed segments of the trace. Because of the motors' internal structure, there are only two valid firing orders that they can perform, and I can only reconstruct motion that exhibits this behavior. The firing orders are associated with forward and backward movement, and the motor controller switches between them to execute a reversal. Figure 7.3 illustrates this behavior.

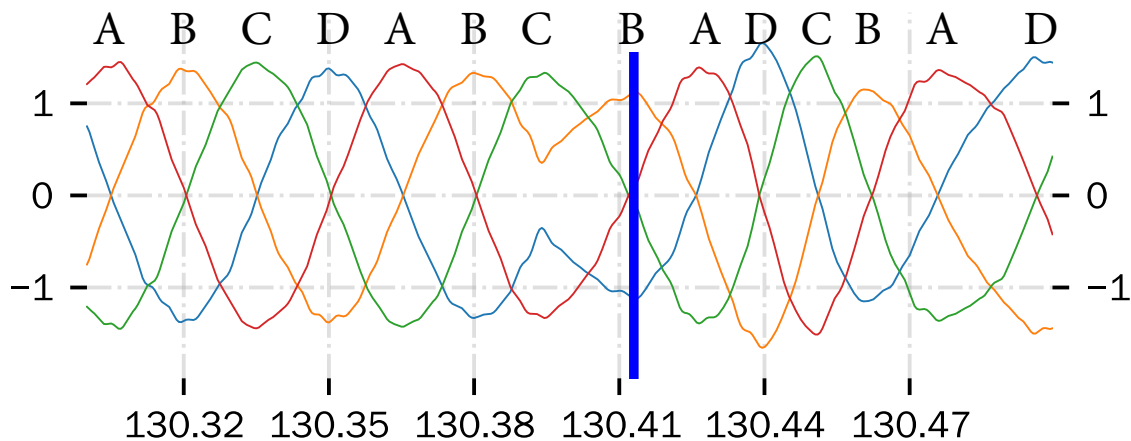


Figure 7.3: A reversal captured in the trace. Peaks are annotated A, B, C, or D to indicate the firing order. Note how the order reverses at 130.41s.

As a result of the noise and other difficulties discussed in Chapter 5, the trace contains both false-negative and false-positive peaks. My observation shows that many of these occur around reversals or dwells, or during periods of inactivity. Either type of invalid peak is incredibly likely to create an invalid firing order¹. Until these are corrected, it is not possible to accurately reconstruct motion. Further, if a section is recognized as correct but not accurate to the actual

¹Peak errors, much like parity error checking, always produce firing order errors in sections with only one bad peak. However, sections with two consecutive bad peaks can produce valid firing orders that are nonetheless not accurate to motor movements.

motor behavior, the reconstructed motion will drift: the estimate of printhead position will accumulate error based on the difference between reconstruction and reality.

I attempt to correct these errors by first segmenting the trace into sections with valid firing orders and sections with invalid firing orders. The valid sections are processed according to the normal logic of motor operation. For invalid sections, contextual information is gathered and saved alongside the section in a list. This list is processed by a heuristic solver, which attempts to find the best weighted reconstruction for that sections by adding and deleting peaks. The best solutions are applied to the trace.

7.1.4 Segmentation into Good and Bad Sections

With all peaks recognized, the approach next identifies the firing order on each axis. The initial synchronization of traces occurs on a long, linear, multi-axial move immediately before the print. This ensures there is a long enough section of consistent movement on every axis to recognize the firing order dynamically.

The pseudocode in Figure 7.4 details the process of identifying the firing order. Identifying the firing order from the data, rather than using a hard-coded order makes the algorithmic approach independent of the physical instrumentation, such as which wire is clamped by which probe. The firing order for an axis is “locked in” after the first 16 consistent peaks have been detected. This firing order is maintained for the remainder of the print, although it may be inverted when reversals occur.

```
axisPeaks = getPeaks(axis)
// Initialize firing order buffer
int FOBuffer[4] = axisPeaks[0:3]

predictionCount = 0
while predictionCount < 16:
    predictionIndex = nextPeakIndex % 4

    if axisPeaks[nextPeakIndex] == FOBuffer[predictionIndex]:
        predictionCount++
    else: // capture new firing order buffer
        FOBuffer = axisPeaks[nextPeakIndex-3:nextPeakIndex]
        predictionCount = 0

    nextPeakIndex++
```

Figure 7.4: Identifying Firing Order.

```

badOrderSections = []

while peakIndex < axisLength:

    nextPeak = getNextPeak(peakIndex)

    if nextPeak == predictForward(FOBuffer, peakIndex):

        if badSection != NULL:
            goodPeaksCounter++
            if goodPeaksCount > goodPeaksNeeded:
                // Close out bad section.
                endBound = FindEndOfBadSection()
                SaveBadSectionInfo(originalFOBuffer, mustReverse, startBound, endBound)
                badOrderSections.append(badSection)
                badSection = NULL

    elif nextPeak == predictReversal(FOBuffer, peakIndex):
        reverseBuffer(FOBuffer)

        if badSection == NULL:
            flagReversal(peakIndex)
        else:
            // Toggle the mustReverse flag
            mustReverse = !mustReverse
            // Good reversals also increment the counter
            goodPeaksCounter++
            // Accumulated enough peaks to end bad section?
            if goodPeaksCount > goodPeaksNeeded:
                // Close out bad section.
                endBound = FindEndOfBadSection()
                SaveBadSectionInfo(originalFOBuffer, mustReverse, startBound, endBound)
                badOrderSections.append(badSection)
                badSection = NULL

        else:
            // Found a bad peak. Begin looking for boundaries.
            if badSection != NULL:
                badSection = createNewBadSection(peakIndex)

            // Any bad peak resets the counter and FOBuffer.
            goodPeaksCounter = 0
            shiftFOBufferToCurrentPeak(peakIndex)

    peakIndex += 1

return badOrderSections

```

Figure 7.5: Identifying Good and Bad Sections.

Next, the algorithm scans across the list of peaks, comparing each in sequence to the firing order. It recognizes three cases: valid forward motion, valid reverse motion, and invalid firing orders. The pseudocode in Figure 7.5 outlines the process. As in the trace diagram of Figure 7.3, if the last peak to fire was, for example, B, then the next peak can be valid if it is on phases A or C. One phase represents forward motion, the other represents a reversal, depending on the history of the firing order going into that peak. Firing on a D peak immediately after a B peak would be an error, since these peaks are nonadjacent.

When two nonadjacent peaks are processed in series, the algorithm treats this as the beginning of an invalid section. Next, it searches for the end of the section, which occurs when any valid sequence is detected for three peaks in a row. Three peaks are necessary because the correction process (handled in the next stage) can require changes as far as three peaks after the last invalid section. Since there is no way of knowing how many valid peaks have occurred since the beginning of the invalid section, I must build up the detected firing sequence one peak at a time, determining for each peak if it is part of a potentially valid firing order. I achieve this by circularly shifting the firing order so that the current peak's phase is in the starting position, then compare the following peaks to the forward and reverse phases of the new firing order. Because there might be reversals in this sequence of three peaks, it may be necessary to invert the firing order again on the second valid peak.

When the end of a bad section is detected, the algorithm saves the endpoints and contextual information of the section to a list. The contextual information contains whether the firing order reversed during the section, the timestamps at the start and end of the section, as well as on which axis the section occurred. The algorithm may then resume normal behavior and continue detecting reversals. This continues until all peaks in the axis are processed. The result is a partially annotated list of peaks and a list of invalid sections. This process is repeated for each axis.

7.1.5 Heuristic Correction of Bad Sections

The invalid sections produced by the previous stage are processed individually in this stage. I recognize two forms of error: missing peaks and duplicate peaks. Missing peaks are flagged whenever two nonadjacent phases are detected in sequence; duplicate peaks are flagged when the same phase is detected twice in a row. Badly ordered sections can contain any number of badly ordered pairs, and I have observed up to 20 in a single section in the test set. Each badly ordered pair has multiple potentially valid corrections: 4 for missing peaks and 3 for duplicate peaks. This produces a large problem space, and the available data does not clearly identify the correct solution. Therefore, I have developed a heuristic approach to search the problem space and identify the solution that best fits a defined set of metrics.

Table 7.1: Solutions considered for badly ordered pairs and their effects on firing order. The beginning *ABCD* exemplifies a possible firing order prior to the badly ordered pair (indicated in red and underlined).

DETECTED	CORRECTED	SOLUTION
ABCD <u>AC</u>	ABCD <u>ABC</u>	Insert Forward Peak
	ABCD <u>ADC</u>	Insert Reverse Peak
	ABCD <u>C</u>	Delete First Peak
	ABCD <u>A</u>	Delete Second Peak
ABCD <u>AA</u>	ABCD <u>ABA</u>	Insert Forward Peak
	ABCD <u>ADA</u>	Insert Reverse Peak
	ABCD <u>A</u>	Delete Peak

Correcting a badly ordered section involves either deleting the first or second peak of a badly ordered pair, or inserting a new peak in between them. The inserted peak can be on either the forward phase or the reverse phase. These four changes represent all of the corrections the heuristic solver considers. All four apply to missing peak pairs. For duplicate peak pairs, deleting the first and deleting the second peak are functionally the same choice with respect to the firing order. Therefore, duplicate peak pairs have only the three solutions of deletion, insertion of a forward peak, and insertion of a reverse peak. This is illustrated in Table 7.1.

The solutions themselves can be compared on the quality of the peaks inserted or deleted. I do this by defining a cost function for each of the solutions that considers the height, prominence, proximity to other peaks, and several other characteristics of the peaks to insert or delete. The insertion cost functions operate on the segment of a trace in between the badly ordered peaks, searching for the best point to insert a new peak on the target phase. The deletion functions evaluate the first and second peaks of the badly ordered pair to determine which is more likely to be a false positive. Both use a relatively complex set of metrics and weightings, which I tuned by gradually identifying error cases in the test set and creating general metrics to correct them.

In addition to the heuristic costs for individual pairs, the solution must consider the entire section and its context to ensure the firing order is still valid. For example: consider the sequence ABCDACD. The badly ordered pair, AC, is identified as a missing peak pair and can be repaired by any of four solutions. However, in the context of the surrounding sequence, different choices could produce an additional reversal, multiple reversals, or none at all. Based


```

// Container Function
def badSectionSolverContainer(badSections)
  for section in badSections:
    // each solution documents the bad section and peaks
    // solutions can be applied to correct the section
    solutions = []

    badPeakIndex = 0
    while badPeakIndex < len(section):

      if missingPeakError(badPeakIndex)
        solutions.append(missingSolutions(badPeakIndex))

      elif duplicatePeakError(badPeakIndex)
        solutions.append(duplicateSolutions(badPeakIndex))

      badPeakIndex++

    bestSolution, bestCost = searchSolutions(solutions, 0)
    applySolutionToTrace(bestSolution)

```

Figure 7.6: Heuristic Solver Container.

on the surrounding sequences, a badly ordered section should contain either an even or odd number of reversals; this is taken into account by the solver, which discards solutions that do not satisfy this constraint.

Pseudocode summarizing the heuristic process is presented in Figures 7.6 and 7.7. The solver consists of two portions, a non-recursive container and a recursive search function. The non-recursive container iterates over all badly ordered sections for each axis. The recursive search explores the solution space for each badly ordered section, eventually returning the solution with the lowest cost. The container then applies this solution to the trace, annotating the appropriate peaks with reversals. After the list of badly ordered sections is exhausted, the trace is fully annotated and can be used to produce a point cloud reconstruction of the 3D printed object.

7.1.6 Point Cloud Generation

Upon completion of the heuristic search and correction process, the trace contains valid time-stamped peaks on the X-, Y-, Z-, and E-axes, all properly tagged wherever a reversal occurs. Although the firing order is now fully correct, there may still be discrepancies between the

```

def searchSolutions(solutions , cost)

    if len(solutions) == 0:
        // We've arrived at a leaf node in the search

        tempFixedPeaks = applySolutionToTrace(solution)
        // Applying solution makes a list of peaks
        // We process these with the same loop in Fig.16
        reversed = countReversals(tempFixedPeaks)

        if reversed % 2 == mustReverse
            // The solution is acceptable; return its cost
            return solution , cost
        else:
            // The solution doesn't match reversal behavior
            // Return infinite cost
            return solution , cost=Inf

    for peakPair in solutions:
        if peakPair.errorType == "missing":
            FWCost = insertCost(peakPair , forward)
            tempSolution = takeBranch(solutions , insertForward)
            FWSolution , FWCost = searchSolutions(tempSolution , cost + FWCost)

            RVCost = insertCost(peakPair , reverse)
            tempSolution = takeBranch(solutions , insertReverse)
            RVSolution , RVCost = searchSolutions(tempSolution , cost + RVCost)

            DCCost = deleteCost(peakPair , current)
            tempSolution = takeBranch(solutions , deleteCurrent)
            DCSolution , DCCost = searchSolutions(tempSolution , cost + DCCost)

            DNCost = deleteCost(peakPair , next)
            tempSolution = takeBranch(solutions , deleteNext)
            DNSolution , DNCost = searchSolutions(tempSolution , cost + DNCost)

            bestCost = minCost(FWCost, RVCost, DCCost, DNCost)
            if bestCost == FWCost
                bestSolution = FWSolution
            elif bestCost == RVCost
                bestSolution = RVSolution
            elif bestCost == DCCost
                bestSolution = DCSolution
            elif bestCost == DNCost
                bestSolution = DNSolution

            return bestSolution , bestCost

    elif peakPair.errorType == "duplicate"
        // Duplicate peaks are handled as above

```

Figure 7.7: Heuristic Recursive Solver.

listed peaks and the real behavior of the printer. In the final stage, I implemented a state machine to track the position and behavior of the print head and translate its motions into a point cloud, with each entry an (X, Y, Z) tuple.

The state machine is initialized with a starting position of (0, 0, 0), then begins to iterate through the peaks in timestamp order. Processing a peak updates the position of the printhead based on the axis and movement direction of the peak. The E-axis is tracked to determine whether a new position is a part of an extruding or non-extruding move. Dwell annotations on the E-axis peaks mark the beginning and end for periods of extrusion. If the E-axis is inactive, the position is updated but not saved to the point cloud. After every peak is consumed, the output from the state machine is the full point cloud representation of the print. By default, it is in units of discrete motor steps; I convert to millimeters to compare the reconstructions to the original models more easily. I used per-axis steps to millimeter ratios of: 6 : 1 for X, 6 : 1 for Y, and 95.2 : 1 for Z. The E-axis is used only to track extrusion and non-extrusion, and does not need to be converted. The point cloud can be exported as a *.XYZ* file, a common format for point clouds and meshes, and manipulated by external modeling programs.

7.2 Experimental Evaluation

For these experiments, I selected a range of models that vary in size and geometric complexity. The simplest is a cube, 10mm on a side, which prints entirely in linear movements (G0 or G1 G-code commands). The remaining figures contain both linear and non-linear movements such as arcs and splines (G2, G3, and G5 G-code commands) and complex surfaces. The models are rendered in Figure 7.8.

These models are sliced² in Cura Lulzbot Edition 3.6.20 for printing on the Lulzbot Taz 6. The slicer settings are consistent across all models. I use an infill parameter of 20 % with 45 degree rotation and the default behavior of thicker base layers and full infill for top and bottom layers. I am therefore reconstructing realistic sliced and printed models, rather than handmade or simplified print patterns.

²Slicer is a common term for the software used to generate a printer-executable toolpath from a model file and printer configuration parameters, controlling elements such as infill patterns and support structures.



(a) Cube



(b) Ninja Star



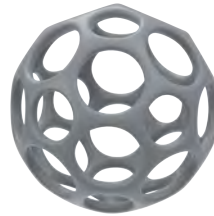
(c) Wrench



(d) Rook



(e) Gear



(f) Bucky Ball



(g) Octopus



(h) Turbine Blade



(i) Stanford Bunny



(j) Stanford Lucy

Figure 7.8: Original STL models used in the experiment.

7.3 Results

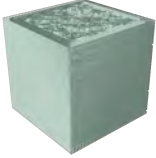




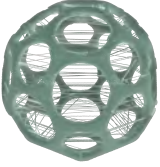

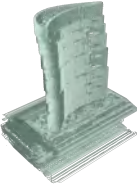
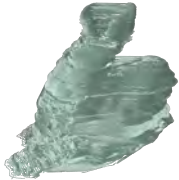

The results of the reconstruction process are given in Table 7.2. Simpler and shorter models such as the Cube and Ninja Star are reconstructed perfectly, while some of the more complex models such as the Stanford Bunny exhibit visible misalignments across layers. Most models are only moderately distorted and remain recognizable. A human engineer could correct the errors and achieve the quality of the simple objects.

Quantifying these errors and determining their meaningfulness has presented a novel challenge not appropriately addressed in the literature. Chhetri et al. [12], who first demonstrated an acoustic side-channel attack on AM systems, quantified their reconstruction using average percentage of axis recognition and average movement distance error. They achieved 78.35% accurate axis recognition and 17.82% movement distance error for their test prints, which were zero-infill perimeter polygons.

Using these distance-oriented metrics, I achieved 100% axis recognition (because all individual motors were instrumented). Across all tested models, I can calculate total distance error as $bad_section_steps/total_steps$. This produces a maximum error of 1.7% for the Rook model, and a minimum of 0.07 % for the Wrench. On average, the error was 0.79%. This figure represents an upper bound of error, because not every step within a bad section is incorrect.

Gao et al. [21] evaluated their reconstruction of the infill path using a metric derived from the Hausdorff Distance. Their approach locates the point in the reconstructed path that is most distant from any point in the original path, and reports that distance as an upper bound on error. However, I believe all approaches to date for measuring accuracy have limited use in the context of AM reconstruction.

Table 7.2: Point cloud renderings and metrics of the reconstructed models. Any support structure is included in the rendering.

METRICS		RENDER	METRICS		RENDER
Name:	Cube		Name:	Ninja Star	
Print Duration:	13.63 min		Print Duration:	4.48 min	
Steps Traveled:	98,098		Steps Traveled:	65,534	
Points in Cloud:	80,196		Points in Cloud:	49,297	
Sections:	2,344		Sections:	1,384	
Bad Sections:	351		Bad Sections:	349	
B.S. Max. Length:	2		B.S. Max. Length:	4	
B.S. Avg. Length:	1.00	B.S. Avg. Length:	1.1		
Name:	Wrench		Name:	Rook	
Print Duration:	44.33 min		Print Duration:	49.98 min	
Steps Traveled:	774,063		Steps Traveled:	429,117	
Points in Cloud:	563,305		Points in Cloud:	340,909	
Sections:	21,445		Sections:	23,735	
Bad Sections:	5,334		Bad Sections:	6,497	
B.S. Max. Length:	7		B.S. Max. Length:	9	
B.S. Avg. Length:	1.05	B.S. Avg. Length:	1.15		
Name:	Gear		Name:	Bucky Ball	
Print Duration:	50 min		Print Duration:	154 min	
Steps Traveled:	728,078		Steps Traveled:	1,731,428	
Points in Cloud:	611,807		Points in Cloud:	1,126,327	
Sections:	25,893		Sections:	68,796	
Bad Sections:	5,572		Bad Sections:	21,269	
B.S. Max. Length:	6		B.S. Max. Length:	7	
B.S. Avg. Length:	1.15	B.S. Avg. Length:	1.09		
Name:	Octopus		Name:	Turbine Blade	
Print Duration:	66.58 min		Print Duration:	84.98 min	
Steps Traveled:	959,332		Steps Traveled:	879,667	
Points in Cloud:	715,662		Points in Cloud:	629,200	
Sections:	19,375		Sections:	25,192	
Bad Sections:	5,684		Bad Sections:	8,386	
B.S. Max. Length:	7		B.S. Max. Length:	5	
B.S. Avg. Length:	1.08	B.S. Avg. Length:	1.07		
Name:	Stan. Bunny		Name:	Stan. Lucy	
Print Duration:	302 min		Print Duration:	242 min	
Steps Traveled:	5,629,158		Steps Traveled:	3,827,019	
Points in Cloud:	4,490,563		Points in Cloud:	2,578,250	
Sections:	82,030		Sections:	98,730	
Bad Sections:	23,167		Bad Sections:	29,193	
B.S. Max. Length:	11		B.S. Max. Length:	7	
B.S. Avg. Length:	1.10	B.S. Avg. Length:	1.13		

7.4 Fundamental Limitations

While my approach achieved significantly higher accuracy on more complex models than prior reconstruction attempts, it still contains uncorrected errors. I suspect that some can be corrected, while others potentially cannot — they may be fundamental to the approach. The most fundamental limitations are discussed below.

The most interesting limitation arises from the disconnect between the toolpath motions and the true shape of the printed part. When the printhead is positioned, the nozzle is placed slightly above the extrusion position to leave room for filament to extrude and attach without collisions. The extruded filament also has a diameter, which changes shape while it solidifies. During printing motions, the elasticity of the molten filament means that it will continue to extrude after the extruder motor stops moving. When planning a toolpath, the slicer attempts to account for these and other physical characteristics of the material and printing system. Many of these compensation techniques are adjusted by the user, vary according to printer and settings, and are largely opaque to side-channel analysis. Producing a mesh directly from the reconstructed point cloud will produce a smaller mesh than the original because of these offsets, and it will contain small gaps due to flow-rate manipulation.

Another key limitation is that the approach cannot distinguish between body and supporting printed material in single-extruder systems such as this. Many objects must be printed with support material to allow for large overhangs and prevent print defects such as sagging or non-adhesion. Given variation in support structure design and use, recognizing it automatically based on geometric characteristics would appear to be a difficult proposition.

Most importantly, it is difficult to estimate how much drift is possible in a reconstruction, and what amount of drift will render a print unusable. Smaller models, such as the Cube and Star, show no drift, but even the larger test set used in this dissertation is not sufficient to tell if this is because they contain fewer steps, or because they do not contain more complex motions that are incorrectly reconstructed in the larger models. Even small amounts of drift can be fatal when minute details are essential to the print: the reconstructed Rook, for example, contains

large and blocky geometries, and would likely print without issue. The reconstructed Octopus has fine detail in the upraised hand, and the reconstructed print would fail there even with only a few steps of drift.

These key limitations are based on the characteristics of the 3D printer, filament, and options available in the slicer. Undoubtedly, further study will uncover more limitations but also the means to overcome them. Improved signal processing, for example, might more accurately detect peaks in the waveform and not only eliminate drift, but the render the entire heuristic portion of the solution unnecessary. The techniques necessary to evaluate and compensate for these are beyond the scope of the present work.

7.5 Addressing Research Question #2

The research presented above in this section was in service of answering a particular research question, reproduced below.

Research Question 2: Can the power side-channel be used to reconstruct a printed object as a method to bypass Digital Rights Management (DRM)?

How can a reconstructed object be represented? Given this representation, how can the accuracy of a reconstruction be measured from it? What accuracy of reconstruction is achievable from the power side-channel alone? What are the factors limiting the accuracy of the reconstruction?

The answer to Research Question 2 is a resounding yes. My reconstruction method, based on a temporal analysis of current trace features, is able to extract the geometry of printed objects with a high level of precision and accuracy. Persistent errors in reconstruction mean that positional drift can propagate through the rest of the print. Subsequent errors can either accumulate in one direction or compensate, and cause random variation about the original point. Even for the largest models printable on the tested printer, the results are recognizable and a human technician could repair and reproduce the model.

To assess the accuracy of the reconstruction, I derived a method-specific measure to place an upper bound on the number of step reconstruction failures. In this system, I report the total number of failed sections where heuristics had to be applied, along with the total number of

steps covered by the section. In many cases, these sections are correctly repaired—the Cube model, for example, contains 351 bad sections but no visible drift. However, reporting the maximum error is the most reasonable approach, since the actual error cannot be detected automatically. All bad sections are marked, and could therefore be manually corrected by a human technician; in larger models with significant error, the number of bad sections may make this extremely labor intensive.

Across the tested models, an upper bound of 0.79% of the total printed steps could have been reconstructed incorrectly. As shown in Table 7.2, the actual error was significantly smaller for many models. Even with these impressive statistics, larger and more complex models are subject to positional drift that would require manual repair before printing. This drift, which comes primarily from the limitations of the implemented signal analysis for peak detection in complex waveforms, points to the inability of accuracy metrics to account for printability in AM.

Through the one-to-one relationship between trace peaks and discrete motor steps, I arrived at a temporally ordered point-cloud representation for the reconstructed object. This representation allows for the maximum degree of accuracy achievable by the printer and is usable with a wide variety of mesh-manipulation toolsets, including those used to generate an STL. While other reconstruction representations are possible, I argue that this one captures both initial model geometry and the sequence of filament deposition in a useful way. Not only does it fully satisfy the requirements for Technical Data Theft, it is suitable for the work on Research Question 3, regarding the forensic examination of sabotage.

7.6 Performance Evaluation of Reconstruction

Once a novel problem has a solution, it is prudent to consider the solution's performance. In the context of this work, there are both baseline and contextual demands on performance. All of the baseline performance needs have been met over the course of dissertation, however, if this approach is to be usable in industrial settings, I must project even greater demands on its performance, which requires developing an understanding of how its performance scales.

Space complexity analysis must be considered in any performance analysis. Owing to software architectural decisions in the code base, only two data structures (the npTrace and output point cloud) scale with the size of the input data. Only one instance of npTrace exists, and all work on it is done by reference; the same is true of the point cloud. The number of entries in npTrace is directly proportional to the input trace, as it is the in-memory representation of the trace. The size of the point cloud is not knowable before computation, but is absolutely bounded above by the size of the trace—there cannot be any more points generated than the number of peaks in the input, and there cannot be more peaks than samples. Therefore, space complexity grows linearly with the input data, which itself is determined by the duration of the print, the sample rate, and the sample resolution, all also linear. The only clear bound here is that, at a certain point, the trace becomes too large to keep in working memory. In very extreme cases, the entire dataset may be too large to store in a single computer's secondary storage. Neither of these limits were encountered in the test set of models, but it is possible that prints conducted with other technologies would need to address these issues.

My approach to time performance analysis will be as follows. First, I will describe the major modules used in my code; these are responsible for logically-divided steps in the process of reconstruction. Each, I expect, will have a particular dominant portion which determines its time complexity. The second step will be to estimate these complexity classes by identifying loops, recursive steps, and calls to external functions with known complexity. Finally, I will profile the time performance of my code across the test set and perform a regression to compare the results to my estimates.

7.6.1 Loading (readCSV)

The readCSV function is composed of a pair of loops, with the first bounded by the size of the input CSV file and the second by the corresponding size of the resulting npTrace object. The primary action of the first loop is to read a line from the file and distribute its contents to the next entry in the npTrace. The second loop simply iterates over the full npTrace, assigning

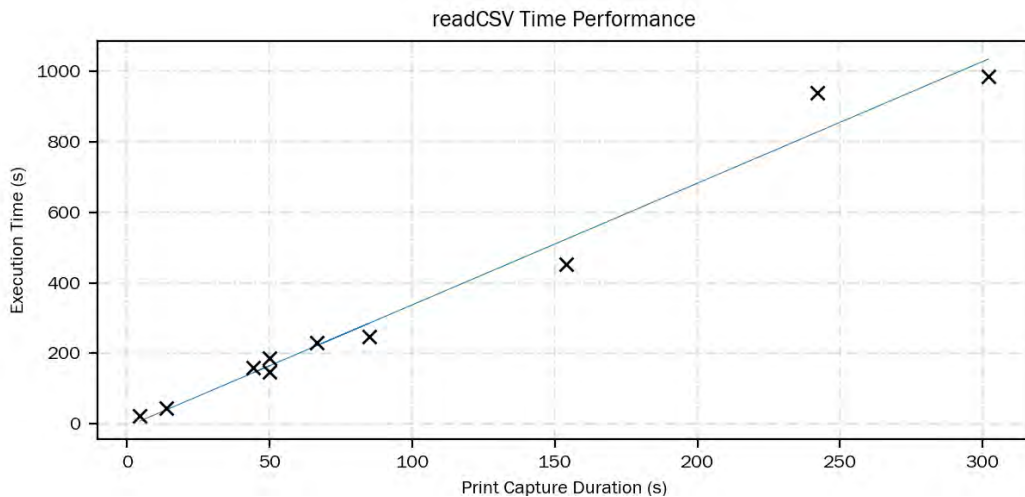


Figure 7.9: Execution time in seconds for the readCSV module, plotted against the duration of the print capture. The plotted line shows a linear regression between the two, with error of 0.177 standard deviations.

consecutive key values to entries. Operations within the loop are of fixed runtime, e.g., assignments and primitive math operations. Accordingly, time complexity should here be linear against the input CSV file size, which corresponds to the total duration of the input trace.

Figure 7.9 plots the runtimes for readCSV against trace duration for the test set. The plotted linear regression resulted in an error of 0.177 standard deviations, supporting my theory of a linear time complexity for this module.

For the test set, the initial loading process (readCSV) is by far the most time consuming module. I have offset this problem by serializing the numPy-formatted version of the trace (as it will be operated on by the remainder of the code) to a file; loading and deserializing the object from this file is much faster than processing the CSV generated by the oscilloscope. As a result, readCSV is still a time-consuming operation, but it only needs to be run once for the entire working lifetime of that trace capture. Looking forward, any prospective runtime operation of this code must similarly ensure it captures data rapidly from the oscilloscope, and renders it into a suitable format as quickly as possible. Since this strategy of saving a full trace in a compatible format cannot be reused for a runtime deployment, a novel solution would be needed.

7.6.2 Finding Peaks (findPeaks)

Moving on to data processing, the `findPeaks` function represents the next-largest contribution to total runtime. It relies, at its core, on the `find_peaks()` function from the `scipy` signals package for python. This implementation performs neighbor-by-neighbor comparison to locate extrema, which are then filtered according to requirements on peak height and peak prominence. In addition to `find_peaks`, this module applies a Butterworth low-pass filter and performs several slicing and assignment operations on the trace. There are no loops or recursive steps in the code as-written, as most of the work in this module is being done within library calls.

Slicing and assignment are both expected to have linear time complexity against the trace size. The operations, which separate the phase data into individual arrays and assigns low-pass-filtered values back to the trace, apply to each entry in constant time.

As with the other signal package functions, `signal.find_peaks()` does not have a known time complexity. Nearest-neighbor peak finding approaches in general operate by traversing the entire input array, comparing each entry against the previous and subsequent. As such I expect it to be of linear time complexity against the number of samples, i.e., the capture duration. The Butterworth filter is applied using the `signal.butter()` and `signal.filtfilt()` functions, also of undocumented time complexity.

I expect the combination of these two library calls to dominate the execution time of this module. Should the time complexity of these calls be linear, the module as a whole will also show linear time complexity. Figure 7.10 plots the execution time of `findPeaks` against the capture duration. A linear regression between the two produced an error of 0.09 standard deviations supporting my assumption.

7.6.3 Heuristic Solver (searchSolutions)

The heuristic solver is implemented using a recursive function `searchSolutions()`, which is called on each potential repair to the bad section to traverse the solution space; once a potential solution is found, the recursive calls “unwind”, returning the weighted cost of that repair for comparison. This is done iteratively for each bad section identified in the print; solutions

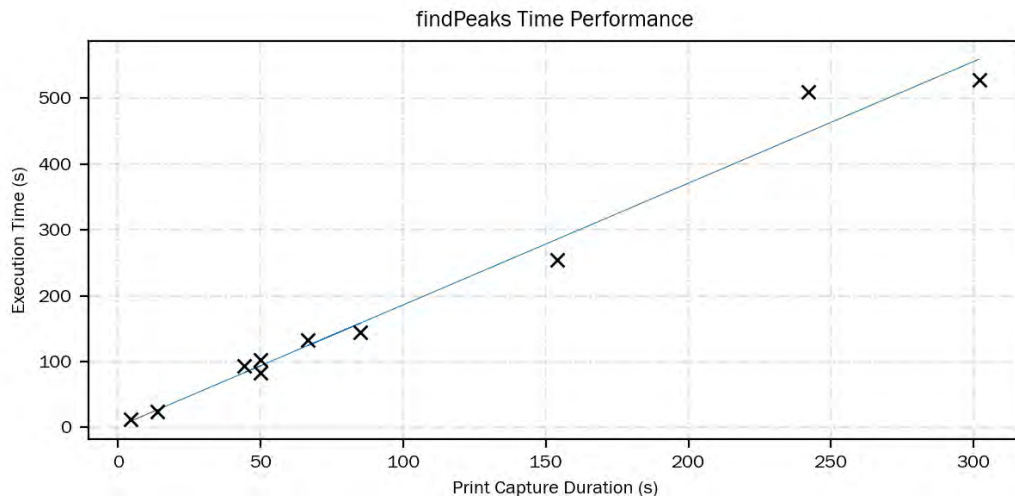


Figure 7.10: Execution time in seconds for the findPeaks module, plotted against the duration of the print capture. The plotted line shows a linear regression between the two, with error of 0.09 standard deviations.

to one section have no effect on other sections. The precise mechanics, weighting of different solutions, and logic used are described in Chapter 7; fortunately, they do not have an impact on the total running time of the module.

The structure of recursion and iteration used in this module is fairly complex. At the top level, the solver iterates over a list of all identified bad sections; this is the first variable determining total execution time. For each bad section, there is a pre-processing loop over each peak that identifies the type of error that occurred; this is expected to contribute another small linear component proportional to bad section length. Then, in the recursive portion, the solver begins to generate a solution set using the finite list of possible corrections for each error type. This list includes either 4 possible fixes, for missing peak errors, or 3, for duplicate peak errors. The composition of any given bad section is probabilistic; there is no way to know how many errors of which type are included until it is processed. For this recursive portion, given a bad section of length N peaks, execution time will be bounded above by 4^N in the worst case, and below by 3^N in the best case.

For this module, then, there are two independent variables influencing execution time: number of bad sections, and the distribution of lengths of bad section. As the input size grows (for this module, as the bad sections grow longer and more numerous), I expect the higher-order complexity to dominate execution time. In this case, that would be the 4^N complexity of

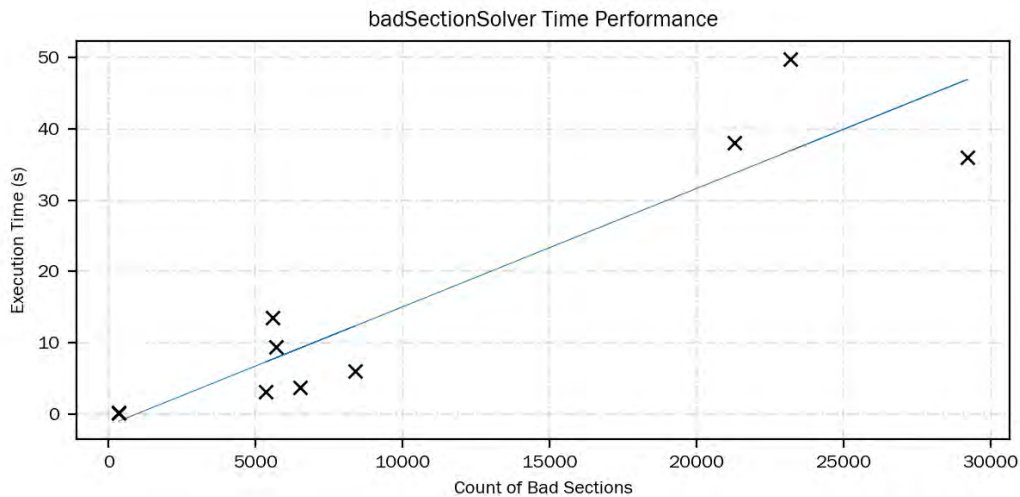


Figure 7.11: Execution time in seconds for the heuristicSolver module, plotted against the count of bad sections. The plotted line shows a linear regression between the two. While this result is lower than with other modules, the trend is still reasonably linear in this region. The artificial bad section with length 15 is omitted from this plot and the linear regression.

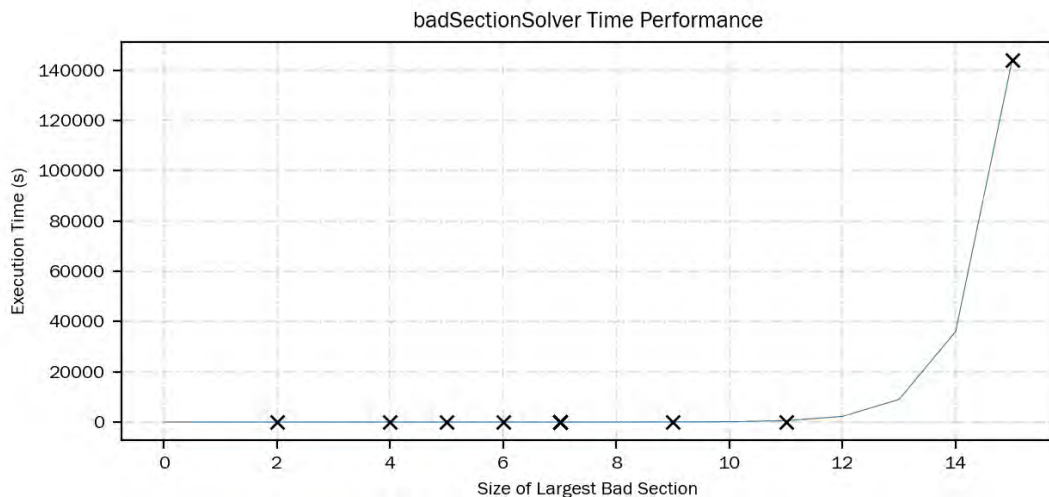


Figure 7.12: Execution time in seconds for the heuristicSolver module, plotted against the largest bad section in each trace. The plotted line shows a regression to 4^N between the two, with error of $1.6e-07$ standard deviations. The rightmost tick mark, with length 15, indicates the results of running this module alone on an artificial bad section.

the recursive step. Figures 7.11 and 7.12 plot the execution time for this module against both variables, with regressions.

The covariance of the linear regression in Figure 7.11, for execution time against number of bad sections, is lower than linear regressions for other modules. This is to be expected, given there is a second major contributor which is non-linear. However, this data alone also

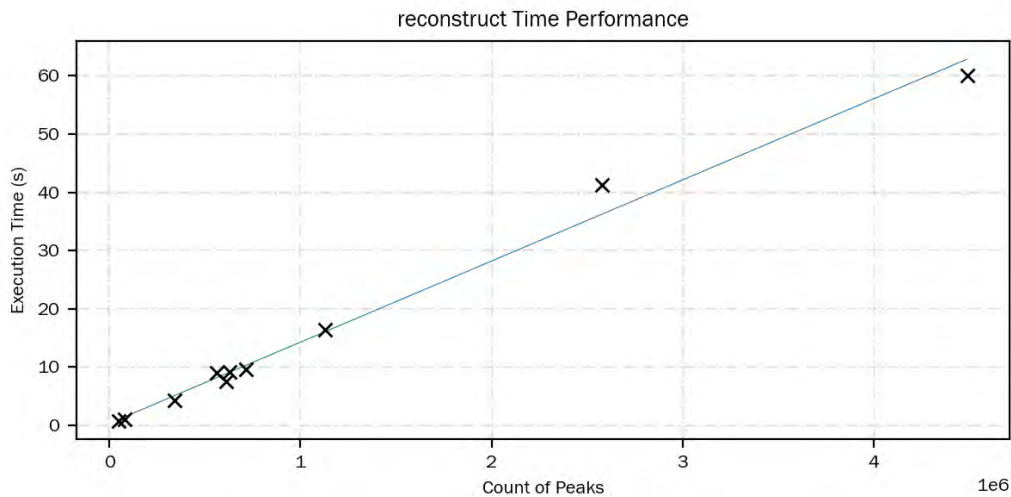


Figure 7.13: Execution time in seconds for the reconstruct module, plotted against the number of peaks in the trace. The plotted line shows a linear regression between the two, with an error of $5.18e-07$ standard deviations.

does not appear to exhibit a 4^N curve for execution time. I believe that this range of input data corresponds to the period before the exponential terms starts to dominate.

To test this assumption, I generated an artificial bad section of length 15, greater than the maximum of 11 which appeared in the test set; this point is included in Figure 7.12. Processing this single bad section took no less than 40 hours on the same hardware as all other tests. This suggests very strongly that the complexity is much greater than linear, and that the ‘knee’ of the curve exists between inputs of length 11 and 15.

7.6.4 Point Cloud Reconstruction (reconstruct)

The final module in the reconstruction package is implemented as a state machine; it takes as input the sequence of all peaks in the trace, yielding a new point in the cloud when the input sequence corresponds to the print-head extruding at a new position. Roughly the same amount of work in modifying internal state must be done regardless of whether a peak occurs during an extruding move or not; the only difference is in constructing and returning a peak object. This means that the operation of the module is a single loop, bounded by the list of peaks, which does a nearly constant amount of work each iteration. I therefore expect a time complexity which is linear proportional to the number of peaks. Figure 7.13 plots the execution time for this module.

Chapter 8

Use: Forensic Examination of Sabotage

Research Question 3 asks if the actuator current side-channel provides enough information to localize and investigate a sabotage attack. Owing to the results of Research Question 1, I can say that the channel certainly contains enough information to identify a sabotage attack, and even to localize it to within a single layer. This alone is a significant improvement over detection only, but is not quite sufficient for investigation.

From a signature-based detection method, one cannot reason about the impact of a sabotage attack. Generating a signature generally reduces complex information about the model down to a handful of metrics, making a very lossy process. In addition, the early state of development for AM sabotage ‘in the wild’ means that no one, at this moment, can predict what real-world attacks may look like; even in the research literature, novel attacks are still being published as late as 2022 [18, 82]. Ideally, the altered geometry could be inspected directly and compared against the expected geometry.

My work on Research Question 2 provides a method for reconstructing geometric information from the power side-channel. In combination with the signature-based detection from Research Question 1, this forms the foundation of side-channel based forensic examination. First, the signature of a possibly sabotaged print is compared against a verifiably benign master signature. If there are discrepancies over the detection threshold, the same current traces can be run through the reconstruction algorithm to generate point clouds for both the known-good and the suspect print.

The work that remains to fully answer Research Question 3 is to develop a method to identify discrepancies between these point cloud representations and to visualize them in a user-friendly manner. Such a system presents a number of advantages for investigators. Expensive destructive testing may be avoided, for one. For another, relying on an expert human analyst allows for a reasonable chance of detecting even previously unknown (also known as zero-day) attacks. In AM, where a body of real-world attack knowledge does not yet exist and even attacks in the literature are regularly custom to the targeted print, this flexibility is almost mandatory.

In light of this human-in-the-loop approach to answering Research Question 3, I pose two followup questions. First, what is the quality of the visualization? A high-quality visualization would adequately identify discrepancies from introduced defects, without a high false-positive or false-negative rate. Secondly, what factors influence the quality? As the input for this stage are point clouds generated by the reconstruction process of Chapter 7, potential factors may be qualities of the point clouds or parameters of the comparison metrics.

In the following section, I present my approach for forensic investigation and visualization based on point cloud rendering and metrics derived from the Hausdorff distance, with the inclusion of additional temporal and dimensional information. In Section 8.3, I outline a series of experiments to test my approach's behavior under different conditions, including the similarity of known-good prints, stability in the presence of reconstruction error, and meaningfully different visualizations for different attacks.

8.1 A Spatial and Temporal Distance Metric for Forensic Investigation in AM

The requirements I place on a comparison metric are as follows.

1. The metric must be calculated between two arbitrary point clouds, a collection of coordinates in R^3 , generated by the reconstruction method outlined in Chapter 7.
2. The metric must distinguish between equivalent clouds, which may be generated by executing the same print twice, and deviating clouds, which come from differing and potentially sabotaged prints.

3. The metric must produce localized results, i.e., have values per-point or per-region rather than across the entire cloud.
4. The metric must be visualizable in a human-readable way.
5. Computing the metric must be reasonably performant.

Some of these requirements demand further explanation. Equivalent, in this context, cannot mean exactly the same. Reconstruction results contain errors which lead to positional drift in the point clouds, so that repeated runs of the print do not produce exactly the same cloud. For this requirement I accept any metric that adequately indicates sabotage-related deviations, even if they also flag drift deviations. Whether these can be distinguished from each other will be examined in the experiments of Section 8.3.

Visualizing the results will follow from the point-cloud visualizations used in Chapter 7. These already present the reconstructed geometry, so the only additional demand on the metric is to modify this system to distinguish between acceptable and deviating points in an equally readable fashion.

As with the other systems presented in this dissertation, performance requirements are dictated by the context of use. For the current purposes of this work, execution times for the test set could acceptably run to multiple hours so long as they're intended for post-incident investigative analysis. As with the reconstruction performance analysis, consideration will be given to potential applications with tighter performance demands or larger inputs.

8.1.1 The Hausdorff Distance

I consider, based on its technical definition [30] and previous applications in the field [21], a modified Hausdorff Distance to meet the requirements for a comparison metric. The Hausdorff Distance, first described in 1914 by Felix Hausdorff, is a measure of closeness between two subsets of a metric space. This already implies it meets my first requirement; point clouds in R^3 are exactly that. It can be defined as the largest minimum distance from a point in set A to a point in set B. This would possibly meet the second requirement, though it requires experimental support. Two identical clouds (A and A) would have a distance of 0. Two equivalent clouds

would have a distance proportional to the maximum drift occurring in the reconstruction. Two differing clouds would have a distance proportional to the geometric deviations introduced by sabotage, in addition to any drift. If the second and third scenarios can be distinguished, then the requirement is met.

The third requirement fails at this point, though the correction is straightforward. The Hausdorff Distance produces a single metric for an entire subset, which does not adequately localize the deviation. Modifying the definition to preserve the minimum distance at each point, which must be calculated in the original Hausdorff, allows it to meet this requirement. This gives a scalar distance value for each point in the cloud, which can be readily visualized as a heatmap. If this visualization is sufficient to identify sabotage under experiment, it meets the fourth requirement.

The fifth requirement demands some calculation. A single distance comparison between two points is a fixed-time operation, involving several additions and square-roots for points in R^3 . In a real implementation, there must also be some assignment operations to store the resulting value. Assume these together take a fixed time δT . A naive implementation of the Hausdorff distance between clouds A and B would have to compare each point in A against each point in B, and then each point in B against each point in A if distances for both clouds are desired. This gives a straightforward but disheartening idea of the time complexity, with the full comparison taking $2 * |A| * |B| * \delta T$.

If I assume that A and B have roughly the same cardinality, as would be the case with an original and stealthily sabotaged print, it simplifies to $2 * |A|^2 * \delta T$ for quadratic time complexity in all cases. This is quite bad, given that the test set of models result in point clouds with cardinalities in the low millions.

This sort of approach is, of course, not realistic. I include it to highlight that some degree of optimization is absolutely necessary when working with inputs of this size. The key optimization I have pursued is to restrict the number of comparisons made for each point by partitioning the point clouds. This I have done in two ways, which I have named the Height-Partitioned Hausdorff (HPH) and the Toolpath-Partitioned Hausdorff (TPH).

8.1.2 Height-Partitioned Hausdorff

On an FDM printer, the reconstructed point cloud naturally separates into layers; the vast majority of prints conduct all of their extrusion in X-Y planes and only traverse on the Z axis with non-extruding moves. In the Height-Partitioned Hausdorff, I take advantage of this to restrict the comparison of points to those within approximately one layer of height. It is approximate, rather than exact, because the Z axis motor trace has the highest error rate due to noise in the entire system. As a result, points that should be on the same plane experience Z-jitter, ranging up and down across a few steps of distance. The average Z-jitter distance, which I consider to be bounded above by the bad-section lengths on the axis, is smaller than the length of a proper layer transition, so including a range above and below will catch the vast majority of points.

Pseudocode outlining the Height-Partitioned Hausdorff is given in Figure 8.1.

```
Vector3[] baseVertices, testVertices;
float[] baseDistances, testDistances;

float distanceThreshold;

// Height-partitioned lists for the base and test clouds
SortedList<float, List<Vector3>> baseYValues, testYValues;

// Sort vertices into height-partitioned lists
foreach vertex in baseVertices
    if vertex.y in baseYValues
        baseYValues[vertex.y].Add(vertex)
    else
        baseYValues.Add(vertex.y, vertex)

foreach vertex in testVertices
    if vertex.y in testYValues
        ... // As in above loop

// Perform distance calculation against matching height-partitioned lists

for i from 0 to baseVertices.Length
    minDistance = inf

    foreach yValue, list in testYValues
        if abs(baseVertices[i].y - yValue) < distanceThreshold
            // List is in partition, compare against that list's points
            for each vert in list
                distance = Vector3.Distance(baseVertices[i], vert)
                if distance < minDistance
                    minDistance = distance

    baseDistances[i] = minDistance

// Repeat above loop for test vs. base comparison
...
```

Figure 8.1: The Height-Partitioned Hausdorff Distance.

8.1.3 Toolpath-Partitioned Hausdorff

An alternative, finer-grained partitioning of the FDM print pattern follows the linear, sequential toolpath. The points in the cloud, after all, are themselves a discrete sampling of extrusion positions along the toolpath. The sequencing of the toolpath is preserved in the ordering of the points. It's therefore possible to partition the Hausdorff distance comparisons into a fixed number of preceding or subsequent points in the cloud. For all points, even those at the very beginning or end of an extrusion, the closest point in the other cloud should occur at the same point in the toolpath. If the comparison system works from the assumption that the two clouds under test should be identical, any failure of this assumption would indicate a deviation.

As always, however, error must be accounted for. For identical clouds only one comparison would be necessary: the same index position in the other cloud. The effects of noise on the reconstruction means that equivalent clouds experience both positional drift and sequential drift, due to the inclusion of false points or the exclusion of real points. The range of comparison should ideally be set to the largest number of peaks of drift between the two clouds, but if it were possible to know this number I could simply eliminate the drift. As it stands, the range should be set to a best estimate of this amount, with some extra to tolerate inaccuracy.

Pseudocode outlining the Toolpath-Partitioned Hausdorff is given in Figure 8.2.

8.2 A Forensic Visualization System

Unlike the development of the modified Hausdorff distance metrics, real-time graphics rendering and manipulating 3D assets would not constitute novel work in this field or in their own; as such, I rely on an existing graphics and rendering engine. There are several requirements on the engine. First, it should be capable of importing 3D assets in the various formats already used in this work: STL files and point clouds. Second, it must also be possible to modify these assets once loaded. Third, the rendering of the assets should be done at an acceptably high framerate even when some elements (such as camera position) are being modified in real time. Fourth, the system must be scalable enough to operate in real-time on large datasets; for this

```

Vector3[] baseVertices, testVertices;
float[] baseDistances, testDistances;

float toolpathRadius;

// Perform distance calculation against range of currentPosition +/- toolpathRadius,
// adjusted for list length and avoiding index-out-of-range

for i from 0 to baseVertices.Length
    minDistance = inf

    // Find the centerpoint at the same relative position in the other cloud
    int otherCloudCenterpoint = ceiling(i / baseVertices.Length) * testVertices.Length

    int leftBound = (otherCloudCenterpoint - toolpathRadius > 0) ?
        otherCloudCenterpoint - toolpathRadius : 0

    int rightBound = (otherCloudCenterpoint + toolpathRadius < testVertices.Length) ?
        otherCloudCenterpoint + toolpathRadius : testVertices.Length

    for j from leftBound to rightBound
        distance = Vector3.Distance(baseVertices[i], testVertices[j])
        if distance < minDistance
            minDistance = distance

    baseDistances[i] = minDistance

// Repeat above loop for test vs. base comparison
...

```

Figure 8.2: The Toolpath-Partitioned Hausdorff Distance.

dissertation, it must at least handle multiple copies from my test set of point clouds. Finally, the system should support human input with standard I/O devices.

Engines intended for 3D games and other real-time audiovisual applications meet all these requirements. I selected the Unity engine [56], as it natively supports STL file import and supports point clouds (in the Stanford PLY format) through a plugin. Although some elements, such as the point-cloud importer, are open source, the core engine is closed-source and proprietary. This limits the scope of performance evaluation significantly, down to code I have developed myself.

Within the engine many of the default rendering behaviors for point clouds, including lighting and coloring, are already acceptable. However, the visualization and rendering of the modified Hausdorff distance metrics must be determined and applied to the imported clouds. I chose to visualize the resulting distances with a combination of heatmaps and point culling.

Heatmaps are a popular tool in 2D and 3D visualization to provide an intuitive view of scalar values that vary over a space or volume. Both of my distance metrics produce a range

of scalar values for each point in the cloud. To assign these a heatmap value, I track the global minimum and global maximum distances found in the cloud and use these to linearly interpolate between two colors. For the base cloud, points range from white (nearest) to blue (furthest); in the cloud under test, points range from green to red in the same manner.

An issue arising from this is that the clouds overlap one another, occupying the same volume in the rendered space. This makes the visualization confusing and difficult to read, especially when deviating sections are “buried” within the interior of the printed volume. To address this issue, I cull a number of points by noting that non-deviating points from the base cloud are not of interest in a sabotage investigation, and this information is already legible in the coloring of the cloud under test. The only points of interest in the base cloud are those that are distant from any point in the test cloud. To provide flexible control of this, my system therefore allows the user to set a threshold distance; points in the base cloud with distances below the threshold are not rendered.

At the top level, I support several other elements of user control over the visualization. Users can rotate and zoom in on the rendered clouds, in a similar fashion to other CAD programs. In addition, users can hide or show either the base or test clouds for solo inspection.

8.3 Experimental Evaluation

In support of answering Research Question 3, on whether the actuator current side-channel provides enough information to localize and investigate a sabotage attack, I have proposed a visualization system for comparing two point clouds. It is now necessary to consider how this question can be broken down into testable hypotheses, and what experiments might be able to examine them.

Can ‘equivalent’ point clouds be distinguished from deviating point clouds? This is the baseline level of performance necessary for an investigative system. If there is no discernable difference between a good-to-good comparison and a good-to-sabotaged comparison, then Research Question 3 can be answered in the negative with no further work.

What qualities of the input point clouds influence the results of the distance metrics and visualization? Prints, and their resulting reconstructions, can vary from one another in

significant ways. Within this context, I expect and wish to test for two significant qualities: density, resulting from infill percentage, and reconstruction error. Both strongly influence the distribution and location of points in the cloud, so I expect them to have an equally sizeable effect on inter-point distances. Density can be directly controlled in the slicer, and so may be used as a variable in my experiments. Reconstruction error, however, is stochastic and uncontrollable. Its effects will instead be carefully observed wherever drift due to error is visible.

What are the key indicators in the visualization of deviations? I arrive to this question with a number of assumptions, guided by an understanding of slicer behavior. Void insertion attacks will likely be localized to a volume, and register as fairly significant deviations in distance proportional to the ‘centermost’ point, the point with the longest minimum distance to the edge of the void. I believe that the insertion of a void before slicing the model introduces significantly broader effects as the slicer accommodates the void in the infill pattern, resulting in different points of extrusion locally and introducing a larger deviation into the global sequence of points. Therefore, I expect that the Height-Partitioned Hausdorff and the Toolpath-Partitioned Hausdorff will be visually distinct here, with the HPH better localizing the void and the TPH showing lasting deviations from the void to the end of the print.

How do the Height-Partitioning and Toolpath-Partitioning optimizations perform for computing the Hausdorff Distance? In a naive implementation, the Hausdorff distance can be computed by comparing every point against every other point. Unfortunately, the point-set sizes achieved during reconstruction regularly number in the millions, so the naive implementation scales poorly. Both of my proposed modifications should reduce execution times substantially by restricting the number of point comparisons made, but the exact degree of improvement is difficult to predict without testing. Examining the time-performance of each approach can be done in conjunction with every visualization experiment.

To examine the above questions, I have designed a series of experiments as follows. A pair of test models, one from the original test set of Chapter 7 and one designed for this purpose, will be printed and their traces captured. The models are presented in Figure 8.3, using an X-ray visualization to show the internal structure. A sabotaged version of each model, containing

a performance-impacting void, will also be printed and captured. Both the standard and void-sabotaged versions will be run twice, once at 20% infill and once at 70% infill. All of these will undergo the full reconstruction process, resulting in reconstructed point clouds.

These clouds will be loaded into my visualization system; for each, both the Height Partitioned Hausdorff and the Toolpath Partitioned Hausdorff will be computed between matching standard and sabotaged captures and the results visualized. I will then examine both these visualizations, attempting to identify deviations and reason about their properties, and noting differences due to infill density and any apparent drift.

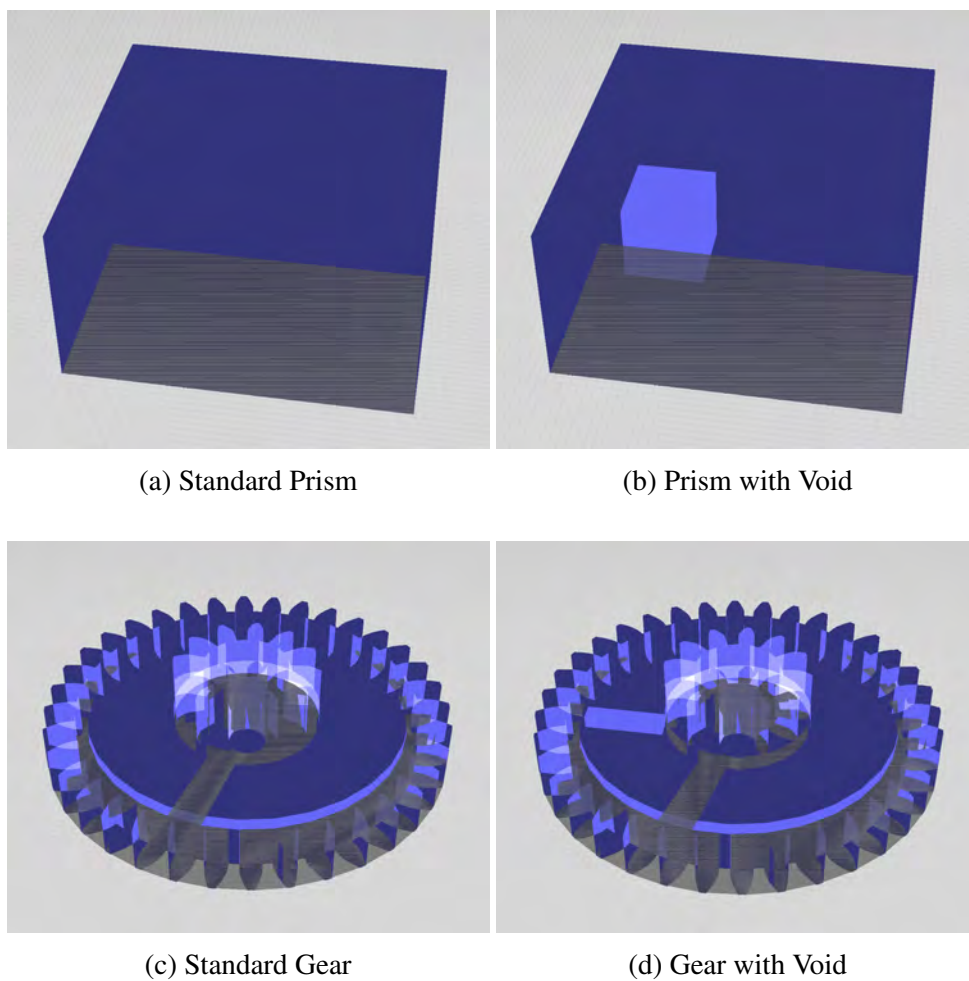
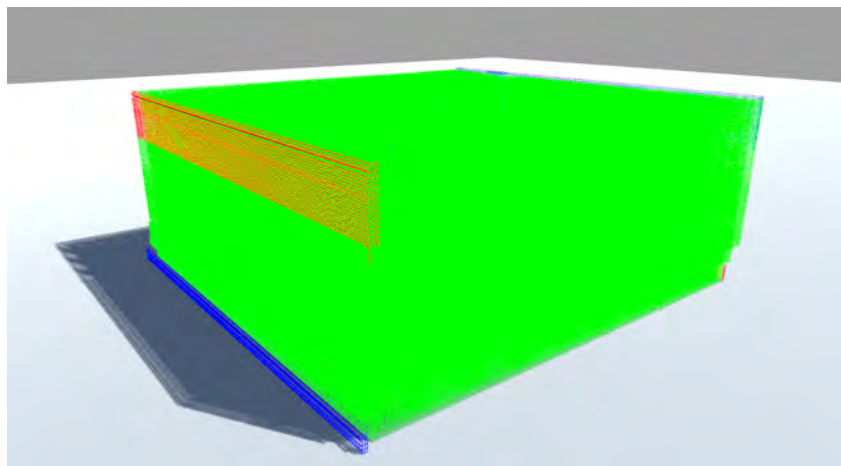


Figure 8.3: Test set of models used for the forensics visualization experiments.

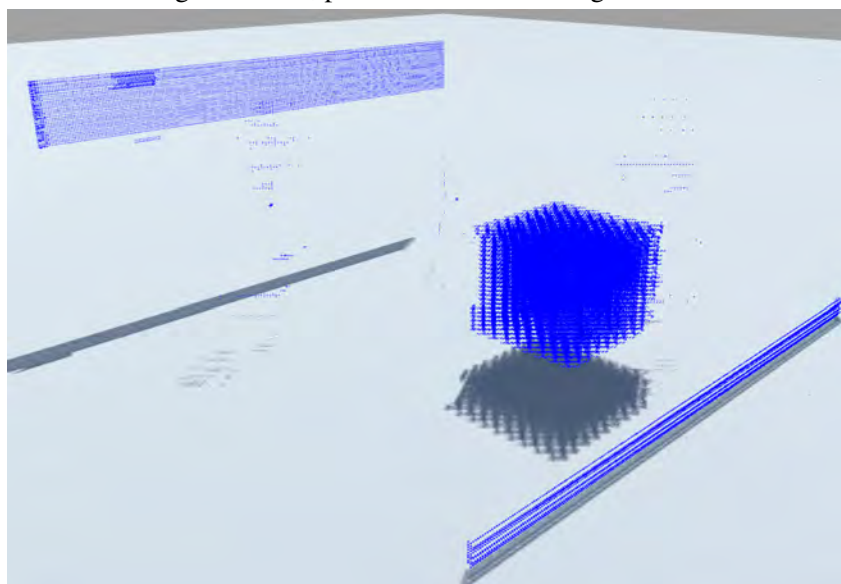
8.4 Results

In this section, I present the results of my experiments in a series of figures, displaying the visualizations under different conditions and highlighting notable elements. I first present the results of the Height-Partitioned Hausdorff, showing the 70% and 20% infill Rectangular Prism, then the 70% and 20% infill Gear. I then present the Toolpath-Partitioned Hausdorff results in the same order.

8.4.1 Prism 70% Infill HPH



(a) The 70% infill Rectangular Prism, processed with the Height-Partitioned Hausdorff metric.



(b) Same as above, with the standard-print point cloud isolated. The bars to the front and rear correspond to drift between the two reconstructions. The cube corresponds to the inserted void.

Figure 8.4: The Prism model HPH comparison, printed at 70% infill.

The 70% infill Prism immediately presents encouraging results: the volume of the void is clearly indicated by blue points, though this requires viewing the base point cloud in isolation due to the density of points overall. Though there is a small amount of shift, 1-2 steps overall, it is easy to determine the cubic shape, size, and position within the print of the void from this visualization.

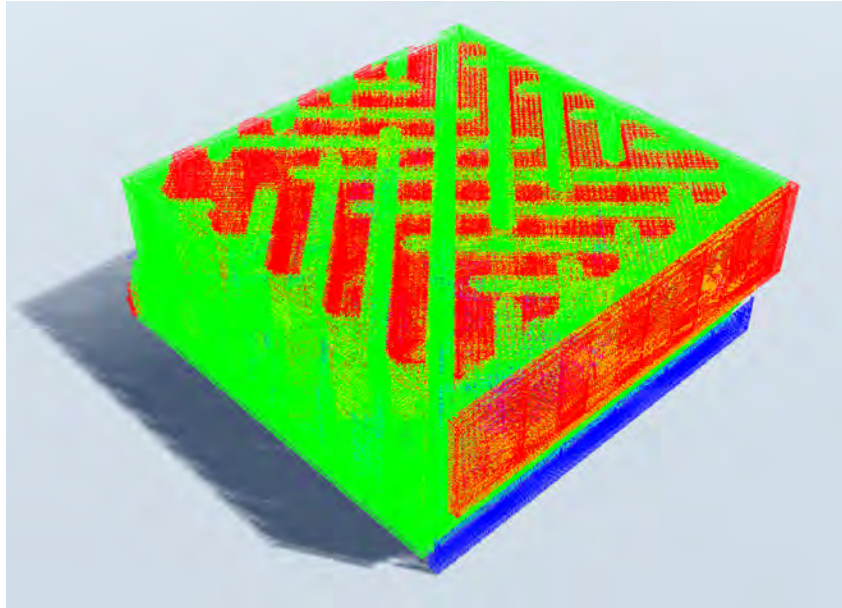
There are two clear bars of blue points visible in addition to the void, which are matched by sections by red points (indicating extraneous points in the test cloud) on opposing sides of the object. These areas correspond to portions of drift, which in this case occurs in the test cloud. While the base cloud reconstruction is very accurate, with no visible drift, the test cloud reconstructed with noticeable drift on the scale of 5 steps occurring twice: once near the base, and once approximately 20 layers from the top. The portions of the test cloud that are shifted further than the culling distance threshold produce these symmetrical bars. Interestingly, the bars do an excellent job of indicating the location, duration, and amount of drift.

Other portions of the print that are known to be equivalent are generally visualized appropriately, with little to no visible red elsewhere in the test cloud. Viewing the base cloud in isolation, some sparse columns of blue points are visible oriented along the infill lines. There are few enough of these to avoid misdirection from the denser void area, but their arrangement in columns is interesting and may indicate a portion of the toolpath which consistently produces minor errors in the reconstruction.

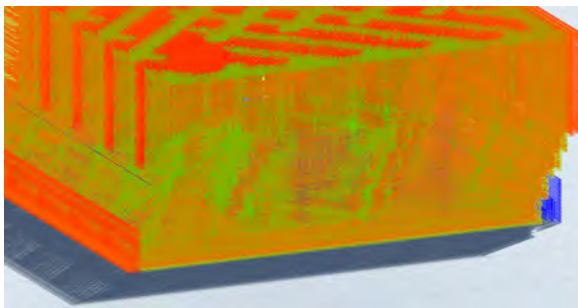
8.4.2 Prism 20% Infill HPH

Results for the 20% infill Prism demonstrate worse cohesion than the 70%; areas that should be equivalent, such as the entire top surface, have substantial patches of red indicating extraneous peaks between what are clearly the infill lines. In this case, I believe the issue is that there is drift present along the Z axis of the test cloud, misaligning it such that the top is lower than the base cloud. The full-infill top layer would then overlap with the infill pattern of the base cloud, leading to this pattern.

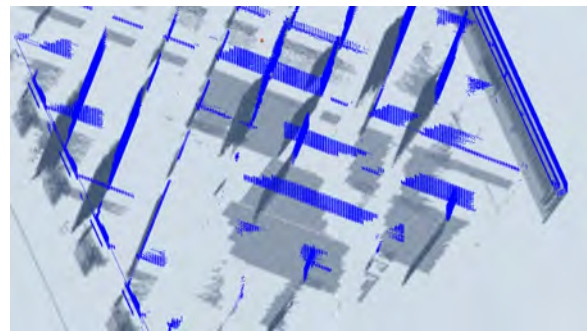
The internal volume fares similarly. Drift on the X and Y axes is substantial enough to misalign major sections of the infill pattern. While it does highlight the internal structure



(a) The 20% infill Rectangular Prism, processed with the Height-Partitioned Hausdorff metric.



(b) Same as above, zoomed to show the internal void.



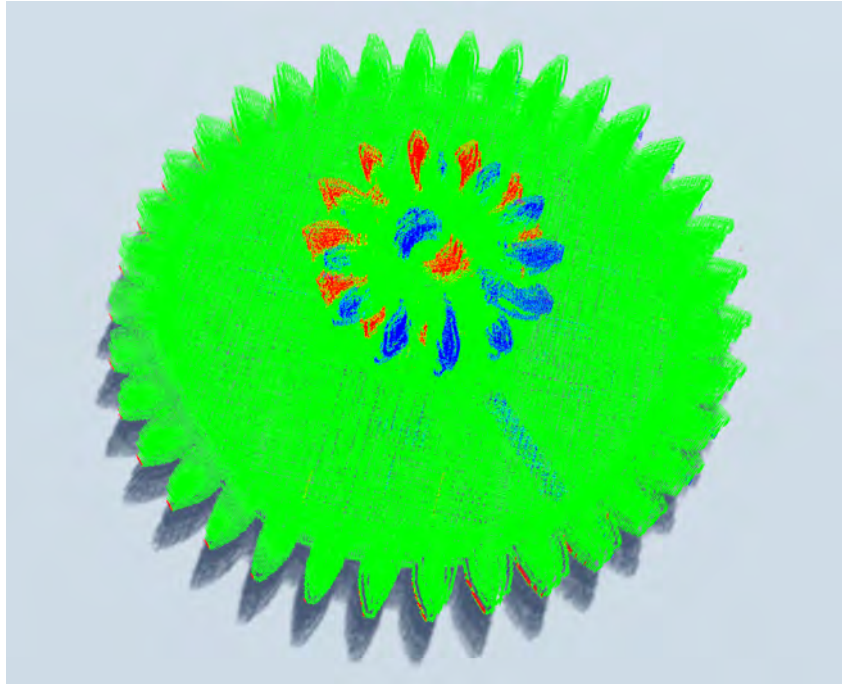
(c) Same as above, with the standard-print point cloud (showing missing points) isolated.

Figure 8.5: The Prism model HPH comparison, printed at 20% infill.

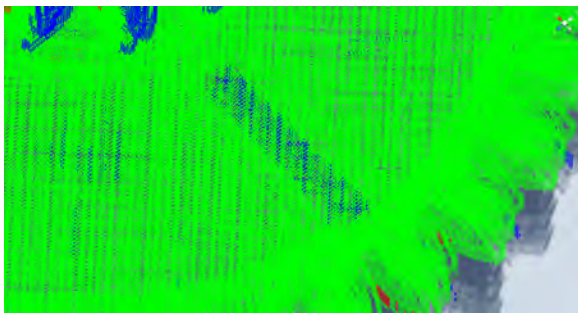
removed by the void attack, the rate of false positives throughout is too high to reliably identify it. The walls of the void itself instead face a false-negative problem; portions are clearly marked as extraneous, but sections which correctly intersect with the infill pattern in the base cloud are not, making it harder to identify as a whole.

8.4.3 Gear 70% HPH

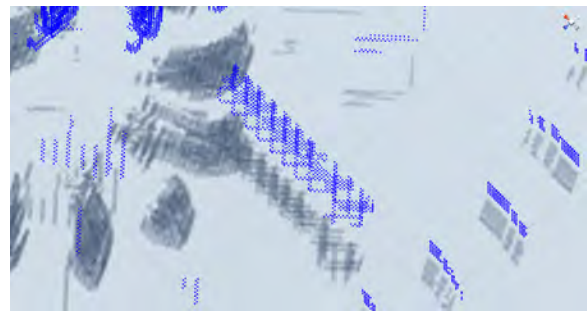
As with the 70% infill Prism, the 70% infill Gear presents encouraging results. The void is clearly visualized, in this instance visible even before isolating the base point cloud. Its dimensions and location are readily determined, even being substantially smaller than the void



(a) The 70% infill Gear, processed with the Height-Partitioned Hausdorff metric.



(b) Same as above, zoomed to show the void.



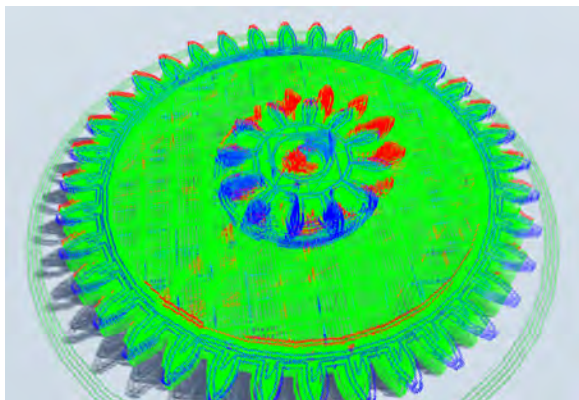
(c) Same as above, zoomed and with the standard-print cloud isolated.

Figure 8.6: The Gear model HPH comparison, printed at 70% infill.

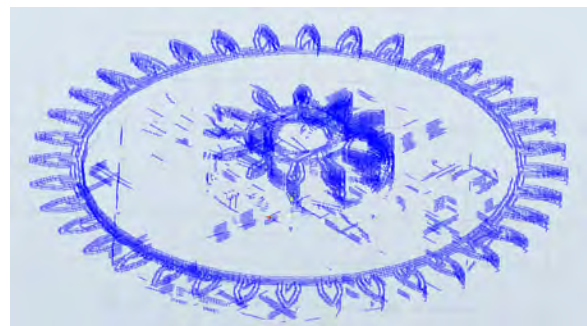
inserted into the Prism. Further, the density of points at the void allow it to be distinguished from minor false-positive points in the surrounding area.

There is still drift in the reconstructions used here, primarily in the top section of the print. In this case, the base and test reconstructions drifted in opposite directions, producing a symmetrical spread of blue and red points around the actual center of the print. Also of interest are areas of drift evident in the teeth of the gear's lower section, which are much smaller but still present. That one side of the gear shows red sections of extraneous peaks and the opposite blue is in keeping with earlier results, but here the behavior isn't global. The lower and upper sections of the print are drifting in different directions, and this is apparent in the visualization.

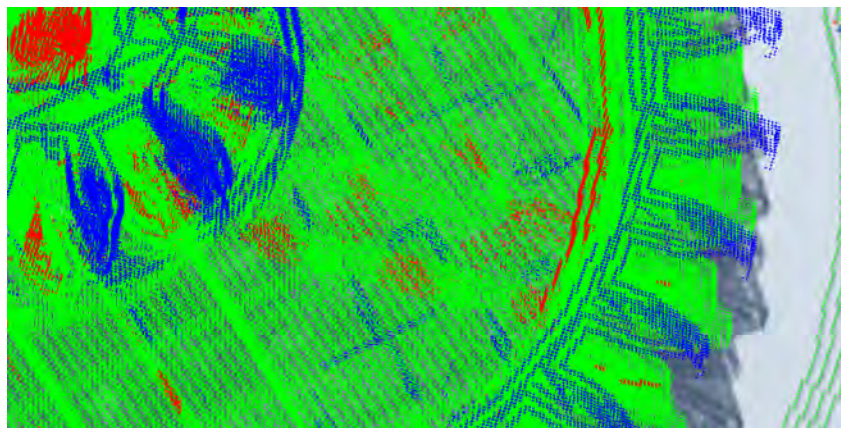
8.4.4 Gear 20% HPH



(a) The 20% infill Gear, processed with the Height-Partitioned Hausdorff metric.



(b) Same as above, with the standard-print point cloud isolated. Without advance knowledge of the void's location, it would be difficult to determine from this image.



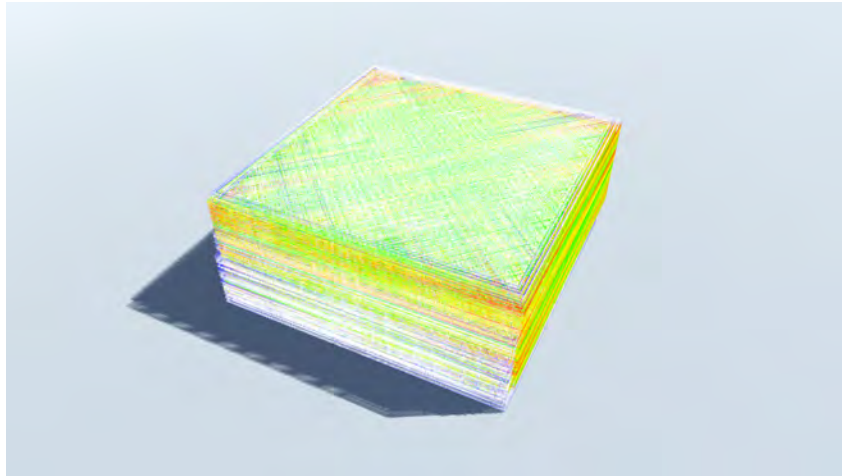
(c) Same as above, zoomed to show the void. Note the red sections of the void walls (indicating extraneous points in the test cloud), and the blue sections within (indicating missing points in the base cloud).

Figure 8.7: The Gear model HPH comparison, printed at 20% infill.

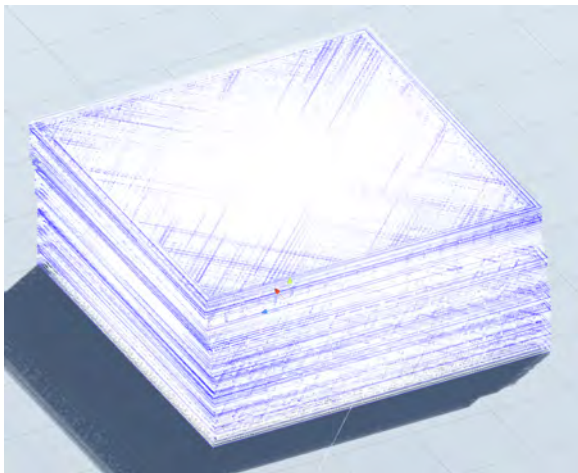
Results for the 20% Gear are an interesting mix: not as poor as the 20% Prism, but still with a high rate of false positives and negatives. The area of the void is mostly correctly visualized, with the interior infill lines highlighted in blue and portions of the void walls in red. However, the isolated base cloud shows that the average density of blue points is too high to easily identify the void. The same is true of red points in the test cloud.

Drift between the clouds appears with the indicators seen in previous tests: opposing red and blue segments, as well as an entire upper layer in blue indicating Z axis drift leaving the base cloud slightly taller. The incidence rate of internal drift is unfortunately high, though the greatest amounts are still present at the borders of the clouds.

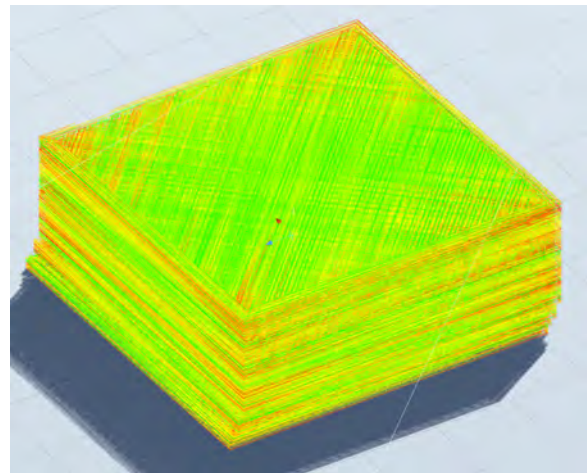
8.4.5 Prism 70% TPH



(a) The 70% infill Prism, compared with the Toolpath-Partitioned Hausdorff metric.



(b) Same as above, with the base cloud isolated.



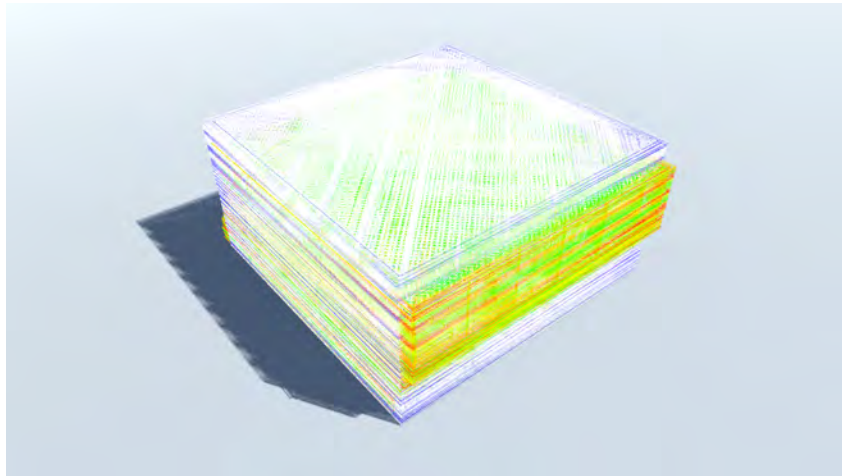
(c) Same as above, with the test cloud isolated.

Figure 8.8: The Prism model TPH comparison, printed at 70% infill.

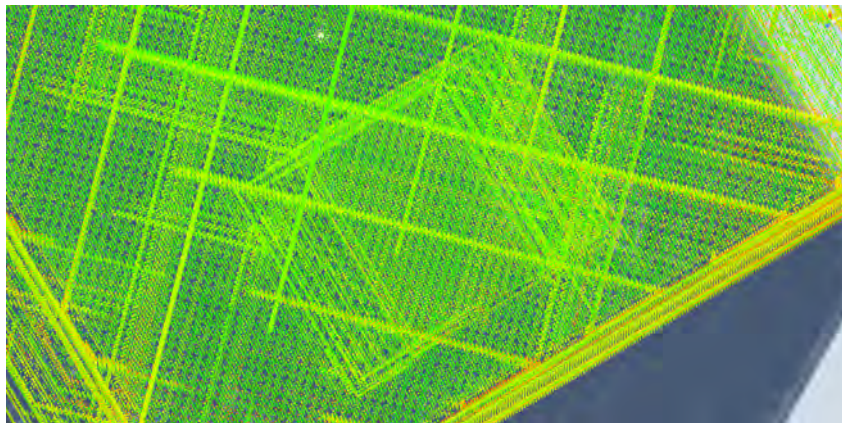
The Toolpath-Partitioned Hausdorff metric produces limited results across the entire range of models; here in the 70% infill Prism the difference between the TPH and HPH is most clear. Deviations are still greater near the borders of the print, but the errors follow a very different pattern. Most importantly, the void is not visualized in a meaningful way. Blue points are instead distributed throughout the body of the cloud, clustering towards the exterior borders. For this and other TPH visualizations, the models are presented without point culling enable to better show the variation.

Drift is also not visualized here in the same manner. There is drift present in these two clouds, but it can't be visually identified. Instead, the drift's effect on point counts between any two destinations on the toolpath is the cause behind the periodic alignment and misalignment of the two toolpaths. This was confirmed using an animated visualization, which follows the toolpaths as processed by the TPH algorithm.

8.4.6 Prism 20% TPH



(a) The 20% infill Prism, compared using the Toolpath-Partitioned Hausdorff.

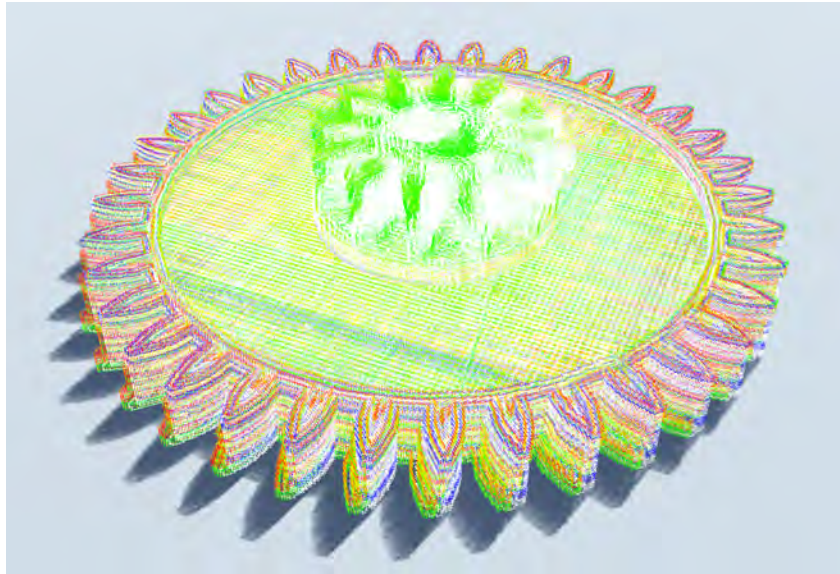


(b) Same as above, zoomed to show the area of the void.

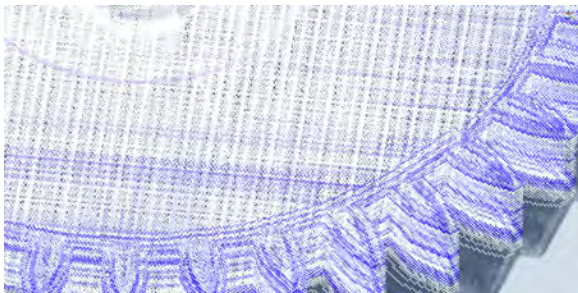
Figure 8.9: The Prism model TPH comparison, printed at 20% infill.

The 20% infill Prism exhibits much the same properties as the 70% infill version, with no apparent change in red or blue point density around the void. Here, the walls of the void in the test cloud are presented from below, showing that the TPH fails to highlight them in red.

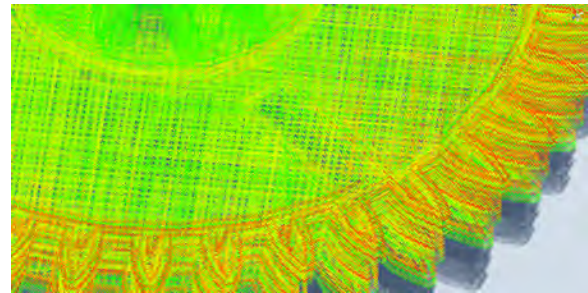
8.4.7 Gear 70% TPH



(a) The 70% infill Gear, compared using the Toolpath-Partitioned Hausdorff metric.



(b) Same as above, with the base cloud isolated, zoomed to show the area of the void.

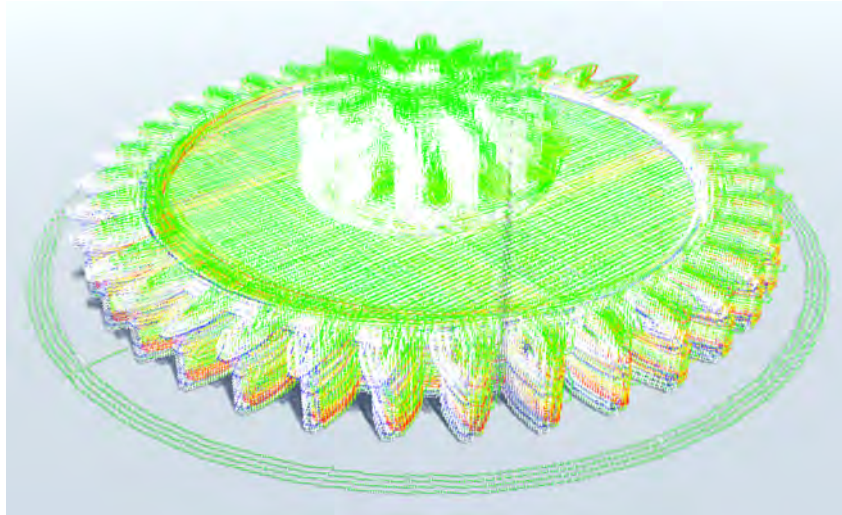


(c) Same as above, with the test cloud isolated, zoomed to show the area of the void.

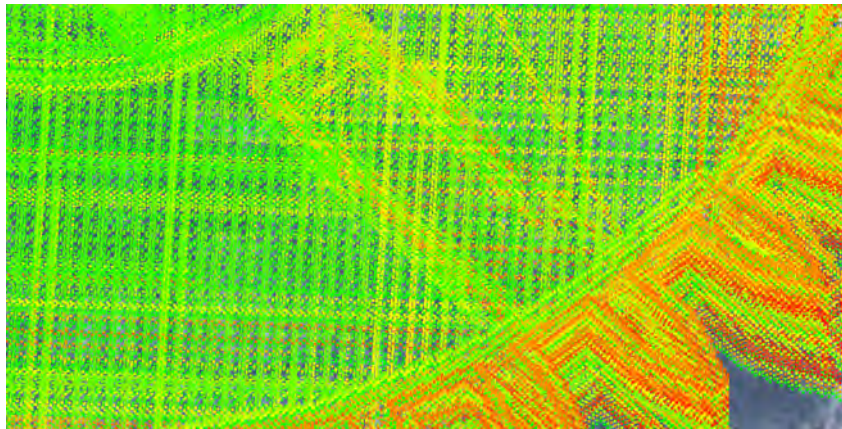
Figure 8.10: The Gear model TPH comparison, printed at 70% infill.

The 70% infill Gear, as with other TPH comparisons, fails to highlight the void area consistently. The test cloud does manage to indicate several portions of the void's walls, but given the high incidence of false positive indications in even the immediate area it would be difficult to identify.

Drift would be difficult to diagnose from this visualization, but its effects are evident. Also worth noting is the extreme incidence of false positive deviations along the gear's teeth. By comparison, the interior is marked as relatively equivalent.



(a) The 20% infill Gear, compared using the Toolpath-Partitioned Hausdorff metric.



(b) Same as above, zoomed to show the void. The highlighting of the void walls in red is the most successful application of the TPH in the test set.

Figure 8.11: The Gear model TPH comparison, printed at 20% infill.

8.4.8 Gear 20% TPH

The 20% infill Gear presents fascinating results-it is the only TPH comparison from which the void might be identified. As Figure 8.11 shows, the walls of the void are consistently highlighted in red with the immediately surrounding area remaining green. This would provide the necessary contrast to distinguish the void. Unfortunately, the base point cloud fails to also highlight the void's interior space in blue.

As before, the drift is present and visible in periodic highlighting of toolpath-sequential cloud segments as deviating, but the degree is much lower here than in other TPH comparisons. Looking back at the results for the HPH comparison on these same reconstructions (Figure 8.7),

it appears that there may be very limited drift in the early sections of the print: the bottom gear teeth have no blue or red highlighting indicating X and Y drift, unlike the 70% infill Gear (Figure 8.6).

8.5 Addressing Research Question #3

The research presented above in this section was in service of answering a particular research question, reproduced below.

Research Question 3: Does the actuator current side-channel provide enough information to localize and investigate a sabotage attack?

Secondary to this question, I also asked:

- Can ‘equivalent’ point clouds be distinguished from deviating point clouds?
- What are the key indicators in the visualization of deviations?
- What qualities of the input point clouds influence the results of the distance metrics and visualization?
- How well do the Height-Partitioning and Toolpath-Partitioning optimizations perform in execution time?

Addressing the first and second together, the visualization results indicate the equivalent point clouds and deviating point clouds can be distinguished using the Height-Partitioned Hausdorff metric, for the higher-density 70% infill prints. In these cases, the differing regions are clearly indicated with minimal noise occurring elsewhere. Equivalent regions of the 70% comparisons show as clear, largely unbroken regions of green. The only muddying factor in these comparisons is drift, which introduces matching deviations on opposite sides of the model, but this is itself visually distinct from the appearance of the tested attack.

For HPH on the 20% infill models, the effects of drift on both interior regions and height alignment serve to obscure the visual signature of the attack. The signature is present, with blue indicating the void space and patches of red through the void walls, but cannot be well

distinguished from the many other infill walls that are highlighted. Further, the reduced density of the cloud means that even under perfect conditions there would be fewer points in the signature.

Toolpath-Partitioned Hausdorff comparisons fared poorly overall, with only the 20% infill Gear comparison managing significant contrast between a correctly-highlighted void space and the surrounding unmodified points. Even here, the lack of blue highlighting for the removed internal structure means identification would be marginal.

As this was a remarkable underperformance, I proceeded to visualize the operation of the TPH as an animated trace to diagnose the issue. What this revealed is that the degree of drift eventually leads to multiple layers of separation between equivalent-sequence points in the base and test clouds. As the infill patterns of every model follow a fairly standard linear-with-rotation strategy, equivalent points in the toolpath have a tendency to cross and then separate towards the center of the cloud, and are likely to be at their most distant towards the boundaries, which have the highest average distance from a randomly-selected point in the cross section. Since the layer separation, even across several layers, is far smaller than distances in the X-Y cross sections, it contributes little to the distance measure. The resulting visualizations generally show near-equivalence towards the center of the models, divergence towards the boundaries, and deviations due to actual changes in the toolpath and resulting point cloud sequencing are lost in the noise.

That the 20% infill Gear reconstructions coincidentally had low enough drift to perform as intended on the walls of the void is a pleasant indicator that this approach can function, but it would need to be much more reliable before any result would be trustworthy.

Addressing the third sub-question, two qualities emerged as major factors in the visualization results: density and drift. High-density models with 70% infill performed consistently well using the HPH metric, with few internal deviations apart from the sabotaged region and dense blocks of color indicating missing or extraneous points. Low-density model with 20% are instead overwhelmed by drift, causing the thin internal structures (infill walls in most slicers are a single extrusion line) to appear as deviating, which obfuscates the already lower-intensity

signature of the attack. It's therefore reasonable to say that higher-density models are both more robust to drift and more suitable to HPH comparison in general.

The effects of drift on TPH comparison are clear: changes due to drift in the point sequence appear to overwhelm the metric in all but one instance. Density, interestingly, has either no or a slight negative effect on TPH results. Logically this tracks, as the comparison range for TPH is both fixed in size and linear along the toolpath. Increased density would only have an effect if the range was so wide as to include both an outgoing line of extrusion and its return in the opposite direction. For a more reasonable comparison range, density only increases the number of points overall in a space, which would provide even more opportunities for drift to occur.

The final sub-question will have a further examination in Section 8.6 on performance analysis, but I will sketch the results in brief here. The naive Hausdorff computation is infeasibly time consuming for the tested point clouds. For the 20% Gear comparison, which averages 1 million points in each cloud, the algorithm would have taken approximately 1,500 days to terminate (this was computed based on average loop execution times). The HPH optimization, which restricts comparisons to the present and adjacent layers, brings this execution time down to 4 hours for the same comparison. The TPH optimization, which restricts comparisons to a fixed sequential range, performs even better, executing in under 30 seconds. For this question, the performance optimizations are an unqualified success over the naive approach, bringing the execution times down to usable scales for forensic investigations.

Overall, Research Question 3 can be answered positively: the results support that the actuator current side-channel, and my reconstructive process for it, can provide enough information to localize and investigate a sabotage attack. The visualizations of the 70% infill Prism and Gear clearly highlight an inserted void, giving a firm idea of its shape, size, and location within the printed object.

Results for the other comparisons indicate the limitations of the approach as it currently exists: drift, already a concern in the reconstruction stage, here places a firm limit on the fineness of details that can be correctly compared and visualized. It particularly restricts the TPH from correct functioning, whose limited comparison range means it is overwhelmed by drift earlier than HPH. While this will be elaborated on in Chapter 10, future work focused on

```

hausdorff(set A, set B)

    float hDistance = 0

    for pointA in A:
        for pointB in B:
            if distance(pointA, pointB) > hDistance:
                hDistance = distance(pointA, pointB)

    return hDistance

```

Figure 8.12: A naive approach to calculating the Hausdorff distance.

reducing or eliminating drift in the reconstruction stage would be a great boon to the forensics application, allowing the rapidly-executing TPH metric to be deployed with reliable results.

8.6 Performance Evaluation of the Forensics Metrics

The basis of my forensic comparison is the Hausdorff distance, as discussed in Chapter 8. The metric can be defined for two point-sets A and B as follows: for all points in A, find the minimum distance traveled to reach a point in set B. The greatest distance found across the set is the Hausdorff distance [30].

This can be computed naively by an exhaustive search; such an approach is described in Figure 8.12. While correct, this approach is slow for realistic print sizes.

In the case of forensic comparison, the point-sets A and B can be assumed to be a pair of similar, if not identical, reconstructed 3D print point clouds. Any difference in the cardinality of A and B, perhaps due to sabotage attack, will make up a small proportion of the total size. I denote the size of the first, known good print as N . The size of the second point set, B, is N' . I make a practical consideration here to ease analysis: in all tested cases in this dissertation, $|N' - N| \ll |N|$. Therefore in performance assessment I substitute N for N' , to reduce the number of variables in reported performance complexities.

In the naive approach (Figure 8.12), two nested loops iterate over every point in A and compare each against every point in B; this results in a time complexity of $O(N^2)$. The test set of models have reconstructed point cloud sizes ranging from 49,000 for a small model up to 4,400,000 for the largest model. The number of points in the cloud is proportional to the total printed volume, itself a result of the overall size of the part and its infill settings. Based

on a handful of runs, the naive approach becomes frustratingly slow and impractical in the low 100,000 point range; looking to future scalability, many important additively manufactured parts are much larger or denser than this test set. Therefore, I opted for modified metrics that allow for better time complexities.

The Height-Partitioned Hausdorff, as described in Section 8.1.2, partitions the point cloud into approximately layer-width slices along its height. The pseudocode for computing this metric across two clouds is reproduced in Figure 8.13. The first two *foreach* loops perform the segmentation; these iterate linearly over the clouds, and contribute a component of $2 * N$. The bottom *for* loop and nested *foreach* and *for* loops perform the bulk of the work in computing the HPH, and are run once for each point cloud.

```

Vector3 [] baseVertices , testVertices ;
float [] baseDistances , testDistances ;

float distanceThreshold ;

// Height-partitioned lists for the base and test clouds
SortedList<float , List<Vector3>> baseYValues , testYValues ;

// Sort vertices into height-partitioned lists
foreach vertex in baseVertices
    if vertex.y in baseYValues
        baseYValues[vertex.y].Add(vertex)
    else
        baseYValues.Add(vertex.y , vertex)

foreach vertex in testVertices
    if vertex.y in testYValues
        ... // As in above loop

// Perform distance calculation against matching height-partitioned lists
for i from 0 to baseVertices.Length
    minDistance = inf

    foreach yValue , list in testYValues
        if abs(baseVertices[i].y - yValue) < distanceThreshold
            // List is in partition , compare against that list's points
            for each vert in list
                distance = Vector3.Distance(baseVertices[i] , vert)
                if distance < minDistance
                    minDistance = distance

    baseDistances[i] = minDistance

// Repeat above loop for test vs. base comparison
...

```

Figure 8.13: The Height-Partitioned Hausdorff Distance.

Picking apart the looping structure, the outermost loop iterates over each point in the first cloud, or N times. The next nested loop iterates over the height partitions: these will vary based

on the print, as they correspond roughly to the number of printed layers, and can be labeled Z . Critically, this loop does no work, with all calculations taking place in the nested loop. The innermost loop iterates over every vertex in each matching height partition, calculating the distance. These are arranged so that, for the typical point, only one partition matches. Points that have drifted along the Z axis may match two layers, above and below. This is by design; there is no easy way to tell which layer they belong to without significant complication of the algorithm. In the typical case, the innermost loop iterates $\frac{N}{Z}$ times; the portion of the total points contained by the partition. For the outlier points, it may iterate $2 * \frac{N}{Z}$ times. Combining these loops, the entire structure gives

$$N * \frac{N}{Z} = \frac{N^2}{Z} \quad (8.1)$$

While this isn't a complete change from the naive Hausdorff approach, which also had $O(N^2)$ time complexity, the introduction of a coefficient Z is of great benefit. For the tested models, Z is approximately 140, which brought execution times down into workable ranges.

The Toolpath-Partitioned Hausdorff, as described in Section 8.1.3, exploits the sequential nature of the point cloud imposed by the toolpath to perform comparisons against a fixed window of points. Pseudocode for computing this metric is reproduced in Figure 8.14.

This algorithm is relatively easier to analyze: the outer loop iterates over the full length of the point cloud, N . The bounds of the inner loop must be calculated, but they have a fixed range of $2 * \text{toolpathradius}$, a constant I will refer to with R . The total structure therefore executes $R * N$ times, a linear scaling with coefficient R . Linear time complexity is a marked improvement over both the naive Hausdorff and the HPH, and this is borne out by the experimental work: TPH runs executed in seconds, where HPH runs executed in hours and naive runs were too time-consuming to run to completion.

Unfortunately, the optimization used for TPH renders the metric much more sensitive to drift. If issues with drift in the reconstruction stage can be tamed, the TPH approach is performant enough for any application up to and including online execution while printing.


```

Vector3[] baseVertices , testVertices;
float[] baseDistances , testDistances;

float toolpathRadius;

// Perform distance calculation against range of currentPosition +/- toolpathRadius ,
// adjusted for list length and avoiding index-out-of-range

for i from 0 to baseVertices.Length
    minDistance = inf

    // Find the centerpoint at the same relative position in the other cloud
    int otherCloudCenterpoint = ceiling(i / baseVertices.Length) * testVertices.Length

    int leftBound = (otherCloudCenterpoint - toolpathRadius > 0) ?
        otherCloudCenterpoint - toolpathRadius : 0

    int rightBound = (otherCloudCenterpoint + toolpathRadius < testVertices.Length) ?
        otherCloudCenterpoint + toolpathRadius : testVertices.Length

    for j from leftBound to rightBound
        distance = Vector3.Distance(baseVertices[i] , testVertices[j])
        if distance < minDistance
            minDistance = distance

    baseDistances[i] = minDistance

// Repeat above loop for test vs. base comparison
...

```

Figure 8.14: The Toolpath-Partitioned Hausdorff Distance.

Otherwise, the HPH optimization allows for offline use, but is unlikely to approach the speed of TPH.

Chapter 9

Discussion

9.1 Advantages and Drawbacks of Power Side-Channel

The clearest takeaway, across all of the experiments performed and findings of this dissertation, is that the power side-channel possesses high-precision and highly useful information about printer behavior. In much the same manner as the initial papers on Simple and Differential Power Analysis [33], I have shown that this information can be used for extremely effective attacks and defense measures in AM. While this dissertation focused on experiments with FDM technology, I see no reason to doubt its applicability to other AM technologies and indeed other modern manufacturing systems, though adjustments for the actuation signals of each new machine will be needed.

These claims are supported by the particular properties of the power side-channel revealed by my research. My examination of stepper motors instrumented via inductive current clamps shows direct correlation between the current delivered and motor actuation, at the full precision of the motor. The current traces proved to be legible both to human observers and to algorithmic approaches, both frequency based [23] and temporal [22]. The time-domain analysis in particular demonstrates the low-noise and high-fidelity of the signal, by enabling step-accurate reconstruction of the printed geometry. These properties clearly elevate the power side-channel above other channels explored in the literature, which are frequently hampered by high environmental noise, limited correlation, or missing essential actuation information.

Several other advantages are inherent to the use of side-channels. By capturing printer behavior at the very last stage of the printing process, long after leaving the cyber domain,

side-channel analysis is able to capture the results of any cyber-domain attack. This includes straightforward attacks, such as the modification of design files or attacks against software, but also includes more sophisticated attacks by compromised firmware.

My chosen instrumentation strategy, inductive current monitoring, is largely non-invasive in the AM system, requiring only physical access to otherwise unmodified wiring. The data-processing stages of all three presented techniques (detection, reconstruction, and forensics) are entirely independent of the computing hardware or software of the printing system. As a result, it is not only portable across many AM machines, but can be retrofitted onto legacy machines that were never designed with security features. Further, the monitoring system is well air-gapped from the monitored and potentially compromised machine, increasing the difficulty of simultaneous compromise.

The power side-channel also presented several difficulties that appear to be inherent to its use, and are worthy of discussion. Looking at the big picture, choosing a system control signal, like the actuator current used in this dissertation, locks one out of a ‘black-box’ approach to analysis. To achieve any meaningful results requires some degree of reverse-engineering of the AM system; in some cases, as demonstrated by the reconstruction algorithm in Chapter 7, that reverse engineering can be a time-consuming and tedious process. It need be performed only once for each model of AM system to be instrumented, but even this may be a huge burden.

Even with a solid engineering understanding of the instrumented system, the measured data requires expertise to work with. Electrical systems, especially complex and dense ones like AM systems, have substantial interference between their subsystems which can generate noise in many signals. The inductive measurement system eliminates a great deal of this, but the final signal still presents high-frequency noise that must be dealt with. In addition to noise, there are inherent stochastic deviations in signals even across identical prints: timing and signal level both can vary. Establishing thresholds for signal events often demands either extensive reverse-engineering or empirical study. The signal-processing knowledge to tame noise and variance without eliminating necessary signal data is essential, as this is difficult or impossible to automate with off-the-shelf solutions; facing this challenge requires as much engineering art as it does technical knowledge. Once in possession of well-filtered, highly accurate data,

there remains the problem of storing all of it. The signals involved in printing operate at fairly high speeds and must be adequately sampled for reliable results; at the same time, full 3D prints are conducted across multiple hours (for other printing technologies, even multiple days). Storing this data presented a challenge even at the lab experiment level, and for a larger-scale manufacturer would represent a significant expense and difficulty.

9.2 Application-Specific Results

In addition to properties of the power side-channel itself, my individual applications of the channel to tasks in AM Security revealed interesting properties of both the approaches and the printing process.

In the initial approach to signature generation and comparison (see Chapter 6), I adapted the use of Principal Component Analysis, a frequency-domain technique. It had previously been applied to the acoustic emanations of stepper motors, with positive results [9]. My results in applying the same approach to stepper motor current traces were also positive, as the resulting signatures were able to detect many categories of attacks. However, using a frequency-based approach masks substantial amounts of information available in the current side-channel that could be of great use in further analysis.

Temporal localization in frequency-based signatures is fundamentally limited to the windowing scheme chosen before converting your data into the frequency domain. Identifying trace features that occur in less time than your window duration is difficult or impossible; my later research demonstrated that key trace features such as reversals occur over periods in the low single milliseconds range, making them impossible to spot in the original signature. Further, the actual deviation in frequency content is often minimal and of even lower duration.

In total, frequency-based signatures on this side-channel are reasonable to apply as an early-detection measure, one demanding relatively low computation time and human effort. Where it fails, though, is in properly localizing or characterizing the detected attack. For this it is necessary to move on to more nuanced forms of analysis.

To make better use of the side-channel's high-quality data I first applied it to the hardest possible test: full reconstruction of the printed object (see Chapter 7). In its initial publication, I

presented this work in the context of a Technical Data Theft attack; it also forms the foundation of my subsequent work on forensic analysis. In pursuit of this research goal I uncovered many interesting and sometimes unexpected characteristics of the reconstruction process.

The earliest interesting result was the incredible precision and reliability of the current side-channel for this actuator type, and that despite this it remains challenging to work with. The relevant features of the current trace are plainly visible when plotted, are easy to distinguish even before filtering high-frequency noise, and map cleanly and unambiguously onto printer behavior. Nevertheless, programmatically recognizing the features and deriving the correct reconstruction is a challenging problem of signal analysis and processing. While the difficulty did not prevent me from producing results exceeding prior attempts in the field, the complexity is worth noting here because it limits offensive use of the side-channel to only the most potent of adversarial actors, and equally places demands on defensive use. As such, it seems more likely that any real-world application of this approach would be produced as a commodity or service, as with legitimate cyber-security providers of tools like firewalls, or offensively as with rootkit packages or spyware systems.

Despite the resultant error of reconstruction being small, it both propagates and accumulates over the course of the reconstructed print. This property is not unique to my reconstruction approach, but is shared by any system that uses a ‘dead-reckoning’ tracking system. As an example; inertial tracking systems (such as accelerometers) have no way of referencing the real position of the measured object, and thus accumulate error. GPS systems, while they may not be able to track positions down to the same precision, are constantly able to re-reference the measured object and thus have bounded, non-accumulating error. In my work, the result of this property is that relatively rare single-step errors produce offsets that endure throughout the rest of the print reconstruction, which are alternately made worse or compensated by subsequent errors. Either near-perfect accuracy (via more sophisticated signal processing) or some reference point to correct drift would substantially improve results for longer reconstructed prints. This was demonstrated by my subsequent work developing the side-channel based forensic analysis system.

Computing and visualizing comparison metrics for the forensics system, the cumulative drift was sufficient to render one method (Toolpath-Partitioned Hausdorff) inoperable in all but one test case. Even the more robust method, Height-Partitioned Hausdorff, was unable to overcome drift in lower-density prints; sparse, single-width infill lines become misaligned and are flagged as deviant. While the parameters of the comparison can be relaxed, such as comparison ranges and warning thresholds, this inevitably reduces the system's sensitivity to attacks. A less error-prone reconstruction stage would enable not only the analysis of sparse-infill prints, but the use of the more performant Toolpath-Partitioned Hausdorff metric.

9.3 Results on Error Stability

The side-channel data is non-deterministic, or rather, has stochastic variance. The data processing system is deterministic (does not rely on randomization), but is overly sensitive to the stochastic variance in the data. Specifically, I believe the heuristic solver is the origin of instability in reconstruction error. This stage is producing heuristic measures of potential peaks and peak removals, a process that relies on the specific height, prominence, and timing of trace sections that were already of low quality. To verify this would require observing that the points of divergence between reconstructions reliably occur within bad sections, rather than in un-flagged sections of trace.

It will be necessary to apply more advanced techniques from signal processing and analysis to reduce reliance on time-domain data in these instances. More adaptive filtering, based on measurements of the trace under analysis, should be better able to filter noise and resolve usable peaks. This will require collaboration with experts from the field of signal processing, for whom this is a domain-native problem and adequately solved by modern techniques.

Chapter 10

Future Work

This dissertation has shown that the actuator current side-channel is a powerful source of information, which can be used for both malicious data theft and sabotage defense. While my work has significantly advanced the state of the art, it has also revealed several further avenues for research.

10.1 Application to other AM Processes

The first and clearest line is to apply this technique, the non-invasive instrumentation of a printing system's actuators, to 3D printers using other AM processes. Fused Deposition Modeling, while both commonly used and presenting readily understood actuators, is not often deployed for functional parts in safety-critical systems: those roles are more often filled by metallic Direct Energy Deposition (DED) and Powder Bed Fusion (PBF) systems.

DED printing systems use a similar actuation strategy to FDM: a print-head, which is movable along multiple axes by a motor system, is used to deposit molten material at a point and form successive layers of an object. The print-head keeps a constant relative position to the point of fusion, much like an FDM print-head is always positioned at a certain height above where the filament will attach. The key difference, in the context of my method, between the two is the nature of the material feedstock and the energy applied to fuse it. Wirefeed DED systems operate almost identically to FDM, with motorized extrusion, but powder-fed DED systems use a gas-flow delivery system that would be controlled by multiple electro-mechanical nozzles and pressurized neutral gases. In addition, rather than a frontend heating element, DED

systems apply either a laser, and electron beam, or an arc-welding system to fuse material. Each of these are actuated differently, both from the electro-thermal nozzles of FDM and from each other. Finally, rather than stepper motors, servomotors are the commonly used positional actuator in industrial systems, owing to their higher precision, speed, and integrated feedback control mechanisms. Understanding and modeling these actuators from the perspective of their electrical control systems would be the main work necessary to adapt my methods to DED.

In PBF, a prepared powder surface across the entire print bed is selectively sintered or melted from a distance-typically by manipulating the path of a fixed-position laser through a galvanometer system, though some gantry-based systems also reposition the laser source. At its core, this is a very different positional actuation strategy from FDM and DED, requiring greater adjustments of the model used in my method to reconstruct the scanning path along the powder's surface. Further, the delay and separation between powder preparation and laser actuation means that two temporally distinct processes have to be correlated to understand the toolpath, where in FDM these processes occur simultaneously. The geometry involved in reaching different areas of the printbed with the laser, as well as the thermal characteristics of both laser path and melt pools, also introduces nonlinearities that must be accounted for.

In both of these cases, and in most industrial AM processes, it will be necessary to solve the problem of feedback control systems. The majority of industrial AM machines integrate some level of feedback control in their systems. Positional actuators such as servomotors, thermal systems like nozzles or chamber temperature, and more exotic systems like chamber atmosphere composition are only a few examples. As these channels are explored as a means of sabotage attack [49], it becomes necessary to integrate them into a sabotage defense.

10.2 Scalability Improvements

While the reconstruction and forensics algorithms produce useful results, their performance restricts them from 'online' forensics applications, that is, being run concurrently with the print. As detailed in the performance analysis section of Chapter 7, reconstruction presents linear scaling on everything but bad section length. Forensics, however, presents a larger issue. While the Toolpath-Partitioned Hausdorff achieves linear scaling, it isn't effective enough given the

current degree of error. The Height-Partitioned Hausdorff is more robust against the error, but only achieves quadratic scaling with a favorable coefficient. Given the additional complexities that online operation would present, such as recalculating distance metrics as new points are detected, quadratic scaling might not be sufficient.

Further, while these runtimes are manageable within FDM printing, other printing technologies often require substantially longer printing times. Some employ different, more sophisticated actuators that require higher sample rates and more data collected. Projecting the current performance of the algorithm out to the multi-day prints that are common in, for example, metallic PBF printing, shows that a significant performance improvement may be necessary for those cases. I see several avenues to achieve this.

First, the use of a more efficient and less error-prone signal analysis method for the initial peak-finding. Discounting the execution time of the CSV file input process, which only needs to be run once for the data's lifetime, the peak-finding process is the largest contributor to execution time in reconstruction. Not only this, the errors introduced by misread peaks are both the main risk for runaway execution times in the heuristic solver, they prevent the use of the optimally efficient Toolpath-Partitioned Hausdorff metric in the forensics stage. Replacing the current peak finder with a more advanced signal analysis approach would be an opportunity to improve the time scaling, remove the need for a heuristic solver, and enable the use of much more efficient comparison metrics all at once.

Beyond simply improving existing portions of the approach, there are more radical options available to improve performance. Many of the operations performed during reconstruction and forensics only require 'local' knowledge, rather than the entire dataset. Done with care, it should be possible to parallelize most of the existing codebase, enabling significant performance improvements even in the face of the larger datasets associated with other printing processes.

10.3 Purposeful Reconstruction Metrics

The metrics proposed for AM process reconstructions, either offensively or defensively, are wildly heterogeneous and mutually incompatible. This makes them ill-suited for comparing

the reported quality of reconstruction approaches. Further, the majority of metrics are tuned to judge the quality of the reconstruction process itself, rather than the reconstructed object. This makes it difficult or impossible to answer the questions raised in this dissertation, such as whether a particular reconstruction is sufficient for purposes like intelligence gathering or part infringement, or of a fine enough resolution to detect and analyse sabotage attacks.

In the traditional definition of side-channels, where it is applied to break cryptography or snoop on computations, the ‘ground truth’ being compared against is digital. It can be defined discretely and absolutely, even though in reality that digital definition is hiding a great deal of hardware complexity that is dealt with by computer engineers. In AM, the digital version of the ground truth, which might be defined as the original STL model or perhaps the toolpath command sequence, is not necessarily the target of a side-channel attack or the most relevant for a defender. The focal point of AM is the physical object itself, its properties, and its similarity to the intended design. The difficulty of designing metrics for physical objects from the perspective of a computer scientist is that, rather than claiming some number of correct bits, there are a vast number of measurable properties in the geometric domain alone.

Defenders have the unenviable task of determining the minimum level of attack that compromises their mission, for all the possible attacks against their chosen technology. The mapping of user concerns (regarding accuracy, physical properties, etc.) onto the effects of known AM attacks is an unsolved problem. There is a gap to be bridged here by researchers, and it must be done from both sides. From one side, reconstruction specialists must push forward to determine what can be detected about a printing process. From the other, offensive researchers must determine the attacks that have to be detected, for safety. Even if there is no way to have these thresholds perfectly overlap, knowing where they are will allow the integration of a safety factor into a design.

10.4 Adversarial Actors in AM

The attacks considered in this dissertation hinge on a small set of theoretical adversaries. I have attempted to keep these both as broad as possible, and reasonable within an attack’s context. However, because there have as yet been no publically disclosed real-world attacks against

AM, these adversaries are purely hypothetical. While it does not detract from the conclusions of this dissertation, or indeed from the many other technical publications in AM Security that have adopted the same approach, I see the need to put technical security research in a realistic adversarial context. As it stands, two clear avenues to improvement stand out to me.

First, it is possible to build upon the adversarial modeling research conducted in other security domains, such as pure cybersecurity, including computer security, network security, and cloud security, as well as more closely related domains such as Industrial Control Systems. In addition, many of the potential real-world adversarial actors in AM have been extensively studied in works of national security, international relations, nuclear non-proliferation, and criminology. All these have identified a variety of adversarial organizations and individuals, their motives, methods, and capabilities. With further study and research, this wealth of knowledge can be projected to provide context to adversarial models in AM, allowing for sound judgement of attack feasibility and more realistic analysis of risk.

While a recent anonymized survey of AM adopters indicates there have been AM-specific attacks [74], all organizations reporting attacks declined to disclose any details. This pattern of behavior is similar to what was seen in the early days of Industrial Control Systems security, and is likely motivated by the same desire to avoid publicity, negative impacts on reputation, and the leaking of proprietary information about impacted systems. This problem was eventually mitigated in the United States when reporting and analysis of such infrastructure-level incidents was brought under the control of the Cybersecurity and Infrastructure Security Agency (CISA). By making reporting mandatory (at penalty under federal law) and providing government support for attack mitigation and information security regarding disclosures, the overall security posture of ICS was greatly improved. While it may not be possible to implement the same system for AM, owing to major differences in stakeholder relationships compared to large, static forms of critical infrastructure, it is clear that some form of information sharing could greatly benefit all participants in the AM industry. Such a system would ideally allow for disclosing security incidents under specific constraints, such as anonymity or the withholding of proprietary information. Allowing information about real-world attacks to spread to security professionals is the only way to strengthen AM security.

Chapter 11

Conclusion

The power side-channel investigated in this dissertation has been proven a potent source of high-detail information for both offensive and defensive applications. My initial signature approach both yielded a viable strategy for sabotage detection in AM and indicated that the power side-channel would be fertile ground for my subsequent research. I then paired the channel with a custom algorithm to yield highly detailed reconstructions of printed objects, then further developed that reconstruction system into a full forensic investigation technique for sabotage attacks.

The presented approach provides a method for ‘last-mile’ defense of AM systems which is applicable to existing AM machines, easily partitioned from other compromisable systems, and independent of the hardware and software in use. While I have evaluated the approach for the FDM process, I have presented a clear path for applying my approach to other existing categories of AM process. Further, my presentation of an attack using the very same channel revealed an overlooked and severe risk in a common industry practice: outsourcing. The shown attack, and my analysis of its properties, will improve the ability of practitioners to evaluate risks in AM production more realistically. Finally, my application of reconstruction to the investigation and analysis of sabotage attacks works to fill a major gap in AM defensive capabilities. Overall, this dissertation is a significant step towards securing the AM manufacturing base.

References

- [1] akhani3D. akhani3D Production Additive Manufacturing Service Bureau Main Webpage. <https://akhani3d.com>, 2020.
- [2] Mohammad Abdullah Al Faruque, Sujit Rokka Chhetri, A Canedo, and J Wan. Forensics of thermal side-channel in additive manufacturing systems. *CECS Technical Report 16-01*, 2016.
- [3] Mohammad Abdullah Al Faruque, Sujit Rokka Chhetri, Arquimedes Canedo, and Jiang Wan. Acoustic Side-Channel Attacks on Additive Manufacturing Systems. In *2016 ACM/IEEE 7th international conference on Cyber-Physical Systems (ICCPS)*, pages 1–10. IEEE, 2016.
- [4] Mohammad Abdullah Al Faruque, Jiang Wan, and Sujit Rokka Chhetri. Defending Side Channel Attacks in Additive Manufacturing Systems, February 19 2019. US Patent 10,212,185.
- [5] Mohammed Albakri, Logan Sturm, Christopher B Williams, and Pablo Tarazaga. Non-Destructive Evaluation of Additively Manufactured Parts via Impedance-Based Monitoring. In *Solid Freeform Fabrication Symposium*, pages 1475–1490, 2015.
- [6] Sakthi Kumar Arul Prakash, Tobias Mahan, Glen Williams, Christopher McComb, Jessica Menold, and Conrad S Tucker. Detection of System Compromise in Additive Manufacturing using Video Motion Magnification. *Journal of Mechanical Design*, 142(3), 2020.
- [7] ASTM. *ASTM F2792-12a, Standard Terminology for Additive Manufacturing Technologies (Withdrawn)*. 2015.

- [8] Christian Bayens, Tuan Le, Luis Garcia, Raheem Beyah, Mehdi Javanmard, and Saman 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)*, pages 1181–1198, Vancouver, BC, August 2017. USENIX Association.
- [9] Sofia Belikovetsky, Yosef A Solewicz, Mark Yampolskiy, Jinghui Toh, and Yuval Elovici. Digital Audio Signature for 3D Printing Integrity. *IEEE Transactions on Information Forensics and Security*, 14(5):1127–1141, 2019.
- [10] Sofia Belikovetsky, Mark Yampolskiy, Jinghui Toh, Jacob Gatlin, and Yuval Elovici. dr0wned – Cyber-Physical Attack with Additive Manufacturing. In *11th USENIX Workshop on Offensive Technologies (WOOT 17)*, page 16, Vancouver, BC, 2017. USENIX Association.
- [11] Sujit Rokka Chhetri, Anomadarshi Barua, Sina Faezi, Francesco Regazzoni, Arquimedes Canedo, and Mohammad Abdullah Al Faruque. Tool of Spies: Leaking your IP by Altering the 3d Printer Compiler. *IEEE Transactions on Dependable and Secure Computing*, 18(2):667–678, 2019.
- [12] Sujit Rokka Chhetri, Arquimedes Canedo, and Mohammad Abdullah Al Faruque. KCAD: Kinetic Cyber-Attack Detection Method for Cyber-Physical Additive Manufacturing Systems. In *Proceedings of the 35th International Conference on Computer-Aided Design*, page 74. ACM, 2016.
- [13] Sujit Rokka Chhetri, Arquimedes Canedo, and Mohammad Abdullah Al Faruque. Confidentiality Breach through Acoustic Side-Channel in Cyber-Physical Additive Manufacturing Systems. *ACM Transactions on Cyber-Physical Systems*, 2(1):1–25, 2017.
- [14] Sujit Rokka Chhetri, Sina Faezi, and Mohammad Abdullah Al Faruque. Fix the Leak! An Information Leakage Aware Secured Cyber-Physical Manufacturing System. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*, pages 1408–1413. IEEE, 2017.

- [15] Sujit Rokka Chhetri, Sina Faezi, and Mohammad Abdullah Al Faruque. Information Leakage-Aware Computer-Aided Cyber-Physical Manufacturing. *IEEE Transactions on Information Forensics and Security*, 13(9):2333–2344, 2018.
- [16] Sujit Rokka Chhetri, Jiang Wan, and Mohammad Abdullah Al Faruque. Cross-domain security of cyber-physical systems. In *Design Automation Conference (ASP-DAC), 2017 22nd Asia and South Pacific*, pages 200–205. IEEE, 2017.
- [17] Joel Dawson, Michael Iannacone, Srikanth Yoginath, Varisara Tansakul, Rob Jordan, Ali Passian, Joel Asiamah, Milton Nance Ericson, and Gavin Long. Control-Theory-Informed Feature Selection for Detecting Malicious Tampering in Additive Layer Manufacturing Processes. In *Proceedings of the 16th International Conference on Cyber Warfare and Security*, pages 55–64, 2021.
- [18] Karim A ElSayed, Adam Dachowicz, and Jitesh H Panchal. Information Embedding in Additive Manufacturing through Printing Speed Control. In *Proceedings of the 2021 Workshop on Additive Manufacturing (3D Printing) Security*, pages 31–37, 2021.
- [19] Sriharsha Etigowni, Sizhuang Liang, Saman Zonouz, and Raheem Beyah. Physics-Aware Security Monitoring against Structural Integrity Attacks in 3D Printers. In *2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 507–518. IEEE, 2021.
- [20] Matteo Frigo and Steven G Johnson. FFTW: An Adaptive Software Architecture for the FFT. In *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on*, volume 3, pages 1381–1384. IEEE, 1998.
- [21] Yang Gao, Borui Li, Wei Wang, Wenyao Xu, Chi Zhou, and Zhanpeng Jin. Watching and Safeguarding your 3D Printer: Online Process Monitoring against Cyber-Physical Attacks. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 2(3):1–27, 2018.

- [22] Jacob Gatlin, Sofia Belikovetsky, Yuval Elovici, Anthony Skjellum, Joshua Lubell, Paul Witherell, and Mark Yampolskiy. Encryption is Futile: Reconstructing 3D-Printed Models using the Power Side-Channel. In *Proceedings of the 24th International Symposium on Research in Attacks, Intrusions and Defenses*, pages 135–147, 2021.
- [23] Jacob Gatlin, Sofia Belikovetsky, Samuel B Moore, Yosef Solewicz, Yuval Elovici, and Mark Yampolskiy. Detecting Sabotage Attacks in Additive Manufacturing using Actuator Power Signatures. *IEEE Access*, 7:133421–133432, 2019.
- [24] General Electric. The FAA Cleared The First 3D Printed Part To Fly In A Commercial Jet Engine From GE. Technical report, 2015.
- [25] Ian Gibson, David Rosen, Brent Stucker, Mahyar Khorasani, David Rosen, Brent Stucker, and Mahyar Khorasani. *Additive Manufacturing Technologies*, volume 17. Springer, 2021.
- [26] L Graves, WE King, P Carrion, S Shao, N Shamsaei, and M Yampolskiy. Sabotaging Metal Additive Manufacturing: Powder Delivery System Manipulation and Material-Dependent Effects. *Additive Manufacturing*, 46:1020–1029, 2021.
- [27] Lynne MG Graves, Joshua Lubell, Wayne King, and Mark Yampolskiy. Characteristic Aspects of Additive Manufacturing Security From Security Awareness Perspectives. *IEEE Access*, 7:103833–103853, 2019.
- [28] N Gupta, F Chen, K Shahin, TS Srivatsan, TS Sudarshan, and K Manigandan. Design Features to Address Security Challenges in Additive Manufacturing. In *Manufacturing Techniques for Materials: Engineering and Engineered*. CRC Press, 2018.
- [29] Nikhil Gupta, Fei Chen, Nektarios Georgios Tsoutsos, and Michail Maniatakos. ObfusCADE: Obfuscating Additive Manufacturing CAD Models against Counterfeiting. In *Proceedings of the 54th Annual Design Automation Conference 2017*, pages 1–6, 2017.
- [30] Felix Hausdorff. *Grundzüge der Mengenlehre*, volume 7. von Veit, 1914.

- [31] Avesta Hojjati, Anku Adhikari, Katarina Struckmann, Edward Chou, Thi Ngoc Tho Nguyen, Kushagra Madan, Marianne S Winslett, Carl A Gunter, and William 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*, pages 883–894, 2016.
- [32] Identify3D. Identify3D Product Statement. <https://identify3d.com/>, 2023.
- [33] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential Power Analysis. In *Annual International Cryptology Conference*, pages 388–397. Springer, 1999.
- [34] Sizhuang Liang and Raheem Beyah. Poster: A Realizable Framework for Intrusion Detection in Additive Manufacturing Systems Using Analog Side-Channels. In *Network and Distributed Security Symposium*, 2019.
- [35] Sizhuang Liang, Xirui Peng, H Jerry Qi, Saman Zonouz, and Raheem Beyah. A Practical Side-Channel Based Intrusion Detection System for Additive Manufacturing Systems. In *2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS)*, pages 1075–1087. IEEE, 2021.
- [36] Samuel B Moore, Jacob Gatlin, Sofia Belikovetsky, Mark Yampolskiy, Wayne E King, and Yuval Elovici. Power Consumption-based Detection of Sabotage Attacks in Additive Manufacturing. *arXiv preprint arXiv:1709.01822*, 2017.
- [37] Samuel Bennett Moore, William Bradley Glisson, and Mark Yampolskiy. Implications of Malicious 3D Printer Firmware. In *Proceedings of the 50th Hawaii International Conference on System Sciences*, pages 6089–6098, 2017.
- [38] MxD. Playbook for CMMC 2.0 Level 1. Technical report, 2023.
- [39] Ltd. New Japan Radio Co. Stepper Motor Basics. Technical report, 2020.
- [40] Zachary Oligschlaeger, Benjamin Baltes, and Jennifer Chin. Secure 3D Printing, October 2020. U.S. Patent no. 20200326683.

- [41] Gregory Pope and Mark Yampolskiy. Presentation: A Hazard Analysis Technique for Additive Manufacturing. In *Better Software East Conference*, 2016.
- [42] Sakthi Kumar Arul Prakash, Tobias Mahan, Glen Williams, Christopher McComb, Jessica Menold, and Conrad S Tucker. On the Analysis of a Compromised Additive Manufacturing System Using Spatio-Temporal Decomposition. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, volume 59193, page V02BT03A018. American Society of Mechanical Engineers, 2019.
- [43] Muhammad Haris Rais, Ye Li, and Irfan Ahmed. Spatiotemporal G-Code Modeling for Secure FDM-Based 3D Printing. In *Proceedings of the ACM/IEEE 12th International Conference on Cyber-Physical Systems*, pages 177–186, 2021.
- [44] Bikash Ranabhat, Joseph Clements, Jacob Gatlin, Kuang-Ting Hsiao, and Mark Yampolskiy. Optimal sabotage attack on composite material parts. *International Journal of Critical Infrastructure Protection*, 26:100301, 2019.
- [45] RapidDirect. RapidDirect Online CNC Machining and Prototype Manufacturing Service. <https://www.rapiddirect.com>, 2020.
- [46] SciPy.org. Documentation of `scipy.signal.find_peaks`. <https://docs.scipy.org/>, 2020.
- [47] NXP Semiconductors. Application Note 2974: Quick Start for Beginners to Drive Stepper Motor. <https://www.nxp.com/docs/en/application-note/AN2974.pdf>, 2005.
- [48] Zhangyue Shi, Abdullah Al Mamun, Chen Kan, Wenmeng Tian, and Chenang Liu. An LSTM-Autoencoder-Based Online Side Channel Monitoring Approach for Cyber-Physical Attack Detection in Additive Manufacturing. *Journal of Intelligent Manufacturing*, pages 1–17, 2022.

- [49] Andrew Slaughter, Mark Yampolskiy, Manyalibo Matthews, Wayne E King, Gabe Guss, and Yuval Elovici. How to Ensure Bad Quality in Metal Additive Manufacturing: In-Situ Infrared Thermography from the Security Perspective. In *Proceedings of the 12th International Conference on Availability, Reliability and Security*, page 78. ACM, 2017.
- [50] Chen Song, Feng Lin, Zhongjie Ba, Kui Ren, Chi Zhou, and Wenyao 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*, pages 895–907, 2016.
- [51] Jinwoo Song, Harika Bandaru, Xinyu He, Zhenyang Qiu, and Young B Moon. Layered Image Collection for Real-Time Defective Inspection in Additive Manufacturing. In *ASME International Mechanical Engineering Congress and Exposition*, volume 84492, page V02BT02A006. American Society of Mechanical Engineers, 2020.
- [52] Jinwoo Song and Young B Moon. Infill Defective Detection System Augmented by Semi-Supervised Learning. In *ASME International Mechanical Engineering Congress and Exposition*, volume 84492, page V02BT02A005. American Society of Mechanical Engineers, 2020.
- [53] Jinwoo Song, Chunxi Wang, Charl lie Saudrais, Matthew K Swanson, Emily Ann Greaney, and Young B Moon. Cyber-Manufacturing System Testbed Development: Adversarial Insider Manipulation. *Procedia CIRP*, 93:180–185, 2020.
- [54] L Sturm, CB Williams, JA Camelio, J White, and R Parker. Cyber-Physical Vulnerabilities in Additive Manufacturing Systems. *Context*, 7:8, 2014.
- [55] Logan Sturm, Mohammed Albakri, Christopher B Williams, and Pablo Tarazaga. In-situ Detection of Build Defects in Additive Manufacturing via Impedance-Based Monitoring. In *2016 International Solid Freeform Fabrication Symposium*, pages 1458–1478, 2016.
- [56] Unity Technologies. Unity Real-Time Development Platform Homepage. <https://unity.com/>.

- [57] thyssenkrupp. thyssenkrupp 3D-Druck TechCenter: Additive Manufacturing. <https://www.thyssenkrupp-additive-manufacturing.com>, 2020.
- [58] Treatstock. Treatstock Smart Manufacturing Platform. <https://www.treatstock.com>, 2020.
- [59] Nektarios Georgios Tsoutsos, Homer Gamil, and Michail 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*, pages 15–20. ACM, 2017.
- [60] Hamilton Turner, Jules White, Jaime A Camelio, Christopher Williams, Brandon Amos, and Robert Parker. Bad Parts: Are our Manufacturing Systems at Risk of Silent Cyberattacks? *IEEE Security & Privacy*, 13(3):40–47, 2015.
- [61] Chief Information Officer U.S. Department of Defense. Cybersecurity Maturity Model Certification 2.0 Overview. Technical report, 2021.
- [62] Mohammad Vaezi and Chee Kai Chua. Effects of Layer Thickness and Binder Saturation Level Parameters on 3D Printing Process. *The International Journal of Advanced Manufacturing Technology*, 53(1-4):275–284, 2011.
- [63] Hannah Vincent, Lee Wells, Pablo Tarazaga, and Jaime Camelio. Trojan Detection and Side-Channel Analyses for Cyber-Security in Cyber-Physical Manufacturing Systems. *Procedia Manufacturing*, 1:77–85, 2015.
- [64] Terry Wohlers. Wohlers Report 2016: 3D Printing and Additive Manufacturing State of the Industry Annual Worldwide Progress Report. Technical report, 2016.
- [65] Terry Wohlers. Wohlers Report 2020: 3D Printing and Additive Manufacturing State of the Industry Annual Worldwide Progress Report. Technical report, 2020.
- [66] Mingtao Wu and Young Moon. Alert Correlation for Cyber-Manufacturing Intrusion Detection. *Procedia Manufacturing*, 34:820–831, 2019.

- [67] Mingtao Wu and Young B Moon. Intrusion Detection of Cyber-Physical Attacks in Manufacturing Systems: A Review. In *ASME International Mechanical Engineering Congress and Exposition*, volume 59384, page V02BT02A001. American Society of Mechanical Engineers, 2019.
- [68] Mingtao Wu and Young B Moon. Alert Correlation for Detecting Cyber-Manufacturing Attacks and Intrusions. *Journal of Computing and Information Science in Engineering*, 20(1):011004, 2020.
- [69] Mingtao Wu, Jinwoo Song, Long Wang Lucas Lin, Noé Aurelle, Yapan Liu, Bingyan Ding, Zhengyi Song, and Young B Moon. Establishment of Intrusion Detection Testbed for CyberManufacturing Systems. *Procedia Manufacturing*, 26:1053–1064, 2018.
- [70] Mingtao Wu, Jinwoo Song, Snehav Sharma, Jupeng Di, Benliu He, Ziming Wang, Jingkai Zhang, Long Wang Lucas Lin, Emily Ann Greaney, and Young Moon. Development of Testbed for Cyber-Manufacturing Security Issues. *International Journal of Computer Integrated Manufacturing*, 33(3):302–320, 2020.
- [71] Mingtao Wu, Zhengyi Song, and Young B Moon. Detecting Cyber-Physical Attacks in CyberManufacturing Systems with Machine Learning Methods. *Journal of Intelligent Manufacturing*, pages 1–13, 2017.
- [72] Xiao Zi Hang (Claud Xiao). Security Attack to 3D Printing. <http://www.claudxiao.net/Attack3DPrinting-Claud-en.pdf>, 2013. Keynote at XCon2013.
- [73] M Yampolskiy, TR Andel, JT McDonald, WB Glisson, and A Yasinsac. Towards Security of Additive Layer Manufacturing. WiP presented at The 30th Annual Computer Security Applications Conference (ACSAC) 2014, 2014.
- [74] Mark Yampolskiy, Paul Bates, Mohsen Seifi, and Nima Shamsaei. State of Security Awareness in the AM Industry: 2020 Survey, 2022. arXiv.

- [75] Mark Yampolskiy, Jacob Gatlin, and Moti Yung. Myths and Misconceptions in Additive Manufacturing Security: Deficiencies of the CIA Triad. In *Proceedings of the 2021 Workshop on Additive Manufacturing (3D Printing) Security*, pages 3–9, 2021.
- [76] Mark Yampolskiy, Wayne E King, Jacob Gatlin, Sofia Belikovetsky, Adam Brown, Anthony Skjellum, and Yuval Elovici. Security of Additive Manufacturing: Attack Taxonomy and Survey. *Additive Manufacturing*, 21:431–457, 2018.
- [77] Mark Yampolskiy, Lena Schutzle, Uday Vaidya, and Alec Yasinsac. Security Challenges of Additive Manufacturing with Metals and Alloys. In *Critical Infrastructure Protection IX*, pages 169–183. Springer, 2015.
- [78] Wei Yang, Jialei Chen, Chuck Zhang, and Kamran Paynabar. Online Detection of Cyber-Incidents in Additive Manufacturing Systems via Analyzing Multimedia Signals. *Quality and Reliability Engineering International*, 38(3):1340–1356, 2022.
- [79] Shih-Yuan Yu, Arnav Vaibhav Malawade, Sujit Rokka Chhetri, and Mohammad Abdullah Al Faruque. Sabotage Attack Detection for Additive Manufacturing Systems. *IEEE Access*, 8:27218–27231, 2020.
- [80] Steven Eric Zeltmann, Nikhil Gupta, Nektarios Georgios Tsoutsos, Michail Maniatakos, Jeyavijayan Rajendran, and Ramesh Karri. Manufacturing and Security Challenges in 3D Printing. *JOM*, pages 1–10, 2016.
- [81] Houliang Zhou, Chenang Liu, Wenmeng Tian, and Chen Kan. Echo State Network Learning for the Detection of Cyber Attacks in Additive Manufacturing. In *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*, pages 177–182. IEEE, 2021.
- [82] Theo Zinner, Grant Parker, Nima Shamsaei, Wayne King, and Mark Yampolskiy. Spooky Manufacturing: Probabilistic Sabotage Attack in Metal AM using Shielding Gas Flow Control. In *Proceedings of the 2022 ACM CCS Workshop on Additive Manufacturing (3D Printing) Security*, pages 15–24, 2022.