SECURING ADDITIVE MANUFACTURING SYSTEMS FROM CYBER AND INTELLECTUAL PROPERTY ATTACKS

A Dissertation
Presented to
The Academic Faculty

By

Sizhuang Liang

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology

December 2021

SECURING ADDITIVE MANUFACTURING SYSTEMS FROM CYBER AND INTELLECTUAL PROPERTY ATTACKS

Thesis committee:

Dr. Raheem Beyah
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Fatih Sarioglu
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Saman Zonouz
Department of Electrical and Computer Engineering
Rutgers University

Dr. Aaron Stebner
George W. Woodruff School of Mechanical Engineering
Georgia Institute of Technology

Dr. Nikhil Gupta
Tandon School of Engineering
New York University

Date approved: October 13, 2021

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my advisor Prof. Raheem Beyah for his continued support for my study and research, for his patience, motivation, insights, and immense knowledge. Without the support of Prof. Raheem Beyah, this research effort would not be possible. His guidance helped me to find important and interesting security challenges that needed to be addressed in the field of additive manufacturing and helped me to tackle those challenges effectively.

I would like to thank my thesis committee for their time and efforts on this thesis. I would like to thank Prof. Saman Zonouz for his continued collaboration throughout my PhD career. I would like to thank Prof. Fatih Sarioglu, Prof. Aaron Stebner, and Prof. Nikhil Gupta for their innovative research that motivated me to think out of the box.

I would like to thank my parents, Taoxiang Chen and Wenbin Liang, for their deep love and unwavering support. I would also like to thank my friends in the lab, David Formby, Qinchen Gu, Celine Irvene, Tohid Shekari, Anni Zhou, and Binbin Zhao, for their help.

Finally, I would like to acknowledge National Science Foundation for their support on this work under grant 1739259 and 1931977. All views and conclusions contained in this thesis are those of the author, and do not necessarily represent the official positions, express or implied, of the funder.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYMS

AM	Additive Manufacturing
BAAM	Big Area Additive Manufacturing
DTW	Dynamic Time Warping
DWM	Dynamic Window Matching
FNR	False Negative Rate
FP	False Positive
FPR	False Positive Rate
IDS	Intrusion Detection System
IP	Intellectual Property
MFCC	Mel-Frequency Cepstral Coefficients
NDT	Non-Destructive Testing
OCC	One-Class Classification
PCA	Principle Component Analysis
PWM	Pulse Width Modulation
RM3	Rostock Max V3
SD	Signal Detection
SNR	Signal to Noise Ratio
TDE	Time Delay Estimation
TP	True Positive
TPR	True Positive Rate
UM3	Ultimaker 3

SUMMARY

Additive Manufacturing (AM), also known as 3D printing, refers to a collection of manufacturing processes where materials are joined together layer by layer to make objects directly from 3D models. Due to many advantages of AM, such as rapid prototyping, massive customization, material saving, and flexibility of designs, there is a trend for AM to replace traditional manufacturing processes. However, AM highly relies on computers to work. As AM systems are gaining popularity in many critical industry sectors, there is an increased risk of cyberattacks on AM systems.

To protect AM systems from cyberattacks that aim to sabotage the AM systems, Intrusion Detection Systems (IDSs) can be used. In recent years, researchers proposed a series of IDSs that work by leveraging side-channel signals. A side-channel signal is typically a physical signal that is correlated with the state of the AM system, such as the acoustic sound or the electromagnetic wave emitted by a 3D printer in a printing process. Because of the correlation between a side-channel signal and the state of a 3D printer, it is possible to perform intrusion detection by analyzing the side-channel signal. In fact, most existing IDSs leveraging side-channel signals in AM systems function by comparing an observed side-channel signal against a reference side-channel signal.

However, we found that these IDSs are not practical due to a lack of synchronization. Many IDSs in the literature do not contain details on how to align two (or more) side-channel signals at their starting moments and their stopping moments. In addition, we found that there is time noise in AM processes. When the same G-code file is executed on the same 3D printer multiple times, the printing processes will have slightly different timing. Because of time noise, a direct comparison between two signals point by point or window by window will not make sense. To overcome this problem, we propose to use dynamic synchronization to find corresponding points between two signals in real time. To demonstrate the necessity of dynamic synchronization, we performed a total of 302 benign printing processes and a total of 200 malicious printing processes with two printers. Our experiment results show that existing IDSs leveraging side-channel signals in AM systems can only achieve an accuracy from 0.50 to 0.88, whereas our IDS with dynamic synchronization can reach an accuracy of 0.99.

Other than cyberattacks to sabotage AM systems, there are also cyberattacks to steal intellectual property in AM systems. For example, there are acoustic side-channel attacks on AM systems which can recover the printing path by analyzing the acoustic sound by a printer in a printing process. However, we found that the acoustic side-channel attack is hard to perform due to challenges such as integration drift and non-unique solution. In this thesis, we explore the optical side-channel attack, which is much easier to perform than the acoustic side-channel attack. The optical side-channel signal is basically the video of a printing process. We use a modified deep neural network, ResNet50, to recognize the coordinates of the printhead in each frame in the video.

To defend against the optical side-channel attack, we propose the optical noise injection method. We use an optical projector to artificially inject crafted optical noise onto the printing area in an attempt to confuse the attacker and make it harder to recover the printing path. We found that existing noise generation algorithms, such as replaying, random blobs, white noise, and full power, can effortlessly defeat a naive attacker who is not aware of the existence of the injected noise. However, an advanced attacker who knows about the injected noise and incorporates images with injected noise in the training dataset can defeat all of the existing noise generation algorithms. To defend against such an advanced attacker, we propose three novel noise generation algorithms: channel uniformization, state uniformization, and state randomization. Our experiment results show that noise generated via state randomization can successfully defeat the advanced attacker.

CHAPTER 1
INTRODUCTION

1.1 Additive Manufacturing and its Security Challenges

Additive Manufacturing (AM), also known as 3D Printing, is gaining popularity in a variety of critical industrial sectors, such as automobile [1], aerospace [2], construction [3], and medicine [4]. For example, researchers in Oak Ridge National Laboratory used a Big Area Additive Manufacturing (BAAM) system to print a motor car [1]. Airbus used AM to produce titanium bracket connectors for its Airbus A350 XWB for improved reliability and reduced weight [2]. Apis Cor built the world's largest 3D-printed building, a two-story municipal administrative building with a height of 31 feet and an area of 900 square feet [3]. Researchers in University of Michigan applied AM to manufacture a biodegradable scaffolding, which can be attached to the outside of a trachea to treat tracheomalacia [4]. According to the Wohlers Report 2019, the AM industry, consisting of all AM products and services worldwide, will reach \$15.8 billion in 2020, \$23.9 billion in 2022, and \$35.6 billion in 2024 [5].

As the application of AM grows rapidly in critical industrial sectors, cyberattacks to sabotage AM systems is becoming a concern. For example, Moore et al. analyzed the vulnerability of common 3D printing software applications, such as Cura, ReplicatorG, and Repetier-Host [6]. They found several vulnerabilities in these applications that can lead to cyberattacks. Do et al. performed cyberattacks on two MakerBot 3D printers [7]. They found vulnerabilities in the network protocol used in these printers and were able to take control of the printers to exfiltrate sensitive data and remotely manipulate the printers. Moore et al. explored the implications of malicious 3D printer firmware [8]. When the firmware of a 3D printer is tempered with by an attacker, it is possible for the printer to print an object with defects even if the printer is sent benign G-code instructions. Sturm et al. demonstrated that, by inserting small defects, such as voids, into the design file (such as an STL file), the structural integrity of the printed object can be degraded [9]. Later, Yampolskiy et al. presented the idea of performing cyberattacks on AM systems in an attempt to compromise the structural integrity of the printed objects [10]. When a printed object with defects is put in operation, its failure could result in the whole system with the object to fail, causing damage to property and life. To put this idea into practice, Belikovetsky et al. performed a complete cyberattack on an AM system that manufactured propellers for drones [11]. They secretly modified the design file to introduce small gaps between the hub of the propeller and the blades. The manufactured propeller looked identical to a normal propeller and could operate normally for a short period of time. The propeller then broke in flight, causing the uncontrollable drone to fall from the sky.

As the application of AM increases rapidly, the protection of Intellectual Property (IP) associated with AM is also becoming a concern [12, 13, 14, 15]. For example, it could take years for a company to design a product to be additively manufactured and then sell the product for a profit [16]. Losing the IP could mean a financial loss. Meanwhile, there are designs that should be strictly controlled. An example is 3D printed firearms [17]. If the design of a firearm is leaked, it may lead to unlawful production of the firearm. In recent years, there emerges a series of side-channel attacks to steal IP in AM systems. In 2016, Al Faruque et al., Hojjati et al., and Song et al. respectively proposed cyberattacks leveraging the acoustic and magnetic side channels to steal IP in AM systems [18, 19, 20]. Although the methods are different in technical details, they can determine the outline of an object that is being printed by analyzing the acoustic and magnetic side-channel signals collected by a cellphone near the printer.

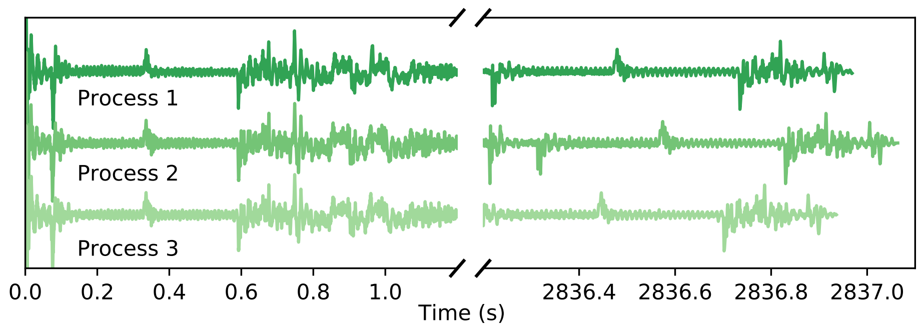


Figure 1.1: Side-channel signals for three printing processes using the same G-code file and the same printer. The signals are aligned at the beginning. The misalignment in the end is caused by time noise.

1.2 Intrusion Detection with Side Channels

Side channels are gaining popularity for intrusion detection in AM systems due to their non-invasiveness and often air-gapped threat models [21, 22, 23, 24, 25, 26, 27]. To perform intrusion detection in AM systems, many existing Intrusion Detection Systems (IDSs) compare an observed side-channel signal against a reference signal point by point or window by window. For each pair of points or windows, a distance is calculated between the signals to determine if they are similar or not [21, 22, 23, 24, 27, 26] (the acoustic layer in [22]), or a classifier can be employed to determine whether the signals are alike or not [22] (the spatial layer in [22]).

This approach to compare signals works well if the two signals are aligned for every pair of points or windows. It appears that when the same printing process is performed on the same AM system, the timing should be the same. If this assumption is true, when an observed signal is aligned with a reference signal at the beginning, they should be aligned at other points. However, our experiments show that this assumption is not true. Figure 1.1 shows side-channel signals from three printing processes with the same G-code file and the same printer. Although the side-channel signals are aligned at the beginning, they do not end at the same time.

AM systems are asynchronous. When executed multiple times, the duration for the

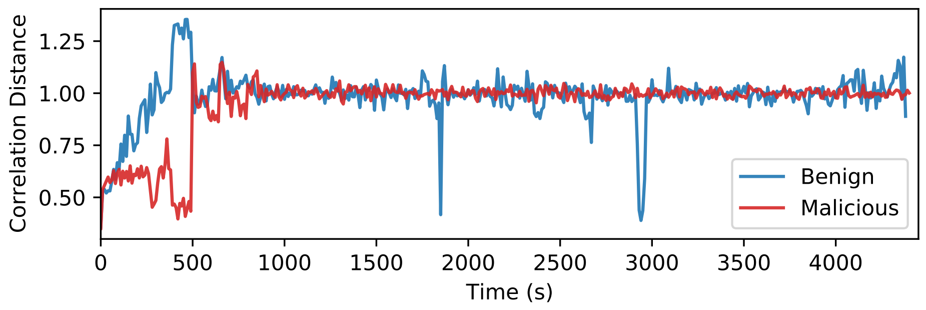


Figure 1.2: Correlation distances of a benign printing process and a malicious printing process. Due to time noise, the distances of a benign printing process are very large and could be larger than the distances of a malicious printing process.

same instruction can vary slightly. In addition, there can be random gaps between instructions. This random variation in timing is referred to as *time noise*. Time noise can be a result of frame drops in data acquisition systems, mechanical and thermal delays in devices, and task scheduling in operating systems (if equipped).

Time noise can invalidate any IDS that is based on comparing a side-channel signal against a reference signal point by point or window by window. When comparing two signals that are out of alignment, the distances can become very large. Figure 1.2 shows the correlation distances of a benign process and a malicious process. Without the consideration of time noise, we can see that the distances of the benign process are as large as the distances of the malicious process.

A practical IDS using side channels must be able to tolerate time noise. One approach to tolerate time noise is to dynamically synchronize two signals. An existing method to do so is called Dynamic Time Warping (DTW) [28, 29]. However, DTW does not natively support real-time operations, consumes an excessive amount of computational resources, and has limited accuracy for side-channel signals in AM systems, as will be demonstrated in section 3.12.

To overcome the problems of DTW, we propose a novel algorithm called Dynamic Window Matching (DWM) to replace DTW. DWM finds the timing relationship between two signals by establishing a sliding window for each signal. As the pair of windows slides across the signals, the relative displacement (a.k.a. the horizontal displacement) between the windows is determined by Time Delay Estimation (TDE). To stabilize the process, we introduce biases in the TDE process, and we introduce inertial when adjusting the relative displacement between the windows. DWM is a window-by-window algorithm. In contrast, DTW is a point-by-point algorithm.

For a complete IDS, we propose NSYNC (Noise SYNC), a framework to practically compare a side-channel signal against a reference signal for real-time intrusion detection in AM systems. NSYNC first of all aligns two signals at the beginning by static synchronization. The horizontal displacements between the two signals are then determined by either DTW or DWM. A comparator then generates the vertical distance for each pair of points or windows. Finally, a discriminator looks at both the horizontal displacements and the vertical distances to automatically determine if there is an intrusion.

In NSYNC, the thresholds in the discriminator are learned by One-Class Classification (OCC) [30]. In contrast, many existing IDSs either use binary classification [21, 22, 25] (the spatial layer in [22]) or magic numbers for thresholds [22, 27] (the acoustic layer in [22]). Some existing IDSs do not have an automatic decision module [24]. Binary classification requires knowing the malicious processes in advance, which can be impractical to achieve. In contrast, OCC does not require such knowledge.

1.3 Defending Against Side-Channel Attacks

Side channels can be used by defenders to protect AM systems but they can also be used by attackers to steal intellectual property in AM systems. In fact, side-channel attacks can be used to break computer systems that are otherwise hard to break [31]. In the literature, there are already many side-channel attacks to steal intellectual property in AM systems. For example, when the acoustic wave in a printing process is recorded by an attacker, the attacker can potentially infer information about the printing process by a variety of techniques, such as signal processing and machine learning [18, 19, 20]. In addition, there are research efforts in the literature to recover the coordinates of the printhead in a 3D printer by analyzing the infrared video observed by an infrared camera [32].

However, existing side-channel attacks on intellectual property in AM systems have a variety of limitations and only work well for a few printing processes. For example, the acoustic side-channel attacks in [18, 20] only work for printing processes where the movement directions are restricted to four or eight cardinal directions, and the acoustic side-channel attack in [19] only works for printing processes where the duration of each G-code instruction is long enough to clearly see the boundaries between G-code instructions in the spectrogram. The infrared video side-channel attack in [32] fails altogether potentially due to a lack of proper methodology.

We propose the optical side-channel attack using a deep neural network. Although the optical side-channel attack requires camera footage of a printing process, which can be hard to acquire compared with audio data in the acoustic side-channel attack, the optical side-channel attack can easily overcome the limitations faced by the acoustic side-channel attack and recover the path of an arbitrary printing process. To our best knowledge, we are the first research group to successfully perform the optical side-channel attack on intellectual property in AM systems and present effective mitigation methods.

To defend against side-channel attacks on intellectual property in AM systems, researchers came up with a variety of methods, such as tuning manufacturing parameters [33, 34], using fake movements [20], balancing loads on motors [18], applying shields [18, 19, 20], and injecting noise [19, 20]. In this thesis, we focus on the noise injection method to defend against our proposed optical side-channel attack, because the noise injection method, compared to many other defense methods, has the least amount of adverse impact on the AM system and the printed objects.

Noise injection as an effective defense method requires an algorithm to determine the pattern of the noise to be injected. There are several existing noise generation algorithms such as random blobs, white noise, full power, and replaying the side-channel signal of another printing process [19]. According to our experiments on the optical side-channel attack, although the existing noise generation algorithms are effective against a naive attacker, they can be easily defeated by an advanced attacker, who is aware of the existence of injected noise and attempts to identify and remove the injected noise in the attack process. To defend against the advanced attacker, in this thesis, we propose three novel noise generation algorithms, and they are channel uniformization, state uniformization, and state randomization. Our experiments show that state randomization can effectively defend against attackers with different prior knowledge.

1.4 Structure of the Thesis

The structure of the thesis is as follows: In chapter 2, we briefly discuss papers that are related to the thesis. In chapter 3, we present the NSYNC framework, which uses side-channel signals in AM systems to perform intrusion detection. In chapter 4, we present the optical side-channel attack as well as methods to defend against the proposed optical side-channel attack. In chapter 5, we summarize important findings in the thesis.

CHAPTER 2
LITERATURE SURVEY

2.1 Cyberattacks to Sabotage Additive Manufacturing Systems

Although there are no reported cyberattack on AM systems that result in a large loss of property or life, the research community is working on potential threats on AM systems.

Moore et al. analyzed the vulnerability of common 3D printing software packages, such as Cura, ReplicatorG, and Repetier-Host, and found several vulnerabilities that could be potentially exploited by attackers [6]. Do et al. analyzed the network protocols in two MakerBot 3D printers and found that they did not check the legitimacy of the SSL/TLS certificates. As a result, they were able to exfiltrate sensitive data and remotely manipulate the 3D printers [7].

To see if it is possible to launch a cyberattack on an AM system to weaken the structural integrity of a printed object without being detected, Turner et al. implanted a virus on a controller PC in an AM system and the virus rewrote G-code instructions for incoming G-code files to maliciously change the geometry slightly to weaken the objects to be printed [35]. They demonstrated with human subjects that the malicious modification could not be noticed by AM operators.

Zeltmann et al. experimented with a Stratasys Connex 500 printer to see if it is possible to insert malicious defects inside a printed object to weaken its structural integrity while evading Non-Destructive Testing (NDT) [36]. They inserted tiny cubes (composed of a soft material) into the structural material in a printed object. Although the ultrasonic inspection could not detect the presence of the tiny cubes, the inserted defects did not cause noticeable degradation of structural integrity. In addition to inserting defects, they experimented with a Stratasys Dimension Elite FDM printer to see if it is possible weaken the structural integrity of a printed object by changing the orientation of the object in the printing process. They found that this method could effectively reduce the structural integrity, but the surface pattern was noticeably changed which could lead to detection.

Sturm et al. performed a cyberattack that is similar to the cyberattack in [35]. They infected a host computer in an AM system with a virus that could automatically insert voids into a design file before it was submitted for printing. The maliciously inserted voids could weaken the structural integrity of the printed object, and their presence was not noticed by human operators.

Slaughter et al. performed a cyberattack on a metal AM system with a real-time process monitoring system using infrared cameras [37]. The infrared images were used for quality assessment and used as inputs to a closed-loop control system to correct temperatures in the AM system. They performed various cyberattacks on the infrared imaging system and the closed loop control system to maliciously alter the temperatures and were able to degrade the quality of the printed objects.

Moore et al. demonstrated with a Printrbot 3D printer that by maliciously modifying the firmware of the printer, an attacker can change the shape of the printed object or change the extrusion rate in the printing process [8]. Normally, the extrusion rate must match the printing speed. Otherwise, the quality of the printed object degrades either in the form of under-extrusion or over-extrusion. This work demonstrates the importance of the security of firmware in AM systems. When the firmware is compromised, a 3D printer can still behave maliciously despite being sent benign G-code instructions.

Belikovetsky et al. demonstrated a complete cyberattack on an AM system that eventually resulted in the damage of physical property. The AM system was manufacturing propellers for drones. An attacker first gained access to the host computer in the AM system by sending an email containing a malicious zip file. Once the victim opened the zip file and clicked an EXE file that looked like a PDF file, a reverse shell was opened in the background for the attacker to secretly control the host computer. The attacker then looked for the design file and strategically inserted defects into important locations in the propeller. When a propeller with the inserted defects was printed, the propeller was able to pass quality inspection, but still broke in flight, causing the drone with the propeller to fall from the sky. This is the most serious type of cyberattacks that can happen to an AM system.

2.2 Intrusion Detection with Side Channels

To fight against cyberattacks that attempt to compromise the structural integrity of printed objects, a series of IDSs leveraging side-channel signals have been proposed in the literature. There are two main advantages of side-channel based IDSs. They are air-gapped from the systems to be protected and inflict zero overhead.

Chhetri *et al.* came up with the idea of using the acoustic side-channel signal in a printing process to detect zero-day cyberattacks on AM systems [21]. They used machine learning to estimate the velocity and distance of a movement from its acoustic side-channel signal. This process is virtually the acoustic side-channel attack in [18, 19, 20]. By interpreting the G-code instruction, the expected velocity and distance of the movement is obtained. An alert is raised if the estimated values are different from the expected values by predefined thresholds. However, an important assumption in the paper is that the side-channel signal can be segmented such that each segment corresponds to a single G-code instruction. This process is highly non-trivial and there are no details in the paper.

Bayens *et al.* presented an IDS with two layers. The acoustic layer compares an acoustic side-channel signal against a reference signal window-by-window with Dejavu, a music retrieval engine that is similar to Shazam [38], to detect malicious infill patterns [22]. The spatial layer compares the position signal (the position of the nozzle with respect to time) against a pre-recorded reference signal to detect intrusion. The problem with the acoustic layer is that Dejavu is optimized for music, not necessarily for acoustic side-channel signals. Our experiment results indicate that the acoustic layer simply classifies all printing processes as malicious. The problem with the spatial layer is that it performs binary classification, which requires knowing the malicious processes to be detected in advance. In addition, the spatial layer requires the installation of a potentiometer onto the nozzle of the printer, which is an invasive sensor.

Moore *et al.* proposed an IDS that measures electric currents delivered to actuators, and compares measured signals against pre-recorded reference signals point-by-point to detect malicious activities [23]. The main problem with this IDS is that it is not aware of the existence of time noise, which can easily render point-by-point comparison invalid. The current sensors are invasive as they need to be installed on the electric wire of steppers in a printer. This can be hard for a lot of 3D printers. In addition, they use an electric wire from the printer as a trigger to determine the starting moment of a printing process, which breaks the air gap between the IDS and the 3D printer.

Gatlin *et al.* improved Moore's method by analyzing the current in the Z-axis stepper motor to identify the moments when a layer change happens [26]. The layer changing moments can not only be used to better align signals to be compared but also be used as indicators for intrusion detection. In addition, instead of comparing raw signals point-by-point, this IDS transforms the side-channel signals into fingerprints and compare fingerprints for intrusion detection. Even though the signals to be compared are aligned at the layer changing moments, since it takes a long time to print a single layer, time noise can still render the point-by-point comparison within a layer invalid. As with Moore's IDS, the installation of current sensors into a printer is invasive and is hard to be performed for a lot of (expensive and enclosed) printers.

Gao *et al.* presented a process monitoring system that observes the acceleration, magnetic, acoustic, and optical side-channel signals to safeguard AM systems against cyberattacks [24]. Based on the observed side-channel signals, state variables, such as the position and velocity of the nozzle, are estimated, along with the height of each layer and the fan speed. To detect intrusion, for each layer, the estimated position signal is converted into a path and compared against the reference path obtained from the G-code file. Other estimated signals are compared against their counterparts derived from the G-code file to detect intrusion. It seems that the reference printing speed and the reference fan speed within a single layer are assumed to be constant. This assumption is usually true for the fan speed, but unlikely to be true for the printing speed. In addition, this paper focuses on how accurate they can recover the state variables from the side-channel signals. They do not provide a discriminator to automatically determine if the side-channel signal to be assessed belongs to a benign process or a malicious process.

Belikovetsky *et al.* presented a framework to transform an acoustic side-channel signal into a spectrogram, which is further compressed by the Principle Component Analysis (PCA) into a signal with only three channels [27]. The transformed signal is compared window-by-window against a reference signal (derived from a verifiable benign printing process) to detect intrusion. They proposed to insert special commands into the G-code file to generate special sounds at the starting and stopping moments of a printing process, which can be easily captured and processed by the IDS. The main problem with this IDS is the lack of awareness of time noise.

Straub built a system with five cameras to capture images of a printing process to monitor the progress of the printing process [39]. Later, Straub applied the same process monitoring system for intrusion detection [40, 41, 42, 42, 43, 44]. However, these IDSs are for post-production verification only. They perform pixel-by-pixel comparison, which makes these IDSs susceptible to the changes of the camera positions and lighting conditions. In addition, these papers only show the difference between reference images and observed images without providing mechanisms to automatically determine if the current printing process is benign or malicious.

Wu *et al.* proposed to use machine learning to perform intrusion detection in AM systems using the optical side channel. In the IDS, a camera captures images of infill patterns in a printing process, and a feature vector of 24 components is derived from each image. They collect a series of images with normal infill patterns. They also collect a series of images containing one of five predefined defects in the infill patterns. A binary classifier is then trained on these images to discriminate images with normal infill patterns and images with abnormal infill patterns. There are two serious problems that may restrict the application of this IDS. First, it is not clear if the IDS can generalize to other types of defects not seen in the five predefined defects. Second, it is not clear if the IDS still works if an image is not fully covered by infill patterns.

2.3 Side-Channel Attacks for Intellectual Property

To perform the acoustic side-channel attack, Al Faruque et al. proposed an algorithm to separate the acoustic signal due to one motor and the acoustic signal due to another motor from the combined acoustic signal [18]. After the signals are separated, they use regression models and classifiers to determine the speed for each motor. However, according to [45], this approach may not be effective when the two motors are moving at the same time.

By assuming the printing speed is constant and known, Hojjati et al. proposed to use matching filters to determine the direction for each movement [19]. As the matching filters typically return multiple directions with equal probabilities, their method further uses the magnetic side channel and human intelligence to determine the most likely directions for all movements. Since this method depends the abrupt changes in the side-channel signal to determine the boundaries of G-code instructions, this method may not work well for a printing process where there are a lot of short and rapid movements.

As with Hojjati's attacker, the method proposed by Song et al. assumes that the printing speed is known in advance and uses magnetic fields for assistance [20]. Instead of using matching filters, their method uses five classifiers to directly determine the nozzle's movement and the extruder's state. Since this method relies on classifiers to work, a major limitation of this method is that it only works well for a printing process where most movements are along the eight cardinal directions.

Other than the acoustic side-channel attack, Al Faruque et al. came up with an attack that uses the infrared side channel to steal IP in AM processes [32]. They proposed a mapping algorithm to convert an infrared video to a speed signal. However, according to [32], the attack was not successful due to low resolution, low sampling rates, limited perspectives, and inability to change the focus of the infrared camera.

2.4 Defending Against Side-Channel Attacks

The authors that proposed the side-channel attacks on AM systems also mentioned potential mitigation methods without comprehensive evaluations in practice though. A complete list of the defense methods is as follows:

Parameter Tuning. Chhetri *et al.* proposed to minimize the mutual information between side-channel signals and G-code instructions by tuning manufacturing parameters, such as object orientation and printing speed [34]. However, the level of protection provided by the method is limited. According to their own performance metric, there is an average reduction of 10% in the success rate, and they acknowledge that this method may not be able to practically defeat side-channel attacks.

Movement Obfuscation. This method injects fake movements in a printing process to confuse an attacker [20], as shown in Figure 2.1. A fake movement moves the printhead at the extrusion speed without filament extrusion. This breaks the important assumption by many attackers that a low printing speed corresponds to an extrusion movement whereas a high printing speed corresponds to a non-extrusion movement [18, 19, 20]. However, this method introduces stringing effects[1], which can adversely affect the quality of the printed object. In addition, this method can significantly extend the printing time.

Signal Power Reduction. This method aims at reducing the power of side-channel signals and thus reduce the Signal-to-Noise Ratio (SNR). The power of acoustic side-channel signals can be reduced by a better design. For example, compared with SeeMeCNC Ro-

[1]In an FDM process, when the nozzle travels from point A to point B without extrusion, the molten filament in the nozzle continues to come out in this process, and forms a string from point A to point B, which is undesirable. To suppress the stringing effect, the extruder retracts the filament at the beginning of the movement and then re-extrudes at the end of the movement.

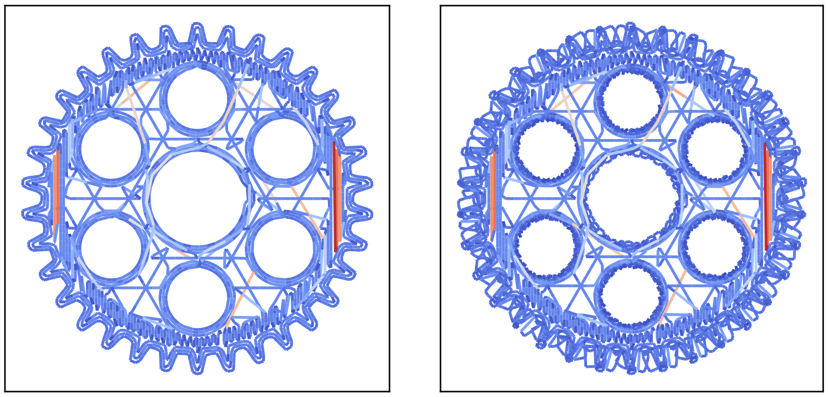

(a) No Fake Movement (b) With Fake Movements

Figure 2.1: Illustration of fake movements. (a) A layer without fake movement. (b) The same layer with fake movements.

stock Max V3, Ultimaker 3 is much quieter and emits much less acoustic waves. The power of optical side-channel signals can be reduced by using less lights in the environment. However, using less lights not only makes normal operation harder but also brings security problems, such as potentially increased theft activities.

Physical Shielding. This method installs physical shields to prevent side-channel signals from reaching the attacker [18, 19, 20]. Examples are acoustic shields for the acoustic side channel and view blocks for the optical side channel. However, view blocks can make it hard to monitor the printing process by human beings or cameras.

Physical Uniformization. The core ideal of this defense method is to make physical changes to the AM system such that the side-channel signals for different states of the AM system are similar. In this way, it will be hard for an attacker to infer the state of the AM system by analyzing the side-channel signal. For example, Al Faruque et al. proposed to balance the loads in the x and y motors to make the acoustic emission from the x motor and that from the y motor similar to each other [18]. For the optical side channel, the printhead and the build plate can be made the same color such that it is hard to see the printhead in relation to the build plate. However, our experiments indicate that this method may not be effective as we were able to reconstruct the printing path by the optical side channel for a printer where the printhead and the build plate were both black.

Noise Injection. This defense method involves using signal generators (such as speakers) to artificially create side-channel signals to interfere with side-channel signals in AM systems in an effort to reduce the SNR and thus thwart side-channel attacks [20, 19]. The artificially created side-channel signals are noise for an attacker, and thus is also referred to as side-channel noise. This paper focuses on this method because, according to our experiments, it is a low cost and effective solution, and it has the least amount of impact on the AM system. The noise injection method requires an algorithm to generate noise. Noise generated by existing algorithms can be easily filtered out by advanced attackers as demonstrated by our experiments later in this thesis.

CHAPTER 3
THE NSYNC FRAMEWORK

3.1 Introduction

The NSYNC framework is a solution to compare two signals to determine if the signals are alike or not. For an AM system, when one of the signals is the side-channel signal of a verified benign printing process and the other signal is the side-channel signal of an unidentified printing process, the NSYNC framework acts as an IDS for the AM system. The main strength of NSYNC is that it is capable of comparing signals that are not perfectly aligned. For an AM system, two side-channel signals from two printing processes with the same G-code file cannot be perfectly aligned due to the presence of time noise in the AM system. Because of this, the NSYNC framework is probably, as of now, the only practical IDS leveraging side-channel signals in AM systems.

In this chapter, we first of all briefly introduce the background information that is needed to understand the NSYNC framework. We then present the threat model that the NSYNC framework is designed for. Next, we discuss the overall structure of the NSYNC framework. Afterwards, we discuss in detail each component of the NSYNC framework. Finally, we present experiment results related to the NSYNC framework.

3.2 Background Information

3.2.1 Additive Manufacturing

Additive Manufacturing (AM) refers to a collection of manufacturing processes where materials are joined together layer by layer to make objects directly from 3D models [46]. AM processes are performed by computers without human intervention. The operation of an AM process is called printing and the machine by which materials are joined together

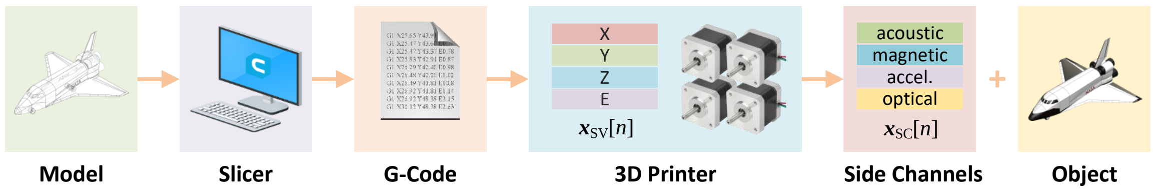


Figure 3.1: A general FDM process. A slicer converts a 3D model into G-code instructions, which are then sent to a printer for execution. The printer interprets the G-code instructions and instructs various actuators in the printer to work. An object is printed, and various side-channel signals (acoustic, magnetic, optical, etc) are generated.

is called a printer. There are seven categories of AM processes, and each category contains many types of AM processes. In this thesis, we focus on Fused Deposition Modeling (FDM) [47], which is the most common AM process.

A general FDM process is described in Figure 3.1. The AM process starts with a 3D model that describes the shape of the object to be printed. A specialized program called a slicer, such as Cura, Slic3r, and MatterSlice, then asks for manufacturing parameters, such as feed rates and temperatures, to be specified. Afterwards, the slicer converts the 3D model into a G-code file, which is a collection of G-code instructions. Finally, the G-code file is sent to a printer for execution. The printer interprets the received the G-code file and instructs various actuators in the printer to work according to the G-code file. An object is printed while various side-channel signals are emitted.

3.2.2 State Variables

From the perspective of control theory, a printer can be represented by a state variable[1]. The components of the state variable may include the position of the printhead, the position of the filament, the temperatures of the nozzle(s) and the build plate, the speed of the fans, etc. The state variable as a function of time forms a state-variable signal, where each component of the state variable corresponds to a channel in the signal. A state-variable

[1]A variable can be a scalar, a vector, or even a matrix. A state variable can be a vector with multiple components.

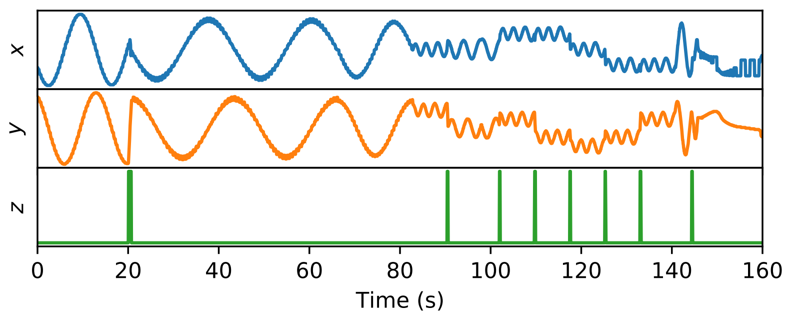


Figure 3.2: Example of a state-variable signal $\boldsymbol{x}_{\text{SV}}$ with three channels x, y, z, where (x, y, z) is the coordinate of the printhead. The state-variable signal could contain more channels such as the displacement of the filament in the extruder, the fan speed, the temperature(s) of the nozzle(s), and the temperature of the build plate.

signal is described by

$$\boldsymbol{x}_{\text{SV}}[n, c], n = 0, 1, \cdots, N-1, c = 0, 1, \cdots, C-1, \tag{3.1}$$

where n is the time index, N is the number of data points, c is the channel index, and C is the number of channels. If the sampling rate is f_s, then the duration of the state-variable signal is N/f_s. Figure 3.2 shows a state-variable signal with three channels corresponding to the x, y, and z coordinates of the printhead.

In this thesis, we may write $\boldsymbol{x}_{\text{SV}}[n, c]$ simply as $\boldsymbol{x}_{\text{SV}}[n]$ or $\boldsymbol{x}_{\text{SV}}$, where $\boldsymbol{x}_{\text{SV}}[n]$ is a vector of C components and $\boldsymbol{x}_{\text{SV}}$ is an array[2] (or matrix) of shape $N \times C$.

Control Variables. There is a concept called control variables in [21]. A control variable is the target of a movement, such as the target speed and the target position. In contrast, a state variable is the instantaneous state during a movement, such as the instantaneous speed and the instantaneous position during a movement. In this thesis, we focus our analysis on state variables as they are more directly related to side channels.

[2]In this thesis, we define arrays and vectors in the context of mathematics. An array is a sequence of ordered elements. There can be an infinite number of elements in an array. A vector is a fixed number of components. In the context of programming languages (C/C++ to be specific), an array is a block of consecutive memory space to store a fixed number of elements. The size of an array, after its creation, cannot be changed. In contrast, the size of a vector can be dynamically increased.

3.2.3 Side Channels

Side channels are unintentional means of communication by which information about a computer or a cyber-physical system can be leaked to an outsider.

AM systems have a variety of side channels. For example, when an AM system is printing an object, the system emits acoustic waves [18, 19, 20] and electromagnetic waves (including quasi-static electric fields and quasi-static magnetic fields) that can be sensed to infer information about the printing process. The acceleration of any moving part in the AM system can be measured by an accelerometer to infer information about the printing process [22, 24]. Other examples of side channels in an AM system include, but are not limited to, power consumption measured by power sensors [23, 26], temperatures measured by thermometers, optical videos captured by cameras [24, 25], and infrared videos captured by infrared cameras [32].

Mathematically, a side-channel signal is represented by

$$\boldsymbol{x}_{\mathrm{SC}}[n, c], n = 0, 1, \cdots, N-1, c = 0, 1, \cdots, C-1, \tag{3.2}$$

where n is the time index, N is the number of data points, c is the channel index, and C is the number of channels. If the sampling rate is f_s, then the duration of the side-channel signal is N/f_s. Typically, f_s is determined by the Analog-to-Digital Converter (ADC) in the data acquisition system that observes the side-channel signal.

As with $\boldsymbol{x}_{\mathrm{SV}}[n, c]$, we may write $\boldsymbol{x}_{\mathrm{SC}}[n, c]$ simply as $\boldsymbol{x}_{\mathrm{SC}}[n]$ or $\boldsymbol{x}_{\mathrm{SC}}$, where $\boldsymbol{x}_{\mathrm{SC}}[n]$ is a vector of C components and $\boldsymbol{x}_{\mathrm{SC}}$ is an array (or matrix) of shape $N \times C$.

3.3 Threat Model

In chapter 2, we saw that there are cyberattacks that seek to compromise the structural integrity of printed objects in AM systems, and there are IDSs that leverage side-channel signals to fight against such attacks. In this section, we describe the threat model that is

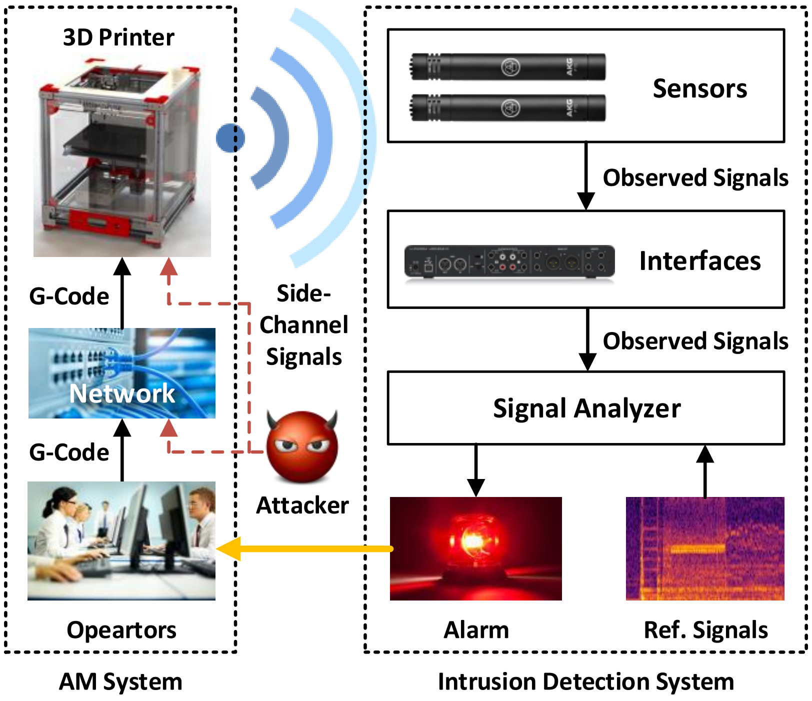


Figure 3.3: Threat model for intrusion detection with side channels. An attacker attempts to undermine the structural integrity of the printed object by modifying the G-code file in the network or the firmware in the printer. An air-gapped intrusion detection system monitors the side-channel signals in the printing process to determine if there is an attack.

shared by the existing IDSs [21, 22, 23, 24, 27, 26].

The threat model is shown in Figure 3.3. In the threat model, an AM system is manufacturing a functional object. An attacker wants to compromise the structural integrity of the printed object without being detected. We assume that the attacker can modify the software (G-code instructions) to be sent to the printer or the firmware of the printer. By modifying the firmware, the printer could behave maliciously despite being sent benign G-code instructions. The attacker knows how to strategically modify the G-code instructions or firmware to weaken the structural integrity of the object and let the object pass existing quality checks, such as the attack demonstrated in [11].

In the AM system, an air-gapped IDS leveraging side-channel signals is deployed. The IDS is composed of an analyzer, sensors (and interfaces) to observe side-channel signals, and reference signals. The analyzer, essentially a digital computer, continuously compares the observed signals against the reference side-channel signals. If an observed signal is considered different from its corresponding reference signal at any moment, then an intrusion is detected and the IDS alerts AM operators, and automatically stops the printing process if necessary.

3.3.1 Acquisition of Reference Signals

Reference signals are recorded side-channel signals from a benign process. A challenge to obtain reference signals is to ensure that they indeed come from a benign process. One way to do this is to subject the printed object to stringent tests and if the printed object passes the tests, the printing process can be considered benign [23, 27, 26].

3.3.2 Requirements of Side-Channel Signals

In order for the aforementioned IDS to work, the selected side-channel signals must be highly correlated with the state-variable signal of the printer. On the one hand, when the state-variable signal of the printer is modified, the changes should be reflected in the side-channel signals. Otherwise, the IDS will result in false negatives. On the other hand, when the state-variable signal of the printer is not altered, the side-channel signals should remain almost the same. Otherwise, the IDS will result in false positives.

3.3.3 Real-Time vs Post-Production

For real-time detection, the analyzer can only see the observed side-channel signals up to the current moment but has access to the whole reference signals. In contrast, for post-production analysis, the analyzer has access to the whole observed side-channel signals as well as the whole reference signals. Real-time detection is more desirable since early detection saves time and reduces material waste when an attack does occur.

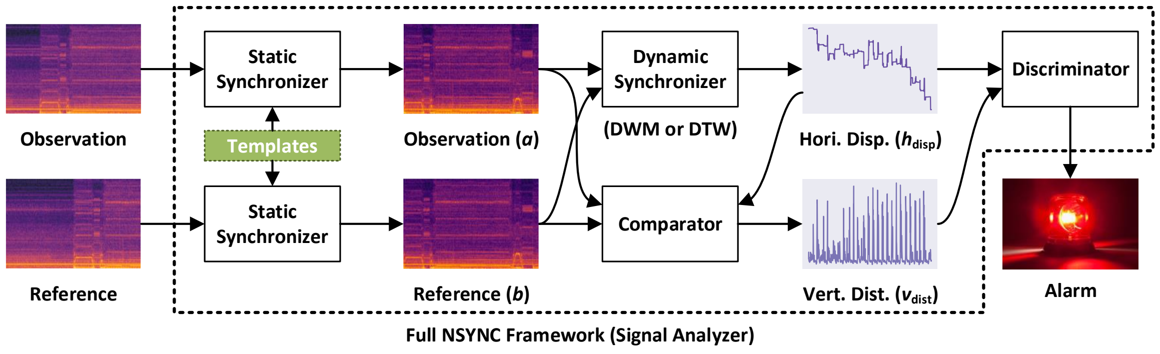


Figure 3.4: Overview of the NSYNC Framework.

3.3.4 Advanced Attackers

We assume that an advanced attacker knows the existence of the IDS in the AM system, and attempts to avoid detection by leveraging the potential weaknesses in the IDS. For example, the advanced attacker can perform an attack such that the duration of the malicious printing process is the same as the duration of the benign printing process. A more advance attacker can perform attacks that result in a lot of tiny changes across the whole printing process. The tiny changes may not be detected by the IDS but their cumulative effect may compromise the structural integrity of the printed object. Nevertheless, we assume that the attacker cannot compromise any component in the IDS, including the observed signals, the analyzer, and the reference signals.

3.4 Overview of the NSYNC Framework

The overall structure of the NSYNC framework is shown Figure 3.4. One of the signals to be compared is referred to as the observed signal (or simply observation) and is denoted by $\boldsymbol{a}$. The other signal to be compared is referred to as the reference signal (or simply reference) and is denoted by $\boldsymbol{b}$.

Process of Operation. First of all, $\boldsymbol{a}$ goes through a static synchronizer to figure out its starting and stopping moments[3]. Meanwhile, $\boldsymbol{b}$ also goes through the static synchronizer

[3]The starting and stopping moments are the starting and stopping moments of the signal comparison process, not the starting and stopping moments of the printing process. See subsection 3.7.1 for more details.

to figure out its starting and stopping moments. Afterwards, $\boldsymbol{a}$ and $\boldsymbol{b}$ are aligned at their starting moments and the signal comparison process is initiated at their starting moments. The aligned $\boldsymbol{a}$ and $\boldsymbol{b}$ then go through a dynamic synchronizer to figure out their timing relationship, described by what is known as the horizontal displacements. After obtaining their timing relationship, a comparator compares the corresponding points or windows in $\boldsymbol{a}$ and $\boldsymbol{b}$ to obtain their vertical distances. Finally, both the horizontal displacements and the vertical distances are used by the discriminator to determine if the signals are alike or not. If the signals are not alike, an intrusion is declared. The signal comparison process is terminated when reaching the stopping moment of $\boldsymbol{a}$ or the stopping moment of $\boldsymbol{b}$.

Two Modes of Operation. Normally, the reference signal is a recorded signal and we can access the whole reference signal before the NSYNC framework is invoked. However, the observed signal can be a recorded signal or a real-time signal. When the observed signal is a real-time signal, we can only access the observed signal up to the current moment.

The NSYNC framework can be operated in two modes, with one being the real-time mode (or the online mode) and the other one being the post-production mode (or the offline mode). In the real-time mode, the NSYNC framework can analyze both real-time signals and recorded signals. In the post-production mode, the NSYNC framework can only analyze recorded signals. The real-time mode is more challenging to implement as the access to the observed signal is more restrictive. To support the real-time mode, all components that a real-time signal goes through must support the real-time signal.

3.4.1 Static Synchronization

Static synchronization refers to a process to identify a group of corresponding points in two or more signals. The corresponding points can be any moments, such as the starting moments, the stopping moments, or any layer changing moments. A moment is defined by a template, an excerpt of a signal around the moment. The primary motivation to develop static synchronization is to align signals at their starting moments and stop the signal comparison process at their stopping moments. The term “static” derives from the fact that only one group of corresponding points is identified for each invocation of static synchronization. Although it is possible to invoke static synchronization multiple times, a unique template is required for each time of invocation.

3.4.2 Dynamic Synchronization

Dynamic synchronization is a process to continuously identify corresponding points or windows in two signals in real time[4]. The relationship between corresponding points or windows in the two signals is described by an array of horizontal displacements.

3.4.3 Comparator

After the corresponding points are determined, the next step is to evaluate the distances or windows between corresponding points. Any valid distance metric (or distance function) can be used, such as the cosine distance and the correlation distance.

3.4.4 Discriminator

A discriminator uses both the horizontal displacements and the vertical distances to determine if the two signals are alike or different. If the discriminator determines that the two signals are different at any moment, an intrusion is declared.

The signal comparison process terminates when we hit the stopping moment of the observed signal or the stopping moment of the reference signal. If no alert is issued during the whole signal comparison process, we declare that there is no intrusion.

[4]Dynamic synchronization cannot be viewed as invoking static synchronization multiple times. First of all, dynamic synchronization only supports a pair of signals (or two signals), whereas static synchronization directly supports multiple signals. Second, dynamic synchronization does not require a template whereas static synchronization requires a template.

3.4.5 NSYNC as a Framework

There is a lot of freedom in the NSYNC framework. In fact, the NSYNC framework does not stipulate a specific method to perform static synchronization or dynamic synchronization. Any method that can accomplish the goal of static synchronization or dynamic synchronization can be used. Similarly, any valid distance metric (or distance function) can be used in the comparator and a variety of mechanisms can be used as the discriminator.

3.5 Structure of a Printing Process

To properly use the NSYNC framework, it is important to understand the structure of a printing process as not all procedures of a printing process are applicable to NSYNC.

3.5.1 Three Parts of a Printing Process

Figure 3.5 shows the structure of a printing process by the Ultimaker 3 (UM3) printer. Each box represents a procedure or a subroutine of the printing process[5]. On a high level, a printing process is composed of three parts, and they are pre-deposition procedures, deposition procedures, and post-deposition procedures. The pre-deposition procedures configure the printer in a state that is ready for material deposition, such as moving the printhead and the build plate to specific positions, heating the nozzle(s) and the build plate to specific temperatures, and priming the nozzle(s) with filaments. The deposition procedures extrude the filaments into the build plate and create the object. The post-deposition procedures put the printer in a state that is suitable for staying idle and getting ready for the next printing job. The post-deposition procedures include moving the printhead and the build plate to specific locations and cooling down all heated components.

Figure 3.6 shows the structure of a printing process by the Rostock Max V3 (RM3) printer. Compared with the UM3 printer, the printing process of the RM3 printer is much

[5]In this thesis, each procedure is referenced by its noun form. For example, “Home Printhead” is referred to as the printhead-homing procedure, “Select Nozzle” referred to as the nozzle-selection procedure, and “Deposit Materials” is referred to as the material-deposition procedure.

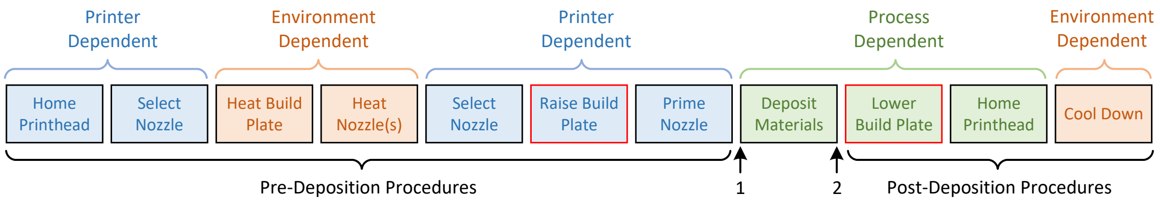


Figure 3.5: Structure of a printing process of the Ultimaker 3 printer. The box labeled as “Deposit Materials” belongs to the deposition procedures. Currently, there is only one procedure in the group of deposition procedures.

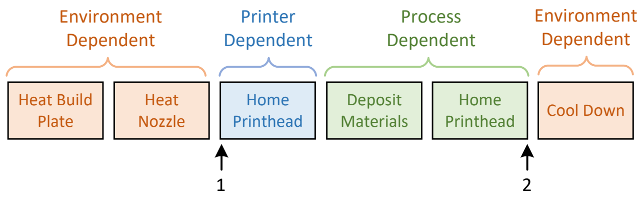


Figure 3.6: Structure of a printing process of the SeeMeCNC Rostock Max V3 printer. The first three boxes belong to the pre-deposition procedures. The box labeled as “ Deposit Materials” belongs to the deposition procedures. The last two boxes belong to the post-deposition procedures.

simpler. Nevertheless, on a high level, the printing process is still composed of three parts. The part before the deposition procedures is referred to as the pre-deposition procedures and includes heating the build plate, heating the nozzle, and putting the printhead to a home position. The deposition procedures are responsible for depositing the materials into the build area and creating the object. The part after the deposition procedures is referred to as the post-deposition procedures and includes putting the printhead to the home position and cooling down all heated components.

Both the UM3 printer and the RM3 printer occasionally move their printheads to their home positions. This is crucial because a printer cannot see the absolute position of the printhead but keeps a record of the relative displacement of the printhead with respect to the home position. If the printhead is accidentally moved by external forces, the printer will not see it and will move the printhead to wrong positions. When the printhead is moved to the home position by a special instruction, switches located at the home position will be triggered to inform the printer that the printhead reaches its home position and the coordinate registers in the printer will be updated accordingly.

3.5.2 Three Types of Procedures

In this thesis, procedures are divided into three types depending on what factor affects the details of the procedures. The three types are environment-dependent procedures, process-dependent procedures, and printer-dependent procedures.

An environment-dependent procedure is affected by the environment such as ambient temperature and wind. For example, the time it takes to heat up a component depends on the initial temperature of the component as well as the ambient temperature and the wind in the environment. An environment-dependent procedure is highly volatile and it is very hard to exactly reproduce the procedure[6].

A process-dependent procedure is affected by the G-code instructions to be executed on the printer. The material-deposition procedure is totally determined by the G-code instructions and hence is process-dependent. In the post-deposition procedures, the build-plate-lowering procedure is affected by the G-code instructions because the build plate is lowered from a position when a printing process completes, and different objects have different heights. Similarly, in the post-deposition procedures, the printhead-homing procedure is affected by the G-code instructions because the printhead is moved to the home position from the last point on the object to be printed. For different objects, the last points are different and hence the homing movements are different. Compared with environment-dependent procedures, process-dependent procedures are less volatile and can be reproduced with the same G-code instructions.

A printer-dependent procedure is not affected by the environment or the G-code instructions. They are only affected by the printer and its configuration. For example, in the pre-deposition procedures, the printhead-homing procedure is the same for different G-code instructions and environment conditions because the printhead should already be at the home position when this instruction is performed. Similarly, the build-plate-raising

[6]By exactly reproducing, we mean reproducing the procedure with exactly the same details, such as the same duration and the same side-channel signals.

procedure in the UM3 printer is the same regardless of the G-code instructions. The build plate is always raised from the lowest position to the highest position. However, this procedure can be affected by the printer and its configuration. When the printer is manually leveled, the height raised in this process can be affected. We also consider the nozzle selection process in the UM3 printer to be printer-dependent because the first nozzle to be used usually is the same for different printing processes, even though this behavior could be overridden by a printing process. Overall, a printer-dependent procedure is highly stable and can be easily reproduced in any printing process.

3.5.3 Signal Comparison Processes

A signal comparison process is a portion of a printing process where comparison of side-channel signals is actively performed by the NSYNC framework. The NSYNC framework is based on signal comparison and hence requires the side-channel signals to be reproducible for different printing processes. As a result, the NSYNC framework cannot analyze environment-dependent procedures in a printing process, and we exclude these procedures in a signal comparison process. It is possible to exclude other procedures in a printing process in the signal comparison process, but the material-deposition procedure should be preserved as this procedure directly affects the printed object.

For the UM3 printer, we only perform signal comparison for the material-deposition procedure because it is very hard to identify the build-plate-raising procedure and the build-plate-lowering procedure in the acceleration side channel. This is because we only installed the accelerometer on the printhead, and it is not affected by the build plate of the printer. The starting and stopping moments of the signal comparison process is identified by two arrows labeled by 1 and 2 respectively in Figure 3.5.

For the RM3 printer, we perform signal comparison for the material-deposition procedure as well as the printhead-homing procedures. Even though the printhead-homing procedure in the post-deposition procedures is process-dependent, the last portion of this procedure is almost process independent. This is because the printer will home the printhead again after the printhead reaches its home position, creating a signature movement that is not affected by the details of a printing process. In this way, we can easily identify the printhead-homing procedures in a side-channel signal.

3.6 Signal Processing in NSYNC

This section introduces necessary background knowledge on signal processing to understand the NSYNC framework.

3.6.1 Signal Notation

A signal (including a side-channel signal) is denoted by $\boldsymbol{x}(t)$, where t is time. At any moment, $\boldsymbol{x}(t)$ is a vector of one or more components. For example, the acoustic pressure measured by a microphone is a vector of one component, whereas the acceleration measured by an accelerometer is a vector of three components (a_x, a_y, a_z). Each component of $\boldsymbol{x}(t)$ as a function of time is defined as a channel. The number of channels in $\boldsymbol{x}(t)$ is denoted by C.

In order for a computer to process a signal $\boldsymbol{x}(t)$, the signal $\boldsymbol{x}(t)$ must be sampled by an Analog to Digital Converter (ADC) into a discrete sequence $\boldsymbol{x}[n] = \boldsymbol{x}(nT_s)$, where $n \in \mathbb{Z}$ is the time index, T_s is the sampling interval. $f_s = 1/T_s$ is defined as the sampling frequency. For each time index n, $\boldsymbol{x}[n]$ is a vector of C components. The whole discrete signal $\boldsymbol{x}[n], n \in \mathbb{Z}$ is simply denoted by $\boldsymbol{x}$. If there are finite samples in $\boldsymbol{x}$, we use N to denote the number of samples in $\boldsymbol{x}$.

We use $\boldsymbol{x}[i]$ to refer to the ith sample in $\boldsymbol{x}$. $\boldsymbol{x}[i]$ contains all the channels and is a vector of length C. We use $\boldsymbol{x}[i, c]$ to refer to the ith sample at the cth channel, where $c = 0, 1, \cdots, C - 1$. We use $\boldsymbol{x}[i_1 : i_2]$ to refer to a slice of $\boldsymbol{x}$ from index i_1 (inclusive) to index i_2 (exclusive). We use $\boldsymbol{x}[:, c]$ to refer to all samples at the cth channel.

3.6.2 Time Delay Estimation

The definition of Time Delay Estimation (TDE) is as follows: Suppose $\boldsymbol{x}$ and $\boldsymbol{y}$ have finite samples and the length of $\boldsymbol{x}$ is longer than that of $\boldsymbol{y}$. TDE is a process to determine the best location of $\boldsymbol{y}$ in $\boldsymbol{x}$, assuming that $\boldsymbol{y}$ appears in $\boldsymbol{x}$ once and only once [48, 49].

Sliding Method. One method to perform TDE is the sliding method. Suppose the length of $\boldsymbol{x}$ is N_x, the length of $\boldsymbol{y}$ is N_y ($N_x \geq N_y$), and the number of channels for both $\boldsymbol{x}$ and $\boldsymbol{y}$ is C. One way to perform TDE is to compare $\boldsymbol{y}$ against $\boldsymbol{x}[n : n + N_y]$ for $n = 0, 1, \cdots, N_x - N_y$. For each n, we measure the similarity between $\boldsymbol{x}[n : n + N_y]$ and $\boldsymbol{y}$ by a score $\boldsymbol{s}[n]$. The similarity scores $\boldsymbol{s}[n], n = 0, 1, \cdots, N_x - N_y$ form a new array with a length of $N_x - N_y + 1$. We have

$$\boldsymbol{s}[n] = f(\boldsymbol{x}[n : n + N_y], \boldsymbol{y}), n = 0, 1, \cdots, N_x - N_y, \tag{3.3}$$

where f is a function to calculate the score, also known as the similarity function.

Since a higher score indicates more similarity, the best location of $\boldsymbol{y}$ in $\boldsymbol{x}$ is given by

$$n_{\text{delay}} = \underset{n}{\operatorname{argmax}}\, \boldsymbol{s}[n], \tag{3.4}$$

which means that $\boldsymbol{y}[0]$ corresponds to $\boldsymbol{x}[n_{\text{delay}}]$.

Similarity Functions. In this thesis, we consider four similarity functions, and they are Inner Product (IP), Covariance (Cov), Cosine (Cos), and Correlation Coefficient (CC) [31]. Suppose $\boldsymbol{u}$ and $\boldsymbol{v}$ are two 1-D vectors of the same length N. The expressions for the four similarity functions are shown in Table 3.1.

where $\|\cdot\|_2$ is the L2 norm operator and

$$\mu_u = \frac{1}{N}\sum_{n=0}^{N-1} \boldsymbol{u}[n], \qquad \mu_v = \frac{1}{N}\sum_{n=0}^{N-1} \boldsymbol{v}[n]. \tag{3.5}$$

Calculation Axes. The similarity functions in Table 3.1 require the two inputs be 1-D,

Table 3.1: Score Functions

Name	**Expression**
Inner Product (IP)	$f(\boldsymbol{u}, \boldsymbol{v}) = \boldsymbol{u} \cdot \boldsymbol{v}$
Covariance (Cov)	$f(\boldsymbol{u}, \boldsymbol{v}) = (\boldsymbol{u} - \mu_u) \cdot (\boldsymbol{v} - \mu_v)$
Cosine (Cos)	$f(\boldsymbol{u}, \boldsymbol{v}) = \dfrac{\boldsymbol{u} \cdot \boldsymbol{v}}{\|\boldsymbol{u}\|_2 \cdot \|\boldsymbol{v}\|_2}$
Corr. Coef. (CC)	$f(\boldsymbol{u}, \boldsymbol{v}) = \dfrac{(\boldsymbol{u} - \mu_u) \cdot (\boldsymbol{v} - \mu_v)}{\|\boldsymbol{u} - \mu_u\|_2 \cdot \|\boldsymbol{v} - \mu_v\|_2}$

whereas $\boldsymbol{x}[n : n + N_y]$ and $\boldsymbol{y}$ are 2-D, unless $C = 1$. When $C > 1$, there are two ways to apply the similarity functions in Table 3.1. The first way is to calculate the similarity score between $\boldsymbol{x}[n : n + N_y, c]$ and $\boldsymbol{y}[:, c]$ for $c = 0, 1, \cdots, C - 1$, and average the similarity scores across the channels. Because the similarity score is effectively applied along the time axis in $\boldsymbol{x}$ and $\boldsymbol{y}$, we refer to this way of calculation as calculating along the time axis. The second way is to flatten both $\boldsymbol{x}[n : n + N_y]$ and $\boldsymbol{y}$ into 1-D vectors and proceed with the calculation. We refer to this way of calculation as calculating along the none axis. Both ways to deal with 2-D inputs will be evaluated in experiments.

3.6.3 Signal Detection

The definition of SD is as follows: Suppose $\boldsymbol{x}$ and $\boldsymbol{y}$ have finite samples and the length of $\boldsymbol{x}$ is longer than that of $\boldsymbol{y}$. We define SD as a process to determine all possible locations of $\boldsymbol{y}$ in $\boldsymbol{x}$. $\boldsymbol{y}$ may appear in $\boldsymbol{x}$ zero or multiple times. The main difference between SD and TDE is that TDE assumes that $\boldsymbol{y}$ appears in $\boldsymbol{x}$ once and only once, whereas SD removes this assumption. SD is in theory very similar to object detection [50, 51].

Sliding Method. As with TDE, we use the sliding method to perform SD. Suppose the length of $\boldsymbol{x}$ is N_x, the length of $\boldsymbol{y}$ is N_y ($N_x \geq N_y$), and the number of channels for both $\boldsymbol{x}$ and $\boldsymbol{y}$ is C. We calculate the similarity score $\boldsymbol{s}[n]$ according to Equation 3.3.

Instead of using Equation 3.4, $\boldsymbol{y}$ is detected in $\boldsymbol{x}$ at index n whenever

$$\boldsymbol{s}[n] > s_c, \tag{3.6}$$

where s_c is a pre-determined critical score.

Critical Score. The main challenge of SD is to accurately determine the critical score. To do so, we need to find many training samples of $\boldsymbol{x}$ such that $\boldsymbol{y}$ appears in $\boldsymbol{x}$ exactly once. Suppose the number of training samples is M and the training samples are expressed by $\boldsymbol{x}_m$, $m = 0, 1, \cdots, M-1$. For training sample $\boldsymbol{x}_m$, we perform TDE to obtain the similarity scores $\boldsymbol{s}_m[n], n = 0, 1, N_x - N_y$ and find the delay of $\boldsymbol{y}$ in $\boldsymbol{x}_m$. Figure 3.7 (a) shows the plot of an array of similarity scores. The upper bound of the critical score is

$$s_{c,\max(m)} = \max_{n} \boldsymbol{s}_m[n]. \tag{3.7}$$

The lower bound of the critical score is

$$s_{c,\min(m)} = \max_{n \in I} \boldsymbol{s}_m[n], \tag{3.8}$$

where $I = \{0, 1, ..., n_{\text{delay},m} - n_{\text{exc}}, n_{\text{delay},m} + n_{\text{exc}} + 1, ...N_x - N_y\}$, $n_{\text{delay},m}$ is the estimated location of $\boldsymbol{y}$ in $\boldsymbol{x}_m$, and n_{exc} is the half width of the window whose points are excluded when calculating the lower bound of the critical score. n_{exc} is a parameter to be tuned. By default, we use $n_{\text{exc}} = \text{round}(0.05 f_s)$.

After obtaining $s_{c,\min(m)}$ and $s_{c,\max(m)}$, we define class A by

$$A = \{s_{c,\min(m)} | m = 0, 1, \cdots, M-1\} \tag{3.9}$$

and we define class B by

$$B = \{s_{c,\max(m)} | m = 0, 1, \cdots, M-1\} \tag{3.10}$$

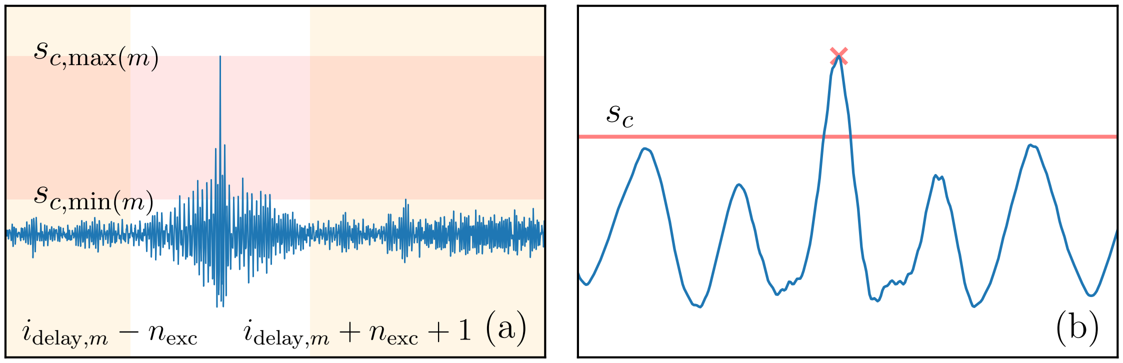


Figure 3.7: Challenges in SD. (a) Determining the range of the critical score for a single training sample of $\boldsymbol{x}$. The range is illustrated by the red region from $s_{c,\min}$ to $s_{c,\max}$. The maximum value of the whole score array is $s_{c,\max}$. The maximum value of the score array inside the orange regions is $s_{c,\min}$. (b) Non-maximum suppression. Any point where the score is above s_c is the detected location of the template. There are multiple points that satisfy this requirement. However, only one point (denoted by a cross in the figure) is the true location of the template.

The next step is to perform binary classification on classes A and B using a single value as the divider. One method to do this is to use the support vector machine with a linear kernel. The divider is then chosen as the critical score s_c.

Non-Maximum Suppression. When performing SD, it is possible for multiple adjacent points to satisfy Equation 3.6, as shown in Figure 3.7 (b). As a result, $\boldsymbol{y}$ could be detected in $\boldsymbol{x}$ multiple times near the same location, although only one instance of $\boldsymbol{y}$ should be detected near that location. To mitigate this problem, we can apply a technique called Non-Maximum Suppression (NMS) [52].

If we know in advance that $\boldsymbol{y}$ appears in $\boldsymbol{x}$ at most once, we can do the following steps. First of all, we check if Equation 3.6 is satisfied by at least one point. If no, $\boldsymbol{y}$ does not appear in $\boldsymbol{x}$. If yes, we know that $\boldsymbol{y}$ appears in $\boldsymbol{x}$ once and only once. The best location of $\boldsymbol{y}$ in $\boldsymbol{x}$ is $\boldsymbol{x}[n_{\text{delay}} : n_{\text{delay}} + N_y]$, where $n_{\text{delay}} = \underset{n}{\operatorname{argmax}}\, \boldsymbol{s}[n]$. This process is tantamount to performing TDE to search for $\boldsymbol{y}$ in $\boldsymbol{x}$.

3.7 Static Synchronization

Static synchronization is defined as a process to find corresponding points in two or more signals to be compared. The corresponding points can be the starting moments, the stopping moments, any layer changing moments, or any characteristic moments[7].

3.7.1 Motivation

Requirements on Signals. The NSYNC framework merely compares two signals to determine if they are alike or not. For the NSYNC framework to act as an IDS, it is important that $\boldsymbol{a}$ and $\boldsymbol{b}$ are similar to each other when their underlying G-code instructions are the same (and the G-code instructions are executed faithfully). For $\boldsymbol{a}$ and $\boldsymbol{b}$ to be similar to each other, their lengths (or duration) should be similar to each other.

However, the data acquisition system to collect side-channel signals starts before a printing process starts and stops after the printing process stops. This is to ensure that the side-channel signals cover the whole printing process. As a result, there are empty parts in a side-channel signal that are irrelevant to the printing process and should be excluded in the signal comparison process. In addition, according to Figure 3.5 and Figure 3.6, there are environment-dependent procedures in a printing process, such as heating and cooling nozzles and build plates, that must be excluded in the signal comparison process.

Starting and Stopping Moments. We define the starting moment of a side-channel signal as the moment when the NSYNC framework starts to perform signal comparison and the stopping moment of the side-channel signal as the moment when the NSYNC framework terminates the signal comparison process. In other words, the NSYNC framework only compares $\boldsymbol{a}$ and $\boldsymbol{b}$ between their starting and stopping moments. According to this definition, the starting and stopping moments do not refer to the starting and stopping moments of a printing process, but the starting and stopping moments of a signal compar-

[7]Notice that the word "moments" is plural. This is because the side-channel signals to be compared are from different printing processes. Each printing process has a starting moment and a stopping moment. We are interested in identifying all of the starting moments and the stopping moments.

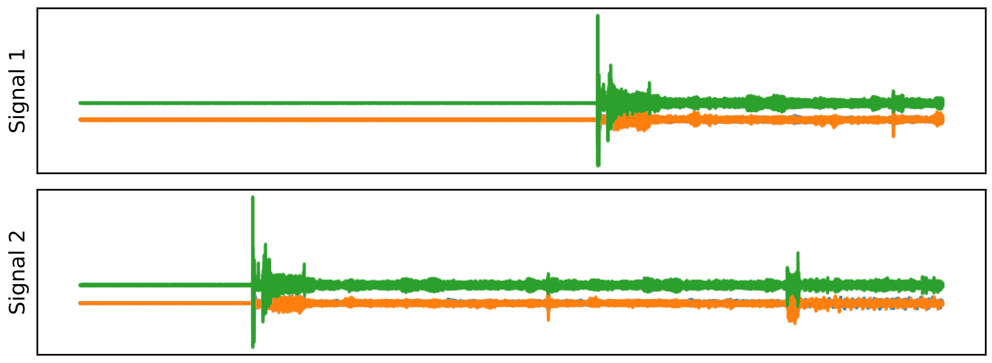


Figure 3.8: Two signals to be compared that are not yet aligned at the starting moments.

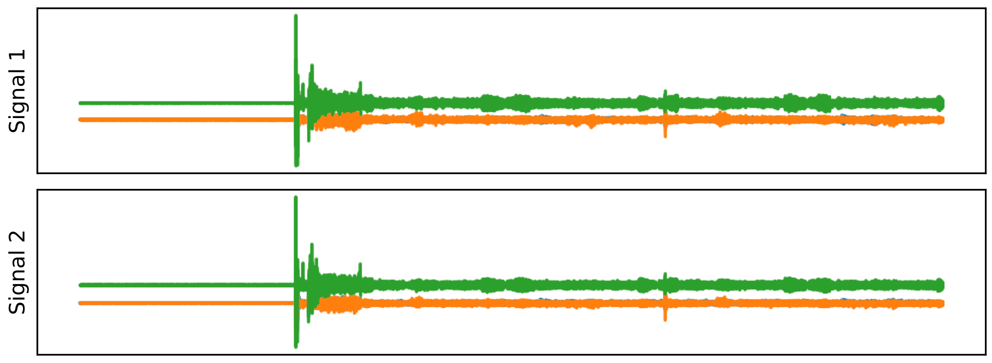


Figure 3.9: Two signals to be compared that are aligned at the starting moments.

ison process. The arrows in Figure 3.5 show the starting and stopping moments for the UM3 printer whereas the arrows in Figure 3.6 show the starting and stopping moments for the RM3 printer.

The primary motivation to develop static synchronization is to automatically and precisely identify the starting and stopping moments in a side-channel signal. In this way, side-channel signals to be compared can be aligned at their starting moments and the signal comparison process can be terminated at their stopping moments. Figure 3.8 shows two signals to be compared that are not aligned at the starting moments, whereas Figure 3.9 shows two signals to be compared that are aligned at the starting moments. If the two signals are directly compared point-by-point or window-by-window without the initial alignment, the comparison will not make sense.

Several researchers in the literature are aware of the importance of aligning side-channel signals before comparison. For example, Moore et al. [23] proposed to use an electric wire to connect the printer and the intrusion detection system. When the warming up process completes, a pulse is sent over the electric wire to the intrusion detection system to initiate the signal comparison process. The main problem with this approach is that the air gap between the AM system and the IDS system is penetrated, and the environment-dependent procedures are included in the signal comparison process.

Belikovetsky et al [27] proposed to use the M300 and G4 instructions to artificially play beeps at the starting and stopping moments to determine the starting and stopping moments in a recorded side-channel signal. However, it is likely that they manually identified the beeps in all of their recorded side-channel signals. A mechanism is still needed to automatically identify the beeps, as it is not practical to manually identify the beeps precisely for a lot of recorded signals or a real-time signal.

3.7.2 Templates

Static synchronization attempts to find corresponding moments in two or more signals, and the moments can be the starting moments, the stopping moments, or any layer changing moments. How to specify the moment for static synchronization to look for? One solution is to use a template, which is a sample of a signal around that moment.

Manually Obtain Templates. When inspecting a side-channel signal, we can manually identify the moments of interest. Figure 3.10 shows an example of the acceleration side-channel signal for a printing process by the UM3 printer. The green vertical line on the left sub-figure shows the manually identified starting moment and the green shaded area shows the template for the starting moment. The red vertical line on the right sub-figure shows the manually identified stopping moment and the red shaded area shows the template for the stopping moment.

It might be trivial to identify the starting and stopping moments by using the acceleration side-channel signal since there is a clear boundary at the starting moment or the

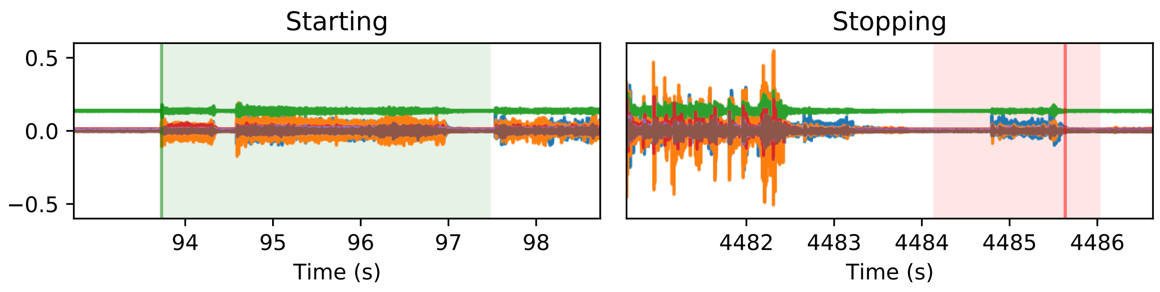


Figure 3.10: Manually obtaining the starting and stopping templates of a printing process.

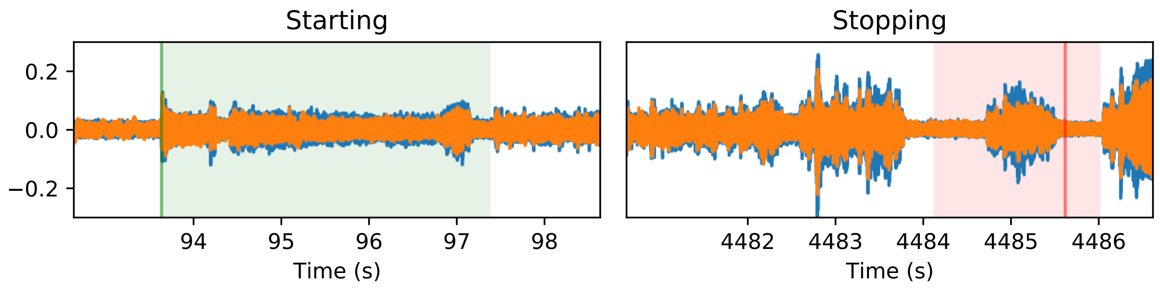


Figure 3.11: Acoustic side-channel signal around the starting and stopping moments.

stopping moment. However, it is non-trivial to do so by using other side-channel signals, such as the acoustic side-channel signal, as shown in Figure 3.11. This is because the acoustic side channel is susceptible to environment noise. To mitigate this problem, we can use the spectrogram of the acoustic side-channel signal to identify the starting and stopping moments, as shown in Figure 3.12. The spectrogram is less susceptible to the environment noise and we can clearly see the patterns and the boundaries in the spectrogram.

Anchor Point of a Template. A moment is typically defined by a single point in a signal whereas a template has a duration. The moment that a template represents is referred to as the anchor point of the template. For example, the anchor point of the template in the

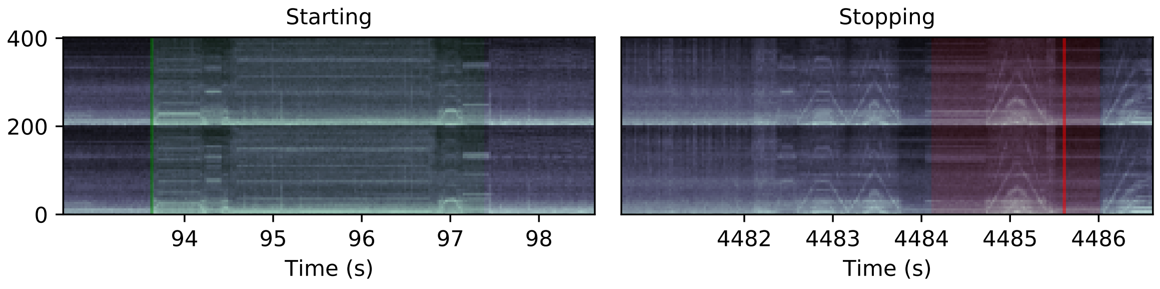


Figure 3.12: Acoustic side-channel signal around the starting and stopping moments.

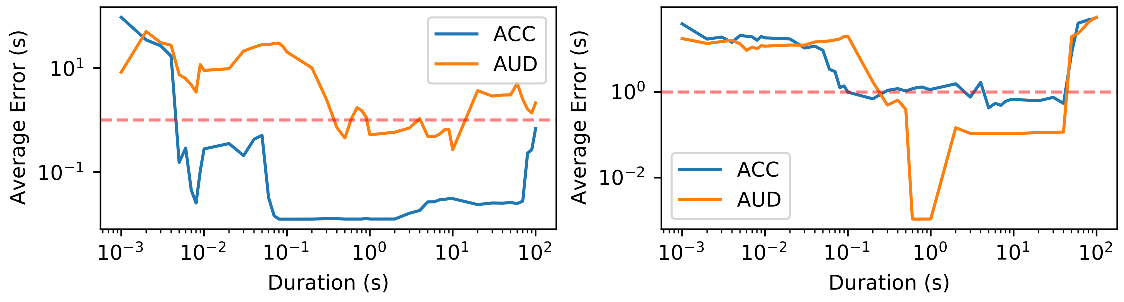


Figure 3.13: Illustration of the influence of the duration of templates on the performance of static synchronization. ACC refers to the acceleration side channel whereas AUD refers to the acoustic side channel. The results were derived with the UM3 printer. The left sub-figure shows the result for estimating the starting moment whereas the right sub-figure shows the result for estimating the stopping moment.

left sub-figure in Figure 3.10 is the left edge of the template. The anchor point of the template in the right sub-figure in Figure 3.10 is 1.5 seconds to the right of the left edge. In this thesis, the location of an anchor point is expressed in terms of its displacement with respect to the left edge of the template. According to this rule, the anchor point in Figure 3.10 is at 0 second, whereas the anchor point in Figure 3.10 is at 1.5 seconds. In general, it is possible for an anchor point to be located at any location in its template. It is even possible for an anchor point to be located outside of its template.

Duration of a Template. The template for the starting moment in Figure 3.10 has a duration of 3.75 seconds whereas the template for the stopping moment in the same figure has a duration of 1.9 seconds. Figure 3.13 illustrates the influence of the duration of templates on the performance of static synchronization and the performance was measured by the average error between the estimated moment and the actual moment. We can see that the best performance is achieved for an intermediate duration. When selecting the duration of a template, there is a trade-off. On the one hand, the template should be long enough to be characteristic. On the other hand, if the template is very long, it may not be the same across different printing processes, causing degradation of performance.

Two Types of Templates. There are two types of templates, and they are process-dependent templates and printer-dependent templates. If a template is only applicable to a specific printing process (the same G-code file and the same printer), then the template is process-dependent. If a template is applicable to all printing processes by the same printer, then the template is printer-dependent. When a template is fully contained in a printer-dependent procedure, the template is printer-dependent. If a template is contained in a process-dependent procedure, there is still a chance for the template to be printer-dependent. For example, the last second of the printhead-homing procedure of the RM3 printer is printer-dependent even though the procedure itself is process-dependent.

It is desirable to have printer-dependent templates since they can be used across different printing processes. For the RM3 printer, we were able to obtain printer-dependent templates for both of the starting moments and the stopping moments. However, for the UM3 printer, we were only able to obtain process-dependent templates for the starting moments and the stopping moments.

3.7.3 Modes of Operation

The NSYNC framework can be operated in two modes, and they are the real-time mode (or the online mode) and the post-production mode (or the offline mode). The main difference between the two modes is the accessibility to the observed signal. For the real-time mode, we have access to the observed signal up to the current moment. For the post-production mode, we have access to the whole observed signal.

As with the NSYNC framework, static synchronization can be operated in two modes. One of the operation modes is the Time Delay Estimation (TDE) mode, whereas the other mode is the Signal Detection (SD) mode. The TDE mode is mainly used for analyzing post-production (offline) signals, whereas the SD mode is mainly used for analyzing real-time (online) signals.

For the TDE mode, the assumption is that the template appears and appears only once in the signal to be analyzed. The task for static synchronization is to determine the best location of the template in the signal. This task can be accomplished by what is known as

Time Delay Estimation (TDE) [48, 49]. When we have access to the whole side-channel signal for a printing process and we know that there is only a single starting moment (or a single stopping moment), we can perform static synchronization in the TDE mode.

For the SD mode, the assumption is that the template may appear in the signal zero or multiple times. We do not know in advance exactly how many times the template will appear in the signal. However, we may know the upper limit on the number of times that the template may appear in the signal. One method to implement this mode of static synchronization is Signal Detection (SD).

3.7.4 Static Synchronization in the TDE Mode

When we are sure that a template appears in a signal to be analyzed once and only once, we can use TDE to figure out the location of the template in the signal. For example, we typically have access to the whole reference signal $\boldsymbol{b}$ and we know that the starting moment should appear in $\boldsymbol{b}$ once and only once. Suppose the template of the starting moment is $\boldsymbol{t}$. We can perform TDE to determine the best location of $\boldsymbol{t}$ in $\boldsymbol{b}$. It is possible for the template of the starting moment and the template of the stopping moment to be very similar to each other. When this is true, we need to restrict the range of $\boldsymbol{b}$. For example, if we know the approximate location of the starting moment in $\boldsymbol{b}$, we can perform TDE to search for $\boldsymbol{t}$ in the vicinity of the approximate location of the starting moment in $\boldsymbol{b}$. If we know that the starting moment in $\boldsymbol{b}$ must be within the first 200 seconds of $\boldsymbol{b}$, we only need to perform TDE to search for $\boldsymbol{t}$ in the first 200 seconds of $\boldsymbol{b}$. Other advantages of performing TDE locally include reduced analysis time and increased accuracy. When NSYNC is operated in the post-production mode and we have access to the whole observed signal $\boldsymbol{a}$, we can perform TDE to find the starting moment in $\boldsymbol{a}$ in the same way.

We can perform TDE to search for the stopping moment in a signal in a similar way. There are two main differences. First, we use the template of the stopping moment, instead of the template of the starting moment. Second, the stopping moment of a signal is typically close to the end of the signal. For example, we may want to search for the stopping moment in the last 100 seconds of the signal.

If the NSYNC framework is operated in the post-production mode and we have access to the whole observed signal $\boldsymbol{a}$, we can use TDE to obtain the starting moments for both $\boldsymbol{a}$ and $\boldsymbol{b}$. We can then align the two signals at their starting moments. This completes static synchronization at the starting moments. However, if the NSYNC framework is operated in the real-time mode, we do not have access to the whole observed signal $\boldsymbol{a}$, and we cannot use TDE to figure out the starting moment for $\boldsymbol{a}$. When this situation happens, we need to resort to SD to figure out the starting moment of $\boldsymbol{a}$.

3.7.5 Static Synchronization in the SD Mode

When we cannot guarantee that a template appears in the signal to be analyzed once and only once, we can use SD to determine if the template appears in the signal to be analyzed, and if so, the locations of all occurrences of the template in the signal. There are two exemplary scenarios where TDE is not applicable and SD is necessary.

One scenario is to look for layer changing moments in an offline signal. This is because there are many occurrences of the layer changing moments in the signal, and this breaks the assumption in TDE that the template appears once and only once in the signal to be analyzed. For this scenario, we can perform SD with NMS to determine the locations of all occurrences of the template in the signal.

Another scenario is when the NSYNC framework is operated in the real-time mode, and we have access to the observed signal $\boldsymbol{a}$ until the current moment. It is possible that the template may not have occurred in the signal up to the current moment. When the NSYNC framework is operated in the real-time mode, the data acquisition system feeds the observed side-channel signal frame by frame into the IDS. Suppose the template of the starting moments is $\boldsymbol{t}$ and its critical score is s_c. We can perform SD to determine if $\boldsymbol{t}$ has occurred in $\boldsymbol{a}$ up to the current moment, and if so, the moment of $\boldsymbol{t}$ in $\boldsymbol{a}$.

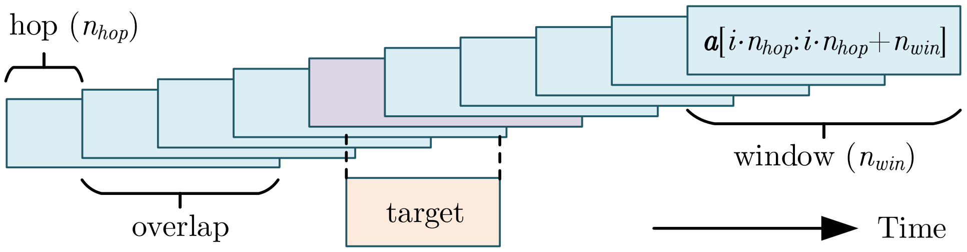


Figure 3.14: Illustration of hopping windows. When the overlap is wider than the target signal, the target signal must be fully included in one of the windows.

An Efficient Implementation. When the NSYNC framework is operated in the real-time mode, there is an efficient method to perform static synchronization in the SD mode. Suppose we are looking for $\boldsymbol{t}$ (the template) in the observed signal $\boldsymbol{a}$. We can access $\boldsymbol{a}$ window by window, and each window is expressed by $\boldsymbol{a}[i \cdot n_{\text{hop}} : i \cdot n_{\text{hop}} + n_{\text{win}}], i = 0, 1, \cdots$, where i is the window index, n_{win} is the width of the window in time index, and n_{hop} is the number of points by which the window moves forward each time. For each window, we perform SD to detect $\boldsymbol{t}$ in $\boldsymbol{a}[i \cdot n_{\text{hop}} : i \cdot n_{\text{hop}} + n_{\text{win}}]$ with the assumption that $\boldsymbol{t}$ appears zero or one time. Once $\boldsymbol{t}$ is detected in $\boldsymbol{a}[i \cdot n_{\text{hop}} : i \cdot n_{\text{hop}} + n_{\text{win}}]$ with a time delay of j, then the left edge of $\boldsymbol{t}$ is $i \cdot n_{\text{hop}} + j$.

As shown in Figure 3.14, as long as the overlap between windows exceed the length of $\boldsymbol{t}$, $\boldsymbol{t}$ will be fully contained in at least one window. This is because the left edge of $\boldsymbol{t}$ must be contained in the hop region of at least one window. Since the overlap region of two adjacent windows is wider than $\boldsymbol{t}$, $\boldsymbol{t}$ must be contained within the window. By default, in NSYNC, we select $n_{\text{win}} = 2n_{\text{hop}}$ and let n_{hop} be equal to the length of $\boldsymbol{t}$.

We can use the aforementioned method to detect the starting moment as well as the stopping moment in $\boldsymbol{a}$, provided that the templates and the critical scores are available. When the anchor point of a template is not located at the left edge of the template, the displacement of the anchor point with respect to the left edge of the template must be added to $i \cdot n_{\text{hop}} + j$.

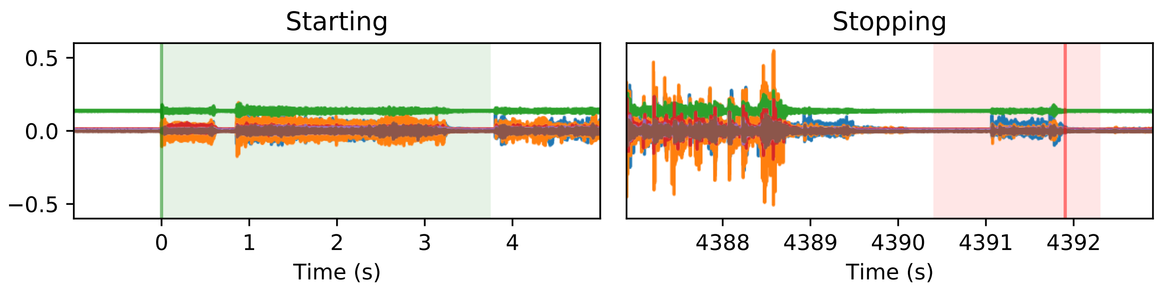


Figure 3.15: Acceleration side-channel signal with the time index readjusted.

3.7.6 Summary of Static Synchronization

The primary purpose of static synchronization is to identify the starting and stopping moments in the observation $\boldsymbol{a}$ and the reference $\boldsymbol{b}$. When the NSYNC framework operates in the real-time mode, we can perform static synchronization in the SD mode to figure out the starting and stopping moments in $\boldsymbol{a}$ and we can perform static synchronization in the TDE mode to determine the starting and stopping moments in $\boldsymbol{a}$. When the NSYNC framework operates in the post-production mode, we can perform static synchronization in the TDE mode to figure out the starting and stopping moments for both $\boldsymbol{a}$ and $\boldsymbol{b}$.

When the starting moments of $\boldsymbol{a}$ and $\boldsymbol{b}$ are determined, they can be aligned at their starting moments and the signal comparison process can be initiated. We typically offset the time axes of $\boldsymbol{a}$ and $\boldsymbol{b}$ such that the starting moments have a time index of zero, as shown in Figure 3.15. The stopping moments of $\boldsymbol{a}$ and $\boldsymbol{b}$ are used for terminating the signal comparison process. To be specific, when any of the stopping moments of $\boldsymbol{a}$ and $\boldsymbol{b}$ is reached, the signal comparison process is complete. It should be noted that we cannot align $\boldsymbol{a}$ and $\boldsymbol{b}$ at their stopping moments. This is because $\boldsymbol{a}$ and $\boldsymbol{b}$ are aligned at their starting moments and they may have different lengths (or duration). As a result, it is impossible to align $\boldsymbol{a}$ and $\boldsymbol{b}$ at their stopping moments.

3.7.7 Duration-Based Intrusion Detection

When we perform static synchronization to determine both the starting and stopping moments for a signal, we can estimate its duration. Suppose the duration of $\boldsymbol{a}$ in seconds is T_a and the duration of $\boldsymbol{b}$ in seconds is T_b. We can compare T_a and T_b to perform intrusion detection. If the absolute difference between T_a and T_b exceeds a certain threshold, an intrusion is declared. In other words, an intrusion is detected if

$$|T_a - T_b| > T_c, \tag{3.11}$$

where T_c is the threshold or the critical duration.

To obtain the threshold T_c, we need a series of training signals $\boldsymbol{a_m}, m = 0, 1, \cdots, M-1$ that indeed come from benign printing processes. Suppose their lengths in seconds are $T_{a,m}, m = 0, 1, \cdots, M-1$. The threshold T_c is then equal to the upper bound of $\{|T_{a,m} - T_b|\}$ plus a margin. The margin is required because the number of training signals is limited and the length of a signal from a new benign printing process may have a larger deviation from T_b. As a simple solution, we can let T_c to be

$$T_c = (1 + r) \max_m |T_{a,m} - T_b| \tag{3.12}$$

and we can start with $r = 0.25$. It can be expected that the margin coefficient r decreases as the number of training samples M increases and vice versa.

The duration-based intrusion detection system can be defeated if an attacker is able to craft an attack such that the duration of the malicious printing process is equal to the duration of the benign printing process. Nevertheless, this duration-based IDS may still defeat a naive attacker who is not aware of the existence of the duration-based IDS.

3.8 Dynamic Synchronization

Dynamic Synchronization (DSYNC) refers to any process that continuously identifies corresponding points or windows in two signals ($\boldsymbol{a}$ and $\boldsymbol{b}$). DSYNC is needed when there is time noise in $\boldsymbol{a}$ and $\boldsymbol{b}$. In this section, we first discuss Dynamic Time Warping (DTW), an existing method to perform point-based DSYNC. We then discuss Dynamic Window Matching (DWM), a novel method to perform window-based DSYNC.

3.8.1 Dynamic Time Warping (DTW)

Point-Based Comparison. Suppose $\boldsymbol{a}$ and $\boldsymbol{b}$ have been aligned at their starting moments[8] with a reasonable accuracy (not necessarily perfect), and we want to compare $\boldsymbol{a}$ and $\boldsymbol{b}$ point by point. One may calculate the distance between $\boldsymbol{a}[n]$ and $\boldsymbol{b}[n]$ for $n = 0, 1, \cdots$. However, due to time noise, the comparison between $\boldsymbol{a}[n]$ and $\boldsymbol{b}[n]$ is meaningless since they may not be corresponding points.

Overview of DTW. Dynamic Time Warping (DTW) is an existing method to find the corresponding points between $\boldsymbol{a}$ and $\boldsymbol{b}$ [29]. DTW requires a distance metric $d(\cdot, \cdot)$ be provided, and then outputs a list of tuples where a tuple (i, j) specifies that $\boldsymbol{a}[i]$ and $\boldsymbol{b}[j]$ are corresponding points.

FastDTW. Due to the time complexity of DTW, a variation of DTW, called FastDTW, is typically used [53]. FastDTW requires an additional parameter called the radius. Because FastDTW is an approximation to DTW, this parameter controls the trade-off between speed and accuracy. We always use the smallest radius for the fastest speed because it takes a very long time to analyze side-channel signals by FastDTW. In this thesis, we simply use DTW to refer to FastDTW.

Online DTW. DTW requires knowing the whole $\boldsymbol{a}$ and the whole $\boldsymbol{b}$ before they can be analyzed. In other words, DTW does not support real-time analysis. Fortunately, there is an ongoing effort to create a version of DTW that supports real-time analysis [54].

[8]The time axes are adjusted such that the starting moments have a time index of zero.

Horizontal Displacement $\boldsymbol{h}_{\text{disp}}$. If there is only one tuple with the first index being i and the tuple is (i, j), we define $j - i$ as the horizontal displacement of $\boldsymbol{b}$ with respect to $\boldsymbol{a}$ at index i. In other words, we have $\boldsymbol{h}_{\text{disp}}[i] = j - i$. If there are multiple tuples with the first index being i, such as $(i, j_1), (i, j_2), \cdots, (i, j_{K_i})$, we define

$$\boldsymbol{h}_{\text{disp}}[i] = \frac{1}{K_i} \sum_{k=1}^{K_i} j_k - i. \tag{3.13}$$

3.8.2 Dynamic Window Matching (DWM)

Window-Based Comparison. Instead of comparing $\boldsymbol{a}$ and $\boldsymbol{b}$ point by point, we can alternatively compare $\boldsymbol{a}$ and $\boldsymbol{b}$ window by window. To be specific, we calculate the distance between $\boldsymbol{a}\{i\}$ and $\boldsymbol{b}\{i\}$, where

$$\boldsymbol{a}\{i\} = \boldsymbol{a}[i \cdot n_{\text{hop}} : i \cdot n_{\text{hop}} + n_{\text{win}}], \tag{3.14}$$

$$\boldsymbol{b}\{i\} = \boldsymbol{b}[i \cdot n_{\text{hop}} : i \cdot n_{\text{hop}} + n_{\text{win}}], \tag{3.15}$$

$i = 0, 1, \cdots$ is the window index, n_{win} is the window width in indexes, and n_{hop} is the number of samples by which the windows move forward each time. $\boldsymbol{a}\{i\}$ is referred to as the ith window of $\boldsymbol{a}$, whereas $\boldsymbol{b}\{i\}$ is referred to as the ith window of $\boldsymbol{b}$. Due to time noise, the comparison between $\boldsymbol{a}\{i\}$ and $\boldsymbol{b}\{i\}$ can be meaningless, even if $\boldsymbol{a}[0]$ is perfectly aligned with $\boldsymbol{b}[0]$.

Overview of DWM. To solve this problem, for each $\boldsymbol{a}\{i\}$, instead of comparing it with $\boldsymbol{b}\{i\}$, we attempt to find a better window of $\boldsymbol{b}$ to compare with. Suppose such a window does exist and it can be expressed by

$$\boldsymbol{b}\{i; \boldsymbol{h}_{\text{disp}}[i]\} = \boldsymbol{b}[i \cdot n_{\text{hop}} + \boldsymbol{h}_{\text{disp}}[i] : i \cdot n_{\text{hop}} + \boldsymbol{h}_{\text{disp}}[i] + n_{\text{win}}] \tag{3.16}$$

where $\boldsymbol{h}_{\text{disp}}[i]$ is referred to as the horizontal displacement of $\boldsymbol{b}$ with respect to $\boldsymbol{a}$ at index

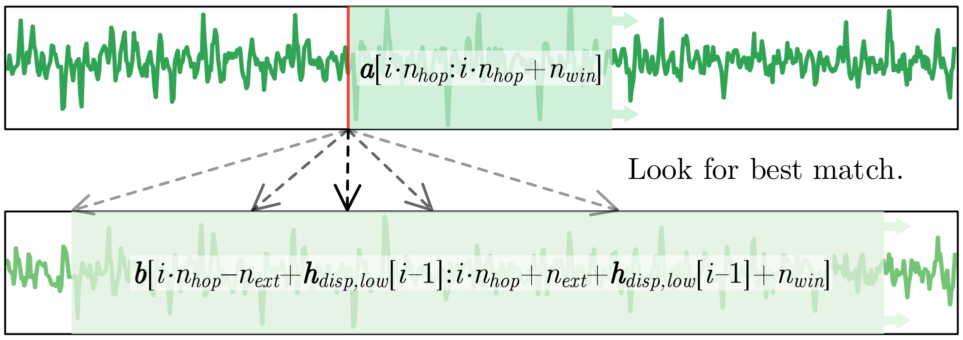

$\boldsymbol{a}[i\cdot n_{hop}:i\cdot n_{hop}+n_{win}]$
Look for best match.
$\boldsymbol{b}[i\cdot n_{hop}-n_{ext}+\boldsymbol{h}_{disp,low}[i-1]:i\cdot n_{hop}+n_{ext}+\boldsymbol{h}_{disp,low}[i-1]+n_{win}]$

Figure 3.16: Illustration of the DWM algorithm. A pair of sliding windows are established on the signals. As the windows slide across the signals, Time Delay Estimation (TDE) is used to determine the relative timing relationship between the pair of windows.

i and $\boldsymbol{b}\{i; \boldsymbol{h}_{\text{disp}}[i]\}$ is referred to as the ith window of $\boldsymbol{b}$ with an offset of $\boldsymbol{h}_{\text{disp}}[i]$. The absolute value of $\boldsymbol{h}_{\text{disp}}[i]$ is the horizontal distance, denoted by $\boldsymbol{h}_{\text{dist}}[i]$.

Dynamic Window Matching (DWM) is a novel algorithm to find the corresponding windows between $\boldsymbol{a}$ and $\boldsymbol{b}$. The core of DWM is to determine the best $\boldsymbol{h}_{\text{disp}}$ such that $\boldsymbol{a}\{i\}$ corresponds to $\boldsymbol{b}\{i; \boldsymbol{h}_{\text{disp}}[i]\}$.

A Basic Algorithm to Find $\boldsymbol{h}_{\text{disp}}$. We present a basic algorithm to find $\boldsymbol{h}_{\text{disp}}$. For each window index i, we look for $\boldsymbol{a}\{i\}$ in the vicinity of $\boldsymbol{b}\{i\}$, as shown in Figure 3.16. To be more precise, we perform TDE to detect $\boldsymbol{a}\{i\}$ in

$$\boldsymbol{b}\{i\}_{\text{E}} = \boldsymbol{b}[i \cdot n_{\text{hop}} - n_{\text{ext}} : i \cdot n_{\text{hop}} + n_{\text{ext}} + n_{\text{win}}], \tag{3.17}$$

where n_{ext} is a new parameter, called the extended window size, and $\boldsymbol{b}\{i\}_{\text{E}}$ is the extended ith window of $\boldsymbol{b}$. Suppose TDE returns a time delay of j. As a result, TDE thinks that $\boldsymbol{a}\{i\}$ is aligned with $\boldsymbol{b}\{i; j - n_{\text{ext}}\}$. We then let

$$\boldsymbol{h}_{\text{disp}}[i] = j - n_{\text{ext}}. \tag{3.18}$$

Extending the Range of $\boldsymbol{h}_{\text{disp}}$. The basic algorithm to find $\boldsymbol{h}_{\text{disp}}[i]$ might work well if the magnitude of the actual $\boldsymbol{h}_{\text{disp}}[i]$ does not exceed n_{ext}. Otherwise, the algorithm

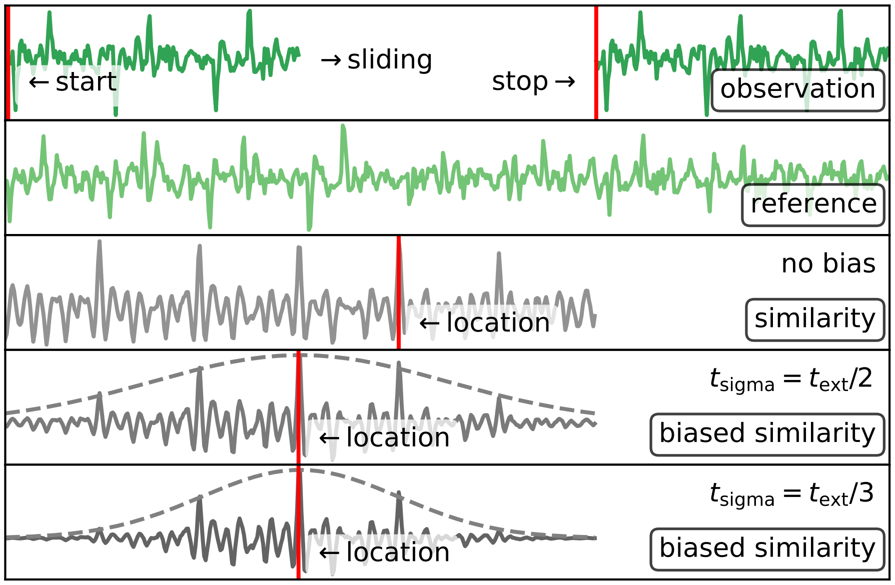


Figure 3.17: Illustration of Time Delay Estimation with Bias (TDEB).

will definitely fail. This is because, according to Equation 3.18, the range of $\boldsymbol{h}_{\text{disp}}[i]$ is $[-n_{\text{ext}}, n_{\text{ext}}]$. Hence, if the magnitude of the actual $\boldsymbol{h}_{\text{disp}}[i]$ exceeds n_{ext}, it is impossible for the basic algorithm to return a correct $\boldsymbol{h}_{\text{disp}}[i]$.

To solve this problem, we perform TDE to detect $\boldsymbol{a}\{i\}$ in $\boldsymbol{b}\{i; \boldsymbol{h}_{\text{disp}}[i-1]\}_{\text{E}}$. Suppose TDE returns j. We then perform the assignment

$$\boldsymbol{h}_{\text{disp}}[i] = j - n_{\text{ext}} + \boldsymbol{h}_{\text{disp}}[i-1]. \tag{3.19}$$

For $i = 0$, we define $\boldsymbol{h}_{\text{disp}}[i-1]$ to be 0.

Time Delay Estimation with Bias (TDEB). As shown in Figure 3.17, when $\boldsymbol{a}\{i\}$ is mainly composed of periodic signals, multiple time delays are returned by TDE with equal probability. Similarly, when $\boldsymbol{a}\{i\}$ is mainly composed of noise, TDE returns a random time delay. In a word, when $\boldsymbol{a}\{i\}$ is periodic or noisy, TDE is unstable.

To solve this problem, we rely on the assumption that $\boldsymbol{h}_{\text{disp}}[i]$ should be close to $\boldsymbol{h}_{\text{disp}}[i-1]$ most of the time. In other words, j should be close to n_{ext} most of the time.

When performing TDE, as an intermediate step, we obtain a similarity array $\boldsymbol{s}[j]$, $j =$

$0, 1, \cdots, 2n_{\text{ext}} - 1$. To increase similarity scores near $j = n_{\text{ext}}$, we multiply the similarity array by a Gaussian window with a standard deviation of n_{sigma} and a length of $2n_{\text{ext}}$, as shown in Figure 3.17. We then continue TDE with the modified similarity array. In this way, we introduce bias towards $j = n_{\text{ext}}$. When $\boldsymbol{a}\{i\}$ is periodic or noisy, $\boldsymbol{h}_{\text{disp}}[i]$ will be close to $\boldsymbol{h}_{\text{disp}}[i-1]$.

Low Frequency Component of $\boldsymbol{h}_{\text{disp}}$. There is a new problem after extending the range of $\boldsymbol{h}_{\text{disp}}$. If, for any reason, the estimated value of $\boldsymbol{h}_{\text{disp}}[i-1]$ deviates significantly from its true value, it may cause $\boldsymbol{h}_{\text{disp}}[i]$ to deviate significantly from its true value, which in turn causes further deviation in $\boldsymbol{h}_{\text{disp}}[i+1]$, etc. In other words, the DWM process may run away. To mitigate this problem, we obtain a low frequency component of $\boldsymbol{h}_{\text{disp}}$ in the following way

$$\boldsymbol{h}_{\text{disp,low}}[i] = \text{round}(\eta(j - n_{\text{ext}}) + \boldsymbol{h}_{\text{disp,low}}[i-1]), \tag{3.20}$$

where η is a parameter that controls how fast $\boldsymbol{h}_{\text{disp,low}}$ can be affected by $j - n_{\text{ext}}$. Now, we perform TDEB to detect $\boldsymbol{a}\{i\}$ in $\boldsymbol{b}\{i; \boldsymbol{h}_{\text{disp,low}}[i-1]\}_{\text{E}}$. Suppose TDEB returns j. We then perform the assignment

$$\boldsymbol{h}_{\text{disp}}[i] = j - n_{\text{ext}} + \boldsymbol{h}_{\text{disp,low}}[i-1]. \tag{3.21}$$

The Final Algorithm to Find $\boldsymbol{h}_{\text{disp}}$. The final DWM algorithm is presented in Algorithm 1, where $\text{TDEB}[\beta](\boldsymbol{x}, \boldsymbol{y})$ is a function that finds the time delay of $\boldsymbol{y}$ in $\boldsymbol{x}$ biased by a Gaussian window with a standard deviation of n_{sigma}.

3.9 Comparators

Once the corresponding points or windows between $\boldsymbol{a}$ and $\boldsymbol{b}$ are identified, a distance value can be calculated for each pair of points or windows, which are further used by the discriminator for intrusion detection.

Algorithm 1 Dynamic Synchronization

```
Input: $\boldsymbol{a}$, $\boldsymbol{b}$, $n_{\text{win}}$, $n_{\text{hop}}$, $n_{\text{ext}}$, $n_{\text{sigma}}$, $\eta$
Output: $\boldsymbol{h}_{\text{disp}}[i], i = 0, 1, \cdots$
 1: Define $\boldsymbol{h}_{\text{disp}}$ as an array that can increase in size.
 2: Define $\boldsymbol{h}_{\text{disp,low}}$ as an array that can increase in size.
 3: Add a special element $\boldsymbol{h}_{\text{disp,low}}[-1] = 0$.
 4: $i = 0$
 5: Wait for the printing process to start.
 6: while the printing process is not over do
 7:     Wait for $\boldsymbol{a}[i \cdot n_{\text{hop}} : i \cdot n_{\text{hop}} + n_{\text{win}}]$ to be become available.
 8:     $j$ = $\text{TDEB}[n_{\text{sigma}}]($
            $\boldsymbol{b}[i \cdot n_{\text{hop}} - n_{\text{ext}} + \boldsymbol{h}_{\text{disp,low}}[i-1] :$
              $i \cdot n_{\text{hop}} + n_{\text{ext}} + \boldsymbol{h}_{\text{disp,low}}[i-1] + n_{\text{win}}]$,
            $\boldsymbol{a}[i \cdot n_{\text{hop}} : i \cdot n_{\text{hop}} + n_{\text{win}}]$
        $)$
 9:     $\boldsymbol{h}_{\text{disp}}[i] = j - n_{\text{ext}} + \boldsymbol{h}_{\text{disp,low}}[i-1]$
10:     $\boldsymbol{h}_{\text{disp,low}}[i] = \text{round}(\eta(j - n_{\text{ext}}) + \boldsymbol{h}_{\text{disp,low}}[i-1])$
11:     $i = i + 1$
12: end while
13: return $\boldsymbol{h}_{\text{disp}}$
```

Vertical Distance $\boldsymbol{v}_{\text{dist}}$. If $\boldsymbol{a}$ and $\boldsymbol{b}$ are synchronized by DTW, we can calculate $\boldsymbol{v}_{\text{dist}}$ between $\boldsymbol{a}$ and $\boldsymbol{b}$ point by point. If DTW returns only one tuple with the first index being i, namely (i, j), we define $\boldsymbol{v}_{\text{dist}}[i] = d(\boldsymbol{a}[i], \boldsymbol{b}[j])$, where f is a function (also known as the distance function) to calculate the distance. If there are multiple tuples with the first index being i, such as $(i, j_1), (i, j_2), \cdots, (i, j_{K_i})$, we define

$$\boldsymbol{v}_{\text{dist}}[i] = \frac{1}{K_i} \sum_{k=1}^{K_i} d(\boldsymbol{a}[i], \boldsymbol{b}[j_k]). \tag{3.22}$$

If $\boldsymbol{a}$ and $\boldsymbol{b}$ are synchronized by DWM, we can calculate the vertical distances between $\boldsymbol{a}$ and $\boldsymbol{b}$ window by window. Suppose $\boldsymbol{a}\{i\}$ is the ith window of $\boldsymbol{a}$ and its corresponding window in $\boldsymbol{b}$ is $\boldsymbol{b}\{i; \boldsymbol{h}_{\text{disp}}[i]\}$. We have

$$\boldsymbol{v}_{\text{dist}}[i] = d(\boldsymbol{a}\{i\}, \boldsymbol{b}\{i; \boldsymbol{h}_{\text{disp}}[i]\}). \tag{3.23}$$

Distance Functions. In this thesis, we consider two distance functions, and they are the cosine distance and the correlation distance. Suppose $\boldsymbol{u}$ and $\boldsymbol{v}$ are 1-D vectors of the same length N. The cosine distance is defined by

$$d(\boldsymbol{u}, \boldsymbol{v}) = 1 - \frac{\boldsymbol{u} \cdot \boldsymbol{v}}{\|\boldsymbol{u}\| \cdot \|\boldsymbol{v}\|}. \tag{3.24}$$

The correlation distance is defined by

$$d(\boldsymbol{u}, \boldsymbol{v}) = 1 - \frac{(\boldsymbol{u} - \mu_u) \cdot (\boldsymbol{v} - \mu_v)}{\|\boldsymbol{u} - \mu_u\| \cdot \|\boldsymbol{v} - \mu_v\|}. \tag{3.25}$$

where $\|\cdot\|_2$ is the L2 norm operator and μ_u and μ_v are defined in Equation 3.5. The second term in Equation 3.25 is in fact the Pearson's correlation coefficient between $\boldsymbol{u}$ and $\boldsymbol{v}$.

Other than the cosine distance and the correlation distance, there are many other distance metrics (or distance functions), such as the Manhattan distance and the Euclidean distance. However, we do not consider the two distance metrics because they are sensitive to the overall amplitude of $\boldsymbol{a}$ and $\boldsymbol{b}$, and the overall amplitude of many side-channel signals is susceptible to changes.[9] If a distance metric that is sensitive to the overall amplitude of $\boldsymbol{a}$ and $\boldsymbol{b}$ is used in an IDS, the overall amplitude of $\boldsymbol{a}$ and $\boldsymbol{b}$ must be strictly controlled (which can be very hard). Otherwise, the IDS will suffer from a lot of false alerts.

Computation Axes. The distance functions in Equation 3.24 and Equation 3.25 require the two inputs be 1-D. When the inputs are $\boldsymbol{a}[i]$ and $\boldsymbol{b}[j]$, the inputs are 1-D. However, when the inputs are $\boldsymbol{a}\{i\}$ and $\boldsymbol{b}\{i; \boldsymbol{h}_{\text{disp}}[i]\}$, the inputs are 2-D, unless $C = 1$. When $C > 1$, there are two ways to apply the distance functions in Equation 3.24 and Equation 3.25. The first way is to calculate the distance value between $\boldsymbol{a}\{i\}[:, c]$ and $\boldsymbol{b}\{i; \boldsymbol{h}_{\text{disp}}[i]\}[:, c]$ for $c = 0, 1, \cdots, C-1$, and average the distance values across the channels. This way of calculation is referred to as calculating along the time axis. The second way is to flatten

[9]For example, the amplitude of an acoustic side-channel signal strongly depends on the distance from the microphone to the printer as well as the gain of the ADC converter, both of which are susceptible to changes.

both $\boldsymbol{a}\{i\}$ and $\boldsymbol{b}\{i; \boldsymbol{h}_{\text{disp}}[i]\}$ into 1-D vectors and proceed with the calculation. This way of calculation is referred to as calculating along the none axis. Both ways to deal with 2-D inputs will be evaluated in experiments to ascertain their performance.

3.10 Discriminators

The discriminator checks $\boldsymbol{a}$ and $\boldsymbol{b}$ in real time to automatically determine if $\boldsymbol{a}$ is significantly different from $\boldsymbol{b}$. If so, an intrusion is declared. The discriminator is composed of three sub-modules. If any sub-module raises an alert, an intrusion is declared. Each sub-module is discussed as follows.

3.10.1 Sub-Module 1: $\boldsymbol{c}_{\text{disp}}$-Based Detection

This sub-module checks $\boldsymbol{h}_{\text{disp}}$ to determine if the dynamic synchronization between $\boldsymbol{a}$ and $\boldsymbol{b}$ is successful. When dynamic synchronization succeeds, $\boldsymbol{h}_{\text{disp}}$ contains a few fluctuations, such as the benign process in Figure 3.18 (b). In contrast, when dynamic synchronization fails, $\boldsymbol{h}_{\text{disp}}$ contains a lot of fluctuations, such as the malicious process in Figure 3.18 (b). To capture this feature, we calculate the Cumulative Absolute Difference of the Horizontal Displacement (CADHD) by

$$\boldsymbol{c}_{\text{disp}}[i] = \sum_{j=0}^{i} |\boldsymbol{h}_{\text{disp}}[j] - \boldsymbol{h}_{\text{disp}}[j-1]|, \tag{3.26}$$

where $\boldsymbol{h}_{\text{disp}}[-1]$ is defined to be zero. Figure 3.18 (a) shows the CADHD arrays for a benign process where dynamic synchronization succeeded and a malicious process where dynamic synchronization failed. An intrusion is detected at index i if

$$\boldsymbol{c}_{\text{disp}}[i] > c_c, \tag{3.27}$$

where c_c is a critical value to be determined later.

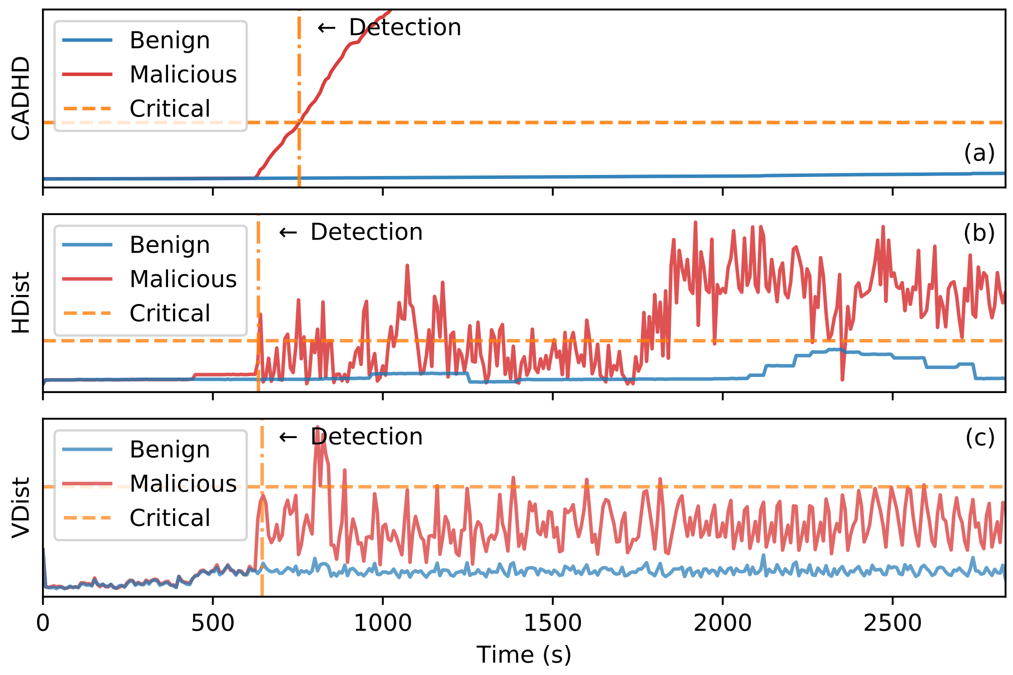


Figure 3.18: Automatic Intrusion Detection. (a) Intrusion is detected if $\boldsymbol{c}_{\text{disp}}[i]$ exceeds c_c. (b) Intrusion is detected if $\boldsymbol{h}_{\text{dist}}[i]$ exceeds h_c. (c) Intrusion is detected if $\boldsymbol{v}_{\text{dist}}[i]$ exceeds v_c.

3.10.2 Sub-Module 2: $\boldsymbol{h}_{\text{dist}}$-Based Detection

For $\boldsymbol{h}_{\text{dist}}$, as shown in Figure 3.18 (b), an intrusion is detected at index i if

$$\boldsymbol{h}_{\text{dist}}[i] > h_c, \tag{3.28}$$

where h_c is a critical horizontal distance to be determined.

3.10.3 Sub-Module 3: $\boldsymbol{v}_{\text{dist}}$-Based Detection

For $\boldsymbol{v}_{\text{dist}}$, as shown in Figure 3.18 (c), an intrusion is detected at index i if

$$\boldsymbol{v}_{\text{dist}}[i] > v_c, \tag{3.29}$$

where v_c is a critical vertical distance to be determined.

Suppressing Spikes. There are spikes in $\boldsymbol{h}_{\text{dist}}$ and $\boldsymbol{v}_{\text{dist}}$ due to time noise and amplitude noise. The spikes could cause false positives. To mitigate this problem, we filter $\boldsymbol{h}_{\text{dist}}$ and $\boldsymbol{v}_{\text{dist}}$ in the following ways before Equation 3.28 and Equation 3.29 are applied:

$$\boldsymbol{h}_{\text{dist},f}[i] = \min(\boldsymbol{h}_{\text{dist}}[i-n:i]), \quad i = 0, 1, \cdots, \tag{3.30}$$

$$\boldsymbol{v}_{\text{dist},f}[i] = \min(\boldsymbol{v}_{\text{dist}}[i-n:i]), \quad i = 0, 1, \cdots, \tag{3.31}$$

where $\boldsymbol{h}_{\text{dist},f}$ and $\boldsymbol{v}_{\text{dist},f}$ are filtered arrays and n is the window size of the filter. By default, we use a window size of 3 for both $\boldsymbol{h}_{\text{dist}}$ and $\boldsymbol{v}_{\text{dist}}$.

3.10.4 Learning the Critical Values

In this section, we describe a method that uses One-Class Classification (OCC) to determine the critical values c_c, h_c, and v_c in the discriminator. For this purpose, for one reference signal $\boldsymbol{b}$, we need to run the benign process M times and observe the side-channel signals $\boldsymbol{a}_m$, $m = 0, 1, \cdots, M-1$, where M is the number of observed signals.

Suppose $\boldsymbol{c}_{\text{disp,m}}$, $\boldsymbol{h}_{\text{dist,m}}$ and $\boldsymbol{v}_{\text{dist,m}}$ are obtained by comparing $\boldsymbol{a}_m$ and $\boldsymbol{b}$. Here, we assume that $\boldsymbol{h}_{\text{dist,m}}$ and $\boldsymbol{v}_{\text{dist,m}}$ are the *filtered* horizontal distance array the *filtered* vertical distance array respectively. We have

$$c_{c,m} = \max_i \boldsymbol{c}_{\text{disp,m}}[i], \tag{3.32}$$

$$h_{c,m} = \max_i \boldsymbol{h}_{\text{dist,m}}[i], \tag{3.33}$$

$$v_{c,m} = \max_i \boldsymbol{v}_{\text{dist,m}}[i]. \tag{3.34}$$

The critical distances are determined by

$$c_c = \max_m c_{c,m} + r\left(\max_m c_{c,m} - \min_m c_{c,m}\right), \tag{3.35}$$

$$h_c = \max_m h_{c,m} + r\left(\max_m h_{c,m} - \min_m h_{c,m}\right), \tag{3.36}$$

$$v_c = \max_m v_{c,m} + r\left(\max_m v_{c,m} - \min_m v_{c,m}\right), \tag{3.37}$$

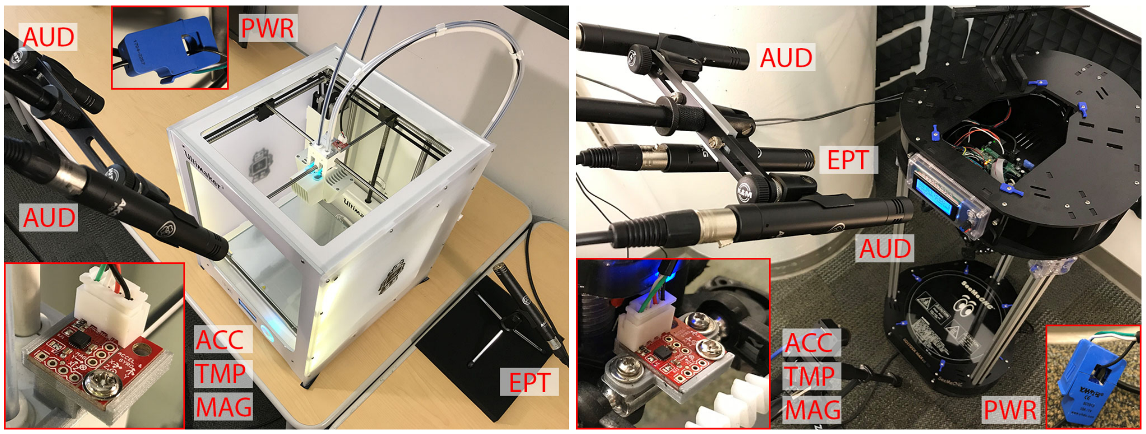


Figure 3.19: Experiment setups. (a) Ultimaker 3 and various sensors, (b) SeeMeCNC Rostock Max V3 and various sensors.

The parameter r determines the False Positive Rate (FPR) and the False Negative Rate (FNR). The higher the value of r, the lower the FPR, but the higher the FNR. The value of r depends on M, the sample size. To maintain the same FPR, r gets smaller when M becomes larger. In NSYNC, we select an r that results in a small FPR (< 0.05) for most scenarios.

3.11 Experiments on Static Synchronization

We performed a series of experiments to evaluate the performance of static synchronization in NSYNC. We first of all present details of the experiment setup. Afterwards, we discuss how to determine the parameters in static synchronization. Finally, we present experiment results on static synchronization and a duration-based intrusion detection system.

3.11.1 Experiment Setup

Printers. We performed experiments on an Ultimaker 3 printer (UM3) and a SeeMeCNC Rostock Max V3 printer (RM3), as shown in Figure 3.19. The UM3 printer was a popular desktop 3D printer [55], whereas the RM3 printer was a popular Delta printer.

Printing Processes. We selected a gear model with a diameter of 60 mm and a thickness of 7.5 mm, as shown in Figure 3.20 (a). For UM3, we used Cura 4.4 as the slicer. For

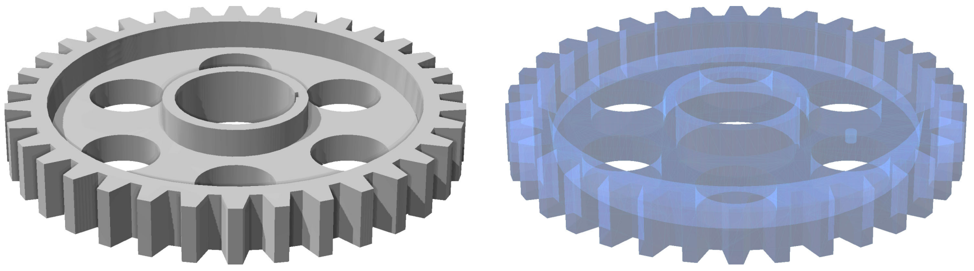


Figure 3.20: Models for printing. (a) The geometry of the gear to be printed. (b) A malicious version of the gear. A void is located inside the gear.

Table 3.2: Printing Processes for Each Printer

B/M	**Process**	**Re.**	**Description**	**Ref.**
B	Benign	1	This is used for the reference.	
B	Benign	50	These benign processes are used for training.	
B	Benign	100	These benign processes are used for testing.	
M	Void	20	A void is inserted.	[9]
M	InfillGrid	20	Infill pattern is changed to grid.	[22]
M	Speed0.95	20	Printing speed is decreased by 5%.	[24]
M	Layer0.3	20	Layer height is changed to 0.3 mm.	[24]
M	Scale0.95	20	The object is shrunk by 5%.	[9]

B/M = Benign or Malicious. Re. = Repetition (for each printer). All the malicious processes are used for testing.

RM3, we used MatterControl 1.7.5 with MatterSlice as the slicer. For both printers, we used the default setting with a layer height of 0.2 mm.

For each printer, the benign process was repeated 151 times. One benign process served as the reference. 50 benign processes were used for learning the critical values (training). The other 100 benign processes were used for testing. We manipulated the benign G-code file in five different ways to simulate five types of malicious printing processes in the literature. All malicious processes were used for testing. The details of the printing processes for each printer can be found in Table 3.2.

Side Channels. We used six different types of side channels in the experiments and the details are in Table 3.3. The locations of the sensors are shown in Figure 3.19. For each printer, we installed the MPU9250 sensor on the printhead and SCT013 measured the total AC current delivered to the printer. Figure 3.21 and Figure 3.22 respectively show samples

Table 3.3: Types of Side Channels

ID	Side Channel	Sensor	*fs* (Hz)	Channels	Bits
ACC	Acceleration	MPU9250	4000	6	16
TMP	Temperature	MPU9250	4000	1	16
MAG	Magnetic	MPU9250	100	3	16
AUD	Audio	AKG170	48,000	2	24
EPT	Elec. Potential	AKG170*	96,000	1	24
PWR	Power/Current	SCT013	12,000	1	24

EPT = Electric Potentials. Channels = Number of Channels.
* The AKG170s for collecting electric potentials were modified by removing their caps, inspired by the method in [56].
We used a Teensy board to relay data from MPU9250 to a host PC.
We used a UMC404HD interface to collect data from AKG170 and SCT013 to a host PC.

of the side-channel signals for the UM3 printer and the RM3 printer.

Spectrograms. Many existing IDSs internally transform a side-channel signal into a spectrogram before further processing [22, 27, 26]. For other IDSs, including NSYNC, in addition to comparing raw signals directly, we also compared their spectrograms. For each side-channel signal, we obtained its spectrogram via Short-Time Fourier Transforms (STFT) [31] and the details are shown in Table 3.4.

Theoretically speaking, the spectrogram of a signal can be considered as a new signal with a reduced sampling rate and an increased number of channels. In fact, each frequency band can be treated as a channel. When the original signal has multiple channels, the frequency bands for all channels can be concatenated. In this thesis, when we refer to a side-channel signal, the signal may be a spectrogram.

We consider spectrograms due to their potentially better performance. For example, Han et al. used electromagnetic radiation to classify programs running in a Programmable Logical Controller (PLC) and their results show that the classification accuracy is significantly higher if the classification is performed in the spectrogram domain instead of the time domain [56]. Figure 3.21 and Figure 3.22 respectively show samples of the spectrograms for the UM3 printer and the RM3 printer.

Transformation. The spectrogram of a signal can be regarded as a transformation

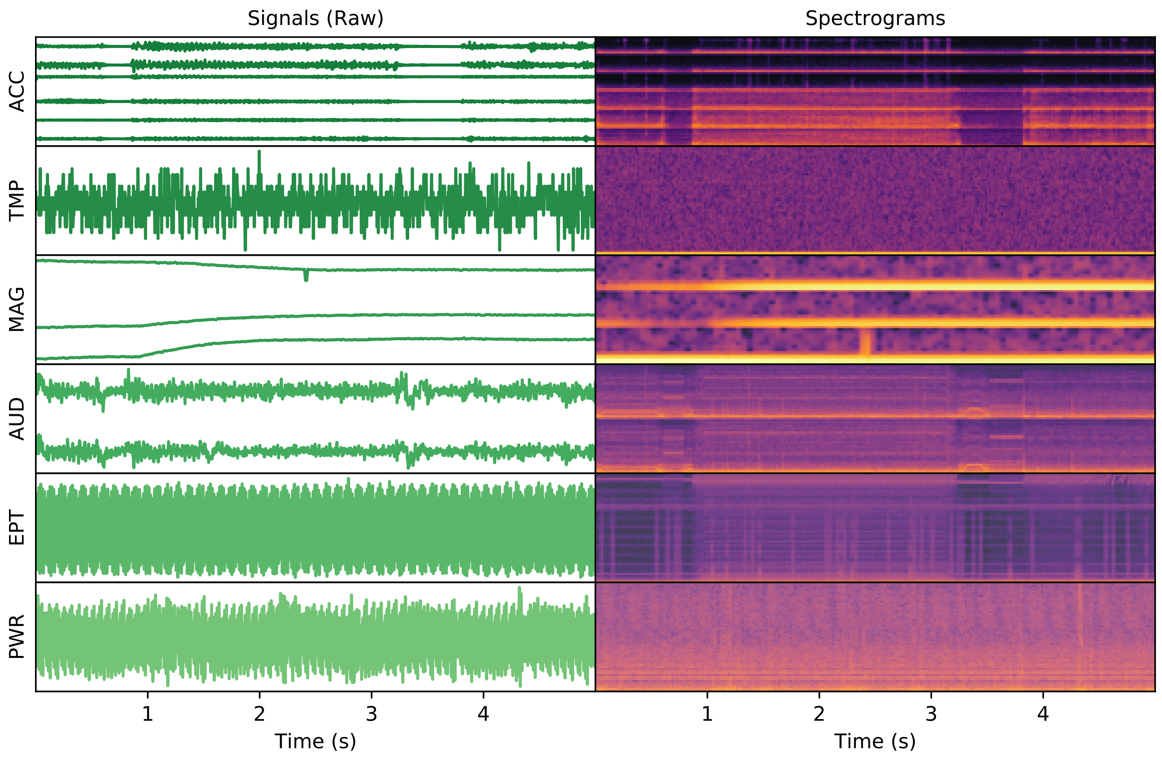


Figure 3.21: Samples of side-channel signals for the UM3 printer.

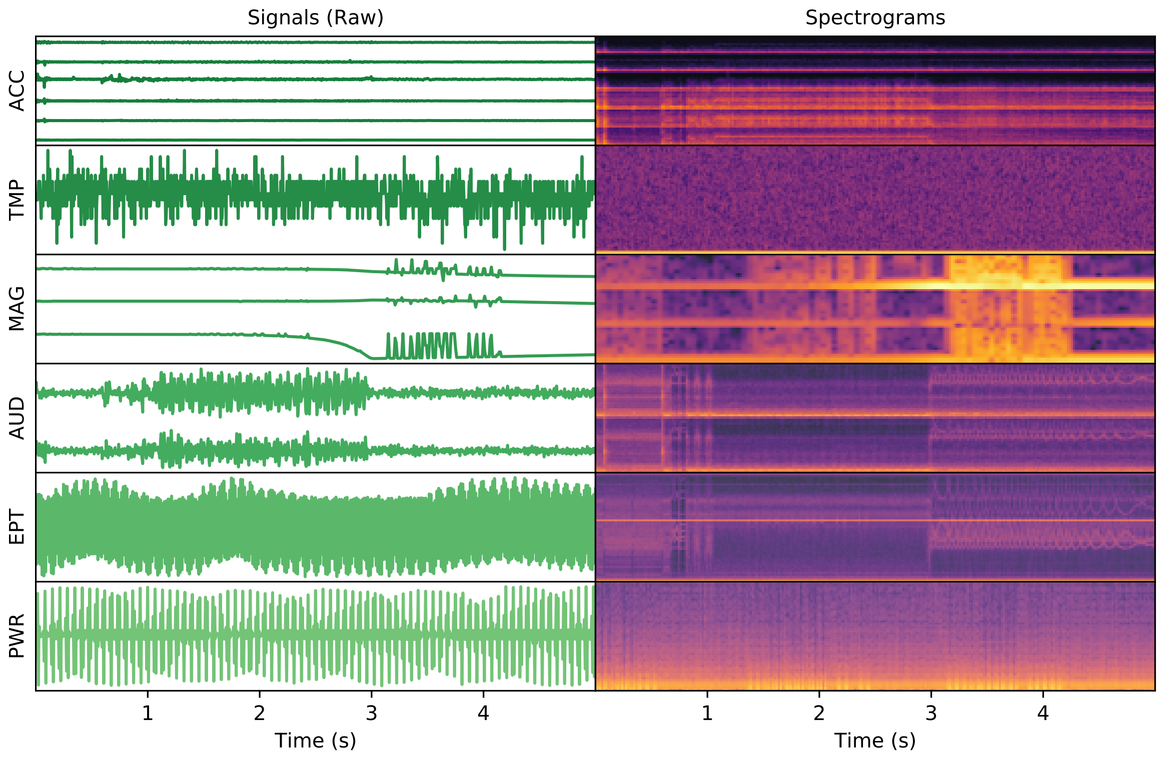


Figure 3.22: Samples of side-channel signals for the RM3 printer.

Table 3.4: Spectrograms for Side Channels

ID	Δf (Hz)	Δt (s)	**Window**	**Channels**	**Bits**
ACC	20	1/80	BH	101×6	16
TMP	20	1/80	BH	101	16
MAG	5	1/20	BH	11×3	16
AUD	120	1/240	BH	201×2	16
EPT	120	1/240	BH	401	16
PWR	60	1/120	Boxcar	101	16

Δf is the spectral resolution, which is equal to the reciprocal of the window size (in seconds) in STFT. Δt is the temporal resolution, which is equal to the time by which the window moves forward each time in STFT. BH = Blackman-Harris.

of the signal, and we add the suffix SPG to identify the transformation. For example, ACC-SPG refers to the spectrogram of the acoustic side-channel signal. Apart from spectrograms, there are other types of transformations that can be performed on a signal, such as Mel-Frequency Cepstral Coefficients (MFCC) as used by [21] and the Principle Component Analysis (PCA) of the spectrogram of a signal as used by [27]. To emphasize a signal without any transformation, we refer to the signal as the raw signal and use the suffix Raw to identify a lack of transformations. For example, an acceleration side-channel signal without any transformation can be referred to as ACC-Raw.

When a transformation suffix is missing, an identifier typically refers to the type (or modality) of the side channel. For example, ACC refers to the acceleration side channel (emphasizing the channel). If the identifier is used in a place where an input to the NSYNC framework is expected, the Raw transformation is assumed by default. For example, ACC refers to the raw acceleration side-channel signal.

3.11.2 Parameters for Static Synchronization

The primary purpose of this section is to figure out the influence of the parameters of static synchronization on its performance. The parameters only include the score function and the calculation axis (collectively referred to as the kernel). The parameters are the same for static synchronization in the TDE mode and static synchronization in the SD mode.

We mainly evaluated the performance in the TDE mode because the parameters should influence static synchronization in the TDE mode and static synchronization in the SD mode in a similar way and it is easy to perform static synchronization in the TDE mode.

Performance Metrics. To quantitatively measure the performance of static synchronization, it is important to properly define performance metrics. In this thesis, we use two performance metrics. The first performance metric is the error between the detected moment and the actual moment[10]. This metric measures the accuracy of static synchronization is. The second performance metric is the elapsed time for the static synchronization to complete. This metric measures the speed of static synchronization.

Evaluation Procedures. To comprehensively understand the performance of static synchronization, we performed static synchronization for each printer, each type of side-channel signals[11], and each template[12]. For each printer, each type of side-channel signals, and each template, there were 150 printing processes (or 150 signals) to be analyzed. We performed static synchronization for the 150 printing processes to obtain the average error and the average elapsed time.

Templates Used for Evaluation. We need to specify the templates to be used when invoking static synchronization. For the UM3 printer, the templates are shown in Figure 3.15. To be specific, the starting template starts at the starting moment and ends 3.75 seconds after the starting moment. The stopping template starts 1.5 seconds before the stopping moment and ends 0.4 seconds after the stopping moment. For the RM3 printer, the starting template starts at the starting moment and ends 1.0 second after the starting moment. The stopping template starts 1.0 second before the stopping moment and stops

[10]We manually obtained the actual starting moment and the stopping moment for all printing processes.

[11]A side-channel signal is a specific signal whereas a type of side-channel signals refers to a collection of side-channel signals that share the same side channel and the same transformation. We have six different types of side channels in the experiments and they are ACC, TMP, MAG, AUD, EPT, and PWR. However, we have 12 types of side-channel signals and they are ACC-Raw, TMP-Raw, MAG-Raw, AUD-Raw, EPT-Raw, PWR-Raw, ACC-SPG, TMP-SPG, MAG-SPG, AUD-SPG, EPT-SPG, and PWR-SPG. A side channel is a means to transmit information whereas a side-channel signal refers to the information in the channel. Technically speaking, in this thesis, a side-channel signal is a combination of a side channel and a transformation.

[12]By each template, we mean each template for the current type of side-channel signals. There are two templates and they are the starting template and the stopping template.

exactly at the stopping moment.

Evaluation Results. The combination of a score function and a calculation axis is collectively referred to as a kernel. We performed static synchronization in the TDE mode with eight different kernels (four different score functions and two calculation axes) in order to compare their performance. The four score functions are Inner Product (IP), Covariance (Cov), Cosine (Cos), and Correlation Coefficient (CC), as shown in Table 3.1. The two calculation axes are none and time. The four score functions are combined with the two calculation axes to form the eight kernels.

Table 3.5 shows the average error as a function of kernels for the UM3 and RM3 printers. We only show the results for ACC-Raw, ACC-SPG, AUD-Raw, and AUD-SPG because they have the best performance and the space is limited. We show the results in tables because there are a lot of numbers to be shown and the numbers span several orders of magnitudes. We can see that the CC-time kernel had the best performance for both printers, both types of templates, and all types of side-channel signals.

However, we also see that the CC-none kernel performed poorly for ACC-Raw. This is because CC-none considers the relationship across channels. For example, the mean (average) of the x or y channel in the ACC-Raw signal over time is approximately zero, whereas the mean of the z channel over time is a non-zero constant due to the gravity. This relationship across channels can result in a large score regardless of the location of the template in the side-channel signal. As a result, the ability for CC-none to selectively find the best location of the template is degraded.

Figure 3.23 shows the elapsed time vs kernels for AUD-Raw and the starting template. The results for other types of side-channel signals or the stopping template are similar and omitted to save space. We can see that, when the calculation axis was none, the elapsed time increased as the mathematical expression becomes more complex. To be specific, the IP score function had the least amount of time consumption whereas the CC score function had the most amount of time consumption. However, when the calculation axis was time,

Table 3.5: Average Error vs Kernel for Static Synchronization

Printer	Template	Kernel	Type of Side-Channel Signals			
			ACC-Raw	ACC-SPG	AUD-Raw	AUD-SPG
UM3	Starting	IP-none	0.018015	72.107050	1.331822	20.974442
		Cov-none	0.018015	29.438300	1.331822	21.838501
		Cos-none	4.642789	0.013450	1.563312	0.004151
		CC-none	4.631124	0.013400	1.563312	0.004167
		IP-time	0.018015	72.107050	1.331822	20.974442
		Cov-time	0.018015	0.012800	1.331822	0.004151
		Cos-time	1.315330	45.559150	1.108091	0.004167
		CC-time	0.015700	0.015600	1.108091	0.004151
	Stopping	IP-none	2.421939	4.960650	0.001083	2.256279
		Cov-none	2.646234	6.849700	0.001083	1.163590
		Cos-none	26.890172	0.011850	0.098144	0.000345
		CC-none	24.708509	0.011850	0.098144	0.000328
		IP-time	2.421939	4.960650	0.001083	2.256279
		Cov-time	2.085349	1.137700	0.001083	0.000328
		Cos-time	2.761774	0.853800	0.001083	0.000410
		CC-time	0.349514	0.012300	0.001083	0.000375
RM3	Starting	IP-none	0.011196	1.479500	0.007065	0.000100
		Cov-none	0.011196	0.000900	0.007065	0.190432
		Cos-none	0.154043	0.000250	0.020888	0.000017
		CC-none	0.011198	0.000250	0.020888	0.000017
		IP-time	0.011196	1.479500	0.007065	0.000100
		Cov-time	0.011198	0.000450	0.007065	0.000017
		Cos-time	0.011191	0.014650	0.022904	0.000100
		CC-time	0.011187	0.000000	0.022904	0.000000
	Stopping	IP-none	0.012368	24.563200	2.201267	11.654659
		Cov-none	0.012368	0.556500	2.201267	0.000000
		Cos-none	0.012368	0.000200	0.167697	0.000017
		CC-none	0.012368	0.000200	0.167697	0.000017
		IP-time	0.012368	24.563200	2.201267	11.654659
		Cov-time	0.012368	0.000200	2.201267	0.000000
		Cos-time	0.012378	15.156100	0.169946	0.000050
		CC-time	0.012369	0.000000	0.169946	0.000000

The units of all numbers are in seconds.

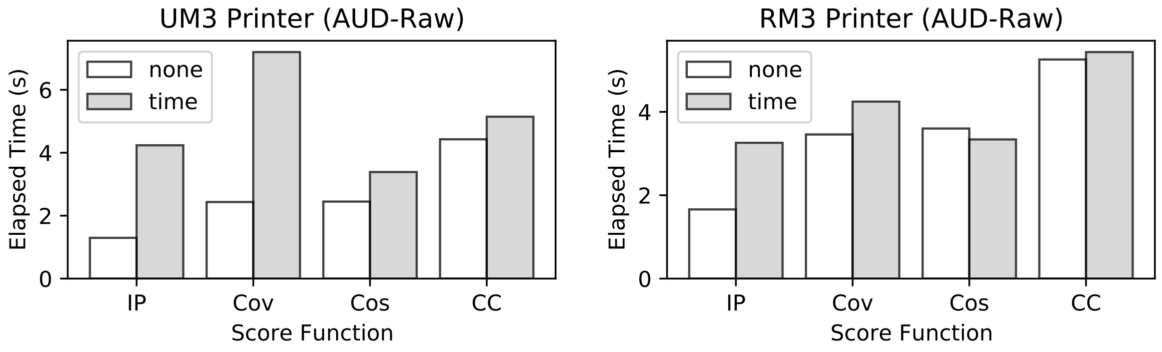


Figure 3.23: Elapsed time vs kernels for the UM3 and RM3 printers. The results are for AUD-Raw only. Results for other types of side-channel signals are similar and omitted.

there was no obvious pattern. We can see that the Cov score function consumed more time than the CC score function did. It is hard to explain the exact reason behind this behavior. It might be a result of the optimization strategies in the computer architecture.

Generally speaking, time consumption is less of a concern because, to analyze a signal of 200 seconds, the maximum elapsed time is less than 10 seconds. Static synchronization can be easily performed in real time.

Because the CC-time kernel had the overall best performance, we will use this kernel for the rest of the paper by default. We will use the CC-time kernel not only for TDE but also for SD. We will use the CC-time kernel not only for static synchronization but also for DWM, an implementation of dynamic synchronization.

3.11.3 Analysis of Properties of Templates

To identify a moment in a signal by static synchronization, we need a template for that moment. There are countless possibilities for a template. For example, we need to determine the duration of the template and we need to determine the location of the moment in the template (also known as the anchor point). This process is referred to as template design. In this subsection, we study how properties of a template, namely the duration and the anchor point, can affect the performance of static synchronization. We use the same performance metrics and the evaluation procedures in subsection 3.11.2.

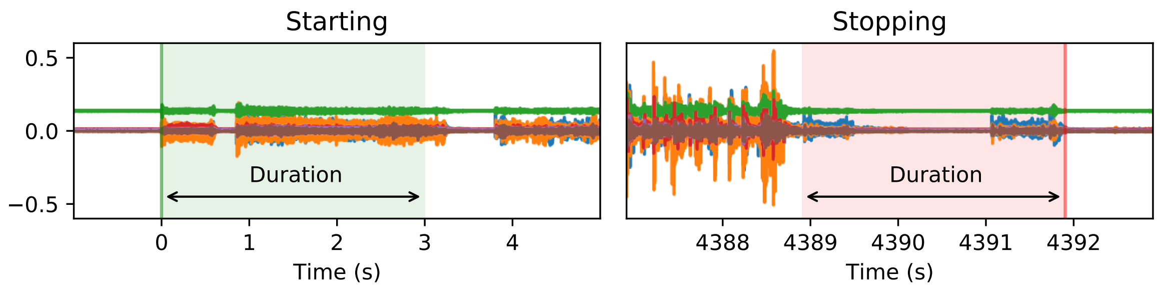


Figure 3.24: Definition of duration for the starting and stopping moments.

Parametric Analysis on Template Duration. We studied the influence of the duration of a template on the performance of static synchronization. Figure 3.24 shows the definition of the duration of a template. For the template of the starting moment (also referred to as the starting template), the anchor point is located on the left edge of the template. For the stopping template, the anchor point is relocated on the right edge of the template. To evaluate the influence of template duration, we need to perform static synchronization for a lot of templates with different duration. In order to perform the experiments within a reasonable amount of time, we only performed experiments with the acceleration side-channel signals (ACC-Raw) and the acoustic side-channel signals (AUD-Raw).

Figure 3.25 shows the average error for the UM3 printer and the RM3 printer and Figure 3.26 shows the elapsed time for both printers. We see that for both printers, both types of templates, and both types of side-channel signals, the average error exhibited the same pattern. That is, the average error was lower in the middle and higher on both ends. Simply put, the best performance was achieved with an intermediate duration[13].

For the UM3 printer, an accept range of template duration is from 1 second to 10 seconds for all involved side-channel signals and all involved templates. By manually inspecting the waves, we decided to use a duration of 3.75 seconds for the starting template because this template is the longest template that is relatively consistent among all printing processes considered in the experiments. By the same token, we decided to use a duration of 1.5 seconds for the stopping template. For the RM3 printer, an acceptable range of tem-

[13]In this thesis, when the word “duration” is used as a countable noun, it means the value of the duration.

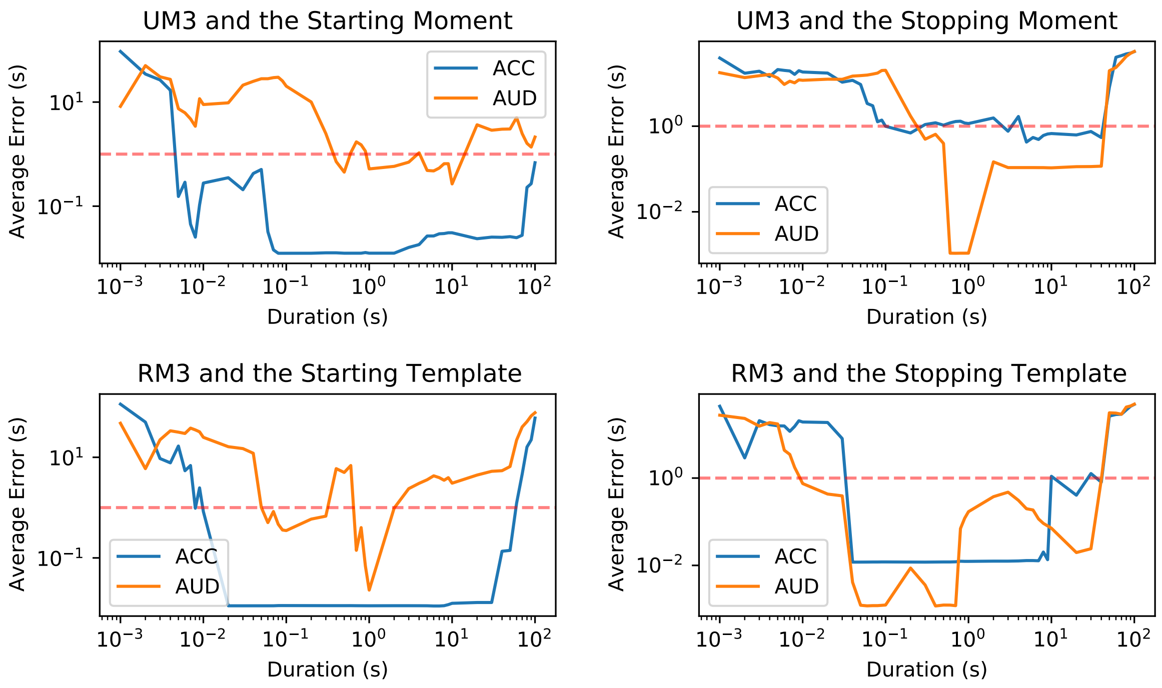


Figure 3.25: Average error as a function of template duration. ACC is the acceleration side-channel signal whereas AUD is the acoustic side-channel signal.

plate duration is from 0.1 second to 10 seconds. We decided to use a duration of 1 second for both of the starting template and the stopping template.

By inspecting Figure 3.26, we see that the elapsed time overall increased as the template duration increased although there was a lot of fluctuation. We see that the patterns were the same for all printers, all types of side-channel signals, and all types of templates. The pattern might be related to the computer architecture of the system that performed the calculation. We also see that, to find a signal of 100 seconds in a signal of 200 seconds, the maximum elapsed time was around 10 seconds. In other words, for a template that lasts 100 seconds, we could still perform static synchronization in real time.

Necessity of Arbitrary Anchor Points. When designing static synchronization for the NSYNC framework, we allow an arbitrary location of the anchor point inside a template (or even outside of the template). This makes it possible for the stopping template in Figure 3.15 to include an empty part of 0.4 second after the stopping moment. Notice that the stopping template in Figure 3.24 does not have this empty part. One may ask if it is

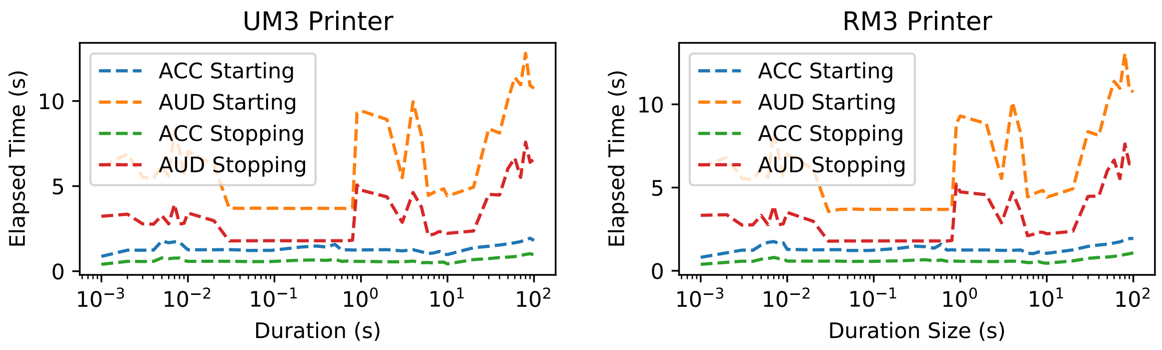


Figure 3.26: Average elapsed time as a function of template duration.

Table 3.6: Average Error with and without Empty Parts

	ACC-Raw	ACC-SPG	AUD-Raw	AUD-SPG
With the Empty Part	0.349514	0.012300	0.001083	0.000375
Without the Empty Part	1.398049	0.188300	0.001083	0.000328

The results are for the UM3 printer and the stopping moment only.

really necessary to include this empty part in a template.

To answer this question, we performed experiments with the 0.4 second empty part present and not present while holding everything else constant. We only performed experiments with the UM3 printer and the stopping template. The results are shown in Table 3.6. We can see that the empty part in the template significantly reduced the average error for ACC-Raw and ACC-SPG although it slightly increased the average error for AUD-SPG. Why does adding an empty part help with the performance for certain scenarios? This is because we are using the CC-time kernel, which also takes the empty part into consideration. When searching for the template in a signal, a match should also contain the empty part. In this way, including the empty part can make the template more selective.

3.11.4 Static Synchronization in the TDE Mode

In the previous two subsections, we determined that the overall best kernel was CC-time and we designed the starting and stopping templates for both the UM3 printer and the RM3 printer. Using ACC-Raw as an example, the templates for the UM3 printer and the RM3 printer are shown in Figure 3.27 and Figure 3.28 respectively. The properties (duration and

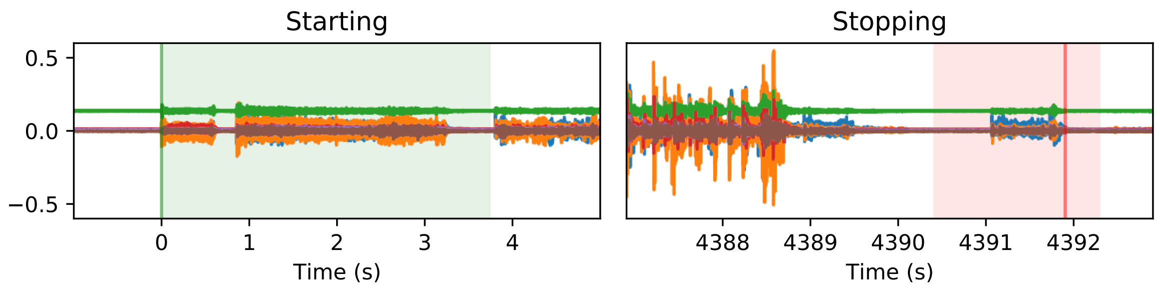


Figure 3.27: Starting and stopping templates (ACC-Raw) for the UM3 printer.

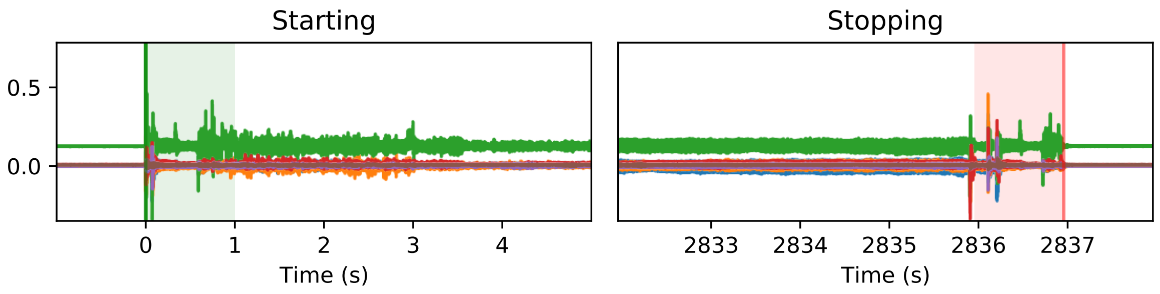


Figure 3.28: Starting and stopping templates (ACC-Raw) for the RM3 printer.

anchor point) of the templates for other types of side-channel signals are the same.

With the parameters and the templates specified, we performed static synchronization in the TDE mode for all types of side-channel signals. The results for both printers (UM3 and RM3) and both templates (starting and stopping) are shown in Figure 3.29, where the dashed red line in each sub-figure corresponds to the duration of the template.

In this thesis, we consider static synchronization unsuccessful if the error is more than the duration of the template. According to this standard, we see that TMP-Raw, TMP-SPG, EPT-Raw, PWR-Raw, and PWR-SPG were useless as static synchronization with these types of side-channel signals unanimously failed. In fact, a visual inspection[14] of these types of side-channel signals reveal that they appeared to be random and did not contain much information about the printing process. In other words, the Signal to Noise Ratio (SNR) of these types of side-channel signals was almost zero. MAG-Raw, MAG-SPG, and EPT-SPG were semi-useful types of side-channel signals as static synchronization with these types of side-channel signals sometimes failed and sometimes succeeded. A visual

[14]Refer to Figure 3.21 and Figure 3.22 for samples of the side-channel signals.

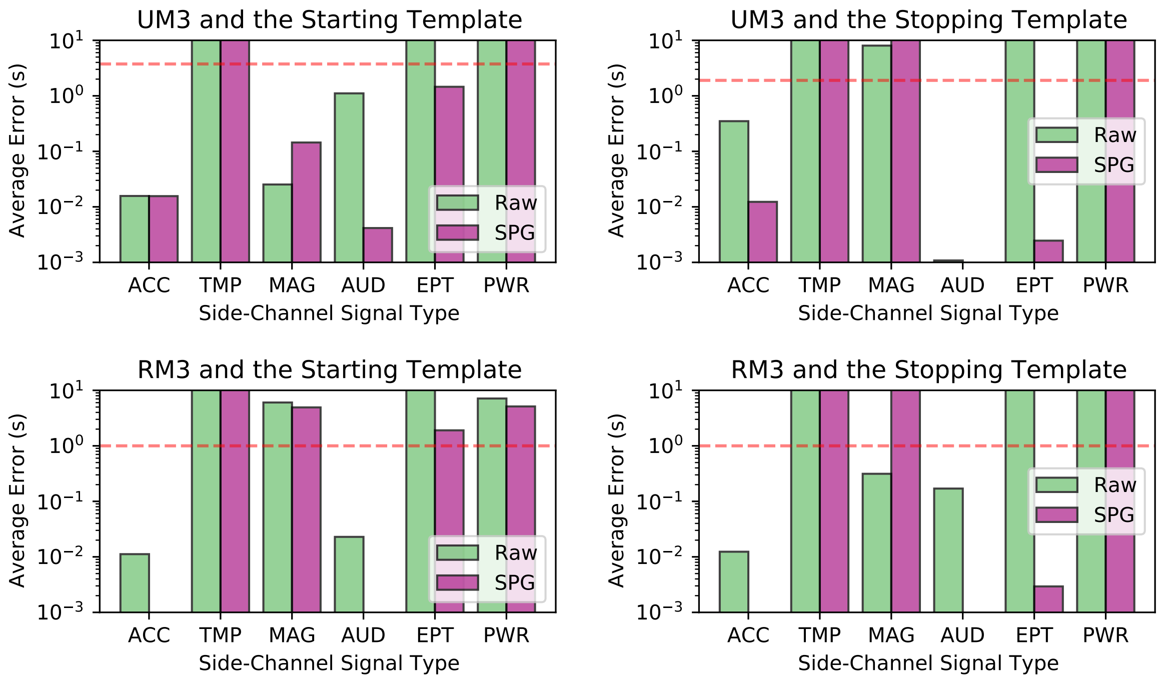


Figure 3.29: Average error of static synchronization in the TDE mode.

inspection of these types of side-channel signals reveal that they were weakly correlated with the mechanical state-variable signal[15] of the printer. In other words, they contained some useful information but the SNR was relatively low. ACC-Raw, ACC-SPG, AUD-Raw, and AUD-SPG were high quality types of side-channel signals. They were strongly correlated with the mechanical state-variable signal of the printer and the SNR was high.

A visual inspection reveals that the MAG-Raw signals from the RM3 printer were pulse width modulated as the signals mainly came from the heating coil in the build plate. The temperature of the build plate was controlled by pulse width modulating the current in the build plate, which created pulse width modulated magnetic fields around the build plate. As a result of this, it is possible to infer the thermal state variables of the printer. However, Pulse Width Modulation (PWM) makes the MAG-Raw signals non-reproducible. This violates the assumption of the NSYNC framework. That is, side-channel signals for the same printer and the same G-code instructions should be almost identical. As a result, MAG-Raw and MAG-SPG from the RM3 printer are not suitable for the NSYNC framework unless

[15]The mechanical state-variable signal contains the position, velocity, and acceleration of the printhead.

measures are taken to filter out the modulated pulses in the side-channel signals.

We see that EPT-Raw was useless whereas EPT-SPG was semi-useful. A close inspection reveals that the EPT-Raw side-channel signals contained a predominant 60 Hz component, which came from the power system. The 60 Hz component makes the side-channel signals look like sinusoidal waves with ripples. As a result, EPT-Raw is not very useful for the NSYNC framework. However, EPT-SPG is still useful because the 60 Hz component is only one of the many frequency components and all the frequency components share the same weight when calculating the scores.

Overall, for ACC-Raw, ACC-SPG, AUD-Raw, and AUD-SPG, we could use static synchronization in the TDE mode to accurately identify the starting and stopping moments of a printing process. The spectrograms performed better than the raw signals. With ACC-SPG or AUD-SPG, the maximum error was around 0.01 second for a template with a duration up to 3.75 seconds. This accuracy is satisfying for practical applications.

3.11.5 Static Synchronization in the SD Mode

As with static synchronization in the TDE mode, we use the CC-time kernel and the templates in Figure 3.27 and Figure 3.28 for static synchronization in the SD mode.

Critical Scores. For static synchronization in the SD mode, we need to obtain the critical scores. For each printer, we used the 50 benign printing processes to obtain the critical scores. A critical score was obtained for each printer (UM3 and RM3), each type of side-channel signals, and each template (starting and stopping). We used the method in subsection 3.6.3 to obtain the critical scores. Specifically, we used a support vector machine with a linear kernel and a C value of 20 in the process of determining the critical scores. The critical scores for both printers and both templates are shown in Figure 3.30.

With the critical scores determined, we performed static synchronization in the SD mode on the 100 benign printing processes for testing purposes and the 100 malicious printing processes. The results for both printers and both templates are shown in Figure 3.31.

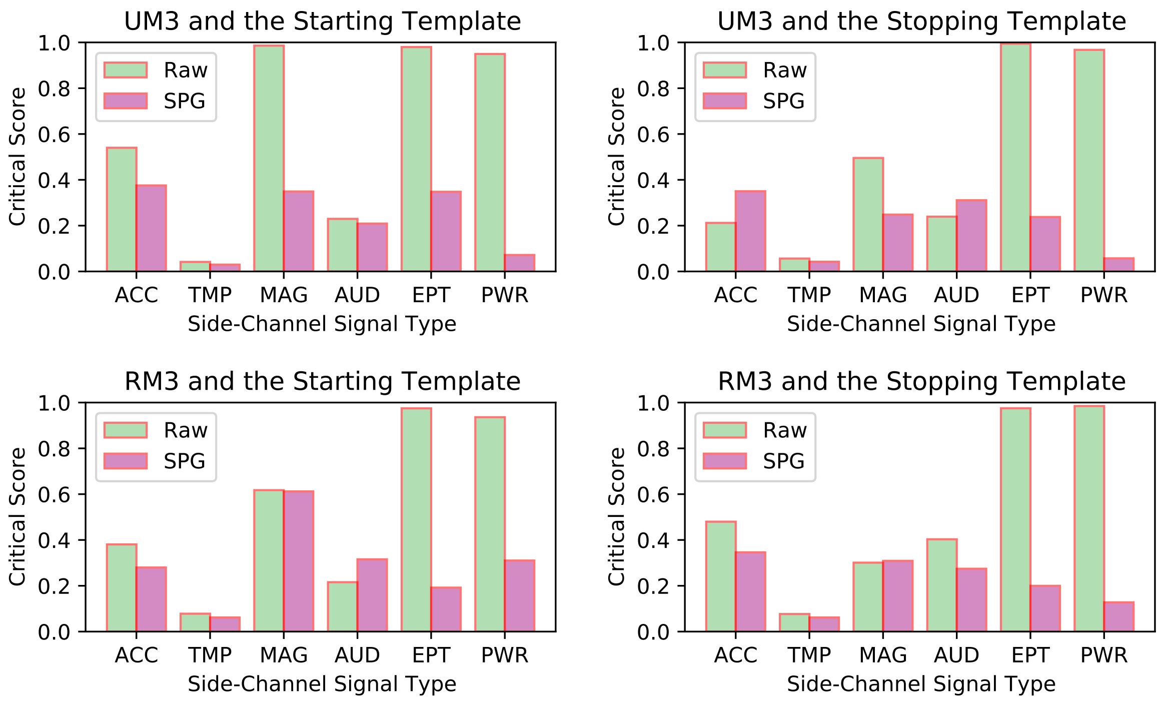


Figure 3.30: Critical scores for static synchronization in the SD mode.

We can see that only the RM3 printer with AUD-SPG could successfully identify both the starting moments and the stopping moments. For the UM3 printer, only ACC-SPG could properly identify the starting moments with a relatively large error (around 1.0 second). The performance of static synchronization in the SD mode was much worse than that in the TDE mode. Static synchronization in the SD mode is inherently more challenging.

One of the potential reasons that static synchronization in the SD mode failed is because the critical scores were only applicable to the benign printing processes. In fact, the starting and stopping templates were derived from the reference printing process. They could match benign printing processes very well, but they did not match malicious printing processes at the same level. Our hypothesis is that static synchronization in the TDE mode tolerates a higher level of mismatch than static synchronization in the SD mode does. To test this hypothesis, we performed static synchronization in the SD mode only on the 100 benign printing processes and the results are shown in Figure 3.32. We see that the performance was greatly improved as ACC-Raw, ACC-SPG, and AUD-SPG could successfully identify

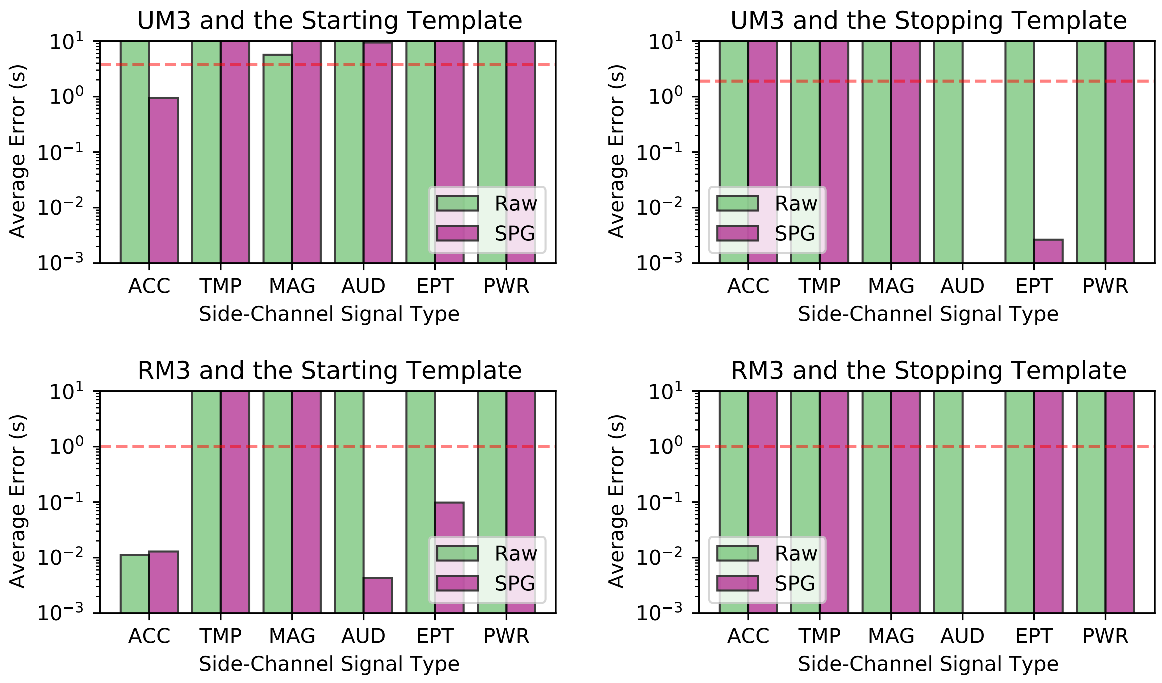


Figure 3.31: Average errors of static synchronization in the SD mode.

the starting and stopping moments for both printers.

In summary, static synchronization in the SD mode is more challenging. We could successfully perform static synchronization in the SD mode on benign printing processes. The best types of side-channel signals were ACC-Raw, ACC-SPG, and AUD-SPG. For the acoustic side channel, spectrograms were required as AUD-Raw failed altogether in the SD mode. With ACC, ACC-SPG or AUD-SPG, the maximum error was around 0.01 seconds for a template with a duration up to 3.75 seconds. We could not successfully perform static synchronization in the SD mode for malicious printing processes. Fortunately, unsuccessful static synchronization will increase the chance for the NSYNC framework to issue an alert and thus properly detect an intrusion event.

3.11.6 Duration-Based Intrusion Detection

By static synchronization, we can determine the duration of a printing process. According to subsection 3.7.7, we can use the duration of a printing process as an indicator for intrusion detection. If the duration of a printing process is very different from what is expected,

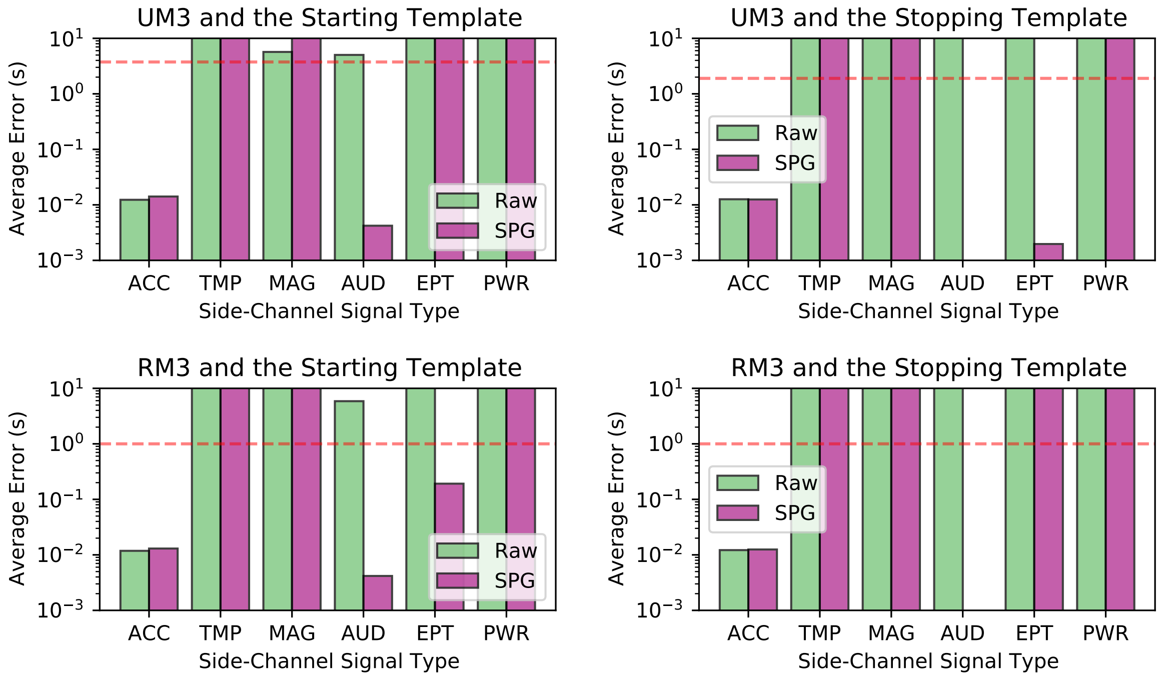


Figure 3.32: Average errors of static synchronization in the SD mode. The results were obtained by testing on the 100 benign printing processes only.

an intrusion can be declared. We established an intrusion detection system according to the procedures outlined in subsection 3.7.7 with a margin ratio of $r = 0.25$. We used static synchronization in the SD mode to determine the duration of a printing process. We used the 50 benign printing processes to obtain the thresholds and tested on the other 100 benign printing processes as well as the 100 malicious processes. The intrusion detection results are shown in Table 3.7. The results are reported in the format of

Accuracy (False Positive Rate / True Positive Rate)

where the accuracy is calculated by dividing the number of correctly predicted printing processes (benign or malicious) by the number of all printing processes, FPR is defined to be the number of False Positives (FPs) over the number of benign printing processes, and True Positive Rate (TPR) is defined to be the number of True Positives (TPs) over the number of malicious printing processes. A FP is a benign printing process that is declared to be malicious by the IDS, whereas a TP is a malicious printing process that is declared to be malicious by the IDS. We see that ACC-Raw, ACC-SPG, and AUD-SPG could reliably

Table 3.7: Duration-Based IDS Results

T	**SC**	**Printer**	
		UM3	RM3
Raw	ACC	1.00 (0.00 / 1.00)	1.00 (0.00 / 1.00)
	MAG	0.83 (0.34 / 1.00)	0.59 (0.64 / 0.82)
	AUD	0.97 (0.07 / 1.00)	0.84 (0.33 / 1.00)
	EPT	0.69 (0.17 / 0.55)	0.64 (0.56 / 0.84)
SPG	ACC	1.00 (0.00 / 1.00)	1.00 (0.00 / 1.00)
	MAG	0.53 (0.95 / 1.00)	0.51 (0.99 / 1.00)
	AUD	1.00 (0.00 / 1.00)	1.00 (0.00 / 1.00)
	EPT	0.87 (0.27 / 1.00)	0.98 (0.04 / 1.00)

T = Transformation. SC = Side Channel (Signal Type).

distinguish malicious printing processes from benign printing processes for both printers.

One of the reasons why the intrusion detection system worked well is because the durations of all malicious printing processes were very different from those of the benign printing process. Table 3.8 shows the average duration and the standard deviation of the duration for all types of printing processes. When the duration of a malicious printing process differs from the duration of the benign printing process by at least the standard deviation of the duration of the benign printing process, there is a good chance for the intrusion detection system to issue an alert.

Nevertheless, it is possible for an advanced attacker to craft a malicious printing process such that its duration is close to the duration of the benign printing process. The advanced attacker can remove or add instructions in a printing process to adjust its duration. The duration-based intrusion detection system will not be able to defeat such an advanced attacker, and we need to resort to the rest part of the NSYNC framework.

3.12 Experiments on Dynamic Synchronization

In this section, we present experiment results related to dynamic synchronization. We used the same experiment setup as in section 3.11. We first of all show the consistency of horizontal displacements. Afterwards, we discuss how to obtain the parameters in dynamic

Table 3.8: Duration of Printing Processes

Process	Printer	
	UM3	RM3
Benign	4393.500 ± 0.897	2836.981 ± 0.034
Void	4404.973 ± 0.806	2844.488 ± 0.034
InfillGrid	4410.381 ± 0.884	2823.446 ± 0.026
Layer0.3	3035.414 ± 0.551	1985.969 ± 0.015
Speed0.95	4455.842 ± 0.991	2981.224 ± 0.049
Scale0.95	3874.998 ± 0.812	2492.245 ± 0.033

The units of numbers are in seconds.

synchronization, including DTW and DWM, with experiments. Finally, we present intrusion detection results using NSYNC with DTW and NSYNC with DWM.

3.12.1 Consistency of Horizontal Displacement Arrays

The consistency of horizontal displacement arrays refers to the following phenomenon. When performing dynamic synchronization between a reference printing process and a specific printing process, if the dynamic synchronization process is successful, the horizontal displacement array should be invariant with respect to the method and the type of side-channel signals that are used to determine the horizontal displacement array. In other words, a horizontal displacement array is the property of two printing processes.

Default Parameters for DTW. In order to perform DTW, we have to specify all parameters. There are two parameters in DTW[16]. One is the radius in the DTW algorithm and the other is the type of distance metrics. We always use a radius of 1 (the minimum allowed radius) because the DTW algorithm is very slow when analyzing side-channel signals in AM systems and the larger the radius, the slower the algorithm. For the distance metric, by default we use the correlation distance as defined in Equation 3.25. We will later explore the performance of different distance metrics in subsection 3.12.2.

Default Parameters for DWM. As with DTW, in order to perform DWM, we have to specify all parameters. By default we use the parameters listed in Table 3.9. These

[16]More precisely, FastDTW. This is because only FastDTW has the radius parameter.

Table 3.9: Parameters in DWM

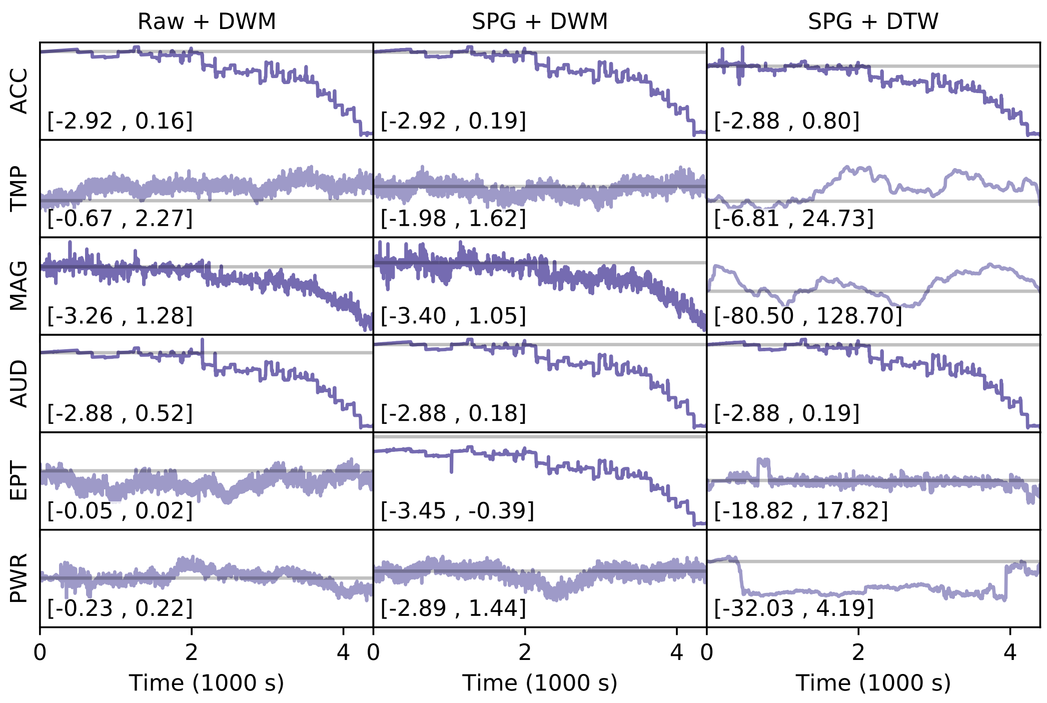
Printer	**Kernel**	t_{win}	t_{hop}	t_{ext}	t_{sigma}	η
UM3	CC-time	4.0 s	2.0 s	2.0 s	1.0 s	0.1
RM3	CC-time	1.0 s	0.5 s	0.1 s	0.05 s	0.1

parameters work well for a variety of side-channel signals and printing processes. We will explain how we obtained the parameters in subsection 3.12.3.

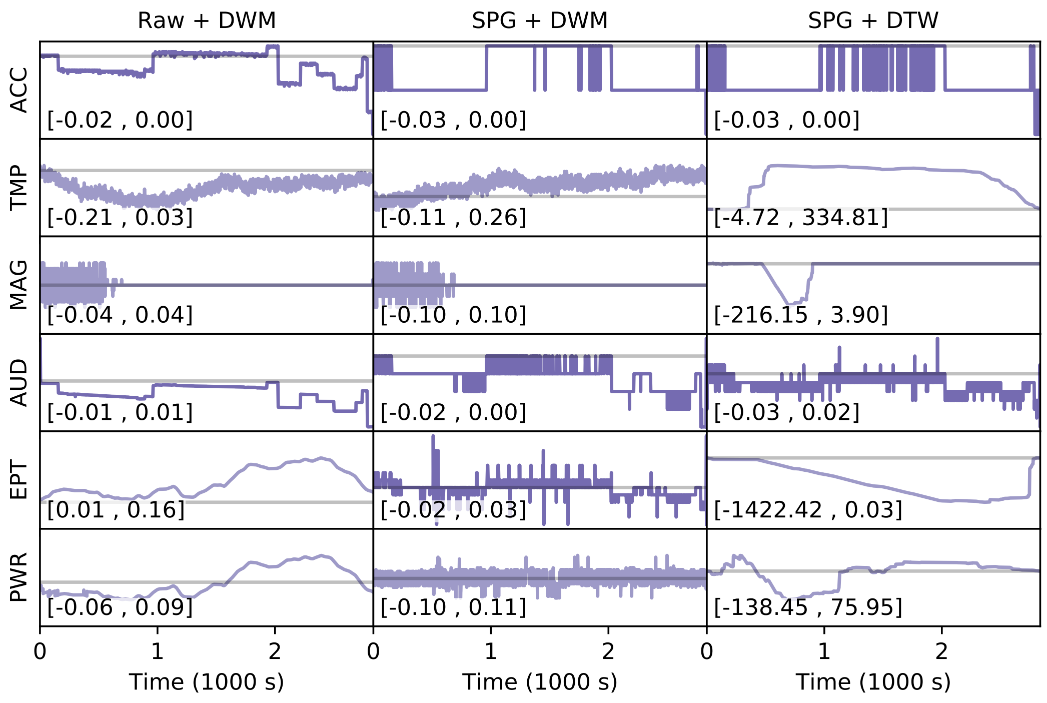
Experiment Procedures. For each printer (UM3 or RM3), each type of side-channel signals (6×2 in total), we performed DWM and DTW to obtain the horizontal displacements between the side-channel signal from the reference printing process and the side-channel signals from all other printing processes. We were only able to perform DTW for signals with the SPG transformation because it took forever to analyze the raw signals. Whereas many of the horizontal displacement arrays looked like random curves, some horizontal displacement arrays exhibited consistent patterns.

Figure 3.33 (a) shows horizontal displacement arrays between all available side-channel signals of the reference printing process and corresponding side-channel signals of a benign printing process for the UM3 printer. We can see that horizontal displacement arrays obtained by ACC-Raw, ACC-SPG, AUD-Raw, and AUD-SPG are the same, regardless of the method (DWM or DTW). The horizontal displacement array obtained by EPT-SPG with DWM also conforms to this same pattern. In addition, a close inspection of horizontal displacement arrays obtained by MAG-Raw and MAG-SPG with DWM indicate that they conform to the same pattern but with an increased level of noise.

Figure 3.33 (b) shows horizontal displacement arrays between all available side-channel signals of the reference printing process and corresponding side-channel signals of a benign printing process for the RM3 printer. We have similar observations as we had for the UM3 printer except two major differences. First of all, horizontal displacement arrays obtained by MAG-Raw and MAG-SPG appear to be random. This may be due to the PWM waves in the magnetic side-channel signals. Second, horizontal displacement arrays obtained with



(a) UM3



(b) RM3

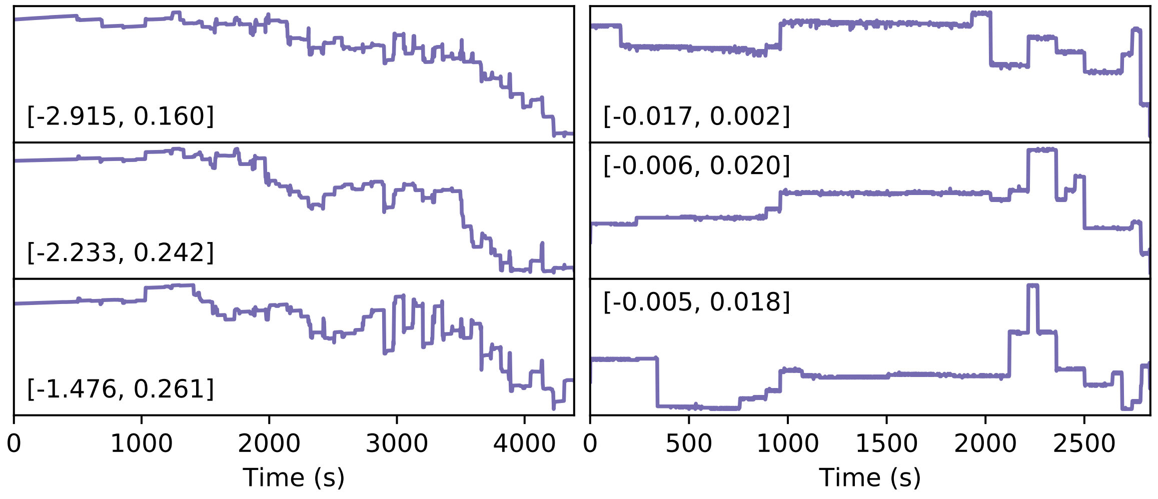
Figure 3.33: Horizontal displacement arrays ($\boldsymbol{h}_{\text{disp}}$) obtained between the reference printing process and a benign printing process with the specified printer by DWM and DTW for all available types of side-channel signals. The numbers in brackets show the range of the horizontal displacements in seconds.

the SPG transformation appear to be ragged versions of their counterparts with the Raw transformation. This is because the range of the horizontal displacements for the RM3 printer is very small and comparable to the temporal resolution of the spectrograms.

Horizontal Displacement Arrays as a Property of Printing Processes. This consistency of horizontal displacement arrays across multiple types of side-channel signals and methods (DWM and DTW) indicate that a horizontal displacement array is a property of printing processes, not a property of side-channel signals. Because of this, we can say the horizontal displacement array between two printing processes, dropping side-channel signals altogether. It should be noted that the consistency of horizontal displacement arrays only occurs between the reference printing process and a benign printing process. There is no well-defined horizontal displacement array between the reference printing process and a malicious printing process.

Success of Dynamic Synchronization. The goal of dynamic synchronization (DWM or DTW) is to find the horizontal displacement array between two printing processes by analyzing side-channel signals. When dynamic synchronization is able to find this horizontal displacement array, we say the dynamic synchronization process is successful. In contrast, when dynamic synchronization returns a random look array, we say the dynamic synchronization process fails. Typically, for high quality side-channel signals (such as ACC-Raw, ACC-SPG, AUD-SPG), DWM and DTW return a horizontal displacement array that is very close to the actual horizontal displacement array with little noise. For low quality side-channel signals, dynamic synchronization typically fails altogether.

Inconsistency of Horizontal Displacement Arrays Across Printing Processes. Figure 3.34 shows the horizontal displacement arrays for three benign printing processes for each printer. We can see that the horizontal displacement arrays are different across different instances of printing processes, even though the printing processes were performed with the same G-code file and the same printer. This means that the time noise in a printing process is random instead of deterministic.



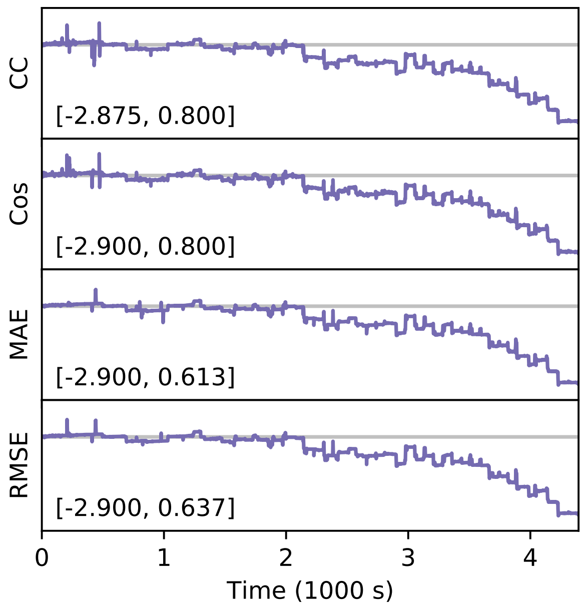
(a) UM3 (b) RM3

Figure 3.34: Horizontal displacement arrays ($\boldsymbol{h}_{\text{disp}}$) for three benign printing processes. The inconsistency of horizontal displacements across different instances of the benign printing process indicates that time noise is highly random. The numbers in brackets show the range of the horizontal displacements in seconds.

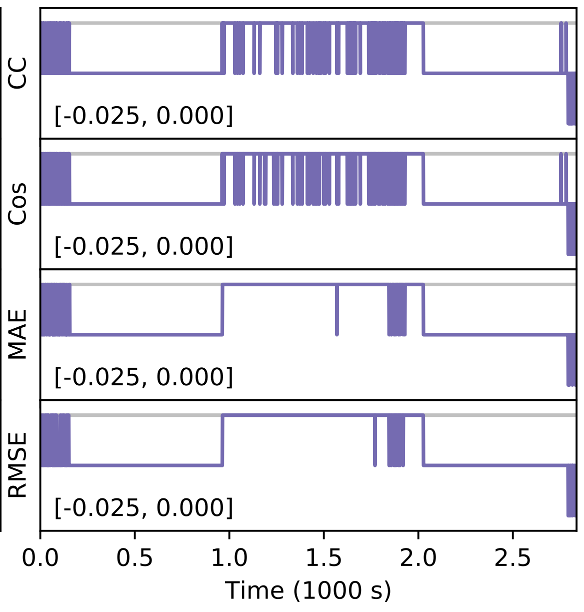
3.12.2 Parameters for Dynamic Time Warping

As previously mentioned, there are two parameters in DTW. One of them is the radius and the other is the distance metric. For the radius, since the calculation speed is the main concern, we always use the smallest allowable radius (which is 1) to expedite the algorithm at a cost of possibly reduced accuracy. In this section, we study how the distance metric affects the performance of DTW.

To save time, we only performed DTW for the 50 benign printing processes with ACC-SPG. Figure 3.35 shows the horizontal displacement arrays obtained by DTW with various distance metrics for one benign printing process. We can see that the distance metric does not significantly affect the horizontal displacement array. In this thesis, we will by default use the correlation distance.



(a) UM3



(b) RM3

Figure 3.35: Horizontal displacement arrays obtained by DTW with various distance metrics for a benign printing process and both printers (UM3 and RM3). The type of side-channel signals used to obtain the horizontal displacement arrays is ACC-SPG. The numbers in brackets show the range of the horizontal displacements in seconds.

3.12.3 Parameters for Dynamic Window Matching

In this section, we explore how parameters in DWM (kernels, n_{win}, n_{hop}, n_{ext}, n_{sigma}, and η) affect the performance of DWM and how each parameter are selected for the best performance. n_{win}, n_{hop}, n_{ext}, n_{sigma}, and η are originally defined in terms of indexes. The five parameters can be alternatively defined in terms of seconds. To be specific, we define $t_{\text{win}} = n_{\text{win}}/f_s$, $t_{\text{hop}} = n_{\text{hop}}/f_s$, $t_{\text{ext}} = n_{\text{ext}}/f_s$, and $t_{\text{sigma}} = n_{\text{sigma}}/f_s$, where f_s is the sampling rate of the side-channel signal.

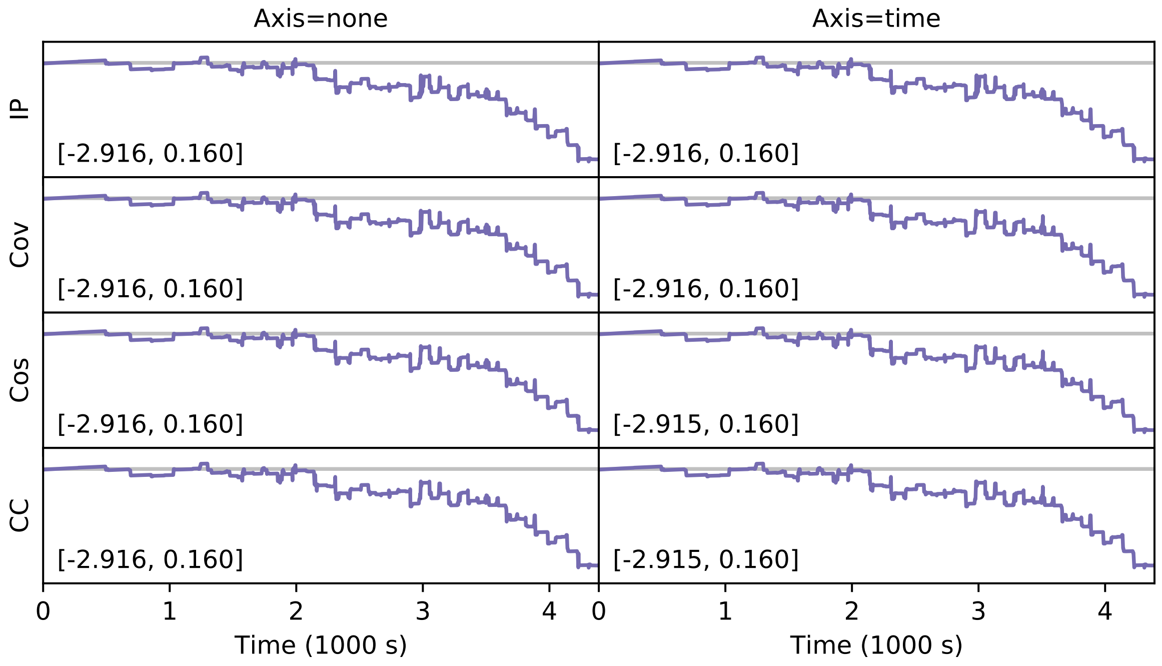
Kernels. We performed DWM with eight different kernels to figure out their performance. The eight kernels were obtained by combining four different score functions (IP, Cov, Cos, and CC) and two different calculation axes (none and time). We performed experiments with both printers, but to save time we only tested ACC-Raw, ACC-SPG, AUD-Raw, and AUD-SPG. For each kernel, each printer, and each type of side-channel signals, we performed DWM on the 50 benign printing processes. By analyzing the results, we find that, for raw signals (ACC-Raw and AUD-Raw), the kernel does not significantly affect the performance of DWM. However, for spectrograms (ACC-SPG and AUD-SPG), Cov-time and CC-time have similar performance and outperform other kernels.

Figure 3.36 shows horizontal displacement arrays obtained by DWM with the eight different kernels for one benign printing process and the UM3 printer. The results for other benign printing processes or the RM3 printer are similar and not shown.

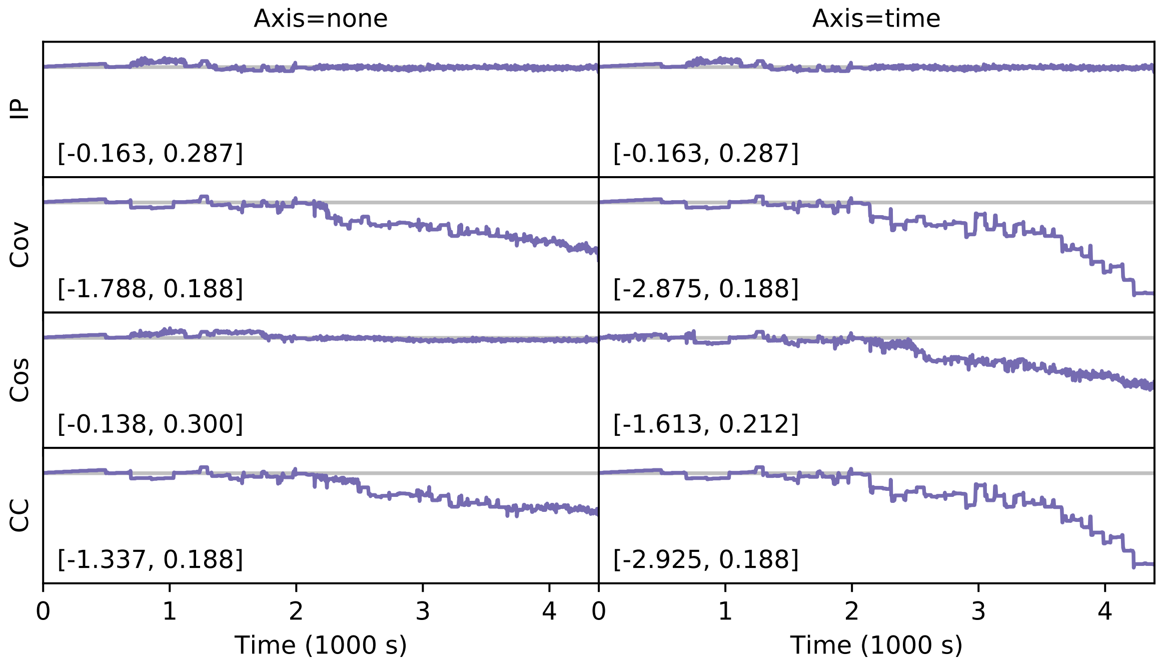
Parameters t_{ext} and t_{sigma}. The ratio $t_{\text{ext}}/t_{\text{sigma}}$ controls the strength of the bias in TDEB (as demonstrated in Figure 3.17), and a higher value of $t_{\text{ext}}/t_{\text{sigma}}$ corresponds to a stronger bias towards the center. When $t_{\text{ext}}/t_{\text{sigma}} < 1$, the bias effect is not significant. When $t_{\text{ext}}/t_{\text{sigma}} > 1$, the bias effect is significant and the extended window size is effectively determined by t_{sigma} instead of t_{ext}. By default, we use $t_{\text{ext}}/t_{\text{sigma}} = 2$ for two reasons. First, bias is desirable. Hence $t_{\text{ext}}/t_{\text{sigma}} > 1$. Second, when $t_{\text{ext}}/t_{\text{sigma}} > 3$, to maintain the same effective extended window size, increasing the ratio is tantamount to increasing t_{ext}. This merely increases the consumption of computational resources without other effects. As a balance, we choose $t_{\text{ext}}/t_{\text{sigma}} = 2$.

With $t_{\text{ext}}/t_{\text{sigma}} = 2$, t_{sigma} effectively determines the extended window size. The influence of t_{sigma} on $\boldsymbol{h}_{\text{disp}}$ is shown in Figure 3.37 (a). To ensure a successful DWM process, t_{sigma} should be larger than the absolute difference of the actual horizontal displacements between any two consecutive windows. At the same time, t_{sigma} should not be too large, as it not only requires more computational resources but also decreases the accuracy of DWM as a wider search area has more distraction.

There is an elegant way to select the best value for t_{sigma}. We first of all estimate the standard deviation of the duration of a benign printing process. This can be accomplished by performing a benign printing process multiple times and measuring the duration of all printing processes. We then select t_{sigma} to be a value that is larger than this standard deviation. Table 3.8 shows the standard deviations for UM3 and RM3 whereas Table 3.9 shows the selected t_{sigma} for both printers. It should be noted that, as the duration of a

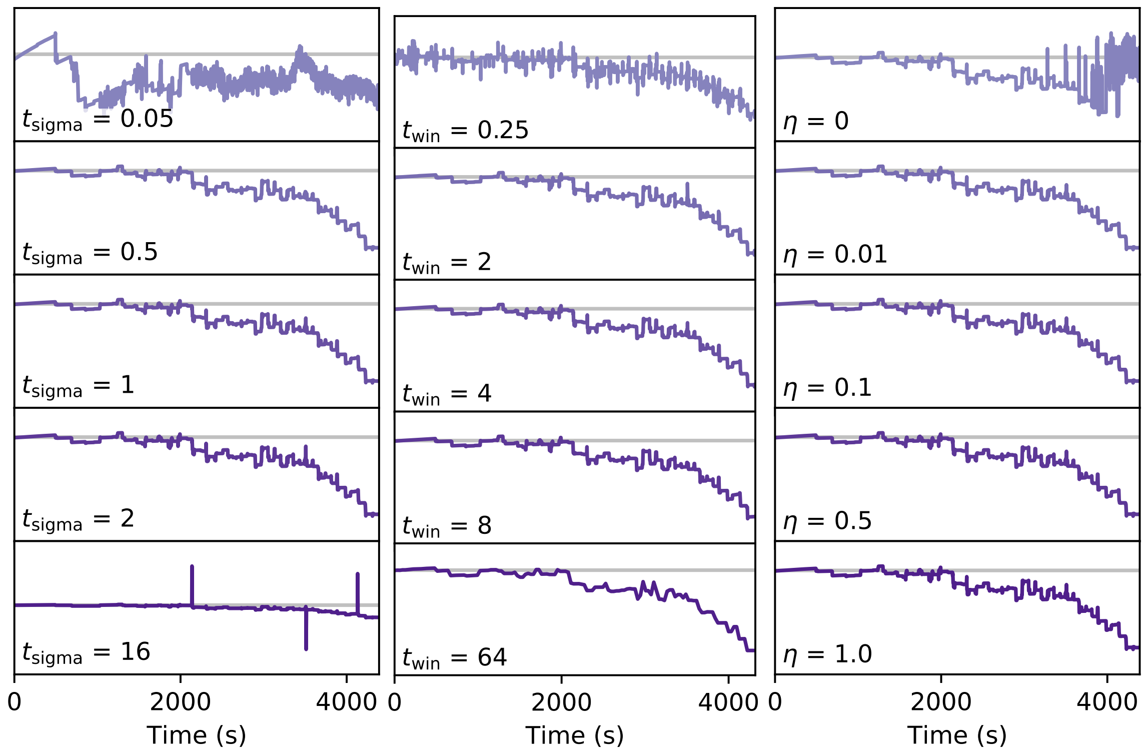


(a) UM3 + ACC-Raw



(b) UM3 + ACC-SPG

Figure 3.36: Horizontal displacement arrays vs kernels for the UM3 printer.



(a) t_{sigma} (b) t_{win} (c) η

Figure 3.37: Parametric analysis results for DWM with the UM3 printer. The ordinates are horizontal displacements in seconds. The brackets in all figures show the range of the horizontal displacements. The results were obtained with ACC-Raw. Results obtained by other types of side-channel signals are similar and not shown. Results with the RM3 printer exhibit similar patterns and are not shown to save space.

printing process increases, the standard deviation of the duration of the printing process may increase and the best t_{sigma} can increase accordingly.

Parameter t_{hop}. t_{hop} controls the temporal resolution of $\boldsymbol{h}_{\text{disp}}$. The maximum value of t_{hop} is t_{win} whereas the minimum value of t_{hop} is $1/f_s$. It is desirable to have a higher resolution by choosing a smaller t_{hop}. However, the computational cost increases significantly as t_{hop} is reduced. As a balance between computational cost and temporal resolution, we choose $t_{\text{hop}} = t_{\text{win}}/2$ by default.

Parameter t_{win}. t_{win} is the window size in the TDE process. Figure 3.37 (b) shows how t_{win} affects $\boldsymbol{h}_{\text{disp}}$. When t_{win} is very small, there are a lot of spikes in $\boldsymbol{h}_{\text{disp}}$. When t_{win} is very large, the temporal resolution of $\boldsymbol{h}_{\text{disp}}$ becomes lower.

In NSYNC, we obtain the best t_{win} by parametric analysis. We sweep t_{win} from a small value to a large value and select the t_{win} such that the change of the overall shape of $\boldsymbol{h}_{\text{disp}}$ is the smallest with respect to t_{win}.

Parameter η. Figure 3.37 (c) shows how η affects $\boldsymbol{h}_{\text{disp}}$. In general, it is necessary to have a positive η. For rare situations, when η is close to 1.0, DWM may run away.

To select the best η, we start with a small value of η, typically $\eta = 0.1$. If DWM is unable to converge, we can crank up this value until DWM converges.

Table 3.9 shows the parameters obtained according to the aforementioned procedures for both printers and we will use these parameters for the following experiments.

3.12.4 NSYNC with Dynamic Time Warping

We evaluated NSYNC with DTW as its dynamic synchronizer. We used $r = 0.3$ (the coefficient for margins of critical values) to bring down the overall FPR close to zero. We were not able to apply DTW on the raw signals because it took forever for DTW to synchronize them. The detection results are shown in Table 3.10. The performance metrics are the same as in subsection 3.11.6. The column $\boldsymbol{c}_{\text{disp}}$ shows the results if CADHD was used alone for intrusion detection. Similarly, the columns $\boldsymbol{h}_{\text{dist}}$ and $\boldsymbol{v}_{\text{dist}}$ show the results if $\boldsymbol{h}_{\text{dist}}$ and $\boldsymbol{v}_{\text{dist}}$ were used alone respectively. We can see that NSYNC with DTW mainly relied on horizontal displacements for intrusion detection, as the $\boldsymbol{v}_{\text{dist}}$-based sub-module totally failed. The acoustic side-channel spectrogram worked well for both printers. The acceleration side-channel spectrogram worked well only for the UM3 printer.

3.12.5 NSYNC with Dynamic Window Matching

We evaluated NSYNC with DWM as its dynamic synchronizer. We used $r = 0.3$ in the One-Class Classification (OCC) training process to bring down the overall FPR close to zero. The detection results are shown in Table 3.11. We can see that $\boldsymbol{c}_{\text{disp}}$ was the most reliable indicator. This means that dynamic synchronization on a benign process will likely

Table 3.10: Detection Results for NSYNC with DTW

P	**T**	**SC**	**Results**	**Individual Sub-Module Results**		
				$\boldsymbol{c}_{\text{disp}}$	$\boldsymbol{h}_{\text{dist}}$	$\boldsymbol{v}_{\text{dist}}$
UM3	SPG	ACC	0.99 (0.02 / 1.00)	0.99 (0.02 / 1.00)	1.00 (0.00 / 1.00)	0.50 (0.00 / 0.00)
		MAG	0.58 (0.10 / 0.26)	0.48 (0.08 / 0.04)	0.57 (0.10 / 0.24)	0.50 (0.00 / 0.00)
		AUD	0.97 (0.06 / 1.00)	0.97 (0.06 / 1.00)	0.97 (0.06 / 1.00)	0.50 (0.00 / 0.00)
		EPT	0.60 (0.04 / 0.24)	0.50 (0.00 / 0.00)	0.61 (0.00 / 0.22)	0.50 (0.04 / 0.04)
RM3	SPG	ACC	0.69 (0.02 / 0.40)	0.69 (0.02 / 0.40)	0.69 (0.02 / 0.40)	0.50 (0.00 / 0.00)
		MAG	0.70 (0.00 / 0.40)	0.70 (0.00 / 0.40)	0.70 (0.00 / 0.40)	0.50 (0.00 / 0.00)
		AUD	1.00 (0.00 / 1.00)	0.95 (0.00 / 0.90)	1.00 (0.00 / 1.00)	0.50 (0.00 / 0.00)
		EPT	0.50 (0.00 / 0.00)	0.50 (0.00 / 0.00)	0.50 (0.00 / 0.00)	0.50 (0.00 / 0.00)

P = Printer. T = Transformation (on Signals). SC = Side Channel. The combination of a side channel and a transformation constitutes a type of side-channel signals. The result format is Accuracy (FPR / TPR), where FPR = False Positive Rate and TPR = True Positive Rate.

succeed whereas dynamic synchronization on a malicious process will likely fail. $\boldsymbol{h}_{\text{dist}}$ and $\boldsymbol{v}_{\text{dist}}$ were less reliable as indicators on their own. When DSYNC fails for a malicious process, the resulting $\boldsymbol{h}_{\text{dist}}$ is meaningless and it may not be larger than the $\boldsymbol{h}_{\text{dist}}$ of a benign process. $\boldsymbol{v}_{\text{dist}}$ is not reliable on its own because amplitude noise can easily affect $\boldsymbol{v}_{\text{dist}}$. For example, acoustic noise (a type of amplitude noise) from the environment (such as talking) directly affects $\boldsymbol{v}_{\text{dist}}$, but this amplitude noise does not significantly affect $\boldsymbol{h}_{\text{disp}}$. Overall, the performance of NSYNC with DWM is excellent as an accuracy of 1.00 was reached for multiple types of side-channel signals. In addition, the $\boldsymbol{v}_{\text{dist}}$-based sub-module worked successfully, and this means that the NSYNC framework with DWM can potentially detect advanced attacks where the timing of the malicious process is preserved by the attacker.

3.12.6 Other Intrusion Detection Systems

This section presents intrusion detection results for IDSs in the literature. First of all, we present evaluation results for three IDSs that do not contain any form of dynamic synchronization, and they are Moore's IDS [23], Bayen's IDS [22], and Belikovetsky's IDS [27].

Moore's IDS [23]. This IDS essentially compares $\boldsymbol{a}[n]$ and $\boldsymbol{b}[n]$ without dynamic synchronization to obtain $\boldsymbol{v}_{\text{dist}}[n]$ for $n = 0, 1, \cdots$, where the distance metric is the Mean

Table 3.11: Detection Results for NSYNC with DWM

P	T	SC	Results	Individual Sub-Module Results		
				$\boldsymbol{c}_{\text{disp}}$	$\boldsymbol{h}_{\text{dist}}$	$\boldsymbol{v}_{\text{dist}}$
UM3	Raw	ACC	0.99 (0.02 / 1.00)	1.00 (0.00 / 1.00)	0.81 (0.02 / 0.64)	1.00 (0.00 / 1.00)
		MAG	1.00 (0.00 / 1.00)	1.00 (0.00 / 1.00)	0.97 (0.00 / 0.93)	0.76 (0.00 / 0.51)
		AUD	0.99 (0.02 / 1.00)	1.00 (0.00 / 1.00)	0.73 (0.02 / 0.47)	0.54 (0.00 / 0.08)
		EPT	0.53 (0.00 / 0.06)	0.53 (0.00 / 0.06)	0.50 (0.00 / 0.00)	0.53 (0.00 / 0.06)
	SPG	ACC	0.99 (0.02 / 1.00)	1.00 (0.00 / 1.00)	0.86 (0.02 / 0.73)	0.90 (0.00 / 0.80)
		MAG	1.00 (0.01 / 1.00)	1.00 (0.00 / 1.00)	0.93 (0.01 / 0.87)	0.78 (0.00 / 0.56)
		AUD	0.99 (0.02 / 1.00)	1.00 (0.00 / 1.00)	0.91 (0.02 / 0.83)	1.00 (0.00 / 1.00)
		EPT	1.00 (0.00 / 1.00)	1.00 (0.00 / 1.00)	0.76 (0.00 / 0.52)	1.00 (0.00 / 1.00)
RM3	Raw	ACC	1.00 (0.00 / 1.00)	1.00 (0.00 / 1.00)	0.90 (0.00 / 0.80)	1.00 (0.00 / 1.00)
		MAG	1.00 (0.01 / 1.00)	1.00 (0.01 / 1.00)	1.00 (0.00 / 1.00)	1.00 (0.00 / 1.00)
		AUD	1.00 (0.00 / 1.00)	1.00 (0.00 / 1.00)	0.79 (0.00 / 0.57)	1.00 (0.00 / 1.00)
		EPT	0.61 (0.00 / 0.21)	0.53 (0.00 / 0.05)	0.50 (0.00 / 0.00)	0.61 (0.00 / 0.21)
	SPG	ACC	1.00 (0.00 / 1.00)	1.00 (0.00 / 1.00)	0.96 (0.00 / 0.91)	0.52 (0.00 / 0.03)
		MAG	1.00 (0.00 / 1.00)	1.00 (0.00 / 1.00)	0.50 (0.00 / 0.00)	0.50 (0.00 / 0.00)
		AUD	1.00 (0.00 / 1.00)	1.00 (0.00 / 1.00)	0.96 (0.00 / 0.91)	1.00 (0.00 / 1.00)
		EPT	1.00 (0.00 / 1.00)	1.00 (0.00 / 1.00)	0.82 (0.00 / 0.63)	0.50 (0.00 / 0.00)

See Table 3.10 for table notes.

Absolute Error (MAE). This IDS uses the standard deviation at every moment to obtain the critical values in the discriminator. The IDS is originally designed for electric currents in motors. However, we were not able to observe this type of side-channel signals. Instead, we applied this IDS on available side-channel signals. The results are shown in Table 3.12. We can see that the IDS could not properly distinguish malicious printing processes from benign printing processes, as the accuracy for most types of side channels was around 0.5, which was basically guessing.

Bayen's IDS [22]. This IDS compares side-channel signals window by window (90 s or 120 s for the window size). This IDS first checks if the windows are in sequence. If not, an intrusion is declared. It then checks the score for each window. If the score of any window is below a pre-defined threshold, an intrusion is declared. However, in [22], there is no detail on how to obtain the threshold for a new printer. As a result, we used the OCC method in NSYNC to determine the thresholds for UM3 and RM3. We used $r = 0.0$

Table 3.12: Detection Results for Moore's and Gao's IDS

P	T	SC	Moore's Method	Gao's Method
UM3	Raw	ACC	0.48 (0.05 / 0.01)	0.51 (0.01 / 0.02)
		MAG	0.49 (0.03 / 0.01)	0.54 (0.01 / 0.08)
		AUD	0.48 (0.05 / 0.01)	0.49 (0.05 / 0.02)
		EPT	0.47 (0.31 / 0.25)	0.48 (0.30 / 0.25)
	SPG	ACC	0.50 (0.01 / 0.00)	0.50 (0.03 / 0.03)
		MAG	0.50 (0.03 / 0.02)	0.50 (0.12 / 0.12)
		AUD	0.50 (0.05 / 0.05)	0.50 (0.05 / 0.05)
		EPT	0.50 (0.00 / 0.00)	0.50 (0.01 / 0.00)
RM3	Raw	ACC	0.50 (0.00 / 0.00)	1.00 (0.01 / 1.00)
		MAG	0.54 (0.08 / 0.15)	0.54 (0.07 / 0.15)
		AUD	0.50 (0.00 / 0.00)	0.50 (0.00 / 0.00)
		EPT	0.52 (0.18 / 0.21)	0.53 (0.31 / 0.37)
	SPG	ACC	0.51 (0.02 / 0.03)	0.52 (0.00 / 0.03)
		MAG	0.52 (0.03 / 0.07)	0.49 (0.10 / 0.08)
		AUD	0.50 (0.00 / 0.00)	0.51 (0.00 / 0.02)
		EPT	0.64 (0.09 / 0.36)	0.57 (0.00 / 0.13)

Results are for AUD only. See Table 3.10 for more table notes.

because the TPRs for the threshold-based sub-module were very low. We tested this IDS only on AUD as this IDS supports only the acoustic side channel. The results are shown in Table 3.13. Overall, the performance of the IDS was unsatisfying as the IDS declared most printing processes as malicious. Especially, for the UM3 printer, the IDS simply declared all printing processes as malicious. The IDS performed better for the RM3 printer than its did for the UM3 printer. This could be due to the fact that the amount of time noise in the RM3 printer was much less than that in the UM3 printer.

Table 3.13: Detection Results for Bayens' IDS

Printer	Win (s)	Results	Individual Sub-Module Results	
			Sequence	Threshold
UM3	90	0.50 (1.00 / 1.00)	0.50 (1.00 / 1.00)	0.57 (0.18 / 0.31)
	120	0.50 (1.00 / 1.00)	0.50 (1.00 / 1.00)	0.54 (0.10 / 0.18)
RM3	90	0.75 (0.51 / 1.00)	0.75 (0.51 / 1.00)	0.95 (0.07 / 0.97)
	120	0.85 (0.30 / 1.00)	0.86 (0.29 / 1.00)	0.80 (0.04 / 0.63)

Results are for AUD only. See Table 3.10 for more table notes.

Table 3.14: Detection Results for Belikovetsky's IDS

Printer	Results
UM3	0.50 (1.00 / 1.00)
RM3	0.85 (0.31 / 1.00)

Belikovetsky's IDS [27]. This IDS applies the PCA to compress the number of channels in the spectrogram of the observed signal down to three. Suppose the result is $\boldsymbol{a}$. The reference signal goes through the same process. Suppose the result is $\boldsymbol{b}$. $\boldsymbol{a}$ and $\boldsymbol{b}$ are then compared point by point (without dynamic synchronization) using the cosine distance metric [27]. Suppose the result is $\boldsymbol{v}_{\text{dist}}$. A window of five seconds is used to calculate the moving average of $\boldsymbol{v}_{\text{dist}}$. If the average distances of four consecutive windows drop below 0.63, then an intrusion is declared. As with the Bayen's IDS, this method supports only the acoustic side channel. The intrusion detection results are shown in Table 3.14. We can see that the IDS totally failed for the UM3 printer and the performance of the IDS for the RM3 printer was unsatisfying as the accuracy was only around 0.85.

The next two IDSs to be evaluated contain a certain level of dynamic synchronization as they align the signals to be compared at the beginning of each layer. Since a layer can be considered as a huge window, this behavior can be considered as a form of dynamic synchronization, but on a very coarse level. The two IDSs are Gao's IDS [24] and Gatlin's IDS [26]. For the two IDS, we mainly present the evaluation results. Detailed analysis is left to interested readers.

Gao's IDS [24]. This IDS is similar to the Moore's IDS except two aspects. First, $\boldsymbol{a}$ and $\boldsymbol{b}$ are synchronized at moments when a layer change happens. Second, there is no discriminator in the Gao's IDS. As a result, we use the discriminator in NSYNC. We used $r = 0.0$ because the TPRs were very low. The results are shown in Table 3.12.

Gatlin's IDS [26]. This IDS first determines the layer changing moments for $\boldsymbol{a}$. If any moment deviates from its expected value by a predefined threshold, an intrusion is declared. Inside each layer, this IDS compares the spectrogram of $\boldsymbol{a}$ against that of $\boldsymbol{b}$ point

Table 3.15: Detection Results for Gatlin's IDS

P	**SC**	**Results**	**Individual Sub-Module Results**	
			Time	Match
UM3	ACC	0.85 (0.30 / 1.00)	0.93 (0.15 / 1.00)	0.73 (0.17 / 0.62)
	MAG	0.74 (0.53 / 1.00)	0.92 (0.16 / 1.00)	0.47 (0.44 / 0.38)
	AUD	0.89 (0.22 / 1.00)	0.93 (0.14 / 1.00)	0.49 (0.09 / 0.07)
	EPT	0.97 (0.05 / 0.98)	0.97 (0.05 / 0.98)	0.51 (0.00 / 0.02)
RM3	ACC	0.86 (0.29 / 1.00)	0.97 (0.07 / 1.00)	0.87 (0.26 / 1.00)
	MAG	0.92 (0.17 / 1.00)	0.98 (0.05 / 1.00)	0.94 (0.12 / 1.00)
	AUD	0.90 (0.20 / 1.00)	0.95 (0.10 / 1.00)	0.95 (0.11 / 1.00)
	EPT	0.96 (0.08 / 1.00)	0.96 (0.08 / 1.00)	0.50 (0.00 / 0.00)

P = Printer. SC = Side Channel. The result format is Accuracy (FPR / TPR), where FPR = False Positive Rate and TPR = True Positive Rate.

by point using a special distance metric [26]. If the cumulative distance in a layer exceeds a predefined threshold, an intrusion is declared. More details of this IDS can be found in [26] or section 2.2. This IDS is originally designed for the electric currents in motors and the layer changing moments are determined by detecting activities in the currents in the Z motor. Since we were not able to access the currents in any motor, we obtained the layer changing moments manually, and we applied this IDS to the side-channel signals that we obtained. The detection results are shown in Table 3.15.

3.12.7 Comparison of Different IDSs

Figure 3.38 summaries the average accuracy for all IDSs that we evaluated. The results were averaged over all printers and all types of side-channel signals (except EPT-Raw). As we can see in Figure 3.38, as the level of dynamic synchronization (DSYNC) increases from none to fine, the overall accuracy of the IDSs increases. Any IDS labeled with the symbol "T" in Figure 3.38 means that the IDS uses time as an indicator for intrusion detection. By analyzing the results of sub-modules in Table 3.10 and Table 3.11, we can see that time is a more effective indicator than amplitude for intrusion detection.

In addition, we measured the average time it took to analyze one second of the side-channel signals for both DWM and DTW. The results are shown in Figure 3.39. We can

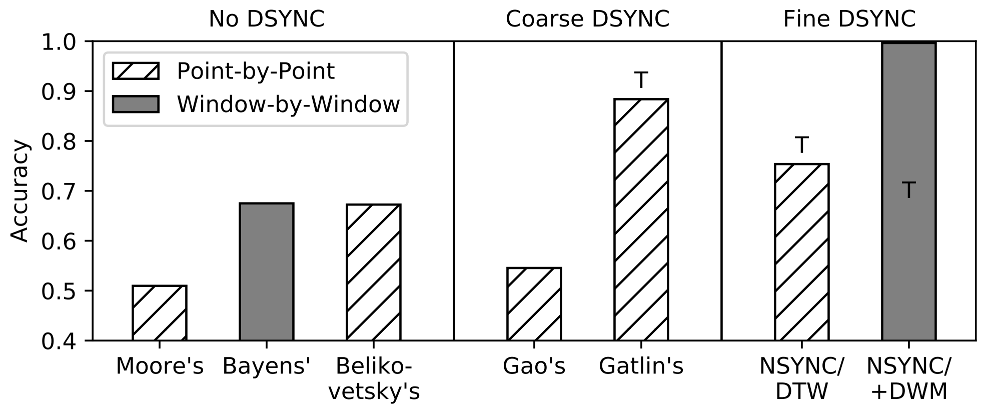


Figure 3.38: Average accuracy of seven different IDSs in our evaluation. The symbol T means that the IDS uses time as an indicator for intrusion detection.

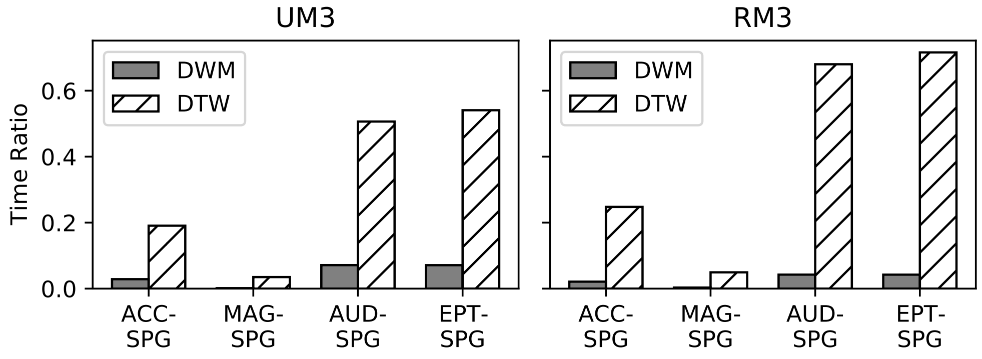


Figure 3.39: Average time it took for DWM and DTW to dynamically synchronize one second of the side-channel signals. This is also known as the time ratio.

see that DTW was much slower than DWM, even if we used FastDTW with the fastest configuration to implement DTW.

3.13 Conclusion

In this chapter, we presented NSYNC, a practical framework of IDSs that leverage side-channel signals in AM systems. The NSYNC framework is composed of four major components, and they are the static synchronizer, the dynamic synchronizer, the comparator, and the discriminator.

3.13.1 Static Synchronizer

The static synchronizer performs static synchronization (SSYNC), which finds specific moments in the signals to be analyzed. The specific moments can be the starting moments or the stopping moments of the signal comparison process. The signals are aligned at their starting moments before the signal comparison process is initiated. The stopping moments in the signals tell the NSYNC framework to terminate the signal comparing process and completes the intrusion detection process. In the literature, only a limited amount of existing IDSs, such as Moore et at. [23] and Belikovetsky et al. [27], are aware of the importance of aligning signals to be compared at their starting moments and explicitly provide solutions to static synchronization.

Our proposed static synchronizer can be operated in two modes. One of them is the TDE mode, which mainly supports post-production analysis or offline signals. The other mode is the SD mode, which mainly supports real-time analysis or online signals. The performance of the SD mode is worse than the TDE mode. Nevertheless, with high quality side-channel signals, our static synchronizer could reliably determine the starting and stopping moments in the SD mode for all benign printing processes.

We implemented an intrusion detection system based on the duration of a printing process and the duration was estimated by the static synchronizer in the SD mode. With high quality side-channel signals, the intrusion detection system could reliably differentiate benign printing processes and malicious printing processes.

Even though the static synchronizer in the SD mode is unable to properly identify the starting and stopping moments of a malicious printing process, it will return a random duration or a status that the moments are not detected at all. When either situation happens, the duration-based intrusion detection system will consider the printing process to be malicious. Additionally, with wrongly identified starting and stopping moments, the dynamic synchronization process in the NSYNC framework will likely fail and issue an alert. In this way, the malicious printing process can still be properly captured.

For our static synchronizer, we tested eight different kernels (four score functions and two calculation axes) for a variety of scenarios, and we find that the Correlation Coefficient (CC) score function with the time axis has the best overall performance. In addition, we explored the design parameters in a template, namely the duration and the anchor point of the template. We found that the best performance of a template is obtained when the template has an intermediate level of duration. Normally, the anchor point is located on one of the edges of the template but allowing for an arbitrary location of the anchor point can improve the performance of static synchronization under certain circumstances.

3.13.2 Dynamic Synchronizer

The dynamic synchronizer in the NSYNC framework performs dynamic synchronization (DSYNC), which finds the timing relationship between two signals to be compared. The timing relationship is described by what is known as the horizontal displacements. Dynamic synchronization is required because there is time noise in a printing process. As we demonstrated in the experiments, an intrusion detection system will fail if it compares two signals directly without any form of dynamic synchronization. An existing method to perform dynamic synchronization is Dynamic Time Warping (DTW). However, our evaluation results show that DTW is not suitable for analyzing side-channel signals in AM systems because DTW not only has limited accuracy but also consumes an excessive amount of computational resources. Our newly proposed method, Dynamic Window Matching (DWM), can successfully overcome the problems of DTW.

One of the most important observations in our experiments is the consistency of horizontal displacement arrays for benign printing processes. Normally, a horizontal displacement array is obtained between two side-channel signals. One of the side-channel signals acts as the reference whereas the other side-channel signal acts as the observation. According to our experiments, the horizontal displacement arrays by different types of side-channel signals are the same provided that the quality of the side-channel signals are high enough for dynamic synchronization to succeed. This means that a horizontal displacement array is the property of two printing processes, not two side-channel signals, even though side-channel signals are needed to obtain the horizontal displacement array.

The consistency of horizontal displacement arrays only occurs for benign printing processes. For a malicious printing process, the dynamic synchronization process typically fails and yields a horizontal displacement array that fluctuates violently. To capture the level of fluctuation in a horizontal displacement array, we use the Cumulative Absolute Difference of Horizontal Displacement (CADHD). A high level of CADHD corresponds to more violent fluctuation in the horizontal displacement array, and a higher likelihood of dynamic synchronization failure.

For our dynamic synchronizer, we explored the influence of each parameter on the performance of dynamic synchronization. We found that the CC-time kernel (the correlation coefficient score function with the time axis) has the overall best performance. The Cov-time kernel (the covariance score function with the time axis) has very similar performance, but we simply use the CC-time kernel. The sigma value, the window size, and the eta value all need to be selected properly as extremely large values or extremely small values can result in poor performance. As a convention, we always select the extended window size to be twice of the sigma value, and we always select the hop size to be half of the window size. The optimal value for each printer is different but in general stays consistent for the same printer across multiple printing processes.

3.13.3 Comparator

While performing dynamic synchronization, we can also obtain the vertical distances between the two signals to be compared. In theory, any distance metric can be used but we use the correlation distance metric with the time axis as it is readily available from the CC-time kernel in the dynamic synchronizer.

3.13.4 Discriminator

The discriminator then takes in the horizontal displacements and the vertical distances to determine if the two signals to be compared are different or not in real time. One of the most important features of the discriminator in the NSYNC framework is that we use One-Class Classification to determine the threshold between benign printing processes and malicious printing processes. The main advantage of this approach is that it uses a series of benign printing processes to determine the threshold and it does not require knowledge of malicious printing processes. A margin ratio is required, and its value decreases as the number of benign printing processes increases.

3.13.5 Experiments

We evaluated the performance of multiple IDSs on the same set of printing processes. We saw that the NSYNC framework performed better than existing IDSs. The NSYNC framework with DWM reached an average accuracy over 0.99, beating all other IDSs, including the NSYNC framework with DTW. We also saw that time-based intrusion detection performed better than amplitude-based intrusion detection. Last but not least, we compared the time consumption of DWM and DTW and DWM was much faster than DTW.

CHAPTER 4
OPTICAL SIDE CHANNEL: ATTACK AND DEFENSE

4.1 Introduction

In chapter 3, we saw how side-channel signals could be used by defenders to perform intrusion detection for AM systems. When a side-channel signal is correlated with the state-variable signal of an AM system, the side-channel signal can be used for intrusion detection. When the state of the AM system is maliciously modified by an attacker, the change will be reflected in the side-channel signal. An intrusion detection system performs its duty by detecting the change in the side-channel signal.

When a side-channel signal is correlated with the state-variable signal of an AM system, the side-channel signal can be used by an attacker to perform a side-channel attack. The attacker analyzes the side-channel signal and attempts to infer the state-variable signal. For example, an attacker may recover the printing path of a printing process by analyzing the acoustic side-channel signal from the printing process [18, 19, 20].

In this chapter, we explore the possibility to recover the state-variable signal from the optical side-channel signal of a printing process. The optical side-channel signal is the video of a printing process and is obtained by an optical camera. Compared with the acoustic side-channel signal, the optical side-channel signal is in general harder to obtain because it is harder to hide a suspicious camera than it is to hide a suspicious microphone. Nevertheless, we focus on the optical side channel for the following reasons.

- First, the optical side-channel attack on AM systems has not been performed. Although there is an infrared side-channel attack in the literature [32], the attack failed in their own evaluation. In fact, there is a good chance for the method in our optical side-channel attack to be applicable to the infrared side channel.

- Second, the acoustic side-channel attack is very challenging. To successfully perform the acoustic side-channel attack, a lot of restrictions apply. There are so many restrictions that it is impossible to perform the acoustic side-channel attack for many realistic printing processes. In contrast, the optical side-channel attack can be easily performed for any printing process.

- Third, it is possible to obtain the optical side-channel signal under several circumstances, such as hacking surveillance cameras and process monitoring cameras.

The structure of this chapter is as follows. We first of all describe the threat model for the optical side-channel attack. We then discuss the details on the optical side-channel attack. Afterwards, we present the optical noise injection method to defend against the optical side-channel attack. Finally, we present and discuss the experiment results. We do not have a dedicated section for background information in this chapter as the background information for this chapter is the same as the one in section 3.2.

4.2 Threat Model

The threat model is illustrated in Figure 4.1. A 3D printer is printing an object of value. An attacker wishes to replicate this object without authorization. For this purpose, the attacker uses a sensor around the printer to collect the side-channel signal in the printing process. The attacker then uses a variety of methods, such as signal processing and machine learning, to recover information about the printing process. The recovered information, namely the intellectual property, can be used to reconstruct the printing process.

4.2.1 Operational Definition of Intellectual Property

Strictly speaking, the intellectual property of an AM process includes all information that is necessary to exactly replicate the printed object. The information includes the make and model of the printer that is used, the composition of the material, the geometry of the

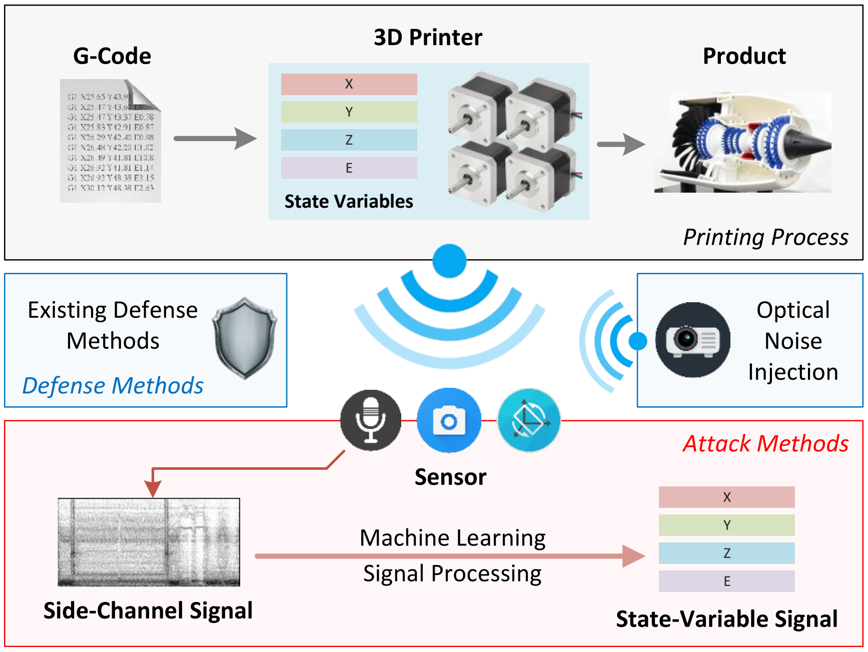


Figure 4.1: Illustration of the threat model. Multiple sensors are shown to make this threat model to be applicable to existing side-channel attacks. Nevertheless, the focus of this thesis is the optical side channel.

design, and the manufacturing parameters [15].

We assume that the attacker knows the make and model of the printer and the composition of the material. The geometry of the design and the manufacturing parameters are embedded in the G-code file. In fact, many papers in the literature define the intellectual property of a printing process as the G-code file [18, 20]. However, we argue that it is not practical to recover the G-code file from the side-channel signal ($\boldsymbol{x}_{\mathrm{SC}}$) of a real printing process as it is not possible to segment the side-channel signal by G-code instructions, as explained in Figure 4.2. As a result of this, in this thesis, we operationally define the intellectual property of a printing process as its state-variable signal ($\boldsymbol{x}_{\mathrm{SV}}$).

To fully replicate a printing process, the state variable should contain at least four channels and they are the x, y, z coordinates of the printhead and the coordinate of the extruder, denoted by e. It is desirable to have other state variables such as the temperature of the nozzle(s), the temperature of the build plate, and the speed of the cooling fan(s). Since the attacker knows the make and model of the printer as well as the filament type, the attacker

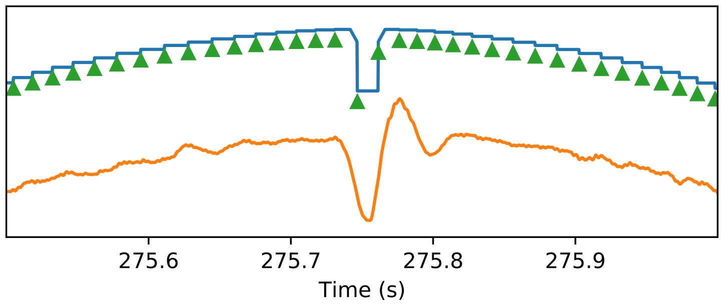


Figure 4.2: Simulated and estimated velocity signals. The upper signal is a velocity signal obtained by simulating a G-code file. The lower signal is the same velocity signal estimated by analyzing a side-channel signal. The green triangles show the boundaries between G-code instructions. If we can only see the estimated velocity signal, it is nearly impossible to determine the boundaries of the G-code instructions.

can infer the temperature and the fan speed based on common practices.

4.2.2 Acquisition of Side-Channel Signals

In this thesis, we are mainly concerned with the optical side channel. There are various ways for an attacker to obtain the optical side-channel signal of a printing process.

- The attacker may have an employee or maintenance contractor working at the factory to install a hidden camera on the ceiling of the factory. The AM factory may have many devices installed on the ceiling, such as fire detectors, sprinklers, motion sensors, light sensors, etc. The hidden camera can be disguised as one of these legitimate devices to evade suspicion.
- The AM factory may have surveillance cameras, which may be connected to the Internet. They can be compromised to become the part of a botnet [57]. The attacker may be able to get a copy of the data from a compromised camera [58].
- In addition, some printers have built-in process monitoring cameras with weak access control and the attacker can access the cameras directly. For example, Ultimaker 3, a very popular desktop 3D printer, has strong security policies for making changes to

the printer and does not allow G-code extraction, but allows anyone to see the feed of its internal camera without authentication.

4.2.3 Analysis of Side-Channel Signals

The attacker uses signal processing and data driven methods such as template filtering and machine learning to recover the state-variable signal from the side-channel signal. For data driven methods, we assume that the attacker has access to the same printer model (and the same filament) in a similar environment. As a result, the attacker can collect a lot of training data, subject to the constraints of time and computational resources.

When a defense method is deployed, we assume that an advanced attacker knows the defense method and tries to evade the defense method by launching more sophisticated attacks. For example, when the noise injection method is used, the attacker can try to learn the pattern of the injected noise and artificially create training samples with the same type of injected noise. The attacker then retrains the machine learning model with the newly created training samples. The retrained machine learning model may be able to automatically reject the injected noise and proceed with the attack.

4.3 Optical Side-Channel Attack

This section describes the details of the optical side-channel attack. We first of all mathematically describe the side-channel attack. We then discuss the challenges in the attack. Finally, we present the implementation of the attack.

4.3.1 Attack Formulation

The optical side-channel attack essentially attempts to recover the state-variable signal ($\boldsymbol{x}_{\mathrm{SV}}$) from the optical side-channel signal ($\boldsymbol{x}_{\mathrm{SC}}$). The process is possible because there is relationship between $\boldsymbol{x}_{\mathrm{SV}}$ and $\boldsymbol{x}_{\mathrm{SC}}$. For example, the position of the printhead directly affects the image seen by a camera. The speed of the printhead also affects the image seen by a camera in the form of motion blur. The relationship between $\boldsymbol{x}_{\mathrm{SV}}$ and $\boldsymbol{x}_{\mathrm{SC}}$ can be mathematically described by

$$\boldsymbol{x}_{\mathrm{SC}}[n] = \boldsymbol{f}(\boldsymbol{x}_{\mathrm{SV}}[n], \frac{\partial \boldsymbol{x}_{\mathrm{SV}}}{\partial t}[n]) + \boldsymbol{e}[n], \tag{4.1}$$

where n is the time index, $\boldsymbol{x}_{\mathrm{SC}}[n]$ is a single frame in the video (an image), $\boldsymbol{x}_{\mathrm{SV}}[n]$ is the state variable at time index n, $\partial \boldsymbol{x}_{\mathrm{SV}}/\partial t$ is the derivative of $\boldsymbol{x}_{\mathrm{SV}}$ with respect to time t, $\boldsymbol{f}$ is the mapping from $\boldsymbol{x}_{\mathrm{SV}}[n]$ and its derivative to $\boldsymbol{x}_{\mathrm{SC}}[n]$, and $\boldsymbol{e}[n]$ is the noise at time index n.

Since it is very hard to mathematically solve for $\boldsymbol{x}_{\mathrm{SV}}[n]$ from $\boldsymbol{x}_{\mathrm{SC}}[n]$, we use data-driven methods such as template matching [19] and machine learning [20].

4.3.2 Challenges of the Attack

There are challenges in the optical side-channel attack. The details of the challenges and the mitigation strategies are described as follows.

Limited Relationship. The relationship between certain channels in the state-variable signal and the optical side-channel signal may be weak. For example, the position of the filament, the temperature of the nozzle(s), and the temperature of the build plate do not affect the optical side-channel signal. As a result, it is nearly impossible to recover these channels in the state-variable signal. In response to this, we only recover the coordinates of the printhead from the side-channel signal. We may guess the temperature of the nozzle(s) and the temperature of the build plate with a high accuracy since the optimal temperatures for a material are typically known constants. However, we need to know the position of the filament in the whole printing process. To solve this problem, we use a common assumption in the literature. That is, the attacker assumes that the printhead mainly travels at two target speeds, a low speed for extrusion movements and a high speed for relocating the printhead. This assumption is used by existing side-channel attacks [18, 20, 19].

Camera Properties. The configuration of the camera can affect the side-channel attack, such as the location and angle of the camera with respect to the printer as well as the focal length, white balance, and exposure of the camera. When the attacker recreates a system to collect training images, we assume that the attacker is able to get the same printer and the same camera with a similar configuration. In addition, we assume that it is possible for the attacker to manually label a limited number of images in videos taken from the camera used in the side-channel attack and use the labeled images for training purposes.

Lighting Conditions. The lighting conditions can change over time, especially when the AM facility has windows that let natural lights through. A robust side-channel attack should be able to tolerate changes in lighting conditions. In this case, machine learning may be a better choice over template matching. To improve the attack's robustness, it is important to augment the training dataset by varying its exposure, brightness, and contrast, to simulate changes in the lighting conditions.

Motion Blur Effect. It takes time for a camera to collect enough photons to register a picture, and this time is referred to as the shutter speed of the camera. If the printhead moves when the camera's shutter is open, the moving printhead gets blurred in the registered picture, and this is referred to as the motion blur effect. To account for the motion blur effect, we need six degrees of freedom in the label of a picture, and they are the x, y, z coordinates of the printhead as well as the three components of the velocity vector of the printhead. Due to the curse of dimensionality in machine learning, we need to collect a huge amount of training images with the printhead at different locations with different velocities. This can be rather challenging. To mitigate this problem, when the ambient light is adequate and the camera has a fast shutter speed, we can ignore the motion blur effect and we have

$$\boldsymbol{x}_{\mathrm{SC}}[n] = \boldsymbol{f}(\boldsymbol{x}_{\mathrm{SV}}[n]) + \boldsymbol{e}[n]. \tag{4.2}$$

Deposited Materials. Materials (filaments) are deposited in a printing process. As a result, for the same location of the printhead, it is possible for the camera to see different images due to the deposited materials. This makes the side-channel attack hard because for

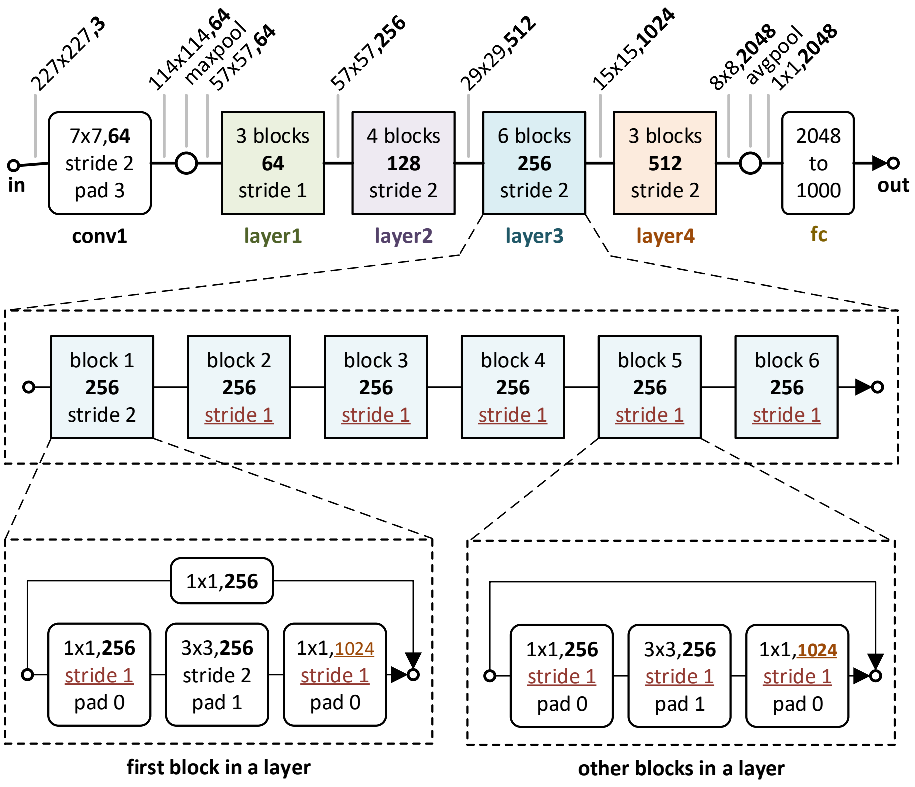


Figure 4.3: Structure of ResNet50.

the same label (the same location of the printhead), the attacker needs to collect multiple images with different distribution of deposited materials on the build plate. To make things worse, changing the deposited materials on the build plate requires human efforts and can be very time consuming. Fortunately, when printing an object that is smaller than the printhead, the object may be totally obscured by the printhead. When this happens, we do not need to consider the influence of deposited materials on the side-channel attack.

4.3.3 Implementation of the Attack

In this thesis, we use ResNet50 [59], a deep convolutional neural network, to recover the state-variable signal from the optical side-channel signal. The structure of ResNet50 is shown in Figure 4.3.

Modifying ResNet50. ResNet50 was originally designed to perform image classification for 1000 classes. The output of ResNet50 contains 1000 numbers. We modified the last fully connected layer in ResNet50 such that it transforms 2048 neurons into 3 neurons, corresponding to x, y, and z respectively. In this thesis, we refer to the modified ResNet50 directly as ResNet50. The input to ResNet50 is a single image showing the printer whereas the output of ResNet50 are the coordinates of the printhead (x, y, and z).

Training Process. To train the deep neural network, we collect images of the printer with the printhead at various locations without filament extrusion[1]. To be specific, we instruct the printhead to sweep the whole printing space with a predetermined interval. To improve the robustness of the attack, we augment the training dataset by randomly rotating the images, randomly cropping and resizing the images, and randomly perturbing the brightness, contrast, and hue of the images. We use the Mean Square Error (MSE) as the loss function and we use the Adam algorithm [60] to train the neural network. We gradually reduce the learning rate in the training process. We use a GPU for training. We select a batch size that can fill up the available GPU memory and we shuffle the training dataset when fetching a batch.

Testing Process. To evaluate the performance of the trained neural network, we use two different testing datasets with each testing dataset having its own purpose.

For the first testing dataset, we collect the testing images in a similar manner that we collect the training images. We execute one G-code instruction at a time, wait for the instruction to be fully completed, take a picture, use the coordinates of the printhead as the label for the picture, and then proceed to the next G-code instruction. The main advantage of this testing dataset is that we have the ground truth label for each image, and we can calculate the Mean Square Error (MSE) between the predicted labels and the ground truth labels as the performance metric of the neural network.

For the second testing dataset, we perform the printing process in real time, record the printing process as a video, and extract frames in the video one by one. The main purpose of this testing dataset is to evaluate the performance of the optical side-channel attack for

[1]In other words, the build plate is clean for all images in the training dataset.

a real printing process. For this testing dataset, we have the motion blur effect (as a side effect) and we do not have the ground truth label for each image. As a result, we only qualitatively measure the performance of the neural network by visually looking at the recovered printing path.

4.4 Optical Noise Injection

The noise injection method uses a signal generator to artificially create a side-channel signal to interfere with the side-channel signal emitted by a printer, in an attempt to thwart the side-channel attack. From the perspective of the attacker, the injected side-channel signal is noise. Compared with many other methods to defend against side-channel attacks as discussed in chapter 2, the main advantage of the noise injection method is its low overhead. This method does not make any change to the existing physical components in the AM system, nor does this method require modification to the G-code instructions. This section discusses the details of the optical noise injection method. We first discuss optical projectors. We then discuss existing and proposed noise generation algorithms.

4.4.1 Problem Definition

Several devices can be used to general optical side-channel signals, such as light bulbs, Light-Emitting Diodes (LEDs), and optical projectors. We use an optical projector to generate optical side-channel signals, because a projector can readily accept a control signal to generate a wide range of optical signals.

Control Signals. For a projector to work, a control signal, denoted by $\boldsymbol{x}_{\mathrm{CS}}$, must be provided. The control signal is typically a video file that is played by the projector. When considering the defender's projector and the attacker's camera as a system, they can be modeled by

$$\boldsymbol{x}_{\mathrm{SC}}[n] = \boldsymbol{g}(\boldsymbol{x}_{\mathrm{SV}}[n], \boldsymbol{x}_{\mathrm{CS}}[n]) + \boldsymbol{e}[n], \tag{4.3}$$

where $\boldsymbol{x}_{\mathrm{CS}}[n]$ is the control signal at time index n, $\boldsymbol{x}_{\mathrm{SC}}[n]$ is the optical side-channel signal at time index n, and $\boldsymbol{e}[n]$ is the noise at time index n. When $\boldsymbol{x}_{\mathrm{CS}}$ is zero, $\boldsymbol{g}$ in Equation 4.3 degenerates to $\boldsymbol{f}$ in Equation 4.2.

Number of Channels. $\boldsymbol{x}_{\mathrm{SC}}$, $\boldsymbol{x}_{\mathrm{SV}}$, and $\boldsymbol{x}_{\mathrm{CS}}$ typically have very different numbers of channels. Figure 4.4 (a) shows an example of a frame of $\boldsymbol{x}_{\mathrm{CS}}$. The projector accepts images of 1024×768 pixels and each pixel contains 3 color values. Hence, there are $1024 \times 768 \times 3$ channels in $\boldsymbol{x}_{\mathrm{CS}}$. Figure 4.4 (b) shows the corresponding image seen by the camera and the image contains 227×227 pixels and each pixel contains 3 color values. Hence, there are $227 \times 227 \times 3$ channels in $\boldsymbol{x}_{\mathrm{SC}}$. Finally, the number of channels in $\boldsymbol{x}_{\mathrm{SV}}$ is equal to the number of components in the state variable that the attacker is attempting to recover, and in this case it is 3 for x, y, and z.

Channel Transformation. Since there are 1024×768 pixels in $\boldsymbol{x}_{\mathrm{CS}}$ and 227×227 pixels in $\boldsymbol{x}_{\mathrm{SC}}$, $\boldsymbol{x}_{\mathrm{CS}}$ and $\boldsymbol{x}_{\mathrm{SC}}$ are not channel compatible. Nevertheless, for every pixel in $\boldsymbol{x}_{\mathrm{CS}}$, it may have corresponding pixels in $\boldsymbol{x}_{\mathrm{SC}}$. Conversely, for every pixel in $\boldsymbol{x}_{\mathrm{SC}}$, it may have corresponding pixels in $\boldsymbol{x}_{\mathrm{CS}}$. The correspondence between pixels in $\boldsymbol{x}_{\mathrm{CS}}$ and pixels in $\boldsymbol{x}_{\mathrm{SC}}$ is referred to as channel transformation. With channel transformation, we can express $\boldsymbol{x}_{\mathrm{CS}}$ using the same channel structure of $\boldsymbol{x}_{\mathrm{SC}}$. In the rest of the thesis, we always express $\boldsymbol{x}_{\mathrm{CS}}$ with channel transformation performed. Hence, $\boldsymbol{x}_{\mathrm{CS}}$ always has the same channel structure as $\boldsymbol{x}_{\mathrm{SC}}$.

Independence of Channels. In this thesis, we in addition assume that channels are independent, although this assumption may not be strictly true due to a phenomenon called leakage. For example, when a projector sends a ray of light that is registered in a single pixel in the camera, the adjacent pixels may also be slightly affected. In this thesis we neglect the leakage phenomenon and we have

$$\boldsymbol{x}_{\mathrm{SC}}[n, c] = \boldsymbol{g}_c(\boldsymbol{x}_{\mathrm{SV}}[n], \boldsymbol{x}_{\mathrm{CS}}[n, c]) + \boldsymbol{e}[n, c]. \tag{4.4}$$

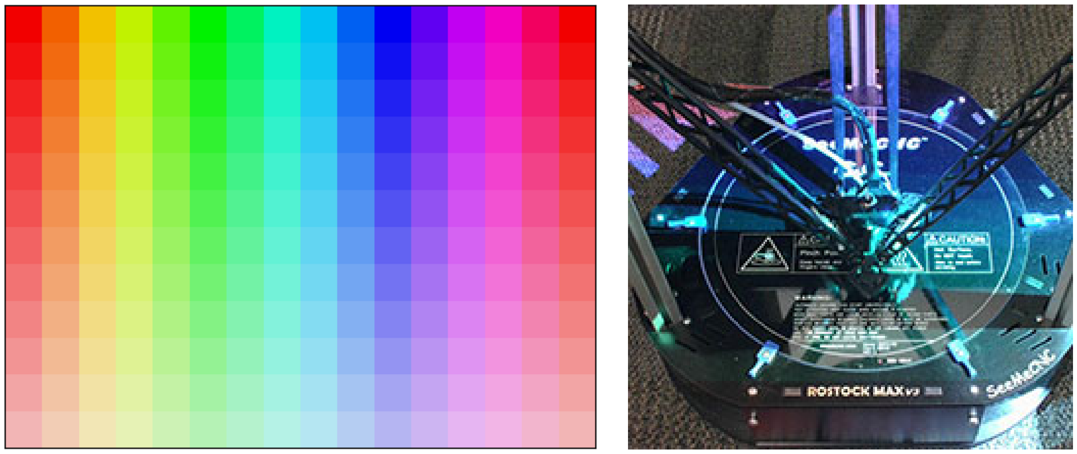

(a) Control Signal (b) Side-Channel Signal

Figure 4.4: Illustration of channel transformation and the capability of the optical projector.

In other words, $\boldsymbol{x}_{\mathrm{SC}}[n, c]$ is only affected by $\boldsymbol{x}_{\mathrm{CS}}[n, c]$. Notice the function here is $\boldsymbol{g}_c$, which takes $\boldsymbol{x}_{\mathrm{CS}}[n, c]$ as an argument. In contrast, $\boldsymbol{g}$ takes $\boldsymbol{x}_{\mathrm{CS}}[n]$ as an argument.

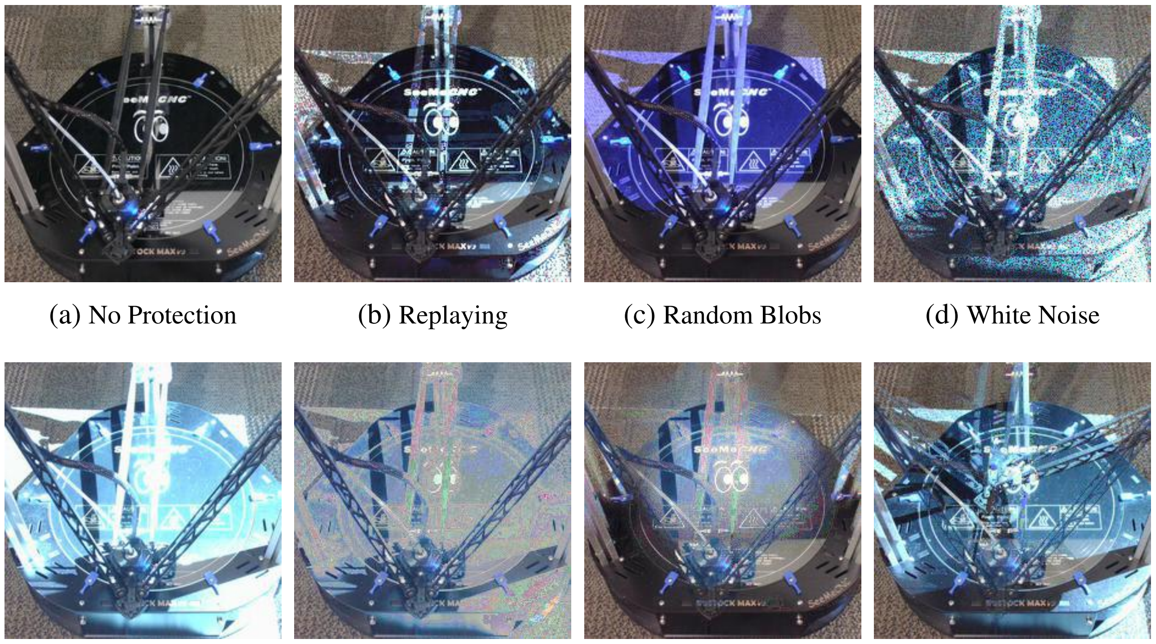
Capability of the Projector. As shown in Figure 4.4, for any pixel in the control signal ($\boldsymbol{x}_{\mathrm{CS}}$), its corresponding pixel in the side-channel signal ($\boldsymbol{x}_{\mathrm{SC}}$) typically takes on a different color. In fact, a pixel in $\boldsymbol{x}_{\mathrm{CS}}$ can take any color in the whole RGB color space. However, the corresponding pixel in $\boldsymbol{x}_{\mathrm{SC}}$ can only reach a (narrow) subset of the whole RGB color space. We refer to the color space that can be reached by a pixel in $\boldsymbol{x}_{\mathrm{SC}}$ as the capability of the projector for that pixel.

If a projector has unlimited capability, we should be able to manipulate $\boldsymbol{x}_{\mathrm{SC}}$ in a way to fully prevent side-channel attacks. One way to do this is to make $\boldsymbol{x}_{\mathrm{SC}}$ a constant value over both time and channel. However, due to the limited capability of the projector, we have to carefully design $\boldsymbol{x}_{\mathrm{CS}}$ to maximize the performance of protection.

4.4.2 Existing Noise Generation Algorithms

A noise generation algorithm refers to a systematic way to create values in $\boldsymbol{x}_{\mathrm{CS}}$. Examples of images with injected noise are shown in Figure 4.5.

Replaying. Suppose $\boldsymbol{x}'_{\mathrm{SC}}$ is the recorded side-channel signal of a printing process that


(a) No Protection (b) Replaying (c) Random Blobs (d) White Noise

(e) Full Power (f) Channel Uniform. (g) State Uniformization (h) State Randomization

Figure 4.5: Example images with noise generated by different noise generation algorithms.

is different from the current printing process. We let $\boldsymbol{x}_{\text{CS}} = \boldsymbol{x}'_{\text{SC}}$. In other words, we replay the side-channel signal of another printing process [19].

Random Blobs. This method fills the pixels in $\boldsymbol{x}_{\text{CS}}$ with a circle that has a random location, a random size, and a random (uniform) color. The blob is maintained for a certain period of time and changes to another random blob.

White Noise. This method uses a random number generator (of a uniform distribution) to independently fill all values in the control signal $\boldsymbol{x}_{\text{CS}}$.

Full Power. This method creates $\boldsymbol{x}_{\text{CS}}$ such that the optical projector constantly outputs the maximum power. To be specific, all values in $\boldsymbol{x}_{\text{CS}}$ are filled with their maximum numbers. This method attempts to blind the attacker's camera.

4.4.3 Proposed Noise Generation Algorithms

In this section, we propose three novel noise generation algorithms. These algorithms rely on the information of the printing process and hence are print-specific.

Channel Uniformization. This method attempts to make $\boldsymbol{x}_{\text{SC}}$ a constant over the channel index c. In other words, this method creates a control signal such that each picture seen by the camera is of a uniform color. In this way, it becomes hard for the attacker to determine the location of the nozzle.

For a specific state (or a specific value of $\boldsymbol{x}_{\mathrm{SV}}[n]$), if the intersection of the ranges of $\boldsymbol{x}_{\mathrm{SC}}[n,c]$ (with respect to $\boldsymbol{x}_{\mathrm{CS}}[n,c]$) for all c is not empty, we can select any value in this intersection as the constant value $C[n]$. We then create $\boldsymbol{x}_{\mathrm{CS}}[n,c]$ by solving

$$\boldsymbol{g}_c(\boldsymbol{x}_{\mathrm{SV}}[n], \boldsymbol{x}_{\mathrm{CS}}[n,c]) = C[n]. \tag{4.5}$$

$\boldsymbol{x}_{\mathrm{CS}}[n]$ is simply a collection of $\boldsymbol{x}_{\mathrm{CS}}[n,c]$ for all c. Both $C[n]$ and $\boldsymbol{x}_{\mathrm{CS}}[n]$ depends on the specific value of $\boldsymbol{x}_{\mathrm{SV}}[n]$.

If the aforementioned intersection is empty, we select a constant value $C[n]$ that is likely to be reached by most $\boldsymbol{x}_{\mathrm{SC}}[n,c]$ (as c is varied). Since $\boldsymbol{x}_{\mathrm{CS}}$ typically makes $\boldsymbol{x}_{\mathrm{SC}}$ brighter, we can select $C[n]$ by

$$C[n] = h(\{\boldsymbol{x}_{\mathrm{SC}}[n,c] | c\}), \tag{4.6}$$

where h is a function to find a large value of a set, such as the 99th percentile of the set. We do not use the maximum value of the set because it may be susceptible to outliers.

State Uniformization. When the state variable ($\boldsymbol{x}_{\mathrm{SV}}[n]$) changes, the image seen by the camera ($\boldsymbol{x}_{\mathrm{SC}}[n]$) changes. It is this relationship that makes it possible to infer the state variable from the image. This method creates a control signal ($\boldsymbol{x}_{\mathrm{CS}}$) in an attempt to make the image ($\boldsymbol{x}_{\mathrm{SC}}[n]$) a constant as the state variable ($\boldsymbol{x}_{\mathrm{SV}}[n]$) changes. In other words, $\boldsymbol{x}_{\mathrm{SC}}[n]$ as a function of $\boldsymbol{x}_{\mathrm{SV}}[n]$ is uniformized.

For a specific channel index c, if the intersection of the ranges of $\boldsymbol{x}_{\mathrm{SC}}[n,c]$ (with respect to $\boldsymbol{x}_{\mathrm{CS}}[n,c]$) for all values of $\boldsymbol{x}_{\mathrm{SV}}[n]$ is not empty, we can select any value in this intersection as the constant value $C[n,c]$. We then create $\boldsymbol{x}_{\mathrm{CS}}[n,c]$ by solving

$$\boldsymbol{g}_c(\boldsymbol{x}_{\mathrm{SV}}[n], \frac{\partial \boldsymbol{x}_{\mathrm{SV}}}{\partial t}[n], \cdots, \boldsymbol{x}_{\mathrm{CS}}[n,c]) = C[n,c]. \tag{4.7}$$

$\boldsymbol{x}_{\mathrm{CS}}[n]$ is simply a collection of $\boldsymbol{x}_{\mathrm{CS}}[n, c]$ for all c. Both $C[n, c]$ and $\boldsymbol{x}_{\mathrm{CS}}[n, c]$ are constants with respect to $\boldsymbol{x}_{\mathrm{SV}}[n]$.

If the aforementioned intersection is empty, we select a constant value $C[n, c]$ that is likely to be reached by most $\boldsymbol{x}_{\mathrm{SC}}[n, c]$ (as $\boldsymbol{x}_{\mathrm{SV}}[n]$ is varied). Since $\boldsymbol{x}_{\mathrm{CS}}$ typically makes $\boldsymbol{x}_{\mathrm{SC}}$ brighter, we can select $C[n, c]$ by

$$C[n, c] = h(\{\boldsymbol{x}_{\mathrm{SC}}[n, c] | \boldsymbol{x}_{\mathrm{SV}}[n]\}), \tag{4.8}$$

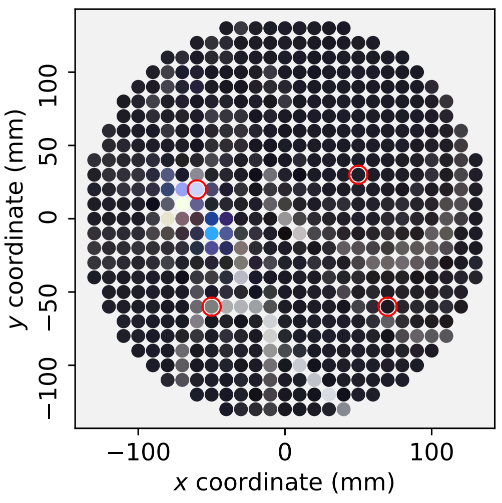
where h is a function to find a large value of a set, such as the 99th percentile of the set.

Figure 4.6 (a) shows the color of a single pixel in $\boldsymbol{x}_{\mathrm{SC}}[n]$ as a function of the first two coordinates of $\boldsymbol{x}_{\mathrm{SV}}[n]$. For each value of $\boldsymbol{x}_{\mathrm{SV}}[n]$ or each dot in Figure 4.6 (a), as the control signal for the specified pixel sweeps all of its possible values, the range of the color of the specified pixel in $\boldsymbol{x}_{\mathrm{SC}}[n]$ forms a color space. Figure 4.6 (b) shows the color spaces for four different states corresponding to the four red circles Figure 4.6 (a). We can see that the intersection of the color spaces is empty.

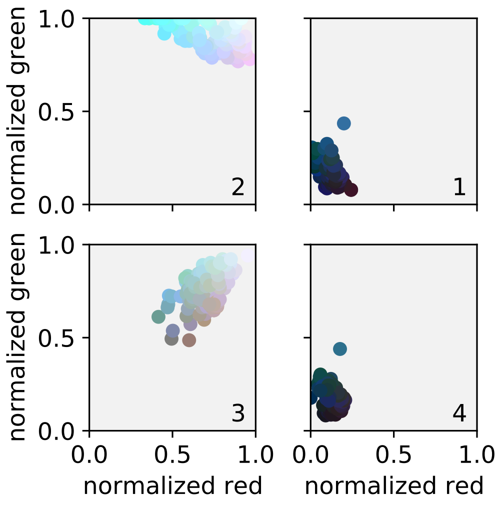
Figure 4.6 (c) shows the color of the corresponding pixel in $\boldsymbol{x}_{\mathrm{CS}}[n]$ obtained by state uniformization and Figure 4.6 (d) shows the color of the specified pixel in $\boldsymbol{x}_{\mathrm{SC}}[n]$ after the obtained $\boldsymbol{x}_{\mathrm{CS}}[n]$ is applied. We can see that the pattern is more uniform, and it may be harder for the attacker to infer (x, y) based on the observed color at this specific pixel.

State Randomization. This method attempts to randomize the relationship between the image ($\boldsymbol{x}_{\mathrm{SC}}[n]$) and the state ($\boldsymbol{x}_{\mathrm{SV}}[n]$). To be specific, we create a control signal ($\boldsymbol{x}_{\mathrm{CS}}$) such that the image ($\boldsymbol{x}_{\mathrm{SC}}[n]$) for a specific state ($\boldsymbol{x}_{\mathrm{SV}}[n]$) appears to be the image ($\boldsymbol{x}_{\mathrm{SC}}[n]$) for another random state ($\boldsymbol{x}_{\mathrm{SV}}[n]$). Since the relationship between $\boldsymbol{x}_{\mathrm{SC}}[n]$ and $\boldsymbol{x}_{\mathrm{SV}}[n]$ is disrupted, we can expect that it will be hard for the attacker to learn the true relationship between $\boldsymbol{x}_{\mathrm{SC}}[n]$ and $\boldsymbol{x}_{\mathrm{SV}}[n]$.

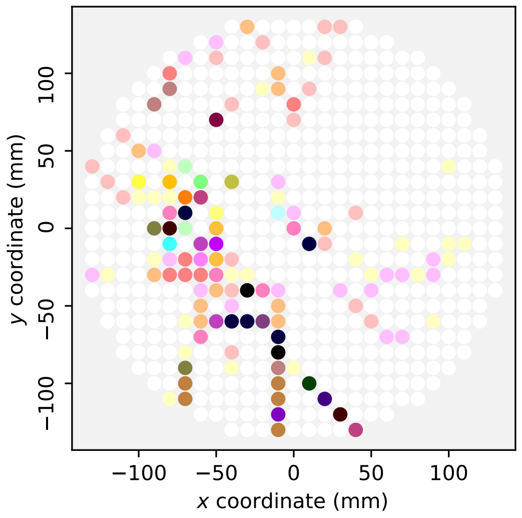
Since a projector can only make pixels in $\boldsymbol{x}_{\mathrm{SC}}$ brighter, we apply a constant value on all pixels in $\boldsymbol{x}_{\mathrm{CS}}[n]$ for all states (or all values of $\boldsymbol{x}_{\mathrm{SV}}[n]$). This constant value is referred to as



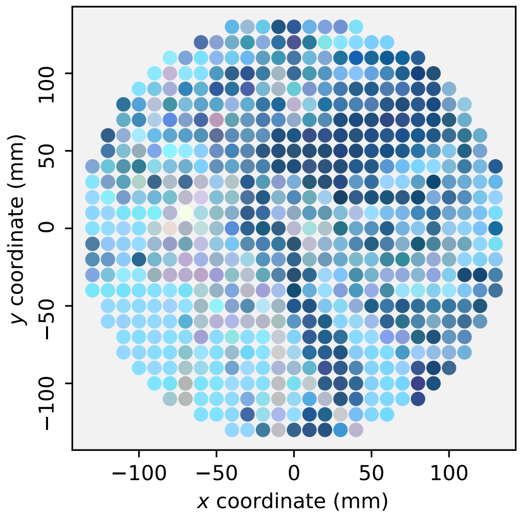
(a) Side-Channel Signal



(b) Color Spaces



(c) Control Signal



(d) Side-Channel Signal

Figure 4.6: Illustration of the state uniformization method. (a) A pixel of $\boldsymbol{x}_{\mathrm{SC}}[n]$ as a function of $\boldsymbol{x}_{\mathrm{SV}}[n, 0 : 2]$. (b) Color spaces for four different states. The four color spaces approximately correspond to the four states with red circles in (a). (c) The control signal obtained by the state uniformization method. (d) A pixel of $\boldsymbol{x}_{\mathrm{SC}}[n]$ as a function of $\boldsymbol{x}_{\mathrm{SV}}[n, 0 : 2]$ after the control signal is applied.

the offset. By using the offset, it is now possible to make pixels $\boldsymbol{x}_{\mathrm{SC}}$ dimmer by reducing the corresponding pixels in $\boldsymbol{x}_{\mathrm{CS}}[n]$ with respect to the offset. We express the strength of the offset as the percentage of the maximum value accepted by an pixel in $\boldsymbol{x}_{\mathrm{CS}}[n]$. By default, we apply an offset with a strength of 50%.

We now discretize the domain of $\boldsymbol{x}_{\mathrm{SV}}[n]$ such that there is a finite number of states. By default, we discretize the domain such that all states are equally spaced. These states are referred to as master states and they serve as the targets to be matched.

For any new state $\boldsymbol{x}_{\mathrm{SV}}[n]$, we randomly select one of the master states as the target state. Suppose the target state is $\boldsymbol{x}'_{\mathrm{SV}}[n]$ and its corresponding side-channel signal is $\boldsymbol{x}'_{\mathrm{SC}}[n]$. We create $\boldsymbol{x}_{\mathrm{CS}}[n]$ such that $\boldsymbol{x}_{\mathrm{SC}}[n]$ gets close to $\boldsymbol{x}'_{\mathrm{SC}}[n]$. To be specific, we find $\boldsymbol{x}_{\mathrm{CS}}[n]$ by determining its value for each channel independently, and $\boldsymbol{x}_{\mathrm{CS}}[n,c]$ is equal to

$$\underset{\boldsymbol{x}_{\mathrm{CS}}[n,c]}{\operatorname{argmin}} \left|\boldsymbol{g}_c(\boldsymbol{x}_{\mathrm{SV}}[n], \boldsymbol{x}_{\mathrm{CS}}[n,c]) - \boldsymbol{x}'_{\mathrm{SC}}[n,c]\right|. \tag{4.9}$$

We repeat this process for all channels and we can obtain $\boldsymbol{x}_{\mathrm{CS}}[n]$. In this way, the optical side-channel signal observed by the attacker, namely $\boldsymbol{g}(\boldsymbol{x}_{\mathrm{SV}}[n], \cdots, \boldsymbol{x}_{\mathrm{CS}}[n])$, will look like $\boldsymbol{x}'_{\mathrm{SC}}[n,c]$. When the attacker analyzes $\boldsymbol{x}_{\mathrm{SC}}[n]$ (with injected noise), he or she will more likely infer $\boldsymbol{x}'_{\mathrm{SV}}[n]$ instead of $\boldsymbol{x}_{\mathrm{SV}}[n]$.

This method is effectively a combination of channel uniformization and an advanced version of replaying[2]. It attempts to hide the real printhead by channel uniformization and creates a fake printhead as best as the projector allows.

[2]The original replaying method simply uses the recorded side-channel signal as the control signal. The advanced replaying method finds a control signal such that the side-channel signal with the injected noise looks like the recorded side-channel signal.

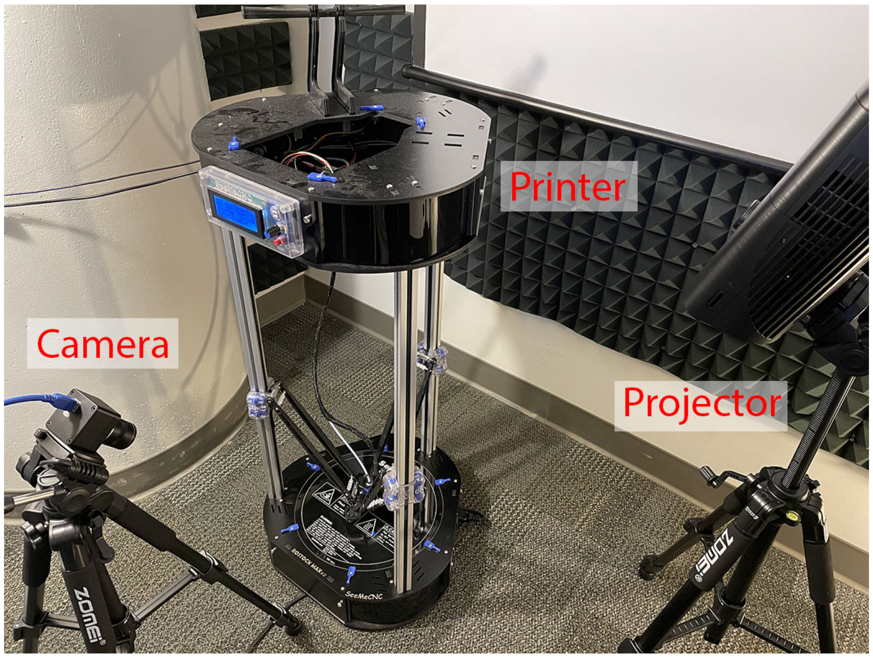


Figure 4.7: Photo of the testbed.

4.5 Evaluation

4.5.1 Experiment Setup

To evaluate the performance of the proposed optical side-channel attack and the proposed defense method, we set up a test bed as shown in Figure 4.7. The test bed was composed of a SeeMeCNC Rostock Max V3 printer, a MOKOSE UC70 camera, and a ViewSonic Pro7827HD projector. The whole test bed was placed in a dedicated room with fully controlled lights (two lamps and no window).

Settings. To ensure the consistency of the images, we disabled auto white balance, auto exposure, and auto focus in the camera. We manually set the white balance to 4600 K and the exposure to -2. The focus length of the camera was mechanically adjusted to clearly show the build plate of the printer. We used default settings for all other parameters in the camera and all parameters in the projector.

Drivers. We used VLC for collecting images and VLC internally used *ffmpeg* to communicate with the camera. The main advantage of this approach is that images can be taken very quickly as they are extracted from the video stream from the camera. We used *ffmpeg* directly to record videos and used *imageio-ffmpeg* for decoding the recorded videos. *imageio-ffmpeg* also internally used *ffmpeg* as the video driver.

4.5.2 Optical Side-Channel Attack

Training Dataset. We collected the training dataset according to the procedures outlined in subsection 4.3.3. The printing area of the printer was a cylinder of 137.5 mm in radius and 404 mm in height. Since most printing processes only involve a fraction of the whole printing space, to save time, we mainly focus on the printing area where z is less than or equal to 5 mm. There are two parts of the training dataset.

- For the first part, we instructed the printhead to sweep the whole xy plane with a step size of 2.5 mm and the z axis with a step size of 1.0 mm. This part covers a large printing space with a coarse resolution. There are 133,365 images in this part.
- For the second part, we instructed the printhead to sweep the xy plane for a radius of $r = 50$ mm with a step size of 1.0 mm and the z axis with a step size of 0.25 mm. This part covers a small printing space with a high resolution. There are 47,385 images in this part.

There are 180,750 images in the training dataset.

Each image is of size 1280x720 and we need to downsample the images to 227x227 before feeding them into the neural network. To enhance the robustness of the neural network, a standard practice is to augment the dataset while downsampling the images [61]. For each image in the dataset, we perform the following procedures:

1. Resize the image into a shape of 480x270.
2. Randomly rotate the image by -3 to 3 degrees.
3. Scale the image randomly by a factor of 0.85 to 1.0.

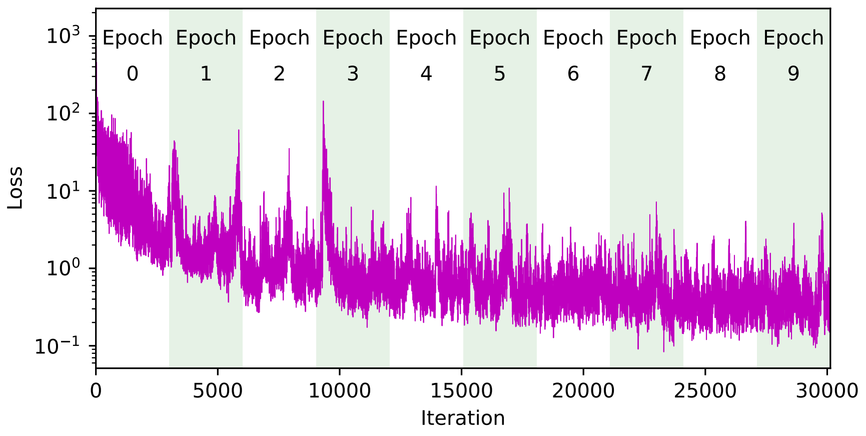


Figure 4.8: Loss as a function of iteration or epoch.

4. Randomly crop the image with a size of 227x227.

5. Randomly change the brightness, contrast, saturation, and hue by 20%, 10%, 10%, and 5% respectively.

Training Process. We started with a ResNet50 network that was pretrained with the ImageNet dataset. We performed training on an RTX 2060 GPU with 6 GB of memory. We used a mini batch size of 40 because it filled up most of the available memory. We used the Mean Square Error (MSE) as the loss function and we used the Adam algorithm [60] for training. The initial learning rate was set to 0.001 and the weight decay was set to 0.0005. For every two epochs, we reduced the learning rate by 10%. We performed the training process for 10 epochs. The loss as a function iteration/epoch is shown in Figure 4.8. We can see that the error converged and the training process was successful.

Testing Datasets. By the procedures in subsection 4.3.3, we collected two testing datasets for a printing process that manufactures a gear, as shown in Figure 4.9 (a). To avoid multiple superimposed layers when displaying the results, we focus on a single layer ($z = 1.3$ mm) for the printing process. Figure 4.9 (b) shows the ground truth of the printing path at this layer.

Testing Results. Figure 4.9 (c) shows the recovered path for the first testing dataset (a contrived printing process with ground truth labels for calculating errors). The average

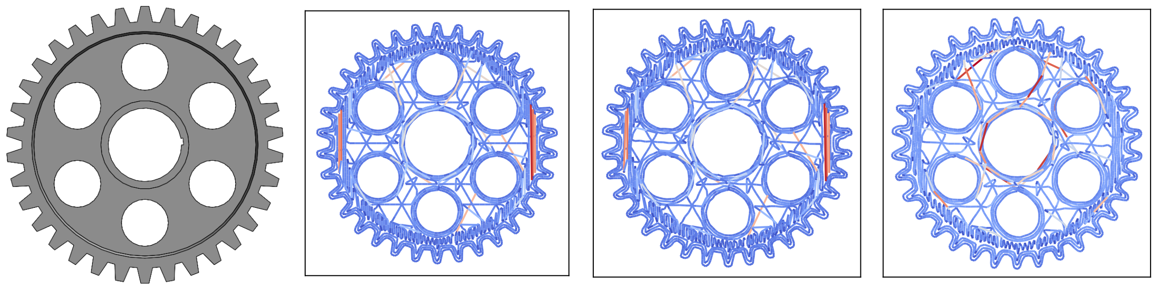

(a) Gear Model (b) Ground Truth (c) Testing Dataset 1 (d) Testing Dataset 2

Figure 4.9: Model and Paths recovered by the neural network for different testing datasets. (a) The gear model. (b) The ground truth path. (c) The recovered path for testing dataset 1. (d) The recovered path for testing dataset 2.

error is 0.71 mm and the maximum error is 1.50 mm. The recovered path is very close to the ground truth. Figure 4.9 (d) shows the recovered path for the second testing dataset (a real printing process without ground truth labels). The shape of the path is very similar to the ground truth. This means that the motion blur effect (as a side effect in the second testing dataset) can be neglected in our experiments.

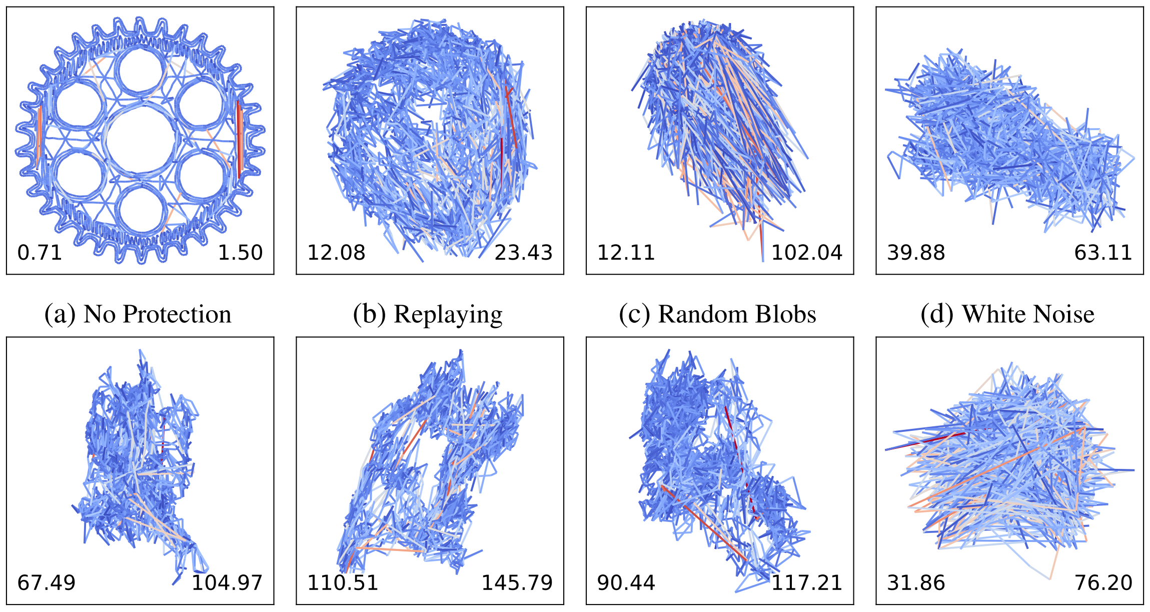
4.5.3 Defending Against the Naive Attacker

We applied the seven noise generation algorithms to the printing process, one at a time, as demonstrated in Figure 4.5. Since the naive attacker was not aware of the injected noise, for each noise generation algorithm, the attacker used the previously trained neural network to directly test on the protected testing dataset[3]. The results and the errors are shown in Figure 4.10. We can see that all noise generation algorithms could effortlessly defeat the attack since the recovered paths are nothing like the true path.

4.5.4 Defending Against the Advanced Attacker

An advanced attacker knows the existence of the injected noise and attempts to defeat the protection method. For this purpose, the advanced attacker collects training images with injected noise and uses the new training dataset to train the neural network, hoping that

[3]To obtain the performance metrics, we collected the protected testing datasets using the same procedure to collect the training dataset or the first testing dataset.



(a) No Protection (b) Replaying (c) Random Blobs (d) White Noise

(e) Full Power (f) Channel Uniform. (g) State Uniformization (h) State Randomization

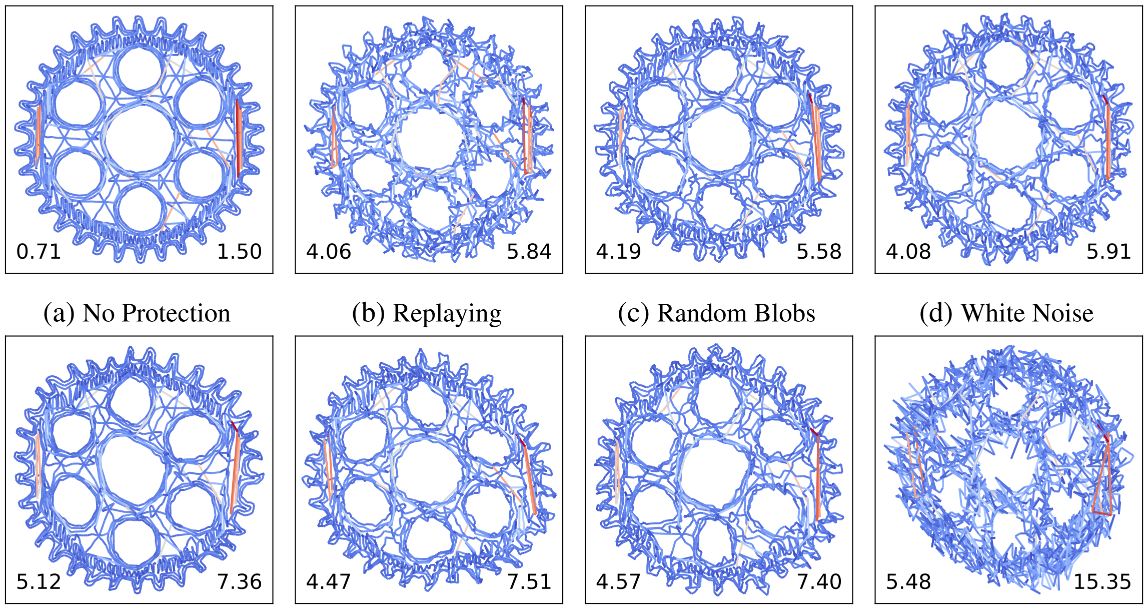
Figure 4.10: Recovered path by a naive attacker when noise generated by various algorithms was applied. The number on the left in each sub-figure shows the mean error in mm whereas the number on the right in each sub-figure shows the maximum error in mm.

the neural network can recognize the noise pattern, reject the injected noise, and properly recover the coordinates in the protected testing images.

To be specific, for each noise generation algorithm, the attacker collects training images with injected noise. There are two potential ways to collect such training images.

- The attacker may setup a testbed with a projector, imitate the noise generation algorithm, and collect the training images at various locations of the nozzle.
- The attacker may extract images with injected noise from previous printing processes and somehow, although very hard, manage to obtain the correct label for each image.

Eventually, the attacker obtains a training dataset that contains injected noise generated by the same algorithm. For noise generation algorithms that contain randomness, such as replaying, random blobs, white noise, and state randomization, the details of the injected noise in the training dataset are different from those in the printing process to be recovered (the protected testing dataset), such as locations and sizes for random blobs, target states



(a) No Protection (b) Replaying (c) Random Blobs (d) White Noise

(e) Full Power (f) Channel Uniform. (g) State Uniformization (h) State Randomization

Figure 4.11: Recovered path by an advanced attacker when noise generated by various algorithms was applied. The number on the left in each sub-figure shows the mean error in mm whereas the number on the right in each sub-figure shows the maximum error in mm.

for replaying and state randomization, and the exact distribution of pixels for white noise. A neural network that is overfitted will not succeed in rejecting the noise, and the neural network has to learn the pattern of the injected noise and reject unseen noise that has the same pattern but with different details.

For each noise generation algorithm, due to the high cost of obtaining a training dataset with injected noise, we collected a training dataset with injected noise covering the whole xy plane with a step size of 5 mm and the z axis for a single point, namely z = 5 mm. The advanced attacker used the new training dataset with injected noise to further train the previously trained neural network for additional 30 epochs, starting with a learning rate of 0.001, which was decreased by 10% for every two epochs. The advanced attacker then used the new neural network to infer coordinates in images from the protected testing dataset. The results are shown in Figure 4.11.

We can see that the advanced attacker did a much better job than the naive attacker in recovering the state-variable signal from the optical side-channel signal. Although the errors in Figure 4.11 (b) to (g) are higher than those in Figure 4.11 (a), a lot of details were clearly recovered such as the number of teeth in the gear, the overall dimension of the gear, and the infill pattern. In Figure 4.11 (h), we can see that state randomization algorithm is the only noise generation algorithm that could withstand the advanced side-channel attack.

4.6 Discussion

4.6.1 Colors in the Paths

One may notice that the color in Figure 4.9 (a) and the color in Figure 4.9 (b) do not match. The colors in the paths show the speed for each movement. A blue color corresponds to a lower printing speed whereas a red color corresponds to a higher printing speed. The colors in the paths do not match because we used a simulator to determine the speed for each movement in the ground truth (and the first testing dataset), whereas the speed in the second testing dataset was determined by the firmware in the printer. Currently, there is a noticeable error between the simulated velocity and the actual velocity.

4.6.2 Performance of the Attackers

Naive Attacker. When no noise was injected, the naive attacker could do a pretty good job in recovering the coordinates from the optical side-channel signal. Although the neural network was trained on a dataset where the nozzle was located in a limited number of locations, the neural network could properly identify the coordinates of the nozzle for a location that was not present in the training dataset. In other words, the neural network could intelligently interpolate with respect to the coordinates of the nozzle. When noise was injected, the naive attacker failed all together. This indicates that the neural network was unable to analyze images with patterns that it had never seen before.

Advanced Attacker. According to experiment results by the advanced attacker, by seeing a limited number of images with injected noise, the neural network was able to filter out the injected noise and proceed with recognizing the coordinates of the nozzle, even though at a reduced accuracy. Notice that for noise generation algorithms with randomness, the details of the injected noise in the training dataset were different from those in the protected testing dataset, but the neural network could still recognize and reject the injected noise. This proves that the neural network was not simply remembering the details of the injected noise but intelligently learned the pattern of the injected noise.

4.6.3 Performance of the Noise Generation Algorithms

According to the evaluation results, only the state randomization algorithm could fight against the advanced attacker. One way to intuitively understand the performance of different noise generation algorithms is to visually inspect the images after the noise is injected. For images in Figure 4.5 (b) to (g), a human being can in general see the location of the nozzle despite the existence of the injected noise.

Full power, channel uniformization, and state uniformization were unable to protect the intellectual property because of the limited capability of the projector. We can expect that, as the projector gets more powerful, the full power noise will blind the camera and the uniformization methods will completely hide the details.

The replaying method failed mainly due to the fact that it did not consider the channel effect caused by the projector, the physical space, and the projector. When a recorded video is directly fed to a projector, the projected motion picture will look different from the real thing. In other words, a human being is able to differentiate the projected nozzle from the real nozzle in Figure 4.5 (b).

The state randomization algorithm had the best performance due to the fact that it is a combination of an improved version of the replaying method and the state uniformization algorithm. Instead of directly feeding a recorded side-channel signal to the control signal, the state randomization algorithm searches for the control signal such that the projected nozzle looks closest to a real nozzle. In addition, the state randomization algorithm attempts to hide the real nozzle similar to what is attempted by the state uniformization algorithm.

Table 4.1: Recorders and Generators

Side Channels	Sensors	Generators
Acoustic	Microphone	Speaker
Optical	Camera	Projector
Thermal	Temperature Sensor	Heater
Magnetic	Magnetometer	Coil
Power	Power Sensor	Resistor
Electromagnetic	Antenna	Antenna
Vibration	Accelerometer	Motor

4.6.4 Generalization to Other Side Channels

The noise injection method may be generalized to other types of side channels, as listed in Table 4.1. For each type of side channels, there is a corresponding signal generator, as demonstrated in Table 4.1. Most of the noise generation algorithms can be directly applied to other side channels. The only exception is the random blobs generation algorithm, which is only applicable to the optical side channel. Our hypothesis is that noise generation algorithms that are effective in fighting against the optical side-channel attack will be effective in fighting against other side-channel attacks. We propose this hypothesis because the optical side-channel attack is an attack that can be easily launched but hard to defend. Many other side-channel attacks cannot be performed at all unless a lot of restrictive assumptions are made. More details can be found in section 4.7.

4.6.5 Limitations and Future Work

Motion Blur Effect. In the experiments, we were able to ignore the motion blur effect by using a camera with a fast shutter speed. However, when the light in the AM facility is low, a camera will suffer from a low shutter speed and we can no longer ignore the motion blur effect. To mitigate this problem, as a direction for future work, we can create a training dataset where the label of each image contains not only the position but also the velocity. The main challenge is that a lot more samples are required as the degrees of freedom are now six instead of three. In addition, it is very hard to precisely control the position and the velocity at the same time for each training image.

Camera Positions and Angles. In the experiments, the position and angle of the camera largely remained stable. The neural network was able to deal with a small amount of variation in the position and angle of the camera. It will be interesting to know if it is possible to train a single neural network that can recognize the coordinates of the printhead from multiple very different positions and angles of the camera, especially from a position or an angle that does not appear in the training dataset.

Large Objects. In the experiments, the printed object was small in size and was completely blocked by the printhead. As a result, we were able to ignore the influence of deposited materials on the side-channel attack. When printing a large object, the deposited materials may affect the side-channel attack. Future work can study how the deposited materials affect the performance of both the native attacker and the advanced attacker.

4.7 Additional Discussion

4.7.1 Acoustic Side-Channel Attack

In section 4.1, we mentioned that the acoustic side-channel attack is hard to perform. The challenges associated with the acoustic side-channel attack are listed as follows.

Integration Drift. The acoustic side-channel signal is strongly correlated with the velocity of the printhead, not the position of the printhead. As a result, an attacker can only recover the velocity of the printhead from the acoustic side-channel signal. To obtain the position, the attacker must integrate the estimated velocity over time. Since the estimated velocity usually contains a lot of errors, there will be more errors in the estimated position. This problem is known as the integration drift problem [62]. This problem is more pronounced if one wants to recover position from acceleration.

Non-Unique Solution. Multiple states of the printer can generate the same side-channel signal. As a result, it becomes very hard to determine the exact state for a given side-channel signal. For example, the acoustic signals for a printhead moving along one direction and its opposite direction are almost identical.

Due to these challenges, it is very hard to perform the acoustic side-channel attack. Existing acoustic side-channel attacks rely on a lot of assumptions to work, and as a consequence, they only work for a very limited amount of contrived printing processes. For example, the attack in [18, 20] restricts the directions of movements to be eight cardinal directions as a way to reduce the search area. The attack in [18] further assumes that multiple layers have identical outlines and the recovered paths are averaged across layers to reach a reasonable accuracy. The attack in [19] assumes that there are no short and rapid movement. Otherwise, it is not possible to identify the boundaries between movements.

In contrast, the optical side-channel attack does not face these problems and can recover the printing path for an arbitrary printing process. This is the primary reason we focus on the optical side-channel attack in this thesis.

4.7.2 Infrared Side-Channel Attack

The authors in [32] claimed that their infrared side-channel attack failed due to the low resolution, the low frame rate, the lack of auto-focus capability, and the fixed camera perspective. However, it is unlikely that these reasons were accountable for the failure because our optical side-channel attack faced the same problem and we had a low resolution (227x227). It is more likely that a root cause of the failure was a lack of proper methodology. There is a great potentially to adapt our optical side-channel attack to the infrared side-channel attack and we may even recover more information such as the temperature(s) of the nozzle(s) and the temperature of the build plate.

4.8 Conclusion

In this chapter, we discussed the optical side-channel attack to steal intellectual property in AM systems. The intellectual property is operationally defined as the coordinates of the printhead as a function of time, a.k.a. the state-variable signal in this thesis. We use a deep neural network to implement the optical side-channel attack. The neural network takes as input an image of the printer and outputs the estimated coordinates of the printhead. The neural network can tolerate a certain level of variation in the camera's position and angle as well as lighting conditions. The neural network contains a certain level of intelligence because it can accurately determine the coordinates of the printhead for an image that is not present in the training dataset.

We experimented with the noise injection method to defend against the proposed optical side-channel attack. We found that any noise generation algorithm could easily defeat an naive attacker, who had no knowledge of the defense method and the trained neural network was never exposed to any image with injected noise. However, an advanced attacker, who knew the existence of the defense method and attempted to filter out the injected noise by adding images with injected noise in the training dataset, could easily defeat existing noise generation algorithms, such as replaying, random blobs, white noise, and full power. To solve this problem, we proposed three novel noise generation algorithms and they are channel uniformization, state uniformization, and state randomization. Our experiment results indicate that only the state randomization algorithm could withstand the attack by an advanced attacker.

CHAPTER 5
CONCLUSION

In this thesis, we mainly studied cybersecurity in Additive Manufacturing (AM) systems with side channels. On the one hand, side channels in AM systems can be used by Intrusion Detection System (IDS) to protect AM systems from being attacked. On the other hand, side channels in AM systems can be maliciously used by attackers to perform side-channel attacks to steal intellectual property in AM systems.

5.1 The NSYNC Framework

We found that existing IDSs leveraging side channels in AM systems are not practical due to a lack of synchronization. On the one hand, side-channel signals to be compared must be aligned at their starting moments before they are compared. On the other hand, there is time noise in a printing process, which is the random variation of timing of instructions in the printing process. If not compensated for, time noise can render signal comparison meaningless. Unfortunately, only a few existing IDSs explicitly provide solutions to align signals at their starting moments and no existing IDS is aware of time noise and they all fail over a long period of time.

To solve the aforementioned problems, we propose NSYNC, a framework of practical IDSs leveraging side-channel signals in AM systems. The most important feature of NSYNC is the inclusion of the static synchronizer and the dynamic synchronizer. The static synchronizer is responsible for aligning side-channel signals to be compared at the starting moments and terminating the signal comparison process at the stopping moments. The dynamic synchronizer is responsible for continuously finding corresponding points or windows in two signals with time noise. Our experiment results indicate that our static synchronizer can find the specified moments (mainly the starting and stopping moments) accurately and efficiently. Our experiment results also indicate that existing intrusion detection systems which are not aware of the existence of time noise have a very limited performance whereas our NSYNC framework with Dynamic Window Matching (DWM) can effortlessly reach an accuracy over 99%.

Other than the synchronization problem, some existing IDSs are not practical due to a lack of practical discriminators, which are responsible for automatically issuing alerts based on the signal comparison results. To be specific, some IDSs lack this component altogether and human efforts are required to manually view the signal comparison results to issue alerts. Some IDSs have discriminators but they require knowing malicious printing processes in advance to determine the thresholds for classifying benign printing processes and malicious printing processes. However, it is impractical to know malicious printing processes in advance as there are countless malicious printing processes in the wild. To avoid this problem, our NSYNC framework uses One-Class Classification (OCC) to build the discriminator. Our experiment results show that the discriminator can reliably differentiate malicious printing processes from benign printing processes when DWM is used as the dynamic synchronizer.

5.2 Optical Side Channel: Attack and Defense

In addition to the NSYNC framework, another big area involved in this thesis is the optical side-channel attack as well as methods to defend against the optical side-channel attack. The optical side-channel attack uses a deep neural network to convert images in a printing process to the state variable (mainly the position of the printhead) of the printer over time. The state variable of the printer over time, also known as the state-variable signal, can be used to reconstruct the printing process. Our experiment results indicate that our proposed optical side-channel attack can successfully recover the path of a printing process despite various challenges, such as limited relationship between the side-channel signal and the state-variable signal, variations in the camera properties and lighting conditions, the motion blur effect, and the influence of deposited materials.

There are a variety of methods to defend against the proposed optical side-channel attack and we explicitly focus on the noise injection method, which does not require any modification to the hardware or software of the AM system and has a very low overhead. To properly use the noise injection method, an algorithm is required to generate the noise. We found that existing noise generation algorithms did not work well for the optical side-channel attack, and we proposed three novel noise generation algorithms. The state randomization algorithm worked very well and could successfully defend against an advanced attacker, who knew the defense method in advance and attempted to circumvent the defense method by training a neural network with images containing the injected noise.

REFERENCES

[1] S. Curran *et al.*, “Big area additive manufacturing and hardware-in-the-loop for rapid vehicle powertrain prototyping: A case study on the development of a 3-D printed shelby cobra,” *SAE Technical Paper*, 2016.

[2] R. Liu, Z. Wang, T. Sparks, F. Liou, and J. Newkirk, “Aerospace applications of laser additive manufacturing,” in *Laser Additive Manufacturing*, ser. Electronic and Optical Materials, M. Brandt, Ed., Woodhead Publishing, 2017, pp. 351–371, ISBN: 978-0-08-100433-3.

[3] D. D. Camacho *et al.*, “Applications of additive manufacturing in the construction industry – a forward-looking review,” *Automation in Construction*, vol. 89, pp. 110–119, 2018.

[4] D. A. Zopf, S. J. Hollister, M. E. Nelson, R. G. Ohye, and G. E. Green, “Bioresorbable airway splint created with a three-dimensional printer,” *New England Journal of Medicine*, vol. 368, no. 21, pp. 2043–2045, 2013, PMID: 23697530.

[5] W. Terry, C. Robert, H. Ray, D. Olaf, and K. Joseph, *Wohlers Report 2019: 3D Printing and Additive Manufacturing State of the Industry*. Wohlers Associates, 2019, 369 pp., ISBN: 978-0-9913332-5-7.

[6] S. Moore, P. Armstrong, T. McDonald, and M. Yampolskiy, “Vulnerability analysis of desktop 3D printer software,” in *2016 Resilience Week (RWS)*, Aug. 2016, pp. 46–51.

[7] Q. Do, B. Martini, and K. R. Choo, “A data exfiltration and remote exploitation attack on consumer 3D printers,” *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 10, pp. 2174–2186, Oct. 2016.

[8] S. B. Moore, W. B. Glisson, and M. Yampolskiy, “Implications of malicious 3D printer firmware,” in *Proceedings of the 50th Hawaii International Conference on System Sciences*, Honolulu, HI, 2017, pp. 6089–6098.

[9] L. D. Sturm, C. B. Williams, J. A. Camelio, J. White, and R. Parker, “Cyber-physical vulnerabilities in additive manufacturing systems: A case study attack on the .STL file with human subjects,” *Journal of Manufacturing Systems*, vol. 44, no. Part 1, pp. 154–164, 2017.

[10] M. Yampolskiy, A. Skjellum, M. Kretzschmar, R. A. Overfelt, K. R. Sloan, and A. Yasinsac, “Using 3D printers as weapons,” *International Journal of Critical Infrastructure Protection*, vol. 14, pp. 58–71, 2016.

[11] S. Belikovetsky, M. Yampolskiy, J. Toh, J. Gatlin, and Y. Elovici, "Dr0wned – cyber-physical attack with additive manufacturing," in *11th USENIX Workshop on Offensive Technologies (WOOT 17)*, Vancouver, BC: USENIX Association, 2017.

[12] S. Bradshaw, A. Bowyer, and P. Haufe, "The intellectual property implications of low-cost 3D printing," p. 27, 2010.

[13] M. Weinberg, *It will be awesome if they don't screw it up: 3D printing, intellectual property, and the fight over the next great disruptive technology*, Nov. 2010.

[14] P. Wright, *The next big fight: 3D printing and intellectual property*, Jan. 2014.

[15] M. Yampolskiy, T. R. Andel, J. T. McDonald, W. B. Glisson, and A. Yasinsac, "Intellectual property protection in additive layer manufacturing: Requirements for secure outsourcing," in *Proceedings of the 4th Program Protection and Reverse Engineering Workshop on 4th Program Protection and Reverse Engineering Workshop (PPREW 4)*, New Orleans, LA, USA: ACM Press, 2014, pp. 1–9, ISBN: 978-1-60558-637-3.

[16] J. F. Hornick and K. Rajan, "Chapter 16 - intellectual property in 3D printing and nanotechnology," in *3D Bioprinting and Nanotechnology in Tissue Engineering and Regenerative Medicine*, L. G. Zhang, J. P. Fisher, and K. W. Leong, Eds., Academic Press, 2015, pp. 349–364, ISBN: 978-0-12-800547-7.

[17] B. Krassenstein, *5 different 3D printed gun models have been fired since may, 2013*, 2014.

[18] M. A. Al Faruque, S. R. Chhetri, A. Canedo, and J. Wan, "Acoustic side-channel attacks on additive manufacturing systems," in *Proceedings of the 7th International Conference on Cyber-Physical Systems*, ser. ICCPS '16, Vienna, Austria: IEEE Press, 2016, 19:1–19:10.

[19] A. Hojjati *et al.*, "Leave your phone at the door: Side channels that reveal factory floor secrets," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16, Vienna, Austria: ACM, 2016, pp. 883–894, ISBN: 978-1-4503-4139-4.

[20] C. Song, F. Lin, Z. Ba, K. Ren, C. Zhou, and W. Xu, "My smartphone knows what you print: Exploring smartphone-based side-channel attacks against 3D printers," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16, Vienna, Austria: ACM, 2016, pp. 895–907, ISBN: 978-1-4503-4139-4.

[21] S. R. Chhetri, A. Canedo, and M. A. A. Faruque, “KCAD: Kinetic cyber-attack detection method for cyber-physical additive manufacturing systems,” in *2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Nov. 2016, pp. 1–8.

[22] C. Bayens, T. Le, L. Garcia, R. Beyah, M. Javanmard, and S. Zonouz, “See no evil, hear no evil, feel no evil, print no evil? Malicious fill patterns detection in additive manufacturing,” in *26th USENIX Security Symposium (USENIX Security 17)*, Vancouver, BC: USENIX Association, 2017, pp. 1181–1198, ISBN: 978-1-931971-40-9.

[23] S. B. Moore, J. Gatlin, S. Belikovetsky, M. Yampolskiy, W. E. King, and Y. Elovici, “Power consumption-based detection of sabotage attacks in additive manufacturing,” *CoRR*, vol. abs/1709.01822, 2017. arXiv: 1709.01822.

[24] Y. Gao, B. Li, W. Wang, W. Xu, C. Zhou, and Z. Jin, “Watching and safeguarding your 3D printer: Online process monitoring against cyber-physical attacks,” *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, vol. 2, no. 3, 108:1–108:27, Sep. 2018.

[25] M. Wu, Z. Song, and Y. B. Moon, “Detecting cyber-physical attacks in cybermanufacturing systems with machine learning methods,” *Journal of Intelligent Manufacturing*, vol. 30, no. 3, pp. 1111–1123, Mar. 2019.

[26] J. Gatlin, S. Belikovetsky, S. B. Moore, Y. Solewicz, Y. Elovici, and M. Yampolskiy, “Detecting sabotage attacks in additive manufacturing using actuator power signatures,” *IEEE Access*, pp. 1–1, 2019.

[27] S. Belikovetsky, Y. Solewicz, M. Yampolskiy, J. Toh, and Y. Elovici, “Digital audio signature for 3D printing integrity,” *IEEE Transactions on Information Forensics and Security*, pp. 1–1, 2018.

[28] S. B. Needleman and C. D. Wunsch, “A general method applicable to the search for similarities in the amino acid sequence of two proteins,” *Journal of Molecular Biology*, vol. 48, no. 3, pp. 443–453, 1970.

[29] H. Sakoe and S. Chiba, “Dynamic programming algorithm optimization for spoken word recognition,” *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 26, no. 1, pp. 43–49, Feb. 1978.

[30] M. M. Moya and D. R. Hush, “Network constraints and multi-objective optimization for one-class classification,” *Neural Networks*, vol. 9, no. 3, pp. 463–474, 1996.

[31] S. Mangard, E. Oswald, and T. Popp, *Power Analysis Attacks: Revealing the Secrets of Smart Cards (Advances in Information Security)*. Berlin, Heidelberg: Springer-Verlag, 2007, ISBN: 0387308571.

[32] M. A. A. Faruque, S. R. Chhetri, S. Faezi, and A. Canedo, "Forensics of thermal side-channel in additive manufacturing systems," in *CECS Technical Report No.16-01*, 2016.

[33] S. R. Chhetri, S. Faezi, and M. A. A. Faruque, "Fix the leak! An information leakage aware secured cyber-physical manufacturing system," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2017*, Mar. 2017, pp. 1408–1413.

[34] S. R. Chhetri, S. Faezi, and M. A. Al Faruque, "Information leakage-aware computer-aided cyber-physical manufacturing," *Trans. Info. For. Sec.*, vol. 13, no. 9, pp. 2333–2344, Sep. 2018.

[35] H. Turner, J. White, J. A. Camelio, C. Williams, B. Amos, and R. Parker, "Bad parts: Are our manufacturing systems at risk of silent cyberattacks?" *IEEE Security Privacy*, vol. 13, no. 3, pp. 40–47, May 2015.

[36] S. E. Zeltmann, N. Gupta, N. G. Tsoutsos, M. Maniatakos, J. Rajendran, and R. Karri, "Manufacturing and security challenges in 3D printing," *JOM*, vol. 68, no. 7, pp. 1872–1881, Jul. 2016.

[37] A. Slaughter, M. Yampolskiy, M. Matthews, W. E. King, G. Guss, and Y. 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 - ARES '17*, Reggio Calabria, Italy: ACM Press, 2017, pp. 1–10, ISBN: 978-1-4503-5257-4.

[38] A. Wang, "An industrial strength audio search algorithm," in *4th International Conference on Music Information Retrieval*, Baltimore, MD, Oct. 2003.

[39] J. Straub, "Initial work on the characterization of additive manufacturing (3D printing) using software image analysis," *Machines*, vol. 3, no. 2, pp. 55–71, 2015.

[40] ——, "A combined system for 3D printing cybersecurity," vol. 10220, 2017, pp. 10220–10220–13.

[41] ——, "An approach to detecting deliberately introduced defects and micro-defects in 3D printed objects," vol. 10203, 2017, pp. 10203–10203–14.

[42] ——, "3D printing cybersecurity: Detecting and preventing attacks that seek to weaken a printed object by changing fill level," vol. 10220, 2017, pp. 10220–10220–15.

[43] ——, "Physical security and cyber security issues and human error prevention for 3D printed objects: Detecting the use of an incorrect printing material," vol. 10220, 2017, pp. 10220–10220–16.

[44] ——, “Identifying positioning-based attacks against 3D printed objects and the 3D printing process,” vol. 10203, 2017, pp. 10203–10203–13.

[45] T. Mativo, C. Fritz, and I. Fidan, “Cyber acoustic analysis of additively manufactured objects,” *The International Journal of Advanced Manufacturing Technology*, vol. 96, no. 1, pp. 581–586, Apr. 2018.

[46] ASTM, “Standard terminology for additive manufacturing – general principles – terminology,” ASTM International, West Conshohocken, PA, Standard ISO/ASTM 52900:2015(E), 2015.

[47] T. J. Coogan and D. O. Kazmer, “Bond and part strength in fused deposition modeling,” *Rapid Prototyping Journal*, vol. 23, no. 2, pp. 414–422, 2017.

[48] C. Knapp and G. Carter, “The generalized correlation method for estimation of time delay,” *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 24, no. 4, pp. 320–327, 1976.

[49] J. Benesty, Y. Huang, and J. Chen, “Time delay estimation via minimum entropy,” *IEEE Signal Processing Letters*, vol. 14, no. 3, pp. 157–160, 2007.

[50] P. Viola and M. Jones, “Rapid object detection using a boosted cascade of simple features,” in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, vol. 1, 2001, pp. I–I.

[51] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05)*, vol. 1, 2005, 886–893 vol. 1.

[52] J. H. Hosang, R. Benenson, and B. Schiele, “Learning non-maximum suppression,” *CoRR*, vol. abs/1705.02950, 2017. arXiv: 1705.02950.

[53] S. Salvador and P. Chan, “Toward accurate dynamic time warping in linear time and space,” *Intell. Data Anal.*, vol. 11, no. 5, pp. 561–580, Oct. 2007.

[54] I. Oregi, A. Pérez, J. Del Ser, and J. A. Lozano, “On-line dynamic time warping for streaming time series,” in *Machine Learning and Knowledge Discovery in Databases*, M. Ceci, J. Hollmén, L. Todorovski, C. Vens, and S. Džeroski, Eds., Cham: Springer International Publishing, 2017, pp. 591–605, ISBN: 978-3-319-71246-8.

[55] T. Alsop, *3D printer market distribution*, 2017.

[56] Y. Han, S. Etigowni, H. Liu, S. Zonouz, and A. Petropulu, “Watch me, but don’t touch me! Contactless control flow monitoring via electromagnetic emanations,” in

Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, ser. CCS'17, Dallas, Texas, USA: ACM, 2017, pp. 1095–1108, ISBN: 978-1-4503-4946-8.

[57] M. Antonakakis *et al.*, "Understanding the Mirai botnet," in *Proceedings of the 26th USENIX Conference on Security Symposium*, ser. SEC'17, Vancouver, BC, Canada: USENIX Association, 2017, pp. 1093–1110, ISBN: 9781931971409.

[58] W. Turton, *Hackers breach thousands of security cameras, exposing tesla, jails, hospitals*, Mar. 2021.

[59] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. arXiv: 1512.03385.

[60] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015.

[61] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, no. 1, p. 60, Dec. 2019.

[62] L. Tingen, S. Juhngperng, and Y. Kerwei, "Application of cascade-connected α-β filter to diminish the drifting effect from integration," in *27th Chinese Control Conference*, 2008, pp. 657–661.