UNIVERSITY OF CALIFORNIA,
IRVINE

Data-Driven Modeling and Analysis for Trustworthy Cyber-Physical Systems

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Computer Engineering

by

Sina Faezi

Dissertation Committee:
Professor Mohammad Al Faruque, Chair
Professor Philip Brisk
Assistant Professor Zhou Li

2021

DEDICATION

To my father, the real world superhero.
To my mother, the endless source of love.
To my brother, for not letting me walk alone in dark.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to express my gratitude to my advisor and committee chair, Professor Mohammad Al Faruque, for all of his help and valuable insights through my research. Without his irreplaceable guidance, ingenious suggestions, and profound belief in my abilities, I would not have been able to complete this thesis today.

I would also like to express my deepest appreciation to my committee members, Professor Philip Brisk and Professor Zhou Li, for dedicating their valuable time to review my work and provide insightful comments.

I would like to extend my sincere thanks to Dr. Sujit Rokka Chhetri and Dr. Korosh Vatanparvar for inspiring me to find my path in research. I also thank my colleagues in the Autonomous and Intelligent Cyber-Physical Systems (AICPS) laboratory, particularly Rozhin Yasaei, Nafiul Rashid, and Anomadarshi Barua Shuvro for all the research related discussions and collaborations that we had throughout my research endeavor.

I would like to thank UCI's Department of Electrical Engineering and Computer Science for their support that allowed me to thrive in this beautiful path. I would also like to thank National Science Foundation (NSF) for partially funding my research under grants CNS-1546993, CMMI-1739503, and CMMI-1763795. Moreover, I would like to thank the Office of Naval Research (ONR) for the award N00014-17-1-2499 to support my research on hardware Trojan detection. Any opinions, findings, conclusions, or recommendations expressed in this thesis are those of the author and do not necessarily reflect the funding agencies' views.

VITA

Sina Faezi

EDUCATION

Doctor of Philosophy in Computer Engineering — **2021**
University of California — *Irvine, California*

Masters of Science in Computer Engineering — **2017**
University of California — *Irvine, California*

Bachelor of Science in Electrical Engineering — **2015**
Sharif University of Technology — *Tehran, Iran*

RESEARCH EXPERIENCE

Graduate Research Assistant — **2015–2021**
University of California, Irvine — *Irvine, California*

Research Intern — **2019**
NEC Laboratories America, Inc. — *San Jose, California*

TEACHING EXPERIENCE

Teaching Assistant, Software Design — **Spring, 2021**
Informatics Department, University of California, Irvine — *Irvine, California*

Teaching Assistant, Boolean Algebra and Logic — **Winter, 2021**
ICS Department, University of California, Irvine — *Irvine, California*

Teaching Assistant, Introduction to Programming — **Fall 2017, 2021**
EECS Department, University of California, Irvine — *Irvine, California*

Teaching Assistant, CPS Design — **Winter 2019, 2020**
ECPS Program, University of California, Irvine — *Irvine, California*

Teaching Assistant, System Software — **Spring 2018, 2019, 2020**
EECS Department, University of California, Irvine — *Irvine, California*

Teaching Assistant, Cyber-Physical Systems Design — **Fall 2016**
EECS Department, University of California, Irvine — *Irvine, California*

PATENTS

Information leakage-aware computer aided cyber-physical manufacturing **2019**
US Patent App. 16/369,993

REFEREED JOURNAL PUBLICATIONS

Brain-inspired golden chip free hardware Trojan detection **2021**
IEEE Transactions on Information Forensics and Security.

Tool of spies: Leaking your IP by altering the 3d printer compiler **2019**
IEEE Transactions on Dependable and Secure Computing

Information leakage-aware computer-aided cyber-physical manufacturing **2018**
IEEE Transactions on Information Forensics and Security

Extended range electric vehicle with driving behavior estimation in energy management **2018**
IEEE Transactions on Smart Grid

Manufacturing supply chain and product life-cycle security in the era of industry 4.0 **2018**
Journal of Hardware and Systems Security

Critical object recognition in millimeter-wave images with robustness to rotation and scale **2017**
Journal of the Optical Society of America A

REFEREED CONFERENCE PUBLICATIONS

HTnet: Transfer learning for Golden Chip-Free Hardware Trojan Detection **Mar 2021**
Design, Automation & Test in Europe Conference & Exhibition (DATE)

QUILT: Quality inference from living digital twins in IoT-enabled manufacturing systems **Apr 2019**
International Conference on Internet of Things Design and Implementation

Oligo-snoop: A non-invasive side channel attack against DNA synthesis machines **Jan 2019**
Network and Distributed Systems Security Symposium (NDSS)

Security trends and advances in manufacturing systems in the era of industry 4.0 **Nov 2017**
IEEE/ACM International Conference on Computer-Aided Design (ICCAD)

Driving behavior modeling and estimation for battery optimization in electric vehicles: Work-in-progress **Oct 2017**
IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis Companion

Fix the leak! an information leakage aware secured cyber-physical manufacturing system **Mar 2017**
Design, Automation & Test in Europe Conference & Exhibition (DATE)

ABSTRACT OF THE DISSERTATION

Data-Driven Modeling and Analysis for Trustworthy Cyber-Physical Systems

By

Sina Faezi

Doctor of Philosophy in Computer Engineering

University of California, Irvine, 2021

Professor Mohammad Al Faruque, Chair

In the age of digitization, a layer of cyber software sits on a hardware circuit and controls the physical systems around us. The tight integration of cyber and physical components is referred to as Cyber-Physical Systems (CPS). The interactions between cyber and physical components brings unique challenges which traditional modeling tools struggle to resolve. Remarkably, they often fail to model the unintentional physical manifestation of cyber-domain information flows (side-channel signals), resulting in trust issues in the system.

This thesis takes a data-driven approach to model CPS behavior when exposed to various information flows. First, we demonstrate how to extract valuable cyber-domain information by recording the acoustic noise generated by a DNA synthesizer. Then, we consider an integrated circuit as a CPS by itself and monitor the chip through electromagnetic and power side-channels to detect hardware Trojans (HT) in the chip.

HT is a malicious modification of the hardware implementation of a circuit design which may lead to various security issues over the life-cycle of a chip. One of the major challenges for HT detection is its reliance on a trusted reference chip (a.k.a golden chip). However, in practice, manufacturing a golden chip is costly and often considered infeasible. This thesis investigates a creative neural network design and training methodology which eliminates the need for a golden chip. Furthermore, it proposes using hierarchical temporal memory (HTM)

as a data-driven approach which can be updated over the chip's life-cycle and uses that for run-time HT detection.

Chapter 1

Introduction

1.1 Cyber-Physical Systems

"A cyber-physical system (CPS) is an orchestration of computers and physical systems [78]." The concept of CPS has grabbed a lot of attention recently as the computation paradigm has shifted from one powerful central computer to distributed embedded computers for an intelligent environment. Initially, a central computer with many users was the available computation platform until the next generation of computers such as PCs and laptops appeared. With further technological advancement, smartphones were proposed as a new computational revolution. Along the same line, embedded computing was developed as the next trend, which introduced new concepts and technologies such as the Internet of Things (IoT) and wearable devices. An intelligent environment is the new computation trend that aims to integrate the physical environment with the embedded computing to remove the human from the loop and build large-scale smart systems. The three key factors leading to this new paradigm are device/data proliferation, integration at scale, and autonomy which points out the limited human ability the consume the increasing information generated by the expo-

nential proliferation of embedded devices. In response to these challenges, the study of CPS is proposed. Some of the main applications of CPSs are automotive, autonomous avionics, smart grid, manufacturing, smart city (smart infrastructure), and health care.

CPS consists of the tight integration of cyber and physical components, which interact with each other in an inseparable manner. Traditionally, there have been various simulation and modeling tools to solve physical complications and read the environment as a deterministic component controlled by algorithmic cyber domain software. Over the past decades, researchers have studied miscellaneous methods for design and automation tools that target hardware/software co-design, physical modeling, networking, control systems simulation, etc. These methods often take a singular obstacle under consideration and provide a solution that perfectly fits the problem. However, the interaction of physical elements with cyber domain modules results in a new set of problems/opportunities which may cause catastrophic failures in the system or favorable circumstance for improving various aspects of CPS systems.

This thesis takes a data-driven approach to study the surface created by CPS components' interactions. Notably, in chapter 2, we show how it is possible to take advantage of a biomedical device (DNA synthesizer) to steal valuable cyber domain information. Then, in chapter 3 and 4 we devise data-driven solutions to model the physical side-channel emissions of integrated circuits (ICs) and use that for hardware Trojan detection.

1.2 Data-Driven Modeling

CPS modeling tools commonly represent CPS as a component-orientated model, actor-oriented model, multi-agent-based model, and event-based models [52]. The type of modeling representations are chosen based on different system characteristics and modelings requirements, and they mostly consist of formal methodologies for design and the engineering of CPS. The majority of available CPS modeling tools rely on first-principle (established laws of physics) derived by the domain experts. The examples of CPS modeling languages and simulation tools are UML [15], CyPhyML [133], Modelica [45], MATLAB Simulnik [21], LabView[13], Ptolemy [109], etc. While these CPS tools create a model with provable deterministic properties that can be used for detecting design defects, they often show a poor capability for capturing stochastic properties of CPS. In particular, when a model is physically implemented, some non-deterministic behavior can be expected from the system due to environmental noise or unaccounted interactions of system components [78]. Traditionally, during the CPS development, this challenge is met by employing rigorous testing of the system after implementation and addressing unseen probabilistic problems only if needed.

In this thesis, we tackle the unpredictable and unaccounted behavior of a CPS using a data-driven approach. The data-driven approach for CPS utilizes the collectible data from a specific system to model/estimate/infer the relationship between different variables of the given system without requiring detailed domain knowledge [127]. Currently, CPSs encompass a large number of sensors providing abundant data that enables data-driven approaches for such systems. Furthermore, throughout this thesis, we show that it is possible to collect side-channel data to expand the data-driven modeling techniques using different CPS cases. We show how it is possible to incorporate the latest advancements in data-driven modeling (artificial intelligence, machine learning, etc.) to solve trustworthiness issues in CPS.

1.3 Trustworthy CPS

The concept of trust has a broad definition that involves various aspects of human life. Over the years, scholars and practitioners have widely acknowledged the importance of trust [97]. The aggressive infusion of CPS into human beings' life in the form of the digitization of everything (from toothbrushes to inter-personal interactions, complex business processes, healthcare) entails the need for a high degree of trust in these systems. While trusting is an act carried out by an end-user, trustworthiness is a quality of the system that has the potential to influence the end-users' trust in the system in a positive way [98]. As a definition, trustworthiness is subject to the interpretation of the end-user. For example, organizations require confidence in the secure handling of their business-critical data, whereas end-users may be more concerned about the usability of the final product [98].

In this thesis, we narrow down the general definition of the trustworthiness of CPS in terms of confidentiality, integrity, and availability (the CIA Triad). The CIA Triad is considered to be the core foundation of information security. In the information age, CPSs play an important role in collecting a high volume of data from various phenomena while consuming a large amount of cyber and physical domain information. In CPS, confidentiality measures are designed to ensure the lockdown of sensitive information from unauthorized access attempts. Meanwhile, CPSs are often employed to carry mission-critical tasks such as saving a human life or landing a spacecraft on Mars. The integrity of CPS involves maintaining the expected behavior of the system while blocking intruders' attempts to change the the behavior of the system over its life cycle. Moreover, in terms of availability, a trusted CPS should guarantee its services to authorized users whenever needed. Along the lines of our definition for trustworthy CPS, through this thesis, we choose various CPS examples to either challenge the CIA properties of those systems or strengthen the trust in the considered system.

1.4 Side-channel Signals

Depending on the nature of a CPS, the designers choose to monitor the system using different mediums such as temperature, pressure, etc. On many occasions, the sensors installed in the system are limited only to those necessary for the system's control and health monitoring. Therefore, not all the observable behavior of a CPS is captured through embedded sensors in the system. In this thesis, we observe the system not only through originally collected observable signals but also via side-channel signals acquired by secondary sensors added to the system. Over the last decades, scholars have shown the use-case of side-channel signals for attacking electronic devices beyond traditional cryptanalysis [62] [93]. The scholars have used side-channel analysis to exploit the information leaking from the physical implementation of electronic devices to discover cryptographic keys or other secrets.

In this thesis, similar to the traditional side-channel analysis for cryptographic devices, we use the analog emission from the physical domain of CPS to infer about different states of the system. We argue that, in CPSs, the cyber domain information flow of the system is often manifested in the form of observable energy flows such as vibration/acoustic, electromagnetic, power, heat, etc. By monitoring these energy flows, we show how it is possible to steal valuable cyber domain information as an attacker, and also show that it can be used to assure integrity of the system.

1.5 Major Challenges and Thesis Contribution

The major challenges for designing a trustworthy CPS are as follows:

- Finding the trust issues which may arise due to the interactions between cyber and physical components in the system.
- Modeling non-deterministic behavior of CPS.
- Choosing modeling methodologies that can be updated as the system ages within the system budget.

To address these challenges, this thesis makes the following contributions:

- Various data-driven approaches are examined to model the behavior of CPS test cases.
- A novel attack methodology is introduced to steal cyber domain information through side-channel analysis of a CPS.
- The malicious activities in ICs as a CPS are detected by a creative neural network model.
- A data-driven approach that can be updated over time is proposed to keep track of ICs during their life cycle.

1.6 Thesis Structure

This thesis is structured as follows: Chapter 2 investigates the commonly used DNA synthesizers' vulnerability against new attack vectors which are possible due to cyber and physical components interactions in these devices; Chapter 3 presents a novel neural network design

(a.k.a HTnet), along with a creative training methodology which can be used for golden chip free hardware Trojan detection; Chapter 4 suggests using a hierarchical temporal memory that can be effectively trained in run-time and used for anomaly detection in integrated circuits behavior.

Chapter 2

Oligo-Snoop: A Non-Invasive Side Channel Attack Against DNA Synthesis Machines

2.1 abstract

Synthetic biology is developing into a promising science and engineering field. One of the enabling technologies for this field is the DNA synthesizer. It allows researchers to custom-build sequences of oligonucleotides (short DNA strands) using the nucleobases: Adenine (A), Guanine (G), Cytosine (C), and Thymine (T). Incorporating these sequences into organisms can result in improved disease resistance and lifespan for plants, animals, and humans. Hence, many laboratories spend large amounts of capital researching and developing unique sequences of oligonucleotides. However, these DNA synthesizers are a CPS with cyber-domain processes and physical domain components. Hence, they may be prone to security breaches like any other computing system. In our work, we present a novel acoustic side-channel

attack methodology which can be used on DNA synthesizers to breach their confidentiality and steal valuable oligonucleotide sequences. Our proposed attack methodology achieves an average accuracy of 88.07% in predicting each base and is able to reconstruct short sequences with 100% accuracy by making less than 21 guesses out of $\mathbf{4^{15}}$ possibilities. We evaluate our attack against the effects of the microphone's distance from the DNA synthesizer and show that our attack methodology can achieve over 80% accuracy when the microphone is placed as far as 0.7 meters from the DNA synthesizer despite the presence of common room noise. In addition, we reconstruct DNA sequences to show how effectively an attacker with biomedical-domain knowledge would be able to derive the intended functionality of the sequence using the proposed attack methodology. To the best of our knowledge, this is the first methodology that highlights the possibility of such an attack on systems used to synthesize DNA molecules.

2.2 Introduction

The ability to rapidly sequence and synthesize DNA has profound implications for society. Large libraries of different DNA sequences play an essential role in genomics research, especially for genetic analysis. Synthetic DNA is poised for widespread consumption if its costs can be lowered dramatically. Based on current trends, the global market for synthetic biology is projected to reach $38.7 billion by 2020 [124]. Beyond biological applications, researchers are beginning to construct DNA-based archival storage systems, which can store up to 215 petabytes of data per gram, with centuries to millennia of endurance if properly stored in a cool and dry environment [114].

Unfortunately, technological advancement often creates new security concerns as technologies mature. To date, the foremost security threat in this field involves the physical safety of synthesized DNA. Present efforts to reduce or eliminate misuse of synthetic DNA include

biosecurity regulations, training and licensing programs for authorized agents, and the embedding of screening chips into DNA synthesizers (modeled on parental control of television access) [102, 18, 118]. However, these threat models implicitly assume that the value is inherent in the DNA itself, as opposed to the information that is encoded in the DNA.

Somewhat more generally, the cyber-physical nature of biotechnology workflows creates new security risks, which the corresponding research community has mostly neglected [104]. One recent example is the now-demonstrated ability to encode information into a DNA sequence that can trigger a buffer overflow error in DNA sequencing software; this exploit can be used to inject malware into the computer running the sequencing algorithm [100]. A subsequent concern is the confidentiality of DNA sequences stored in human biobanks. If the genetic information of the earth's population is exposed, then an attacker may be able to create a contagious virus that is fatal to individuals or a small group, but is otherwise benign to the general population [96].

Confidentiality concerns also extend to synthetic DNA sequences. In synthetic biology, the objective is often to engineer an organism with desired traits or functions. Investors only reap the rewards of their investments *after* the engineered organism passes all regulatory requirements and the investor obtains intellectual property ownership in the form of a patent or copyright. However, while the organism is still under development, the research remains vulnerable to industrial espionage or academic intellectual property theft [125]. In this case, the actual secret to be protected may be an amino acid sequence within a protein (which is derived from DNA) as opposed to the DNA itself. Within this larger context, knowledge of the DNA can still help an attacker determine the amino acid sequence, and the attacker can further benefit if he or she has knowledge of the desired traits or functions of the organism under development. The content of this chapter is provided from [37].

2.2.1 Motivation and Overview

This thesis presents *Oligo-Snoop*: a novel, acoustic, side-channel, analysis-based attack model that can breach the confidentiality of DNA synthesizers. The attack model leverages the physical implementation of the synthesizer to infer the DNA sequence being synthesized. By publishing this attack, we hope to encourage commercial DNA synthesizer manufacturers to strengthen their confidentiality, especially to protect against attack vectors that may be discovered in the future.

As a motivating example, Gibson *et al.* synthesized the genome of a living bacterium out of one million bases of synthetic DNA [50]; eavesdropping on that DNA synthesis run would provide the attacker with the blueprints of a complete organism. More often, instead of synthesizing an entire genome from scratch, researchers add synthetic DNA to an existing organism's genome, thereby imparting desired traits to that organism. For example, for many years the anti-malaria drug artemisinin was available only from a rare plant; however, in 2006 Ro *et al.* added DNA to yeast cells, inducing the modified yeast to produce artemisinin [113], which dramatically reduced the cost of producing a lifesaving drug. In a more recent (and rather controversial) example, Galanie *et al.* added DNA to yeast cells to force them to create prescription opioid drugs [46]. Synthetic DNA plays a key role in each of these examples, and for these and similar efforts to remain secure, it is necessary to develop further protection against eavesdropping.

From a different perspective, the ability to eavesdrop on a DNA synthesizer could be useful in the fight against bioterrorism. Although DNA synthesis has several beneficial applications, there are many ways that it can be used maliciously. Since pathogens are composed of DNA, synthesis methods can be used for artificial pathogen creation. For years, researchers and government agencies have warned that an aspiring terrorist could use synthetic DNA and the techniques of synthetic biology to create deadly pathogens [18] . For example, the

deadly Ebola virus has a genome of only about 18,960 bases [19] and could be built from scratch using synthetic DNA, as could genes from the eradicated disease smallpox, which was responsible for 300-500 million deaths in the 20th century alone [138] . The risk of synthetic, pathogenic DNA is significant enough that in 2010 the US Department of Health and Human Services issued a statement to commercial DNA synthesis companies, warning them to be on the lookout for customers ordering "sequences of concern," or snippets of DNA from the genomes of anthrax, Ebola, smallpox, and several other deadly pathogens [1]. With second-hand DNA synthesizers available on the online auction site eBay for less than $1000, it is feasible that an aspiring bioterrorist could try to use synthetic DNA to create their own tools of biological warfare. The ability to eavesdrop on a suspected terrorist's DNA synthesizer could potentially allow an intelligence or law enforcement officer to ascertain whether or not the suspect is trying to manufacture a deadly biological weapon.

2.2.2 Research Challenges

Technical challenges associated with DNA synthesizer confidentiality are as follows:

- Understanding the DNA synthesis process and its physical implementation.
- Identifying vulnerable components of a DNA synthesizer which can be leveraged under a practical threat model.
- Analyzing attack methodologies which an attacker may utilize.
- Understanding the ways in which an attacker may post-process side channel data to accurately reconstruct the DNA sequences that were synthesized.

2.2.3 Technical Contributions

In this chapter, this thesis makes the following technical contributions, which directly address the challenges listed above:

- We provide a feasibility analysis (**Section 2.5**) to identify potential sources of side-channel information leakage in the system which have never been considered before.
- We present an attack model and propose a practical design approach (**Section 2.4** and **2.6**) that an attacker may use to breach the confidentiality of the DNA synthesizer and reconstruct the DNA sequences using information leaked by the acoustic side-channel.
- We propose an algorithm (**Section 2.7**) that allows an attacker to obtain the other most likely reconstructions of the synthesized DNA sequence if the originally reconstructed DNA sequence is faulty.
- Due to the uniqueness of the proposed attack, in **Section 2.8**, we propose new methods to evaluate our work in terms of performance. For instance, in **Section 2.8.5** we show how to model the distance between the DNA synthesizer and microphone without exhaustive experimentation.
- We propose using a free tool designed for a different purpose to map imperfect attack model predictions onto more meaningful DNA sequences.

2.2.4 Chapter Organization

The chapter is organized as follows: Section 4.3 presents the background necessary to understand the system and process and summarizes related work on confidentiality in cyber-physical systems; Section 2.4 presents the threat model used to breach DNA synthesizer

confidentiality; Section 2.5 analyzes potential sources of acoustic emissions from the synthesizer; Section 2.6 presents the proposed attack methodology; Section 2.7 explains how an attacker can reconstruct synthesized DNA sequences from acoustic measurements with a small number of guesses; Section 2.8 reports experimental results; Section 2.10 discusses potential countermeasures to the attack; and Section 4.8 concludes the chapter.

2.3 Background & Related Work

2.3.1 Oligonucleotide Synthesis

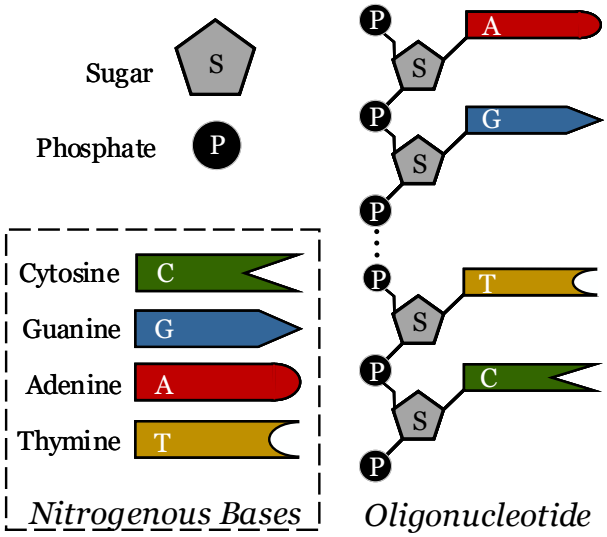


Figure 2.1: Nucleotide bases and oligonucleotide sequence.

Oligonucleotides are the building blocks of DNA and RNA molecules. As shown in Figure 2.1, an oligonucleotide is a sequence of nucleotides. Each nucleotide comprises one of four nitrogen-containing nucleobases (Adenine (A), Cytosine (C), Guanine (G), and Thymine (T)) attached to a sugar (deoxyribose) and a phosphate group. The oligonucleotide is formed by constructing an alternating sugar-phosphate backbone, which joins the nucleotides to one another in a chain of covalent bonds. DNA, which is double-stranded, is formed by joining

133

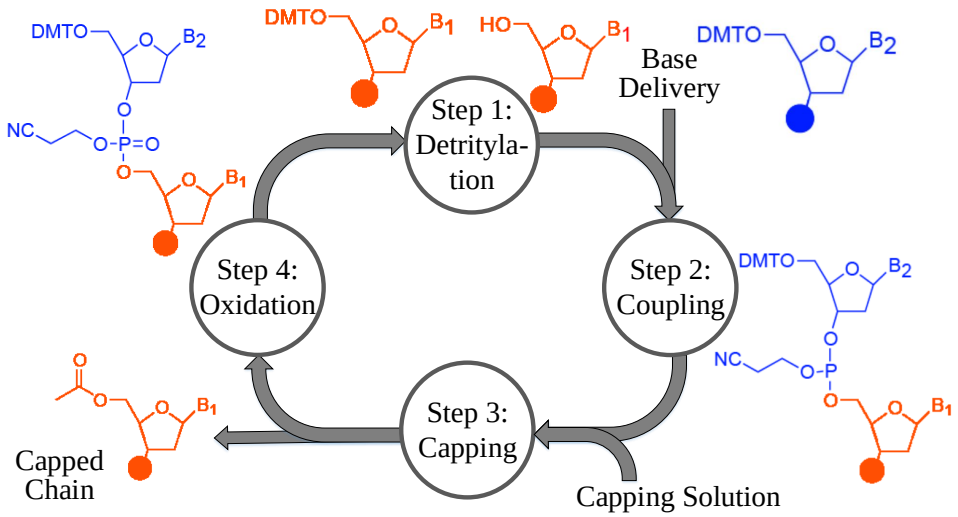


Figure 2.2: Oligonucleotide synthesis cycle.

two complementary oligonucleotides according to base pairing rules (A with T and C with G) where complementary base pairs are joined by hydrogen bonds.

The term "DNA synthesis" is somewhat of a misnomer: so-called DNA synthesizers typically produce oligonucleotides, not double-stranded DNA. If DNA is desired, the user must synthesize two complementary oligonucleotides (typically multiple copies of each) and induce bonding via chemical or enzymatic means. Oligonucleotide synthesis produces short chains of nucleic acids with a defined sequence of bases. The most common form of oligonucleotide synthesis uses the phosphoramidite method [95], which produces multiple chains simultaneously by anchoring bases to a solid support and building upwards. DNA or RNA molecules have groups of 5 carbon atoms in the deoxyribose backbone. These carbon atoms are numbered 1' to 5'. While building the chains, a protective dimethoxytrityl (DMT) group is attached to the open 5' end of each chain. This prevents them from reacting or bonding with undesired materials before a new base is attached.

Figure 2.2 illustrates the process of adding a new base to an oligonucleotide:

- **Detritylation**: The protective DMT groups of the current chains are stripped away so

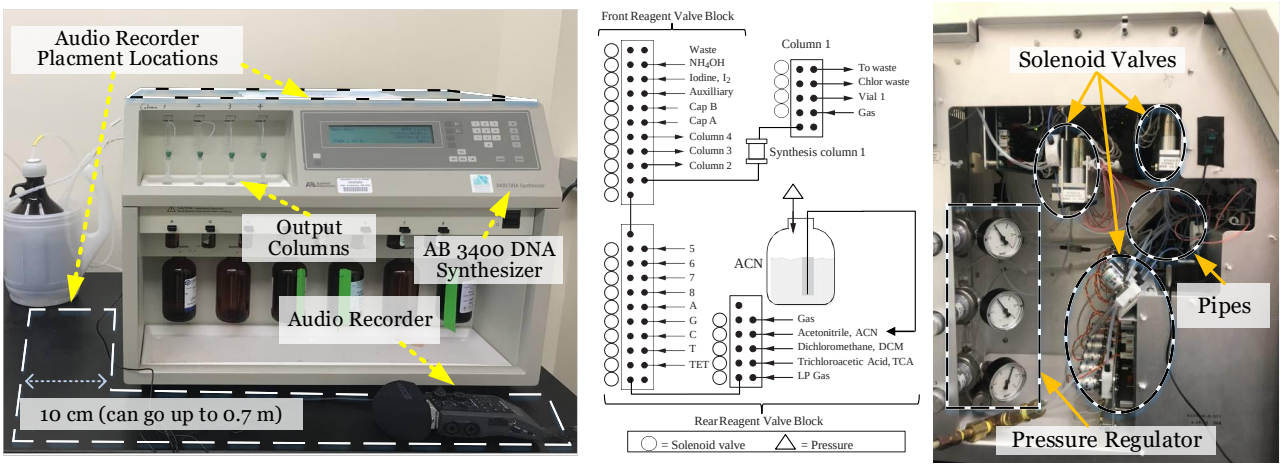
the 5'-terminal can bond to the next base.

- **Delivery**: The next nucleoside phosphoramidite base to be attached is delivered to the solution.
- **Coupling**: A coupling agent, which contains a catalyst that causes the nucleoside to bond with the existing oligonucleotide, is delivered to the solution.
- **Capping**: A small percentage of the chains do not react in the coupling stage and thus do not receive a new base. The base support holding the oligonucleotide is treated with a capping solution that suppresses the addition of further nucleosides.
- **Oxidation**: The attachment point between the current oligonucleotides and the newly added base takes the form of a tricoordinated phosphate triester linkage. This structure is not natural and has limited stability. To improve the stability of this attachment point, the oligonucleotides are treated with iodine and water in the presence of a weak base to oxidize the phosphate triester, transforming it into a tetracoordinated phosphate triester. This form of linkage is natural, stable, and protected.

The oligonucleotides are now ready to receive their next base. The process repeats for every new base addition. Once all the bases have been attached, the oligonucleotides are cleaved from their solid support structures and collected for use.

2.3.2 DNA Synthesizer

DNA synthesizers use a pressure-driven system, shown in Figure 2.3, to deliver chemicals to the output columns where synthesis takes place. A pressurized inert gas pushes chemicals through common pathways and delivers them to the synthesis columns. Blocks containing solenoid valves open and close certain pathways to route chemicals to the synthesis columns.



(a) (b) (c)

Figure 2.3: (a) Experimental setup. (b) Simplified schematics of a DNA synthesizer. Refer to [6] for a detailed explanation of this schematic. (c) The internal structure of the DNA synthesizer.

During each iteration of the synthesis procedure, the common pathways are flushed to remove any leftover residue from the previous iteration.

Our experiments use an Applied Biosystems (AB) 3400 DNA Synthesizer [6]. This specific machine performs 48 valve operations during each synthesis cycle (Figure 2.2). Each step of the synthesis cycle requires multiple valve actuations to clean and prime the delivery lines before issuing step-specific valve operations that deliver the requisite chemicals to the columns.

Some large-scale DNA synthesis machines deliver multiple bases per delivery operation or deliver the same base to different columns at the same time. These machines can complete batch synthesis operations with higher throughput than single-operation synthesis machines such as the AB 3400. However, they still follow the same sequence of steps as shown in Figure 2.2 and produce similar results, albeit in larger quantities.

2.3.3 Side-channel and Information leakage

Side-channel analysis has been studied extensively in various systems to determine their vulnerability. Analog emissions (acoustics, power usage, electromagnetic emissions, vibrations, etc.) are one of the primary side-channels known to leak information. Acoustic emissions have been used to infer fill patterns in additive manufacturing systems [41, 10, 22, 36, 23] and carry out physical attacks on magnetic hard disks [16]. Authors in [131] provided extensive results on the usability of light, seismic and acoustic side-channels by providing their channel characteristics such as rate and path loss. A power side-channel was used in [25] to detect malware in medical embedded systems. Authors in [149] utilized a memory side-channel to detect the activity of unwanted co-resident's virtual machine and the authors of [150] presented cache-based side-channel attacks which can be mounted on existing commercial clouds to steal cross-tenant information. Electroencephalography (EEG) signals obtained from a brain-computer interface were used in [92] to infer private user information. Authors in [20] demonstrated how network traffic based side-channels can be quantified for securing web applications. Timer interrupts and cache based side-channels were used in [54] to achieve a higher success rate than page-fault based side-channel attacks on untrusted operating systems. Authors in [7] demonstrated how accelerometer based side-channels can be used to infer user login details on smartphones. Each of these works demonstrates how various side-channels can be utilized to either infer information or provide better defense mechanisms in various systems.

2.3.4 Acoustics

Vibration of a system in contact with air molecules generates a mechanical wave called an acoustic signal. Commonly, the intensity of this acoustic signal is expressed in terms of

sound pressure level (SPL) using

$$\boldsymbol{L_p = 20log(\frac{p}{p_0}),} \tag{2.1}$$

where $\boldsymbol{p}$ is root-mean-square (RMS) of acoustic pressure and $\boldsymbol{p_0}$ is the minimum hearable acoustic level by human ears [40]. For a quiet office $\boldsymbol{L_p = 50}$ dB; normal conversations $\boldsymbol{L_p = 60}$ dB; vacuum cleaner $\boldsymbol{L_p = 70}$ dB; and hair dryer $\boldsymbol{L_p = 80}$ dB. Human ears barely detect a **3** dB difference in SPL while a **5** dB change can be easily noticed under most conditions.

2.4 Attack Model

Figure 2.4 depicts an attack model that can breach the confidentiality of a DNA synthesizer via information leaked in the acoustic side-channel. The components of the attack model are described as follows:

Adversary Intent: Industrial espionage for stealing intellectual property (IP) and monitoring for bioterrorism activities are covered by this attack model but speak to the intent rather than the identity of the attacker.

Outcome of an Attack: The attacker recovers the sequence of states of the target DNA synthesizer, $\hat{\boldsymbol{S}}_{\boldsymbol{target}}$, which translates to the order and types of the bases that are synthesized by the machine.

Target System: DNA synthesizers can connect to computers, external drives, and Ethernet cables. However, operators generally keep the machine disconnected from the Internet and local networks or use secured protocols to eliminate the possibility of cyber-attacks. In addition, we assume that tampering with the machine or accessing the output DNA sequence

is not possible so the attack must be non-invasive. This is because fluids in the machine are sealed and are driven by pressurized argon. Any exposure to air would result in significant quality degradation, raising an alert.

Existing Vulnerabilities of the System: The DNA synthesizer is vulnerable to physical emissions that can leak system data during operation. Minimizing system observability can hide the system states from the attacker; however, DNA synthesizers are optimized for efficiency (throughput) and accuracy, not security.

Attack Medium: The attacker acquires information about the state of the DNA synthesizer via acoustic side-channel ($\boldsymbol{A}$).

Attacker Capabilities: We assume that the attacker has the capability to place at least one microphone within close physical proximity to the DNA synthesizer. Such an attacker could be a disgruntled employee or a visitor with low-level access to the machine (meaning physical proximity, and no access to cyber components). In this scenario, the attacker can surreptitiously and non-intrusively place an audio recording device (such as a phone) near (or on) the DNA synthesizer. Placing the recorder requires one-time access if we assume that it has wireless transmission capabilities; otherwise, a second physical visit is required to recover the recorder and the data it has collected. Since the authorized users of the target DNA synthesizer are unaware of the attack model introduced in this section, most probably, they would neglect the security implications of any recording device around the machine. Furthermore, if an attacker is able to breach the other systems in the same laboratory (i.e. remote monitoring systems [120], employee phone/laptop, etc.), he will be able to record the information leaked in the acoustic side-channel of the DNA synthesizer through existing microphone(s) of those systems.

Attacker Resources: We assume that the attacker not only has domain knowledge about the synthesis process, but also has access to the user manual of the machine explaining the

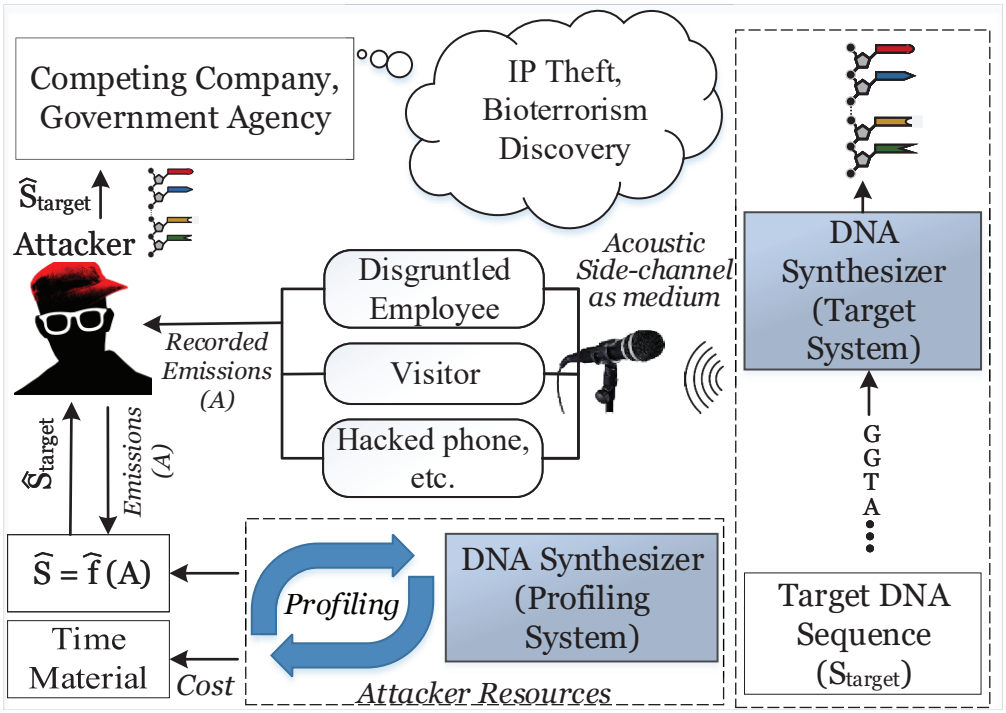


Figure 2.4: Attack model.

machine specific procedures for DNA synthesis. Furthermore, we assume that the attacker has the opportunity to carry out as many experiments as needed to *profile* the machine and build an accurate model ($\hat{\boldsymbol{S}} = \hat{\boldsymbol{f}}(\boldsymbol{A})$, see Figure 2.4) that can infer the order of synthesized bases based on recorded acoustic signals collected from the target machine. If the attacker is a disgruntled employee who has unlimited access to the target system, the profiling DNA synthesizer can be same as the target machine. However, if access is not provided, the attacker could use a replica for profiling purposes (the same model with a structure that is identical to the target machine). For the rest of this chapter, we consider the former scenario where the target and profiling DNA synthesizers are the same machine.

Cost: For an attacker to profile the target machine, the cost of an attack is just the value of the chemical materials used and the time spent during the profiling process (estimating ($\hat{\boldsymbol{f}}$)).

2.5 Feasibility Analysis

While oligonucleotide synthesis is in progress, the physical activity of different components of the system such as solenoid valves, cooling system fans, pressure regulators, and fluids flowing in pipes causes vibration which results in structural acoustic noise emission from the system. We hypothesize that various solenoid valves opening/closing and the flow of fluid through various pipes emit information about the various states of the oligonucleotide synthesis cycle, and that we may be able to identify which nitrogenous bases (A, G, C or T) are deposited during the delivery state of the oligonucleotide synthesis cycle. An attacker may thus eavesdrop on the acoustic emissions, that behave as side-channels, to infer the cycles and the type of the base being delivered.

2.5.1 Structural Acoustics caused by the pipes

DNA synthesizers use plastic pipes (lines) to deliver the nucleotide bases and other chemical materials from the source reservoirs to the output columns and other containers attached to the system. The internal turbulence of the fluid flow running in the pipe causes the walls of the pipe to vibrate, resulting in acoustic noise (i.e. vibro-acoustic) radiation from the pipes. Over the last few decades, a substantial body of research has been dedicated to modeling fluid-structure interactions to simulate and predict vibro-acoustic signal emissions of pipe structures [81, 139]. Work in this area has shown that the magnitude and frequency of a generated vibro-acoustic signal can be determined based on the spatial structure of the pipes, pipe wall thickness, internal pipe pressure, internal fluid speed, the mass density of the fluid and the pipe, and several other features. Authors in [126] have demonstrated how minute changes in the curvature of the elastic pipes can result in different vibro-acoustic footprints. As shown in Figure 2.3c, the delivery lines in the DNA synthesizer have different spatial shapes and curvatures and also deliver fluids with different mass densities. Based on the

work in [126], we suspect that these changes will result in close but unique wavenumbers.

2.5.2 Structural Acoustics caused by the solenoids

DNA synthesizers employ electric solenoids to open and close the valves that control the flow of chemicals to each column; each valve opening or closing operation emits an audible click. Although each solenoid emits a near-identical sound, the DNA synthesizer is an enclosed structure and each valve is located at a different position within the machine. As stated in [146], the enclosed space creates measurable reverberations when a valve emits sound. The equation to calculate reverberation time is given in [146] as:

$$\boldsymbol{T = 0.049\frac{V}{Sa}}, \tag{2.2}$$

where $\boldsymbol{V}$ is the volume of the enclosure, $\boldsymbol{S}$ is the surface area which reflects sound, and $\boldsymbol{a}$ is the average Sabine coefficient of the enclosure. As each valve occupies a distinct position within the DNA synthesizer, the surface area that causes reflections, which impacts the reverberation time, is unique for each valve. Consequently, the collected acoustic signals are likewise unique for each valve due to their unique channel distortions. Similar to the sound generated from the fluids moving through the machines piping, the distinctions between valve noises may be near-inaudible for human listeners. However, a properly trained algorithm should be able to identify key features that distinguish different valve operations.

2.6 Attack Model Design

Here we present the design of the attack model, which we introduced earlier in Section 2.4. In order to accurately infer the physical and cyber-domain states of the system ($\boldsymbol{S}$) from the acoustic side-channel ($\boldsymbol{A}$), an optimal attack could first use principle-based equations to de-

rive a function ($\boldsymbol{A = f(S)}$) to explain the sound produced by the individual components of the system based on the DNA sequence. Then, using techniques like finite element analysis, the attacker may acquire an accurate acoustic emission profile of the DNA synthesizer. Afterward, the attacker may use the inverse function to estimate the sequence ($\boldsymbol{\hat{S} = f^{-1}(A)}$). However, this approach would require an attacker to have complete design details of the individual components, their chemical composition, etc., to accurately simulate the acoustics from the system. To overcome this problem, we propose to use a data-driven approach, by treating the DNA synthesizer as a black-box, to estimate the function ($\boldsymbol{\hat{S} = \hat{f}(A)}$). This approach requires less domain knowledge and achieves faster attack model implementation time for an attacker.

As shown in Figure 2.5, to estimate the function that describes the relationship between the acoustic signal and the oligonucleotide sequences, our proposed attack model consists of two main phases: the training phase and the attack phase. This function may be abstracted as $\boldsymbol{S = \hat{f}(A, \theta)}$, where $\boldsymbol{\theta}$ is the parameter that needs to be trained, $\boldsymbol{S = [S_1, S_2, \ldots, S_n], S_i \in \{A, G, C, T\}}$ is the sequence of oligonucleotides with length $\boldsymbol{n}$, and $\boldsymbol{A}$ represents the acoustic signal gathered from the side-channel. In the training phase, an attacker randomly selects an arbitrary number of training oligonucleotide sequences ($\boldsymbol{S_{train}}$). These sequences are then passed to a profiling DNA synthesizer. Then for each of the $\boldsymbol{S_i \in \{A, G, C, T\}}$, the attacker collects the corresponding acoustic emission ($\boldsymbol{A_{i_1}, A_{i_2}, \ldots, A_{i_k}}$), where $\boldsymbol{k}$ is the length of the acoustic emission. The attacker must initially find an optimal location to place the acoustic sensors, which, in our work, are placed next to the DNA synthesizer in close proximity to the solenoids and the pressure valves. We then perform preprocessing and feature extraction on these acoustic emissions and label them with their corresponding nucleotide bases $\boldsymbol{\{A, G, C, T\}}$. Using a supervised learning approach [115], a classifier function is estimated to predict the particular nucleotide base given the acoustic emission $\boldsymbol{\hat{S}_i = \hat{f}(A_i, \theta)}$. In the attack phase, the attacker surreptitiously places sensors on the target DNA synthesizer and collects the acoustic emissions, and infers

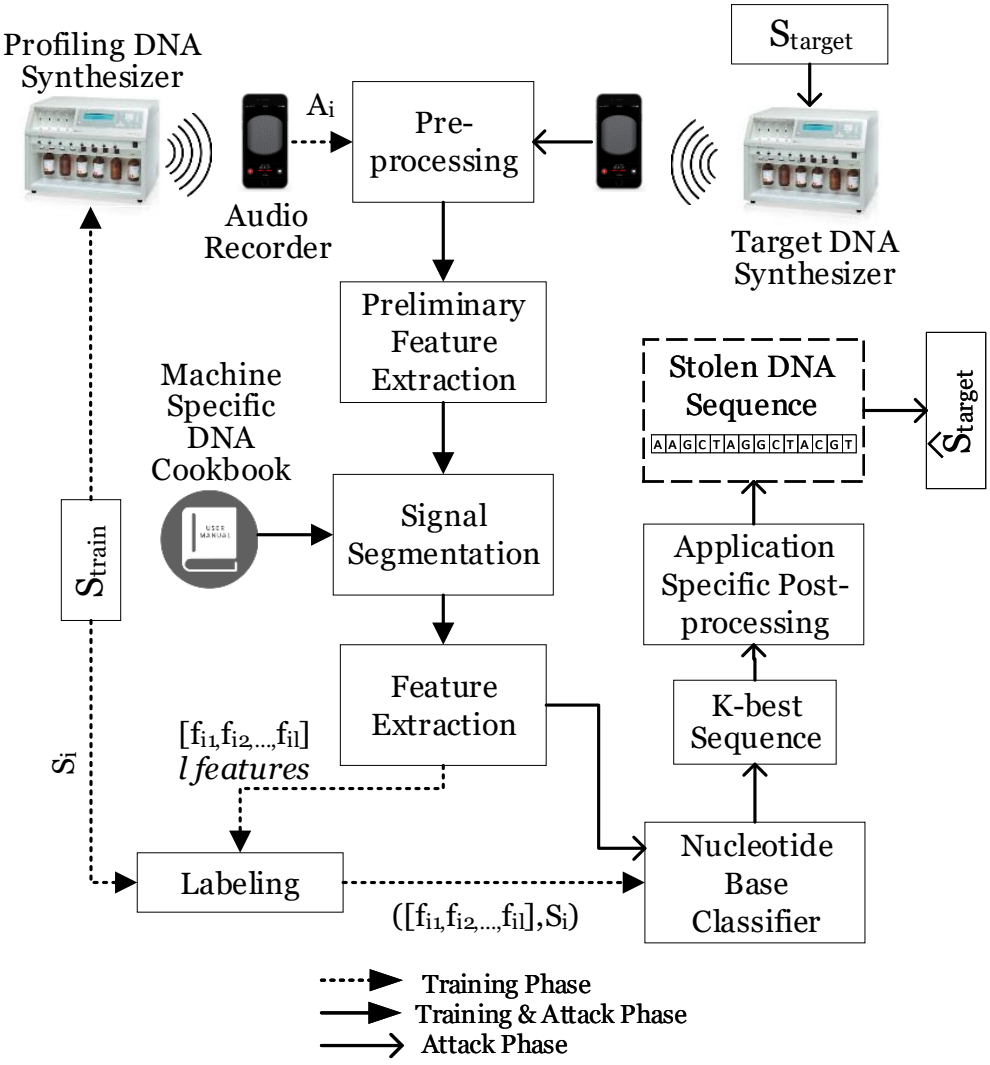


Figure 2.5: Acoustic side-channel attack methodology.

the target oligonucleotide sequence ($\boldsymbol{S_{target}}$). The details of the each of the steps of the attack model design are as follows:

Preprocessing. In this stage, the attacker uses a set of bandpass filters combined with heuristic methods to reduce the effect of background environmental noise, which is added to the acoustic signal generated by the DNA synthesizer. For instance, the attacker may use

$$\boldsymbol{A_{normalized}} = \boldsymbol{diag}(\frac{\mathbf{1}}{\sqrt{\boldsymbol{diag(Rnn)} + \boldsymbol{\epsilon}}} \times \boldsymbol{A}) \tag{2.3}$$

similar to what it has been used in [60] to model the background noise and normalize the signal in relation to it. In this model, $\boldsymbol{Rnn}$ is the background noise covariance matrix based on a portion of a recording when the machine is idle; $\boldsymbol{A}$ is the recorded signal, and $\boldsymbol{\epsilon} = \mathbf{1} \times \boldsymbol{e^{-10}}$ is used to avoid division by zero. In our experience, the background environmental noise is usually more prominent in lower frequency ranges. Hence, if the attacker determines that the DNA synthesizer does not leak information in lower frequencies, he/she can simply use a high-pass filter to eliminate low-frequency components from the signal.

Preliminary feature extraction. Once background noise is removed from the recorded signal, the attacker needs to extract the portions of the signal that correspond to base deliveries $\boldsymbol{\{A, G, C, T\}}$. As shown in Figure 2.2, the oligonucleotide synthesis process goes through various stages, base delivery being one of them. An attacker needs to know the time taken by various stages in order to accurately segment the acoustics for just the nucleotide base delivery stage. As shown in Figure 2.6, opening and closing the solenoid valves introduces peaks in the acoustic signal. These peaks may help an attacker track various stages.

However, difficulties may arise when multiple valves open and close at the same time, which would result in peaks with variable intensity. To mitigate this issue, an attacker could use an approach proposed in [30] to detect the peaks by using their shape with the help of

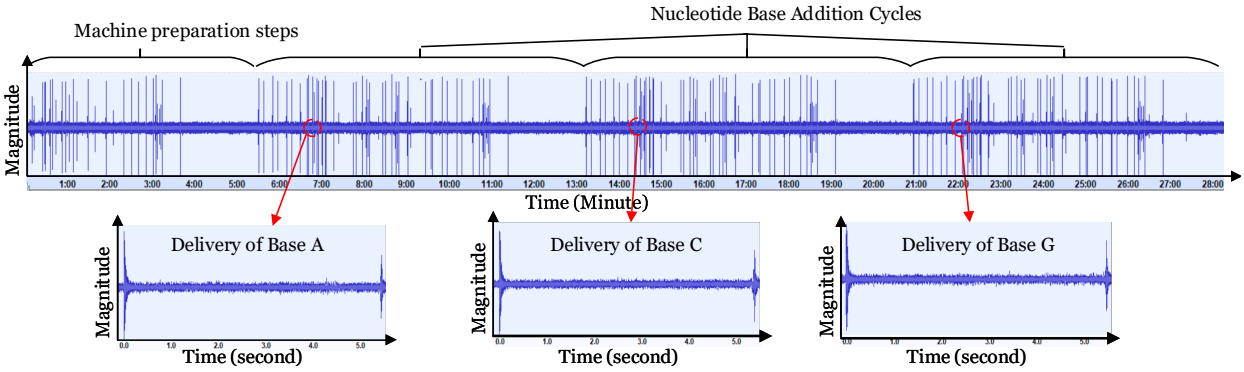


Figure 2.6: Sample acoustic signal emission from DNA synthesizer.

wavelet transforms. Furthermore, since the properties of the solenoid valves are assumed to be known to the attacker, he/she can specify the minimum distance between peaks to increase the accuracy of the peak detection algorithm.

Signal segmentation. Since an attacker has access to the user manual for the DNA synthesizer, he/she knows duration of each stage. Then, using the peak detection algorithm and the timing data from the user manual, the attacker can segment the nucleotide delivery stage. Since all the other stages of the synthesis remain the same for adding each new base, an attacker only needs the base delivery stage to reconstruct the sequence. Although this step can be done manually for shorter sequences, an attacker who wishes to reconstruct long DNA sequences may consider more sophisticated techniques, such as Hidden Markov Models (HMMs) [31] or Long Short-Term Memory (LSTM) neural networks [59], which have historically been used in voice recognition, to track the different states of the machine. Since the relative distance between the peaks is a known constant (as is described in the user manual), both of these models will achieve high accuracy; however, our experience has shown that neither of these models is perfect, and that a successful attacker will need to manually segment the data to obtain 100% accuracy. 100% accuracy is needed, since any error in this stage will jeopardize the attack model's subsequent steps.

Feature extraction. As shown in Figure 2.6, the base delivery segment of the signal consists

of three sections: the first peak, which is the result of opening the valve that controls the flow of a certain base to the output column; a longer section in which the pipes deliver the base; and a final peak, which is the result of closing the valve whose opening generated the first peak. Since the duration of each solenoid valve operation is known, the attacker can divide the delivery segment into three sections and engineer a specific set of features for each. Extractable features range from simple calculations such as the standard deviation of the signal to complex calculations such as coefficients of Fourier and wavelet transforms. Since the length of the signal is short for a delivery segment (less than 5.5 seconds in our experiments), we can assume that the attacker has enough computational power to calculate any of these features for all three sections of the delivery segment. However, extracting all possible features will significantly affect the convergence rates of the classifiers that will use them. Hence, in the training phase, the attacker creates models using a subset of all available features, either by feature projection (e.g. PCA [137], LDA[89]) or feature selection (e.g. [24, 9]). Based on our experiments, the latter approach works much better for acoustic side-channel attacks on the DNA synthesizer because even small environmental background noises mask most of the useful features when PCA or LDA projects them onto lower dimensions. The outcome of this stage is the conversion of an acoustic signal $(\boldsymbol{A_{i_1}, A_{i_2}, \ldots, A_{i_k}})$ into a set of features $(\boldsymbol{f_{i_1}, f_{i_2}, \ldots, f_{i_l}})$ with $\boldsymbol{l \ll k}$.

Nucleotide base classifier. In this stage, the attacker selects and trains the best classification algorithm to estimate the function $(\boldsymbol{S_i = \hat{f}(f_{i_1}, f_{i_2}, \ldots, f_{i_l}, \theta)})$ that correlates a given set of features to one of the four nucleotide bases. To find the best algorithm, he/she trains multiple classifiers such as neural networks [58] and random forests [17] and calculates the accuracy of each classifier. Each of these functions will have a certain method of training $\boldsymbol{\theta = \{\theta_1, \theta_2, \ldots, \theta_m\}}$, where $\boldsymbol{m}$ depends on the type and architecture of the classification algorithm used. The accuracy of a classifier is defined as the percentage of correct predictions divided by the total number of predictions made over the test data-set. To ensure that enough training samples have been provided to the models, the attacker monitors the

corresponding accuracy of predictions in terms of the number of training points. If the accuracy stops improving when new samples are added to the training dataset, the attacker can assume that the classifiers have converged. Once the attacker identifies the most accurate classification algorithms, then he/she can use an ensemble of algorithms to improve the accuracy even further. To do this, the attacker computes the normalized probability distribution for each base ($\boldsymbol{A}, \boldsymbol{G}, \boldsymbol{C}, \boldsymbol{T}$) for each classification algorithm and selects the most probable base as a result. The caveat of having a classifier only predict a single nucleotide base is that an attacker still needs to reconstruct the whole chain, which can introduce additional complexities. In order to tackle this issue in our attack methodology, we propose using an algorithm that produces the k-best sequences of oligonucleotides based on the output of the nucleotide classifier. The details of the k-best algorithm are presented in Section 2.7. In the training phase, an attacker constructs a nucleotide classifier; in the attack phase, the attacker can use the k-best sequencing algorithm, along with domain-specific post-processing, to reconstruct the sequence. Finding the first best sequence is trivial since the attacker only needs to choose the type with the highest probability for each delivery to achieve the highest confidence in the whole sequence prediction. However, finding the next best sequences are not straightforward, and he/she can use the *K-best DNA sequences* algorithm.

Post-processing. The classification algorithms described in the previous stage provide results based on the features present in the given segment of the signal; they ignore any relation of the current base delivery to past or future base deliveries. In practice, the order of the bases in an oligonucleotide sequence follows certain rules based on the synthesis technology, machine capabilities, and the specific domain for which it is being synthesized. For instance, authors of [100] identify three major limitations for DNA synthesis. The first limitation involve homopolymers, which are repeated sequences of the same base; this eliminates the chance of synthesizing an oligonucleotide sequence more than $\sim$10 consecutive instances of the same base as a substring. The second limitation is that a reasonable ratio of G and C bases should always appear in the sequence. The last limitation involves secondary

structures, in which an oligonucleotide sequence contains multiple complementary subsequences that may bind to one another, creating a physical loop. In addition, domain-specific knowledge can improve the accuracy of predictions. For example, there are well-known sequences in synthetic biology that are responsible for certain functions. An attacker with this knowledge will be able to correct errors in the algorithm if he/she notices similarities in the extracted sequences to those mentioned. The same can be found in most other applications of synthesized DNA. For example, in the case of storing data in DNA molecule, if the data is coupled with error detection/correction bits, then identifying the errors will be possible for the attacker as well.

The classification and domain-specific post-processing schemes report a reconstructed oligonucleotide sequence, which the attacker has effectively stolen. Further laboratory experiments can then confirm the correctness of the reconstructed sequence. If the attacker concludes that the reconstructed sequence is incorrect, he/she will want to consider other probable sequences, until he/she discovers a reconstructed sequence that he/she believes to be correct.

2.7 K-best DNA sequences

A nucleotide base classifier predicts $\boldsymbol{q = 4}$ possible output classes (A,G,C,T) for each base. The classifier first estimates the conditional probability distribution of possible outputs ($\boldsymbol{Y = \{c_1, ..., c_q\}}$) for the set of given input features $\boldsymbol{f}$. Then, the classifier reports the class $\boldsymbol{S = c_k \in Y, k \in (1, 2, \ldots, q)}$ with the highest probability as the result:

$$\boldsymbol{f} \text{ is assigned to class } \boldsymbol{c_k} \Longleftrightarrow \boldsymbol{p(c_k|f) \geq p(c_r|f) \forall k \neq r.} \tag{2.4}$$

where $\boldsymbol{r \in (1, 2, \ldots, q)}$. The confidence of a classification algorithm for a certain prediction is defined to be equal to the probability of the predicted class. Since the prediction for each

base delivery is independent from the others, the classifier computes the *confidence* value for the sequence as

$$\boldsymbol{Confidence} = \prod_{\boldsymbol{i=1}}^{\boldsymbol{n}} \boldsymbol{p}(\boldsymbol{c}_{\boldsymbol{k}_i}|\boldsymbol{f}_i), \tag{2.5}$$

where $\boldsymbol{S_i} = \boldsymbol{c}_{\boldsymbol{k}_i}$ is the predicted class for the $\boldsymbol{i^{th}}$ nucleotide base, based on the input features $\boldsymbol{f_i}$. The *confidence* defined in Equation 2.5 represents the chance of predicting the target sequence with 100% accuracy. An attacker would want to maximize this value. Choosing a class such that $\boldsymbol{p}(\boldsymbol{c}_{\boldsymbol{k}_i}|\boldsymbol{x_i}) \geq \boldsymbol{p}(\boldsymbol{c}_{\boldsymbol{r}_i}|\boldsymbol{x_i}) \forall \boldsymbol{k_i} \neq \boldsymbol{r_i}$, will maximize the *confidence* value, thereby increasing the probability that the predicted sequence exactly matches the original sequence. However, as explained in the previous section, there exist scenarios where the attacker would prefer more candidate sequences in addition to the best-predicted one. In response, we propose an algorithm to predict the K-most probable orders of bases in a sequence by keeping the value of *confidence* as close as its possible to its maximum value. Our algorithm is inspired by the Viterbi algorithm [43] which is commonly used to find the most probable sequence of hidden states in an HMM for a given sequence of observations.

Algorithm 1 accepts as input a two-dimensional array $\boldsymbol{P}$ which contains the conditional probability distribution of the four base types with $\boldsymbol{n}$ delivery stages. The value of array entry $\boldsymbol{P(i,j)}$ is equal to $\boldsymbol{p}(\boldsymbol{c_j}|\boldsymbol{f_i})$ where $\boldsymbol{c_j} \in$ {A, G, C, T} and $\boldsymbol{f_i}$ is the given input (set of features $[\boldsymbol{f}_{i_1}, \boldsymbol{f}_{i_2}, \ldots, \boldsymbol{f}_{i_l}]$) to the classifier for the $\boldsymbol{i^{th}}$ nucleotide base prediction. Algorithm 1 converts $\boldsymbol{P}$ into a Directed Acyclic Graph (DAG). The first step is to instantiate two dummy nodes to represent the beginning and end of the sequence. Second, four nodes with {A, G, C, T} labels are added as a layer between the start and end nodes to represent the four possible outputs of each classification step. The start node is connected to the four nodes labeled in the first layer by directed edges with weights set to $\boldsymbol{p}(\boldsymbol{c_j}|\boldsymbol{f_i})$, where $\boldsymbol{c_j}$ corresponds to the label of the destination node. The process repeats iteratively to add subsequent layers: the nodes in layer $\boldsymbol{i}$ are connected to the nodes in layer $\boldsymbol{i+1}$ by instantiating a directed

edge with a weight equal to $\boldsymbol{p(c_j|f_{i+1})}$. Directed edges with weight **1** are added from the four nodes in the final layer to the end node. The DAG enables simple and intuitive way to calculate the confidence of the reconstructed sequence.

Algorithm 1: DAG generation algorithm.

Input: Probability distributions of base type prediction ($\boldsymbol{P}$)

Output: Directed acyclic graph ($\boldsymbol{G}$)

```
// node (<label>,<index>)
// edge (<from_index>,<to_index>,<weight>)
// Order of probability distribution rows in P :  AGCT
```

Procedure *GenerateDAG*

- $\boldsymbol{n\_deliveries} \leftarrow$ length of $\boldsymbol{P}$
- $\boldsymbol{n\_nodes} \leftarrow 4 \times \boldsymbol{n\_deliveries}$
- `// Creating the nodes`
- $\boldsymbol{G} \leftarrow$ node(start,-1), node(end,n_nodes)
- **for** $\boldsymbol{i = 0; i < n\_deliveries; i = i + 4}$ **do**
  - $\boldsymbol{G} \leftarrow$ node(A,i),node(G,i+1),node(C,i+2),node(T,i+3)
- `// Adding the first and last layer edges`
- **for** $\boldsymbol{i = 0; i < 4; i++}$ **do**
  - $\boldsymbol{G} \leftarrow \text{edge}(-1, i, P(0, i))$ `// from source`
  - $\boldsymbol{G} \leftarrow \text{edge}((n\_nodes - 1) - i, n\_nodes, 1)$ `// to end`
- `// Adding internal layers edges`
- **for** $\boldsymbol{t = 0; t < n\_nodes - 4; t = t + 4}$ **do**
  - $\boldsymbol{i\_offset} \leftarrow \boldsymbol{t}$
  - $\boldsymbol{j\_offset} \leftarrow \boldsymbol{t + 4}$
  - **for** $\boldsymbol{i = 0; i < 4; i++}$ **do**
    - $\boldsymbol{i\_idx} \leftarrow \boldsymbol{i\_offset + i}$
    - **for** $\boldsymbol{j = 0; j < 4; j++}$ **do**
    - j_idx $\leftarrow \boldsymbol{j\_offset + j}$
    - $\boldsymbol{delivery\_id} \leftarrow \frac{t}{4} + 1$ `// next delivery`
    - $\boldsymbol{G} \leftarrow \boldsymbol{edge(i\_idx, j\_idx, P(delivery\_id, j))}$
- **return** $\boldsymbol{G}$

Assumption 1: Algorithm 1 generates a DAG.

Remark 1: Algorithm 1 generates a graph layer-by-layer and only adds edges between layers $\boldsymbol{i}$ and $\boldsymbol{i+1}$.

Assumption 2: A path from the start to the end node in the DAG represents a candidate reconstructed sequence. The *confidence* of the corresponding sequence is equal to the product of the weights of the edges along the path.

Remark 2: A path in the DAG emanating from the start node will pass through each layer exactly once. Choosing the node's label as be the delivery base type yields a nucleotide sequence whose length is equal to the number of layers in the graph. All input edges to a node with a given label will weight equal to the probability of that label being correct for its corresponding base delivery. Considering that the value of the last edge in the path, which terminates at the end node, is **1**, the product of the weights on the path is equivalent to the definition of the *confidence* shown in Equation 2.5.

We first call Algorithm 1 to generate DAG $\boldsymbol{G}$ from input $\boldsymbol{P}$. We then replace all the weight values with their corresponding $\boldsymbol{log}$ base values for mathematical simplicity in later steps. We also define the length of a path to be equal to the summation of edge weights on the path. In this case, the k longest paths in the graph will represent the k-most probable sequences. Notice that the summation of the $\boldsymbol{log}$ of a set of values is equal to calculating the log of the multiplication of those values.

There exist multiple algorithms that compute the K longest paths from a source node to sink node in a DAG [34]. The algorithm presented in [35] achieves an optimal asymptotic time complexity of $\boldsymbol{O(m + nlogn + k)}$, where $\boldsymbol{n}$ is the number of nodes and $\boldsymbol{m}$ is the number of edges in the DAG. This algorithm has an $\boldsymbol{O(1)}$ time complexity per pathfinding attempt, after a preprocessing stage that uses Dijkstra's Algorithm to identify the shortest path. This is a substantially more efficient than a brute-force approach, which would enumerate all $\boldsymbol{4^n}$

length-$\boldsymbol{n}$ base sequences and compute their *confidence* values in $\boldsymbol{O(n)}$ time per sequence, yielding an overall time complexity of $\boldsymbol{O(n4^n)}$.

Table 2.1: Probability distribution of base types for three consecutive deliveries.

Base Name	Delivery #1	Delivery #2	Delivery #3
A	**0.9**	0.03	0.12
G	0.05	**0.8**	**0.4**
C	0.01	0.15	0.35
T	0.04	0.02	0.13

Table 2.1 presents a sample probability distribution of base types for a DNA synthesis procedure consisting of three delivery stages. Based on this table it is easy to infer that the most probable sequence is AGG with *confidence* of $\boldsymbol{0.9 * 0.8 * 0.4 = 0.288}$. However, since the probability of delivering base C in the last stage is very close to the probability of delivering base G, if the sequence does not support our requirements, then we would intuitively consider the sequence AGC. If the attacker determines that both AGG and AGC are incorrect sequences, then he/she would turn to Algorithm 1 to generate the DAG shown in Figure 2.7. A top-11 analysis of the DAG generates the following sequences, in order: AGG, AGC, AGT, AGA, ACG, ACC, ACC, ACT, ACA, GGG, GGC.

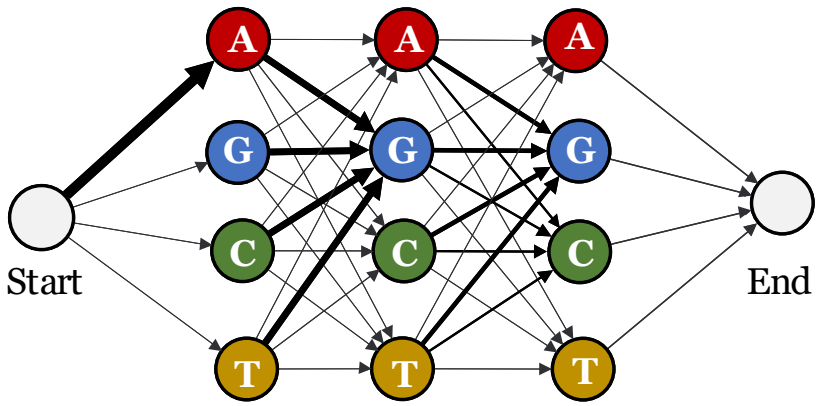


Figure 2.7: A DAG corresponding to the probability distribution provided in Table 2.1 (Thicker arrows represents higher probabilities for the destination nodes.)

2.8 Results and case study

This section presents the experimental results obtained by implementing the proposed attack methodology on an Applied Biosystems Inc. (ABI) 3400, one of the most widely used commercial DNA synthesizers. To validate this choice, we contacted a DNA synthesizer sales expert who stated: *"I strongly recommend you consider either the ABI 394 or the ABI 3400. They are by far the workhorses of the industry. 95% of our customers use the ABI 394."* - [Email]). This section also presents a test cases where we reconstructed the complete oligonucleotide sequences using the proposed K-best DNA sequences algorithm and post-processing steps.

2.8.1 Test Bed

As shown in Figure 2.3, the experimental setup consists of an AB 3400 DNA Synthesizer [6] and a Zoom H6 audio recorder to acquire the acoustic signal. We record the signals through three channels simultaneously at a sampling frequency of 48 kHz with a resolution of 24 bits per sample. For every DNA synthesis run, we randomly place a recorder with two condenser microphones near the DNA synthesizer (on top of the machine or on the setup desk, no further than 10 cm from the machine). We also use a contact microphone to record acoustic signals with almost no environmental noise. Our attack methodology is implemented in Python 3.6. We use the tsfresh [24] library package for feature extraction; scikit-learn [105] for profiling model generation; and networkx [53] for modeling the network discussed in Section 2.7. We also use MATLAB [94] to represent the Short-Time Fourier Transform (STFT) results.

2.8.2 Evaluations Assumptions

We used the Zoom H6 portable handy recorder tool kit, which has publicly available coil condenser microphone characteristics, to carry out our attack. The Zoom H6 is similar to an iPhone 4, which contains two similar internal microphones. As a ubiquitous consumer product, an iPhone 4 placed in a discreet location near a DNA synthesizer, would seem innocuous to the typical user of such a machine, and could easily collect days' worth of data without detection.

We assume that the DNA synthesizer is used exclusively for oligonucleotide synthesis. DNA synthesizers have various cycle scripts for producing different polymerases. If a user intends to synthesize DNA, the same script should be used, without modification, regardless of the target sequence; modification of the cycle script can cause erroneous synthesis behavior. (Over six months of studying this machine in an active biomedical laboratory, we observed that the settings were never changed. We verified that the cycle script that was run repeatedly by different users always matched the cycle script for oligonucleotide synthesis in the AB 3400 synthesizer manual. This assertion was subsequently confirmed by direct communication with the machine operators).

Although the AB 3400 can synthesize four columns in parallel, in practice, the deliveries to output columns do not occur simultaneously, as described in Chapter 6 of the user manual [6]; for each output column, the same solenoid valves and pipes are used. The only difference is setting the *Front Reagent Block* to a different output column before the next delivery cycle. Since the same script is used for every column, extracting the delivery stages for each output is straightforward. To avoid unnecessary complication, we focus on single output column DNA synthesis.

2.8.3 Training and Evaluation

The attack model described in Section 2.6 consists of functions ($\hat{\boldsymbol{f}}(\boldsymbol{.}, \boldsymbol{\theta})$) that need to be trained before they are used for a specific synthesizer. Since the objective of each model is known, we use supervised learning to estimate the functions. We initially synthesized seven different 60-base oligonucleotide sequences, each consisting of 15 A's C's G's and T's in varying orders. An attacker could increase the number of synthesis runs if the classification results do not converge, however, as shown in this section, the initial runs were sufficient. Each synthesis run took 7 hours, 29 minutes, and 53 seconds. As an attacker, we label the acquired signals into different stages: 'initialization', 'repetitive cycle', and 'base delivery'. The labeling is possible because the user manual for the synthesizer machine lists the operations which take place during synthesis and the corresponding duration of each operation [6]. Based on the manual, it is easy to infer that the DNA synthesizer initialization stage takes approximately 787 seconds and only occurs at the beginning of the synthesis procedure. The 'repetitive cycle' stage takes approximately 463 seconds, and the 'base delivery' stage takes approximately 5 seconds inside the 'repetitive cycle' stage.

To pre-process the signals, first the STFT spectrum is calculated. The analysis of the acquired signals reveals there is no difference in the magnitude of frequency components below 300 Hz between the portion of the signal which belongs to environment noise versus actual DNA synthesis (see Figure 2.8). Hence, we chose 300 Hz as a cutoff frequency and filter the signal frequency components below this limit using a high-pass filter. We apply Equation 2.3 to normalize the signal before further processing.

In Figures 2.6 and 2.8, each valve operation produces an audible click from the machine which is clearly visible as a peak in the recorded waveforms. When operational, the DNA synthesizer executes the cycle script, for which the sequence and duration of valve operations during the synthesis run are known. By correlating valve operation timings between the

Table 2.2: Accuracy of the classification models.

Classifier	**Settings**	**Accuracy (%)**			
		Raw Signal	All Features	Selected Features	Improved Selected Features
AdaBoost [55]	_	18.54	39.58	57.22	69.46
Support Vector Machine (SVM) [26]	kernel= 'linear', penalty_error(C) = 0.025	**26.8**	17.63	57.77	84.05
Naive Bayes [147]	_	12.12	27.63	58.61	75.31
Neural Network[58, 119]	architecture= 'fully connected feed forward', activation_func='relu', num_hidden_layers=100, num_training_iteration=1000, solver = 'adam'[72]	14.59	34.99	**64.44**	87.51
Random Forest [17]	num_estimators=100	23.4	46.66	60.69	78.46
Voting [107]	classifiers = {RandomForest, NeuralNetwork}	24.18	**51.25**	62.5	**88.07**

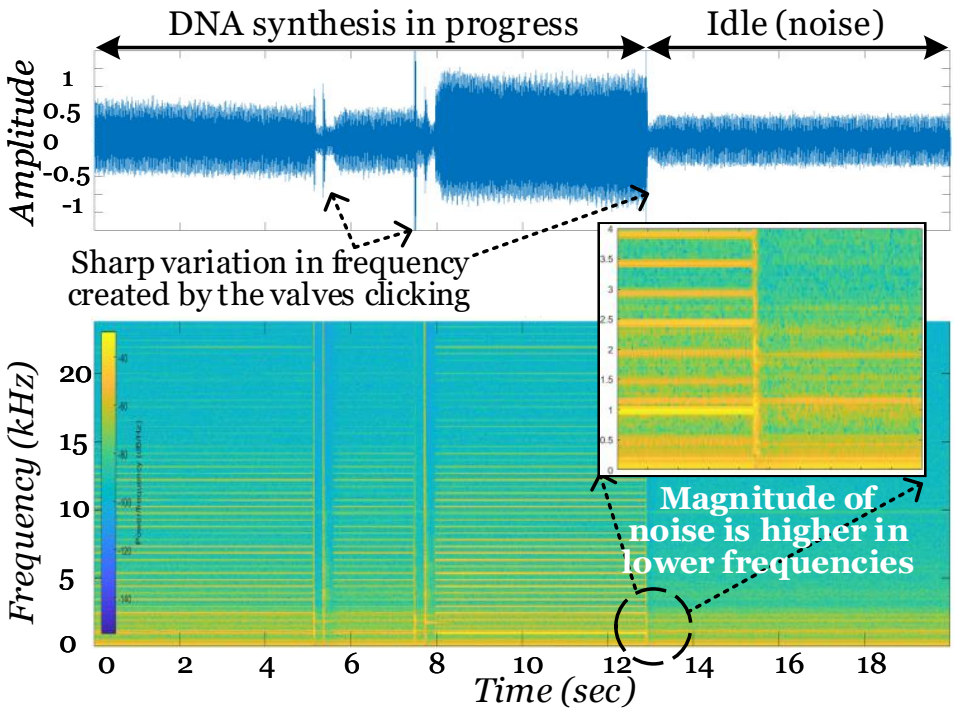


Figure 2.8: Recorded acoustic signal from the DNA synthesizer during synthesis followed by an idle state (top). Short-Time Fourier Transform (STFT) of the corresponding signal (bottom).

waveform and cycle script, sections of the waveform can be labeled with the corresponding valve operations. On the AB 3400 DNA synthesizer, the base delivery stage contains six valve operations with a unique sequence of timings relative to other stages in the synthesis process. With practice, the base delivery stage can be visually identified and extracted from the waveform. Additionally, the specific valve operation that delivers the base always occurs at the same time in the delivery stage. Therefore, this operation can be extracted and used for base identification in the classification step of the attack model. Since a human can manually extract the base delivery operation using these features, we implemented an algorithm to extract the base deliveries using the same process. First, we identify the peak locations in the signal using continuous wavelet transforms [30]. Then, based on the cycle script, the algorithm identifies the sequence of distances that correspond to the base delivery stage. For each stage, the algorithm references the cycle script again and extracts the segment that corresponds to the base delivery valve operation.

Once the base delivery segment is extracted from the signal (similar to what is shown in Figure 2.6), we train six classifiers as shown in Table 2.2. Feeding the raw base delivery acoustic signal to these classifiers results in random classification ($\leq$ 25% accuracy), so a feature extraction step is required before classification. We select the best set of features to be used for classification in two steps. First, we extract all the features introduced with the tsfresh [24] library. These features consist of the time domain, frequency domain, and wavelet-based features. Next we calculate the significance of each feature and carry out multiple test procedures [11] to select the most relevant features with the lowest dependency score [24].

This procedure reduced the number of features from 57,018 to 75 (*selected features* in Table 2.2). The selected features include the magnitude of the Fourier transform components of the input signal in certain frequencies as well as the autocorrelation of the signal with a lag of 2 and 3 samples. The selected set of features matches what was expected from the feasibility analysis of the attack described in Section 2.5. The structural differences between the pipes used for different base deliveries causes each base delivery to generate slightly different frequencies. However, for practicality, the tsfresh library with default settings does not generate all of the frequency components. To discover all of the possible features in the frequency domain while keeping computational resources fixed, we calculate the frequency components with an accuracy of 200 mHz, but only at frequencies above 300 Hz with local peaks in the frequency transform (see Figure 2.8). We reran the same feature selection algorithm and identified 310 features within those frequency bands (*improved selected features* in Table 2.2).

To ensure that the amount of training data is sufficient for the models, we analyze the prediction accuracy based on the number of base delivery samples in the training dataset. We selected *AdaBoost* [55], *linear Support Vector Machine* (SVM) [26], *Naïve Bayes* [147], *Neural Network* [58], and *Random Forest* [17] classifiers to estimate the function $\hat{\boldsymbol{f}}(\boldsymbol{.}, \boldsymbol{\theta})$. We

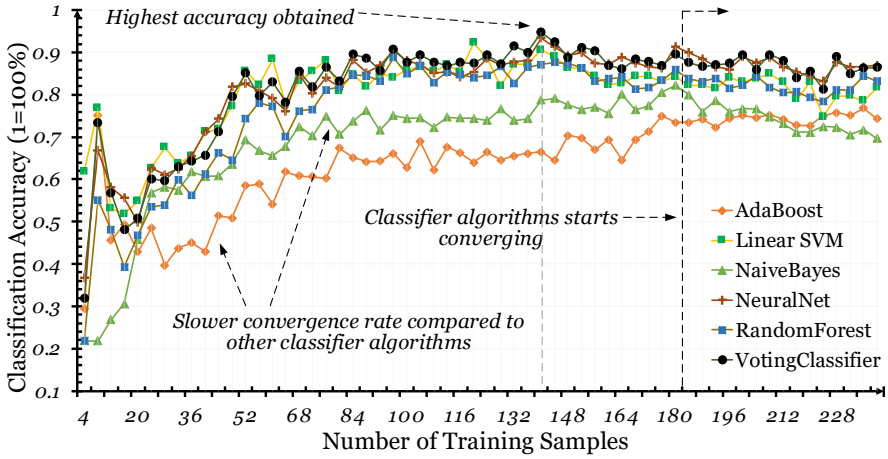


Figure 2.9: Learning curve of various classifiers for nucleotide base classification.

also implemented a weighted majority rule voting [107] based ensemble that uses *random forest* and *neural network* as the base classifiers; for simplicity we set the weights of both classifiers to be equal. As shown in Figure 2.9, around 200 samples is sufficient to train the models to achieve maximum accuracy when classifying nucleotide bases based on the selected features. We used 80% of the dataset for training and 20% for validation, coupled with 10-fold cross validation [73] to produce the results reported in Figure 2.9 and Table 2.2. The reported accuracy numbers shown are averaged across the 10 folds. Figure 2.9 and Table 2.2 show that the majority rule voting-based ensemble of classifiers achieves faster convergence and higher classification accuracy than the other classifiers for the *improved selected features*. This can be explained by the fact that each individual classifier effectively searches a space $\mathcal{H}$ of hypotheses in search of the hypothesis with the highest accuracy. Different classifiers identify different hypotheses with similar, if not the best overall, accuracy. If each classifier's hypothesis has uncorrelated errors with error rates exceeding 50%, then the voting-based ensemble method is likely to increase overall accuracy [29]; however, this improved performance is obtained at the cost of increased computing power and training time for the ensemble of classifiers used in voting.

The next step in the attack model, as discussed in Section 2.6, is post-processing the recon-

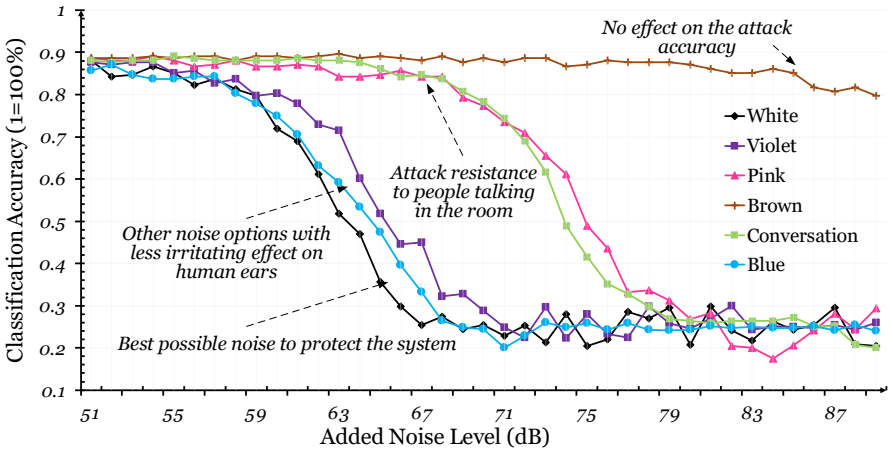


Figure 2.10: The impact of added noise effects on the side-channel attacks against DNA synthesizers.

structed DNA sequence with domain-based knowledge after generating the K-best sequences. Since the accuracy of the classifiers is reported based on random delivery samples, we did not integrate this stage into our initial experiments. Instead, we evaluate the value of such techniques for reconstructing meaningful DNA sequences in Section 2.8.6.

2.8.4 Noise Effect on Accuracy

Although the pre-processing stage used in the attack reduces the effect of environmental noise and normalizes the acquired signals, significant ambient noise might obfuscate the information leaked in the side-channels, reducing the effectiveness of the attack. Hence, it is important to evaluate the robustness of the trained models against potential environmental noises. To this end, we generate six types of different noises and add them to the recorded raw signals for the test samples: brown noise, which has very high intensity in lower frequencies ($\boldsymbol{\beta} = \mathbf{2}$); pink noise, which has high intensity in lower frequencies ($\boldsymbol{\beta} = \mathbf{1}$); white noise, which has same intensity in all frequencies ($\boldsymbol{\beta} = \mathbf{0}$); blue noise, which has high intensity in higher frequencies ($\boldsymbol{\beta} = \mathbf{-1}$); violet noise, which has very high intensity in higher frequencies ($\boldsymbol{\beta} = \mathbf{-2}$); and conversation noise, which is a recorded conversation between two persons

(the power spectral densities of the color noises are proportional to $\frac{1}{f^{\beta}}$).

As shown in Figure 2.10, adding any noise with a high-decibel sound pressure level (SPL) reduces the attack model accuracy. We observe that the noises, which have a medium to high emphasis on their higher frequency components, can mask the leaked signal with lower SPL. If a noise generator is added as a countermeasure against acoustic side channel attacks, then to be effective, it must generate high-frequency noises in close proximity to the synthesizer. General noises in the laboratory, such as employee conversations or air conditioner emission (pink), are unlikely to be effective countermeasures.

2.8.5 Microphone Distance Effect on Accuracy

The SPL induced by the DNA synthesizer is inversely proportional to the distance between the DNA synthesizer and the microphone. If an SPL is measured to be $\boldsymbol{L_1}$ at distance $\boldsymbol{r_1}$, it will reduce to $\boldsymbol{L_2}$ at distance $\boldsymbol{r_2}$ according to the following equation [40]:

$$\boldsymbol{L_2 = L_1 - |20log(\frac{r_1}{r_2})|.} \tag{2.6}$$

We used this equation to evaluate the effect of microphone distance to the target DNA synthesizer. During the delivery stages, our experiments yielded an average SPL of **81.15** dB and **77.1** dB for the contact and condenser microphones. We assume that the acoustic signal collected by the contact microphone is free of environment noise. In this scenario, decreasing the SPL of the recorded signal by the contact microphone while adding a constant room noise is equivalent to placing the recorder at a further distance.

Figure 2.11 shows the result of decreasing the SPL of a recorded signal near the contact microphone in 1 dB increments while keeping the added room noise level constant. Since we know the SPL of the signal within 10 cm of the machine, we add a secondary horizontal axis

to the top of the chart to show the distance of the microphone from the DNA synthesizer using Equation 2.6. If we assume that there is no noise in the environment, the degradation of the SPL of a signal at a further distance would have no effect on the accuracy of a given voting classifier, since a normalization step can revert the SPL back to its expected value; in practice, environment noise is unavoidable. As shown in the figure, increasing the distance decreases the accuracy for all types of classifiers. Once the distance between the microphone and DNA synthesizer exceeds 0.7 meters, the accuracy of all classifiers noticeably degrades, although the exact amount of degradation varies among the classifiers. For example, the Neural Network classifier has a higher accuracy than the Random Forest classifier in close proximity to the DNA synthesizer; however, the Random Forest classifier has higher accuracy when the microphone is placed further away from the synthesizer. The ensemble voting classifier used in the attack methodology can often reach and exceed the accuracy of these two classifiers individually, regardless of the distance.

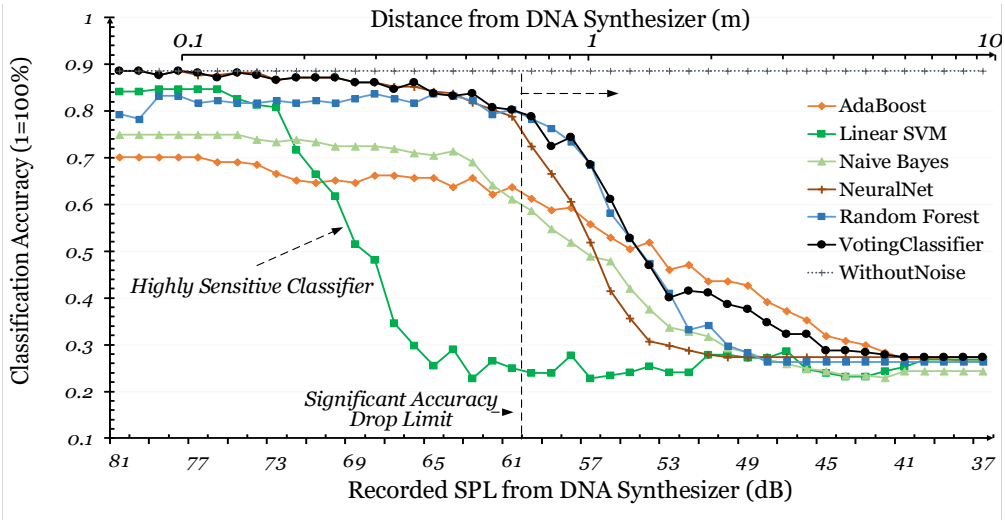


Figure 2.11: Effect of microphone distance on various classifiers used in the attack methodology.

Table 2.3: Results for reconstructing the test cases.

Case #	**Original** Oligonucleotide sequence **Predicted** Oligonucleotide sequence	Sequence Length	Accuracy (%)	BLAST match	Number of guesses to have **N or less** mispredicted amino acid				Brute Force Complexity
					N=3	N=2	N=1	N=0	
1	CGCAA**G**TACTCCTG**C** CGCAA**T**TACTCCTG**A**	15	86.67	Yes	1	1	3	21	15x4^15
2	GGAATAGTAGAAG**AA**TGCTGCACAA**G**CATATGCAGCCTA**T**ACGAACTAGAAGAC**T**ACTGCGAC GGAATAGTAGAAG**CG**TGCTGCACAA**T**CATATGCAGCCTA**C**ACGAACTAGAAGAC**G**ACTGCGAG	63	**90.48**	Yes	12	29	>100	>100	63x4^63
3	TGGCGACAT**G**ATAACCCGTCGGA**G**GATCCGGG**G**CG**G**G**GG**CACCTC TGGCGACAT**T**ATAACCCGTCGGA**T**GATCCGGG**T**CG**T**G**TT**CACCTC	45	86.67	Yes	1	3	19	>100	45x4^45
4	TTTT**T**CGACCGGT**A**T**GA**T**T**CCGCCCGTGACCCAGGACGCTTGCTT TTTT**G**CGACCGGT**C**T**TC**T**G**CCGCCCGTGACCCAGGACGCTTGCTT	45	88.89	Yes	1	3	35	>100	45x4^45

The bold colors with larger font size in oligonucleotide sequence represents the misclassified nucleotide bases

2.8.6 Test Case Evaluation

This section evaluates the impact of our proposed attack methodology by reconstructing four DNA sequences which were synthesized using the DNA synthesizer shown in Figure 2.3. To ensure fairness, the test cases considered here were chosen by an author different than the attacker, and the original sequence was provided to the attacker only after the results were submitted for comparison. The attacker makes only one assumption: the sequence is later going to be implanted in a living organism to create some type of protein. Every three oligonucleotide bases translate to a certain amino acid (the building block of a protein) based on this table [123]. As discussed in Section 2.6 this assumption may help during post-processing. The DNA sequence test cases evaluated are:

1- Conotoxins. We synthesized part of a DNA sequence that translates to a lethal protein: conotoxin. Conotoxins are recognized by the US Government as potential agents of bioterrorism [42]. We assume that the attacker is potentially a government agency or a similar entity.

2- Human insulin. We synthesized the DNA that encodes the alpha chain of human insulin. Insulin was originally extracted from pig pancreases; in 1979 Goeddel *et al.* added DNA encoding human insulin to bacteria to produce actual human insulin [51]. This was the first major drug produced by synthetic biology and led to the founding of Genentech,

a multi-billion-dollar pharmaceutical company.

3 & 4- Peptide. We synthesized two DNA sequences which encoded peptides that were isolated by *in vitro* selection to bind the protein target streptavidin. The peptides have been characterized in [77] and function as high-affinity ligands to the protein target. These peptides could be used as protein affinity tags to purify other proteins from crude cellular lysate or cell-free translation systems.

For the test cases, we use the majority voting rule-based ensemble of classifiers trained in Section 2.8. We first collect the acoustic signal generated by the machine while synthesizing the aforementioned sequences. After preprocessing and background noise elimination, we manually ensure that the correct signal delivery segments are extracted from the given signals. After segment extraction, the model extracts the same set of features that were used for training. Next, the trained classifier predicts the probabilities for each base type for each delivery. As shown in Table 2.3, choosing the base type with the highest probability results in average accuracies close to the classifier accuracy which is calculated in Section 2.8. Since we assume that the reconstructed sequences will be used in a biological application, we also provide the number of errors in terms of mispredicted amino acids. We use the K-best sequence algorithm described in Section 2.7 to show the number of trials that an attacker would need to reconstruct the original sequence with perfect accuracy. The results show that, for short sequences, it is possible to reconstruct the sequence with a reasonable number of trials. However, as sequences grow in length, due to the limited accuracy of the classifiers, finding the exact location of the errors in the sequence becomes more difficult. Hence, we conclude that if the number of bases is long enough, achieving 100% accuracy for reconstructing the whole sequence with the given attack methodology solely based on acoustic side-channel data may not be possible. However, failing to reconstruct the sequence with perfect accuracy does not translate to the absolute confidentiality of the data passed to the machine. An attacker can acquire enough information to determine the intended purpose

of a reconstructed DNA sequence even if some base predictions are incorrect. As it turns out, reading the sequence of bases in a DNA molecule have always been error-prone.

Publicly available software such as BLAST [5] store DNA sequence, their functionality, and can readily determine the most similar known DNA sequences, along with their application, for a given amino acid sequence. If the attacker is a government agency, tools like BLAST may be used to determine if a hostile user is synthesizing DNA sequences that have structural similarities to known hazardous sequences.

In an industrial setting an attacker may work for a competitor whose objective is industrial espionage; in this case, the attacker can use tools like BLAST to derive the likely amino acid sequence of a protein being developed by the company under attack. To quantify the relevant information in the reconstructed sequences, we import amino acid sequences that correspond to the original and reconstructed test cases to the BLAST and then compare output reported for each sequence. Table 2.3 summarizes the result of this experiment. If BLAST reports a similar set of candidates for the original and the reconstructed sequences, Table 2.3 reports YES in the "BLAST match" column.

2.9 Discussion

2.9.1 Attack Cost and its Implications

We spent 56 hours collecting training data on the AB 3400 DNA synthesizer; during this time, human supervision was required for less than one hour in total. We dedicated 5 hours to understanding the structure of the AB 3400 and the scripts used for oligonucleotide synthesis. Manual segmentation of the signal took 20 minutes per synthesis run (less than 3 hours in total); an Intel Core i7-7820X CPU with 16 gigabytes of RAM extracted the features,

selected the best features, and trained 6 models in 14 minutes and 28 seconds. Although access to the DNA synthesizer was granted by its operator (a university laboratory) at no cost, the synthesis runs consumed $300 worth of raw chemical materials. Once the models are trained, the attack phase requires 20 minutes to manually segment 60 base deliveries and less than one second per delivery to predict the base.

These costs are negligible in comparison to the cost of real-world drug production in industry. In 2004, for example, the Bill and Melinda Gates Foundation dedicated $42.6 million to the development of an anti-malaria drug [12]. The recipients of the award published their initial results two years later, and completed the research project by the end of the third year. This particular drug was not patented and its recipe was made freely available as a humanitarian gesture; however, the key point is that the drug cost tens millions of dollars to produce, while an attacker could steal its DNA sequence (if kept a secret) in less than one week of time and at a cost of several hundred dollars. This could easily doom a for-profit private sector drug development project.

2.9.2 Limitations of Attack Methodology and Experiments

The AB 3400 DNA synthesizer is a widely used commercial product, but is not the only one on the market. The feasibility analysis in Section 2.5 implies that the proposed attack methodology is could be applied to any DNA synthesizer that employs solenoid valves and pipes for chemical delivery; future work will validate this attack methodology against similar machines.

This thesis used the same machine for training and validation of the attack model and evaluated its robustness to minute differences between the profiling system and target through the additional of artificial noise (Section 2.8.4). Further investigation is required to evaluate the effectiveness of the attack model when different AB 3400 machines are used for profiling

and targets of the attack.

2.10 Countermeasures

Secured structure: One way to prevent acoustic side-channel attacks is to ensure that all the physical components responsible for the delivery stage are similar: identical solenoids, pressure valves, and pipe lengths must be chosen, and they must be placed in a geometrically identical manner; for example, bends in fluid pipes must be identical to eliminate variations in accoustic emissions. Additionally, anti vibration pads (a.k.a. vibration isolators) could be integrated into the inner layers of the DNA synthesizer to reduce the intensity of any emitted observable acoustic noise.

Artificial noise: Redundant physical components may be added to introduce additional noise and decrease the signal to noise ratio, making it difficult to infer the cyber and physical states of the DNA synthesizer. Although intuitive, adding loud noise may bother employees who work in the same environment as the DNA synthesizer. Thus, proper methods, such as those described in Section 2.8.4, will be needed to search for low intensity noises that can mask information emitted from the acoustic side channel.

Delivery segment obfuscation: The DNA synthesizer's delivery stage is the critical point of vulnerability for this particular attack. A number of opportunities exist to potentially obfuscate this stage exclusively, such as adding redundant steps of varying time length, which are benign to the DNA synthesis process, and randomly selecting them prior to delivery; however, this may increase DNA synthesis time. Another possibility is to randomly select and execute steps unrelated to base delivery, such as cleaning other pipes or waste disposal, concurrently with base delivery. Although the system will remain observable, the cyber and physical-domain states will be obfuscated, which will increase the difficulty of using

data-driven approaches to infer the actual states.

Secured laboratory environment: The most effective practice to assure confidentiality of the synthesized DNA sequences is to prevent any visitor or unauthorized personnel from entering any room that contains a DNA synthesizer. Along similar lines, any unauthorized device found in the same room as a DNA synthesizer should be reported as a security threat. Furthermore, the cyber-security of every electronic device with recording capabilities that enters the laboratory environment, even if authorized, should be considered; the device itself may be compromised by a malicious adversary who can remotely activate acoustic signal recording.

2.11 Conclusion

We proposed and implemented a novel acoustic side-channel attack methodology on DNA synthesizers to steal the type and order of the bases which were synthesized. We tested our attack model against one of the most widely used DNA synthesizers and showed that ignoring such a confidentiality vulnerability can result in a significant research investment loss. We were able to predict the type of each base delivery with an average accuracy of 88.07%. We introduced a novel way to evaluate the effect of microphone distance without exhaustive experiments which revealed that high accuracy can be expected from the proposed attack methodology with up to a 0.7 m gap between the microphone and the DNA synthesizer. In addition, we showed how a reconstructed sequence can be leveraged using a post-processing approach in a biological domain (such as using the publicly available tool BLAST) to identify the protein that the original sequence intended to encode, despite any errors.

Chapter 3

HTnet: Transfer Learning for Golden Chip-Free Hardware Trojan Detection

3.1 Abstract

Design and fabrication outsourcing has made ICs vulnerable to malicious modifications by third parties known as hardware Trojans (HT). Over the last decade, the use of side-channel measurements for detecting the malicious manipulation of the ICs has been extensively studied. However, the suggested approaches often suffer from three major limitations: 1) reliance on a trusted identical chip (i.e. golden chip), 2) untraceable footprints of subtle hardware Trojans which remain inactive during the testing phase, and 3) the need to identify the best discriminative features that can be used for separating side-channel signals coming from HT-free and HT-infected circuits. To overcome these shortcomings, we propose a novel neural network design (i.e. HTNet) and a feature extractor training methodology that can be used for HT detection in run time. We create a library of known hardware Trojans and collect electromagnetic and power side-channel signals for each case and train HTnet to learn

the best discriminative features based on this library. Then, in the test time we fine tune HTnet to learn the behavior of the particular chip under test. We use HTnet followed by an anomaly detection mechanism in run-time to monitor the chip behavior and report malicious activities in the side-channel signals. We evaluate our methodology using TrustHub [117] benchmarks and show that HTnet can extract a robust set of features that can be used for HT-detection purpose.

3.2 Introduction

The CPSs are not only limited to digital machines with moving parts. An Integrated Circuit (IC) which hosts a cyber-domain software/algorithm is also considered to be a CPS by itself. Given the growing demand for low-cost ICs, companies tend to have their chips designed and fabricated by untrusted third-party entities over the globe. This has raised security concerns about intentional malicious modification of the integrated circuits, referred to as Hardware Trojans (HT).

The ICs are deployed in many sensitive applications, such as medical devices, critical defense technologies, and municipal support systems, which highlights the paramount importance of resolving this security issue. The attacker may tamper with the circuit in order to change the functionality, degrade performance, leak information, or deny service [121]. For instance, the malicious circuit manipulation in implantable medical devices can cause physical harm either by altering the functionality or through a Denial-of-Service attack. The HTs inserted in military devices, can leak sensitive data through side-channels and compromise the confidentiality of the information inside the chip such as encryption keys. In the last decade, HT detection has been deeply investigated in the literature; nevertheless, the proposed techniques have several shortcomings. The investigated techniques include destructive and non-destructive methods. Destructive methods have the IC decapsulated and delayered

to obtain the images of physical layers of the chip and utilize reverse engineering techniques to detect malfunction in the circuit. Although this method is able to provide high levels of confidence, it is costly and time-consuming, and it can authenticate only a single chip which is often not usable after the invasive process. Thus, the non-destructive approaches, which rely on the HT impact on host circuit functionality or side-channel parameters, are typically better choice if proven to be effective. HTs are designed to have a small area and power consumption. They usually contain two primary part: trigger and payload. After a series of particular events in external or internal signals, the trigger activates the payload to perform malicious behavior. An inactive Trojan does not alter the functionality and its effect on chip parameters is not significant enough to be distinguished from process variation and measurement noise.

Side-channel signal analysis is another approach for HT detection based on the malicious circuit side-effects on the side-channel parametric profile of the chip. In this method, side-channel parameters are measured and compared to the expected values to identify the presence of additional structure. Most of the side-channel based works depend on a trusted chip, called golden chip, to create the reference model for comparison. However, in practice, the golden chip does not exist since the integrated circuits supply chain is susceptible to HT insertion in any stage [142].

Various works in the literature have studied golden chip-free approaches to address this limitation. In [85], the authors propose the idea of utilizing the Process Control Monitors (PCM) and statistical side-channel fingerprinting to construct the reference for chip verification. It is assumed that the PCM are reliable since manipulating them is very difficult for attackers and any systematic modification of them results in notable deviation. Another approach introduced in [99] is self-referencing. When the HT is triggered, it creates uncorrelated temporal variation in the transient current signature of the chip which is detectable by side-channel measurement.

Due to the stealthy nature of HTs, they may evade detection in the IC testing phase with drastic consequences that are not acceptable in critical application. Thereafter, the idea of run-time HT detection is discussed to act as the last line of defense. The chip behavior is monitored during run-time to guarantee proper and secure IC operation. For instance, the method described in [33] provides an early alert for triggered HTs by monitoring performance counters. The advancement of semiconductor technology has brought about the issue of deterioration of circuit profile over time, known as circuit aging. This phenomenon can cause the run-time HT detection frameworks to fail since a predefined constant model cannot address the impact of the circuit aging on the circuit parameters and it is not considered in most approaches.

In this chapter, we develop a novel neural network design (i.e. HTNet) and a training methodology for HT detection in run time without the golden-chip requirement. We create a library of known HTs and collect electromagnetic (EM) and power side-channel signals for each case and train HTnet to learn the best discriminative features based on this library. Then, during the testing phase, we fine tune the HTnet to learn the behavior of the particular chip under test. We use HTnet followed by an anomaly detection algorithm in run-time to monitor the side-channel signals emitted from the chip and report malicious behaviors related to a triggered HT. The content of this chapter is provided from [38].

3.2.1 Research Challenges

In a nutshell, the state-of-art approaches for HT detection often experience the following challenges:

- Reliance on a reference golden chip, which in practice is costly to obtain or unavailable in some HT threat models such as untrusted IPs.

- Detection of HTs often requires activating them while the probability of triggering an HT during testing is very low due to their stealthy nature.
- For each new IC design, a new feature space needs to be defined to extract the most relevant features in the side-channel signals that can be used for detecting HTs.

3.2.2 Technical Contributions

This chapter makes the following technical contributions to address the aforementioned challenges:

- We devise a custom neural network architecture that performs the same or better than any available approach in the literature for HT detection.
- We propose a methodology to transfer the knowledge learned based on known HT benchmarks to a new circuit which may or may not contain an HT.
- With this work, we publish our collected electromagnetic and power side-channel data for hardware implementation of AES infected by various forms of HTs. We provide our source codes for our model implementation.[1]

3.3 Background

3.3.1 Related Works

The potential threat of HT has motivated many studies where side-channel analysis is a widely used strategy for HT detection. The idea is to measure side-channel parameters

[1] https://github.uci.edu/AICPS/HTnet

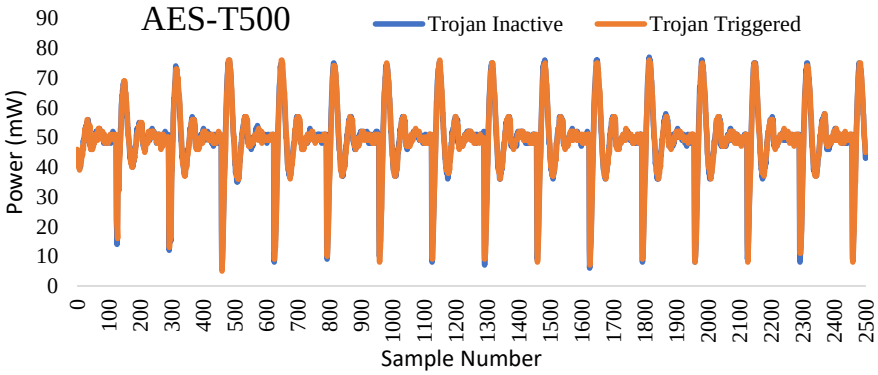


Figure 3.1: Two power time series samples for a benchmark.

including supply power leakage [111, 4, 108, 141], transient current [99], delay [68, 143], temperature[63], or EM emission[154, 129] and determine if the IC is contaminated by comparing the measurements with the expected reference values. Although in some side-channel analysis approaches, golden chip-free methods such as temporal self-referencing [99] are utilized for HT detection, they mainly depend on availability of a golden chip. To overcome this shortcoming, reference-free methods are introduced which rely on the other characteristic of the chip. The authors of [116] use controllability and observability analysis of gate-level netlist along with unsupervised clustering to provide the HT related signal lists and assert the insertion of the HT. In [112], the authors propose that they used deep learning to extract Trojan features from gate-level netlist and detect infected nodes by clustering the nodes. The approach explained in [154] modifies the fill cells in a standard cell library to be observable in an optical image taken through the backside of the chip. Using the optical image, the circuit layout is extracted and used malicious manipulation detection.

3.3.2 Time Series Classification and Anomaly detection

Feature Engineering Based Models

As depicted in Figure 3.1, distinguishing between an inactive HT and a triggered HT signal is not a trivial task as the differences are very small. The majority of the machine learning approaches used in the literature such as Support Vector Machines (SVM) define a distance function for comparing the similarity/dissimilarity between two signals. Given that a side-channel signal with $\boldsymbol{T}$ samples is defined as $\boldsymbol{X} = [\boldsymbol{x_1}, \boldsymbol{x_2}, ..., \boldsymbol{x_T}]$, the distance functions are typically in the form of $\boldsymbol{L_p}$ norm defined as,

$$\boldsymbol{L_p} = ||\boldsymbol{X} - \boldsymbol{X'}||_p = (\sum_{\boldsymbol{i}} |\boldsymbol{x_i} - \boldsymbol{x'_i}|^p)^{1/p} \tag{3.1}$$

where $\boldsymbol{p} = \boldsymbol{1}$ and $\boldsymbol{p} = \boldsymbol{2}$ denote Manhattan distance and Euclidean distance, respectively. However, in practice, the length of the two signals might be different, small shifts in the signal may occur, and it is computationally expensive to determine the $\boldsymbol{L_p}$ value for large values of $\boldsymbol{T}$. To overcome these challenges, various techniques have proposed using a set of features for representing the time series signal. Time domain features such as local min/max/average, frequency domain features extracted by Short-Time Fourier Transform (STFT), and features extracted by Discreet Wavelet Transform (DWT) have been among the most widely used features for the time series representation. Besides the spectral approaches, representative eigenvalue analysis methods such as Principle Component Analysis (PCA) [86] and Singular Value Decomposition (SVD) have been widely used in the literature to compress the signal and focus on the most relevant features in the signal. Often, some of these features are pruned with human supervision to select the best set of features. In most cases, the accuracy of the models heavily rely on the chosen features.

End-to-end Models

In recent years, deep learning has gained incredible popularity. In particular, Convolutional Neural Networks (CNN) have been widely used in various applications due to their capability of extracting the most relevant features under most scenarios automatically. The details of CNN formulation for time series classification is provided in [152]. Under this formulation, each convolutional layer acts as a filter with trainable weights that can adjust to the input shape during the training phase. In contrast to the other methods, neither training nor using CNN based models does not involve a separate feature-engineering step; for this reason they are referred as end-to-end models.

3.4 Methodology

In this section, we describe our methodology for designing and training a neural network to extract the most relevant features from side-channel signals in a compact form. Then, we use a self-referencing approach to detect anomalous behavior.

3.4.1 Library Construction

The first step in our methodology is to gather a dataset of side-channel signals. We use TrustHub [135] benchmarks that are implemented in the FPGA platform as the reference circuits. For each benchmark, we collect $\boldsymbol{N}$ EM and power traces under two scenarios: HT-inactive and HT-triggered. In the HT-inactive scenario, the HT is disabled to stay inactive during data collection. In the HT-triggered scenario, we enforce the trigger condition and the HT is always active. The side-channel signal dataset collected for the $\boldsymbol{j^{th}}$ benchmark is defined as $\boldsymbol{D_j = \{(X_1^j, Y_1^j), (X_2^j, Y_2^j), ..., (X_{2N}^j, Y_{2N}^j)\}}$ which is a collection of pairs

$(\boldsymbol{X_i^j}, \boldsymbol{Y_i^j})$ where $\boldsymbol{X_i^j}$ is a univariate time series signal resulted from concatenation of various side-channel signals (EM and power for this chapter) and $\boldsymbol{Y_i^j} \in \boldsymbol{\{0, 1\}}$ as it is corresponding label (0 - HT-inactive, 1 - HT-triggered).

3.4.2 HTnet Model design

To design a virtually perfect neural network structure for the detection of triggered HTs, we perform an exhaustive search on the available state-of-the-art architectures for CNNs to find the best design for classifying the collected signal into Trojan triggered and inactive classes. Figure 3.2 represents HTnet that is the result of this exhaustive search with fine-tuned hyperparameters. HTnet consists of two Convolutional Layers (CL) with ReLU activations, each immediately followed by a dropout and max-pooling mechanisms to avoid overfitting. The output of these CLs is a preliminary discriminative set of features from the input signal. To further purify the extracted features, we implement an attention mechanism after CLs. We divide the output of the second CL into half, apply a softmax on half of the features and multiply it to the other half. Afterward, a CL with a small number of filters followed by a dense layer with a small number of nodes is added to the network to compress the extracted features. For the rest of this chapter, these compressed features will be referred to as latent features. In the end, there is one more dense layer with a softmax activation function for the classification purpose.

3.4.3 Modeling Side-channel Footprints of Known HTs

In this chapter, we propose a transfer learning approach to avoid the need for a golden chip. Inspired by [106], we first train the HTnet $\boldsymbol{H}$ to learn our previous knowledge about the existing HTs. Assuming that we are going to evaluate our methodology over the $\boldsymbol{m^{th}}$

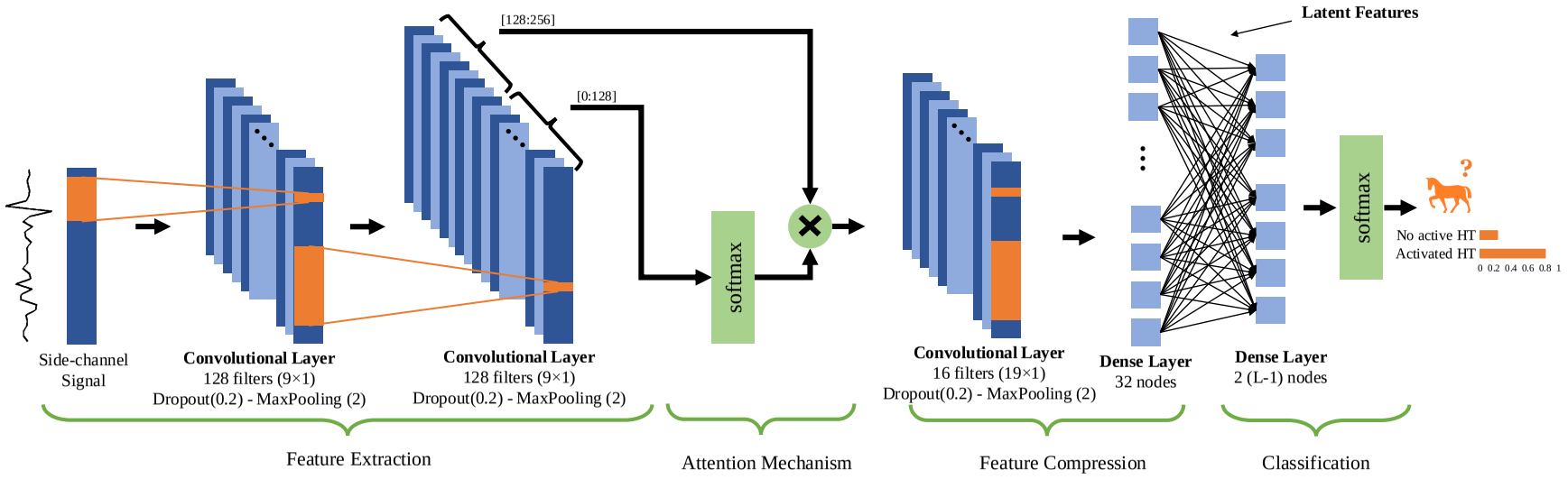


Figure 3.2: HTnet structure when used as classifier side-channel signals.

benchmark, we train HTnet to perform a $\mathbf{2(L-1)}$ classification task over $\boldsymbol{R_m}$ defined as,

$$\boldsymbol{R_m} = \bigcup_{\boldsymbol{i=0}}^{\boldsymbol{i=L}} \boldsymbol{D_i} - \boldsymbol{D_m} \tag{3.2}$$

where $\boldsymbol{L}$ is the total number of benchmarks (each benchmark has inactive and triggered samples). During training, the weights in $\boldsymbol{H}$ are tuned such that it extracts a set of descriptive latent features that can be used to discriminate between previously known HT-inactive and HT-triggered samples. We consider the weights of $\boldsymbol{H}$ as our initial knowledge for previously known HT-infected circuits.

3.4.4 Knowledge Transfer for One-class Feature Learning

Once an IC is manufactured, we can only acquire unlabeled side-channel signals from the chip. In the case of HTs with a trigger mechanism, it is often assumed that they will not get triggered during the chip test phase otherwise they would be detected. This means that during this phase we can assume that we have access to a number of side-channel signals that are mostly likely labeled as HT-inactive. Based on this assumption, we construct a model that learns to extract features for only one class of available data which is HT-inactive during the chip testing phase.

Architecture

As shown in Figure 3.3, we discard the last dense layer in $\boldsymbol{H}$, call it $\boldsymbol{H'}$, and create two new models: a reference model $\boldsymbol{H_r}$ and a target model $\boldsymbol{H_t}$. $\boldsymbol{H_r}$ consists of $\boldsymbol{H'}$ followed by a new softmax dense layer with $\mathbf{2(L-1)}$ nodes. $\boldsymbol{H_t}$ consists of only $\boldsymbol{H'}$ which shares the same weights with $\boldsymbol{H'}$ part of the reference model and is responsible for one-class feature extraction. To train $\boldsymbol{H_r}$ and $\boldsymbol{H_t}$ we use a reference dataset $\boldsymbol{R'_m} \subset \boldsymbol{R_m}$, and a target dataset

$\boldsymbol{D'_m} \subset \{(\boldsymbol{X}, \boldsymbol{Y}) : (\boldsymbol{X}, \boldsymbol{Y}) \in \boldsymbol{D_m} \text{ and } \boldsymbol{Y} \neq \boldsymbol{1}\}$ accordingly.

Loss functions

The extracted features by the network should be both compact and descriptive. A compact feature space will result in a similar feature representation for samples of our target class (HT-inactive - $\boldsymbol{D'_m}$). A descriptive feature space will produce distinct representations for samples of different classes (other benchmarks, HT-triggered, and HT-inactive -$\boldsymbol{R'_m}$). To ensure these two qualities, we incorporate two loss functions: compactness loss ($\boldsymbol{l_C}$) and descriptiveness loss ($\boldsymbol{l_D}$). We formulate, our optimization objective to be,

$$\hat{\boldsymbol{g}} = \min_{\boldsymbol{g}} \boldsymbol{l_D}(\boldsymbol{R'_m}) + \boldsymbol{\lambda} \boldsymbol{l_C}(\boldsymbol{D'_m}) \tag{3.3}$$

where $\boldsymbol{\lambda}$ is a constant. Here, $\boldsymbol{l_C}$ computes the mean squared intra-batch distance within a given batch of target class samples. Define $\boldsymbol{X} = \{\boldsymbol{x_1}, \boldsymbol{x_2}, ..., \boldsymbol{x_n}\} \in \boldsymbol{R^{n \times k}}$ to be the input to the loss function, where $\boldsymbol{n}$ is number of samples in the batch and $\boldsymbol{k}$ is length of each sample. The distance of $\boldsymbol{i^{th}}$ sample with other samples can be defined as,

$$\boldsymbol{z_i} = \boldsymbol{x_i} - \boldsymbol{mean_i} \tag{3.4}$$

where $\boldsymbol{mean_i} = \frac{\boldsymbol{1}}{\boldsymbol{n-1}} \sum_{\boldsymbol{j \neq i}} \boldsymbol{x_j}$ is the mean of the rest of samples in the batch. From there, compactness loss is defined as average Euclidean distance as in,

$$\boldsymbol{l_C} = \frac{\boldsymbol{1}}{\boldsymbol{nk}} \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \boldsymbol{z_i^T z_i}. \tag{3.5}$$

We calculate $\boldsymbol{l_C}$ over target network ($\boldsymbol{H_t}$) outputs. To perform back-propagation using this loss function on this network, it is necessary to assess the contribution of each element over the final loss. Thereby, let $\boldsymbol{x_i} = \{\boldsymbol{x_{i1}}, \boldsymbol{x_{i2}}, ..., \boldsymbol{x_{ik}}\}$ and $\boldsymbol{m_i} = \{\boldsymbol{m_{i1}}, \boldsymbol{m_{i2}}, ..., \boldsymbol{m_{ik}}\}$. Then,

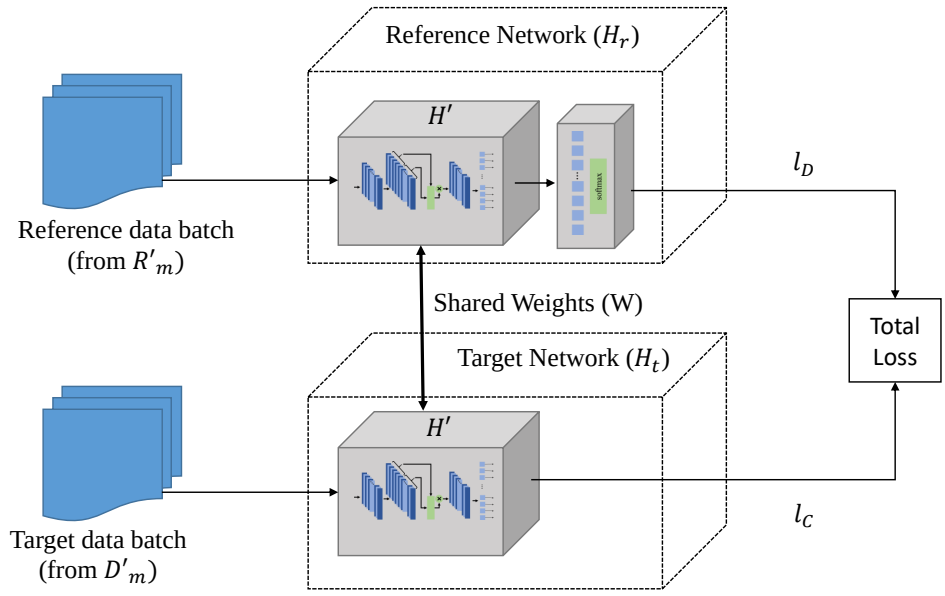


Figure 3.3: Training reference and target networks.

the gradient of $\boldsymbol{l_C}$ with respect to the input $\boldsymbol{x_{ij}}$ can be calculated as,

$$\frac{\boldsymbol{\partial l_C}}{\boldsymbol{\partial x_{ij}}} = \frac{\boldsymbol{2}}{\boldsymbol{(n-1)nk}}[\boldsymbol{n \times (x_{ij} - m_{ij}) - \sum_{k=1}^{n}(x_{ik} - m_{ik})}]. \tag{3.6}$$

For descriptiveness loss $\boldsymbol{l_D}$ we use classical cross-entropy formulation of multiclass classification tasks with regard to the reference dataset and calculate it over the reference network.

Training

Initially, both $\boldsymbol{H_r}$ and $\boldsymbol{H_t}$ contain the pre-trained model $\boldsymbol{H}$ weights. During training, the first two CLs in $\boldsymbol{H'}$ are frozen as commonly done for network fine-tuning and only later layers in the network can be trained. Furthermore, a relatively lower learning rate is chosen for the model training process to avoid sudden deviations from the initially trained model. During every iteration of training, in the forward pass, two sample batches each from the target dataset ($\boldsymbol{D'_m}$) and reference dataset ($\boldsymbol{R'_m}$) are feed to the $\boldsymbol{H_t}$ and $\boldsymbol{H_r}$ respectively to calculate $\boldsymbol{l_C}$ and $\boldsymbol{l_D}$. Since the weights $\boldsymbol{W}$ are shared between the two networks, the

composite loss can be defined as,

$$\boldsymbol{l(R'_m, D'_m) = l_D(R'_m|W) + \lambda l_C(D'_m|W).} \tag{3.7}$$

During training, we back-propagate over two networks and learn the network parameters to minimize this composite loss function which leads to the minimization of the optimization objective in 3.3.

3.4.5 HT detection through self-referencing

In this chapter, our major contribution is constructing a model that can extract the most useful features from side-channel signals for HT detection. Once these features are extracted, any self-referencing anomaly detection approach can detect HTs with high accuracy based on these features. Here, we use One-class Support Vector machines (OC-SVM), K Nearest Neighbors (KNN), and Local Outlier Factor (LOF) as candidates for anomaly detection in the side-channel signal. These models are trained during the testing phase using our feature extractor and used in run time for HT detection.

3.5 Results

3.5.1 Experimental Test Bed

We build an automated testbed (Figure 3.4) to measure the EM and power side-channel signals of various circuits. The device under test is a Sakura-G Board on which two Spartan-6 FPGA are integrated. One controller FPGA handles the USB connection with computer for data transmission and the main FPGA is used to implement the target circuit which

is AES-128 encryption core along with a variety of hardware Trojan circuits in our tests. The board provides the power consumption of target circuit as an output voltage that is measured by the Tektronix TDS2022C oscilloscope. We use H-field probe to measure and amplify the magnetic emissions of FPGA. Then, the Agilent Spectrum Analyzer N1996A reads the signal. The computers sends random test vectors to AES core and collect the power and EM data from the devices. The measurements are synchronized by a trigger from board that determine the start of encryption process of each input.

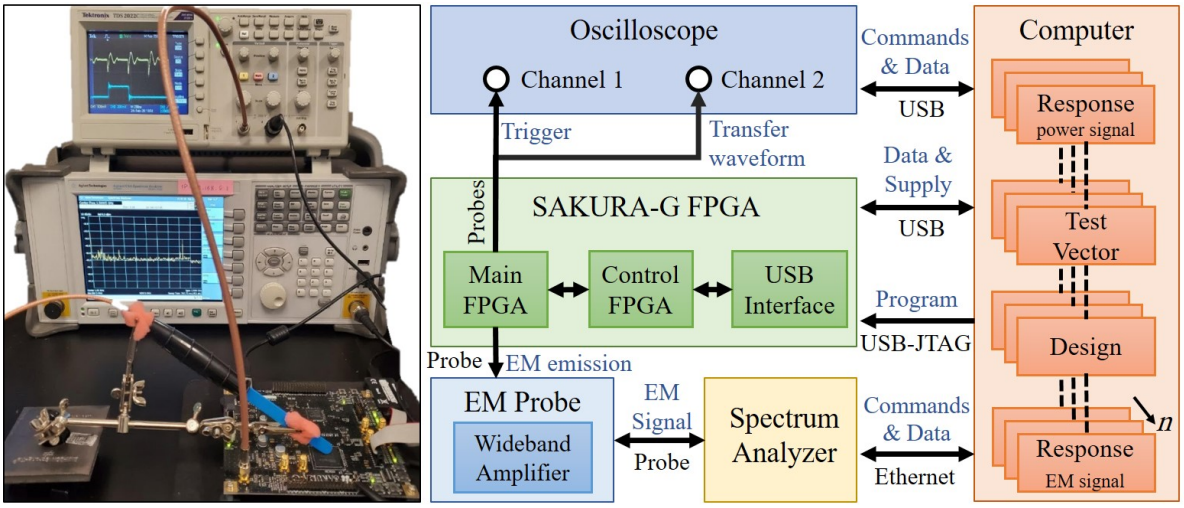


Figure 3.4: The automated testbed for side-channel signals collection.

3.5.2 Evaluation Method

For each benchmark, we collect 1000 HT-inactive and active samples ($\boldsymbol{N = 1000}$) and utilize **80%** of data for training the feature extractors and use the rest of the data to perform the tests (validation) in rest of this chapter. We incorporate same number of HT-inactive and HT active samples for our evaluations. Hence, we simply report our results in terms of accuracy defined as,

$$\boldsymbol{Accuracy} = \frac{\boldsymbol{N_{TP} + N_{TN}}}{\boldsymbol{N_{TP} + N_{TN} + N_{FP} + N_{FN}}} \tag{3.8}$$

Table 3.1: HTnet comparison with the methods in [48] to classify power side-channel signals.

Benchmark	SVM	Naive Bayes	Random Forest	HTnet
AES-T400	0.6646	0.7400	0.7012	0.8975
AES-T500	0.9967	0.9971	0.9900	0.9793
AES-T600	0.6738	0.7321	0.7175	1
AES-T700	1	1	1	1
AES-T800	1	1	1	1
AES-T1000	0.6800	0.7213	0.6925	0.7135
AES-T1100	0.6783	0.7371	0.6929	0.7758
AES-T1300	0.6746	0.9833	0.9721	1
AES-T1400	0.6738	0.7392	0.6704	0.8525
AES-T1600	0.6629	0.7421	0.7008	0.7570
AES-T1800	0.6596	0.7379	0.7204	0.9345
AES-T2000	0.6575	0.7267	0.7225	0.9295
Mean	0.7518	0.8214	0.7984	0.9033

where $\boldsymbol{N_{TP}}$, $\boldsymbol{N_{TN}}$, $\boldsymbol{N_{FP}}$ and $\boldsymbol{N_{FN}}$ are number of true positive, true negative, false positive and false negative accordingly. A *positive* side-channel signal segment means that an HT is triggered, and a *true* indication means that the detection method has correctly identified the type of that segment.

3.5.3 HTnet Evaluation

Given that we have access to both HT-inactive and HT active samples for each benchmark, we first evaluate the performance of HTnet to classify these two types of samples for each benchmark. Note that in this scenario, access to a golden chip is guaranteed and the purpose of this evaluation is to compare HTnet to the other methods available in the literature. Table 3.1 compares the performance of HTnet to the methods proposed in one of the latest works in the literature [48] (2020). This table shows the accuracy of each method using only power side-channel signals. To improve the accuracy of each of the rival methods, we first extract and select the best set of features using [24] and report their best performance. The results show that HTnet outperforms any of these methods over AES benchmarks even when the best type of features are extracted for them.

3.5.4 Golden Chip-Free Feature Extraction Evaluation

We set the value of $\boldsymbol{\lambda}$ to $\mathbf{0.1}$ and follow the training mechanism described in Section 3.4. We use Stochastic Gradient Descent (SGD) optimizer with learning rate of $\mathbf{0.001}$ for training $\boldsymbol{H}$ and learning rate of $\mathbf{0.000001}$ for $\boldsymbol{H_r}$ and $\boldsymbol{H_t}$. Figure 3.5 shows the outputs of $\boldsymbol{H_t}$ for some of the HT-inactive and the HT-triggered samples. As shown in this figure, each sample input (EM concatenated with power) is converted to a compact form of 32 features. In this feature space, all the HT-inactive samples posses a similar set of features while multiple features of HT-triggered samples vary from the HT-inactive samples. These variations can be used for reporting the sample as a HT-triggered case using an anomaly detection mechanism.

3.5.5 HT Detection Evaluation

As described in Section 3.4.5, we incorporate various off-the-shelf anomaly detection mechanisms from [153] to evaluate the effectiveness of our extracted set of features from EM and power side-channel signals for HT detection. As shown in Table 3.2, our proposed methodology for extracting relevant features can increase the accuracy of anomaly detection methods in most of the cases. Particularly, LOF and OC-SVM cannot detect any anomalous behavior in side-channel signals without using our feature extraction mechanism for AES-1100 benchmark. Note that collecting EM side-channel signals highly depends on the probe placement over chip. Thereby, here, we only provide evaluations for benchmarks that we were able to collect reliable EM traces. As shown in 3.2, EM and power side-channel signals fusion can boost the accuracy of our golden chip-free approach to pass the results presented in Table 3.1 in some cases.

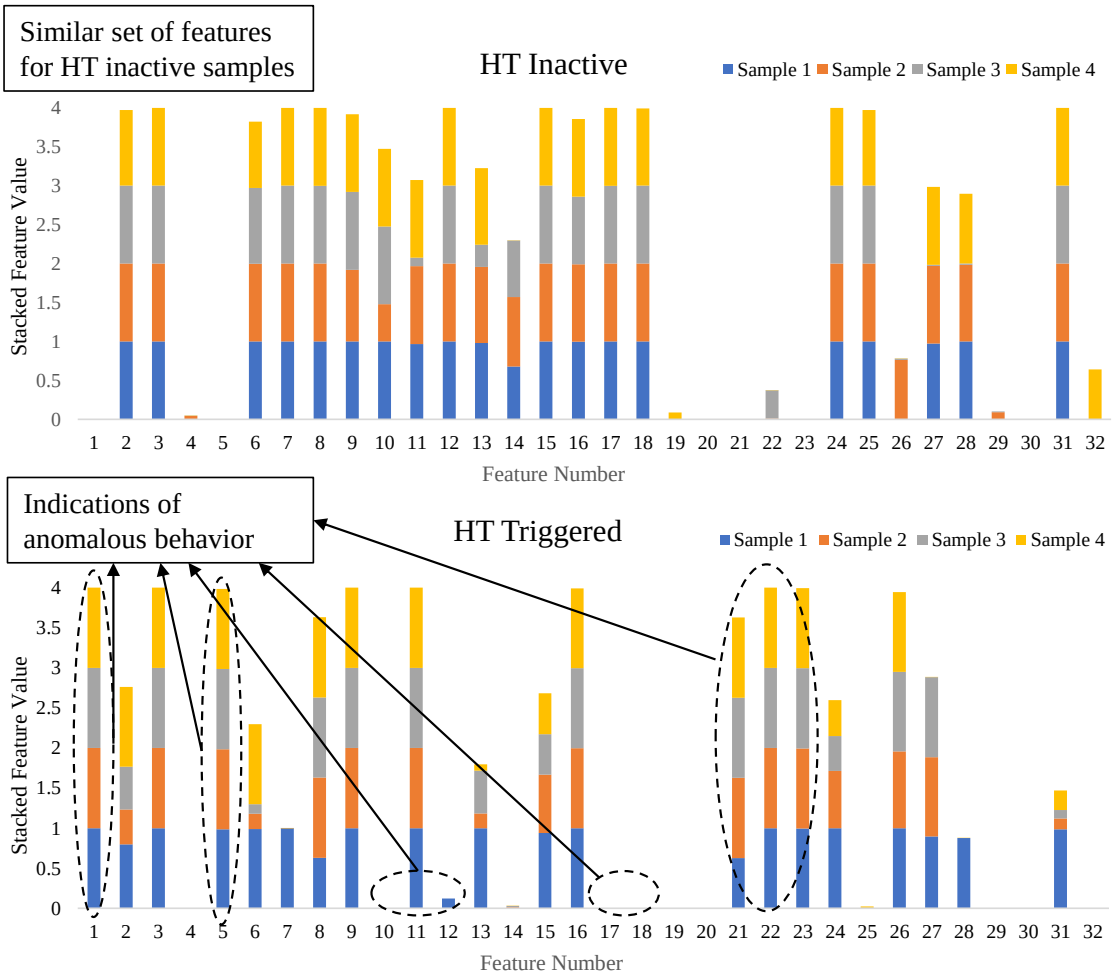


Figure 3.5: Extracted features for four HT-inactive and HT-triggered samples from AES-500 benchmark.

3.5.6 Robustness of Extracted Features

The side-channel signals tend to vary over the life span of an IC due to various reasons, such as aging and temperature change. To evaluate the robustness of our extracted features, we add a synthetic white noise $\boldsymbol{\mathcal{N}(0, \sigma^2)}$ to the scaled (0 to 1) test samples. Figure 3.6 shows the changes in the accuracy of KNN when it is using HTnet extracted features versus when it is directly monitoring the input signal in two benchmarks. The results show that small variations in the side-channel signal can significantly affect a trained anomaly detector accuracy if it does not use our approach for feature extraction. In other terms, an HTnet en-

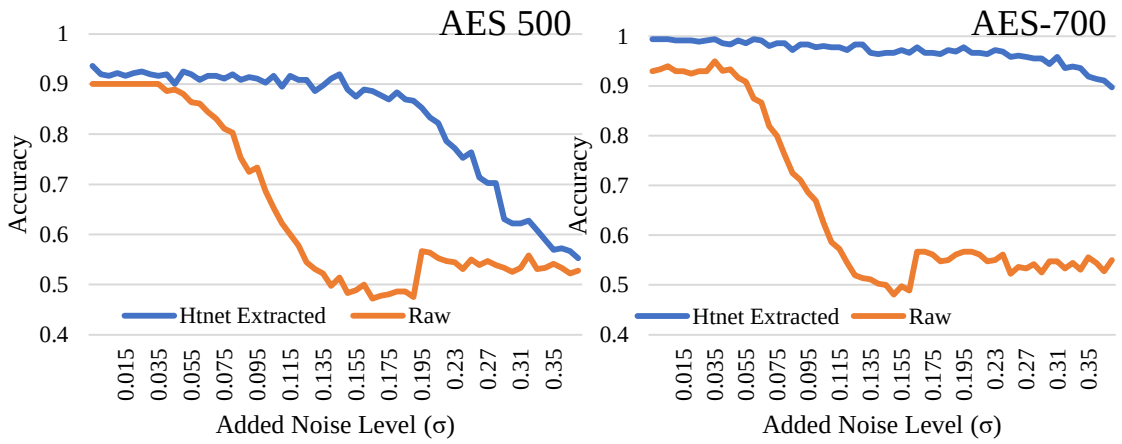


Figure 3.6: Robustness evaluation of KNN with two different input types.

abled anomaly detection method can endure more noise in the collected side-channel signals during run time (more than two times resiliency for these two benchmarks).

3.6 Discussion

As a last line of defense, run-time HT detection approaches play a crucial role in detecting subtle HTs which are not detected in the verification phase. In this chapter, we provide evidence that it is possible to extract robust features that can be used for HT detection using our previous knowledge about HTs. However, running a neural network along with the main circuit will consume considerable amount of energy which will make our approach

Table 3.2: Accuracy of various anomaly detection algorithms over concatenated EM and Power side-channel signals.

	KNN		**LOF**		**OC-SVM**	
Model Input	HTnet	Raw	HTnet	Raw	HTnet	Raw
AES-T500	0.936	0.895	0.928	0.858	0.928	0.919
AES-T700	0.986	0.927	0.975	0.901	0.972	0.946
AES-T800	0.983	0.937	0.961	0.941	0.981	0.916
AES-T1000	0.972	0.914	0.975	0.953	0.983	0.962
AES-T1100	0.803	0.798	0.656	0.491	0.822	0.478
Mean	**0.936**	0.894	**0.899**	0.829	**0.937**	0.844

mostly suitable for applications which do not rely on limited power resources. Furthermore, the self-referencing approach that we discussed in this chapter is a naive approach which can be significantly expanded using the other works discussed in the related literature. Here, we provided a proof of concept for golden chip-free HT detection using recent developments in artificial intelligence. We leave the details of efficient implementation of HTnet and using more advanced self-referencing approaches to our future work.

3.7 Conclusion

In this chapter, we introduced HTnet which outperforms any existing approaches for detecting triggered HTs through side-channel signals given that a reference golden chip is available. Furthermore, we introduce a novel training methodology to utilize our previous knowledge about known HT-infected circuits to design a robust HT-related feature extractor. We show that it is possible to use off-the-shelf anomaly detection algorithms and incorporate HTnet as a feature extractor to detect triggered HTs in run time with 93.7% accuracy on average.

Chapter 4

Brain-Inspired Golden Chip Free Hardware Trojan Detection

4.1 Abstract

The subtle changes in the side-channel measurements of integrated circuits are not always indicator of triggered hardware Trojan. The variations in the side-channel signal may occur due to various other reasons such as temperature change, aging, etc. In this chapter, we propose using a brain-inspired architecture called Hierarchical Temporal Memory (HTM) for hardware Trojan detection. Similar to the human brain, our proposed solution is resilient against *natural* changes that might happen in the side-channel measurements while being able to accurately detect abnormal behavior of the chip when the Hardware Trojan gets triggered. We use a self-referencing method for HT detection, which eliminates the need for the golden chip. The effectiveness of our approach is evaluated using TrustHub [117] benchmarks, which shows 92.20% detection accuracy on average.

4.2 Introduction

As discussed in the previous chapter, companies tend to use third-party IP cores and outsource the chip fabrication process to the foundries worldwide, which has raised security concerns referred to as *Hardware Trojan* (HT). One of the incidents that attracted the attention of the research community toward the threat of HT is the failure of Syrian radars in 2007. As a result of this failure, a suspected Syrian nuclear installation was bombed by Israel. Further investigation revealed that the commercial off-the-shelf microprocessors inside the radar were infected with HT, which was triggered through a hidden back door and disabled the system [2].

Due to their stealthy nature, most HTs are designed to be minuscule and remain inactive with a negligible impact on the circuit specification until a rare specific event triggers them. Ideally, any drift from the original circuit design should be detectable by post-fabrication testing and verification. However, these methods fall short to detect HT because the probability of triggering the HT is usually low.

Therefore, some other methods are required to ensure circuit security. To this end, two major paths are pursued in literature: i) imaging techniques, and ii) side-channel and covert-channel analysis. Destructive approaches mainly involve following steps: de-layering the chip, imaging the die, reverse-engineering the image of the circuit, and conducting element by element comparison. Although these methods are relatively capable of guaranteeing *trust* on the fabricated IC [132, 27, 32], they are destructive, impractical, time-consuming, expensive, and inapplicable to detect the contaminated third-party IPs. In contrast, the second approaches are non-destructive and assess the behavior of the chip through side-channel (e.g. power, temperature, electromagnetic, and timing) or embedded sensor measurements. In these methods, the parameters are measured and compared to the expected values to identify the presence of additional structure.

The fundamental shortcoming of the majority of side-channel based HT detection methodologies is the reliance on a trusted chip (a.k.a golden chip) to create a reference model of the expected side-channel values. Reliance on golden chip is a problem since, in practice, a trusted supply chain to manufacture the golden chip is not available, or it would be too expensive and unaffordable for most of SoC designers [79]. A few works in the literature have tried to resolve this issue by using self-referencing techniques [61], or using accurate trusted simulations combined with embedded sensors in the chip to analyze the side-channel emissions [85]. However, similar to other side-channel based HT detection methodologies they often perform poorly when HT is not triggered. Their poor performance occurs because inactive HT has an insignificant footprint that fits in the process variation margin[64]. Despite the efforts for triggering the HT during the testing of the chip [64], the HT may remain inactive. Therefore, complimentary run-time monitoring and validation mechanisms such as [14, 69, 44, 74] are introduced to provide a last line of defense against HTs in the mission critical systems. Nevertheless, these methods come with the area overhead. Additionally, they will have a high false-positive detection rate if they cannot adapt to the acceptable alteration of the IC side-channel profile over its life span [71, 74].

The HTnet model introduced in Chapter 3 provides a great medium to extract features that can be used for run-time HT detection. However, continuously updating HTnet over the lifetime of the chip is costly and infeasible. In this chapter, we develop another novel golden chip-free method for post-silicon HT detection. We have the premise that the culprit can be in any entities in the IC supply chain, including the design vendors and foundries. We construct the model based on the power consumption of the chip, which monitors the side-channel emissions during the testing and at the run-time. Our proposed model is hierarchical temporal memory (HTM), which mimics human brain behavior to interpolate the side-channel data and pinpoint anomalies that indicate the existence of a HT in the IC. The content of this chapter is provided from [39].

4.2.1 Research Challenges

In a nutshell, the state-of-art ideas for HT detection research have shortcomings to address the following challenges:

- Reliance on a reference golden chip, which is costly and even impossible in some cases such as untrusted IPs.
- HT detection methods are doomed to fail due to process variation in chip fabrication if proper measures are not taken to eliminate this factor.
- Probability of triggering HT during testing is very low due to the stealthy nature of them.
- Run time HT detection mechanisms are often considered to be unfeasible due to their costly hardware overhead and changes in chip characteristics.

4.2.2 Technical Contributions

This chapter makes following technical contributions to address the aforementioned challenges:

- We devise an HTM architecture that performs the same or better than state-of-the-art approaches for HT detection.
- Our proposed method is trained on side-channel data from the Device Under Test (DUT) rather than a golden chip.
- The process variation has no effect on our proposed detection mechanism since it relies on DUT for training.

- HTM training has virtually no cost in terms of power consumption. Hence, sporadic retraining of the model is feasible to cope with aging/temperature effects on the side-channel data.

- With this work, We publish our dataset of power side-channel signals for AES and RS232 circuits infected by various HTs.[1]

Rest of the chapter is organized as follows: Section 4.3 presents the background for understanding IC fabrication supply chain, related works for HT detection, and why it is absolutely necessary to use adoptable models for HT detection; Section 4.4.1 defines the HT problem that we solve in this thesis; Section 4.5 explains the HTM solution that we propose to use for HT detection; Section 4.6.4 presents our test bed and reports the experimental results; Section 4.7 discusses the limitations of this chapter and possible future works; and Section 4.8 concludes the chapter.

4.3 Background

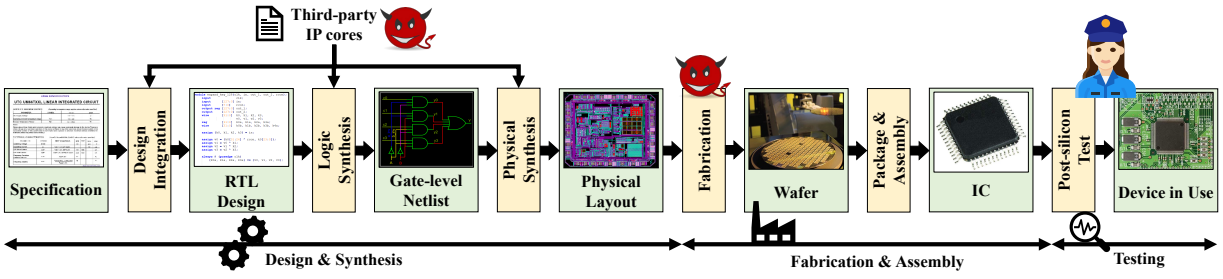


Figure 4.1: Semiconductor supply chain, HT injection points, and our detection mechanism stage in this supply chain.

[1] http://ieee-dataport.org/3599

4.3.1 Semiconductor Supply Chain

The semiconductor supply chain, as shown in Figure 4.1, involves many stages which are distributed among entities around the world. The design flow starts with designing the components in Register Transfer Level (RTL) based on the specification. In addition to in-house designs, third-party vendors provide IP cores in different level of abstraction. The different design components are integrated and go through multiple stages of synthesis and verification using the Electronic Design Automation (EDA) tools and eventually the physical layout is generated. The foundry fabricates the IC which is tested and packaged in the next stages.

4.3.2 Same IC, Different Power Traces

The total power consumption of an IC can be summarized as,

$$\boldsymbol{P_{total} = P_{dyn} + P_{short} + P_{leak}} \tag{4.1}$$

where $\boldsymbol{P_{dyn}}$ is the dynamic power usage; $\boldsymbol{P_{short}}$ is the power consumption contributed by shorting the supply to ground for a brief moment when transistors switch; and $\boldsymbol{P_{leak}}$ is due to the static current leakage from transistors. Each one of these components may change over time. In the following, we set the physical bases on why it is absolutely necessary to evaluate the side-channel data using only models which are frequently updated to match the existing state of the system under monitoring.

Voltage Level Effect. A change in voltage of power supply source ($\boldsymbol{V_d}$) directly affects

the total power consumption:

$$\begin{aligned} \boldsymbol{P_{dyn}} &= \boldsymbol{V_d^2 * f * C_{eff}};\\ \boldsymbol{P_{leak}} &= \boldsymbol{V_d * I_{static}};\\ \boldsymbol{P_{short}} &= \boldsymbol{V_d} * \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \boldsymbol{\alpha_i * I_{sc_i}}, \end{aligned} \tag{4.2}$$

where $\boldsymbol{f}$ is the operating frequency of the IC, $\boldsymbol{C_{eff}}$ is the effective capacity observed by the power supply, $\boldsymbol{\alpha_i * I_{sc_i}}$ are the current leaking when both transistors are on during switching, $\boldsymbol{n}$ is number of transistors switching in the IC on average, and $\boldsymbol{I_{static}}$ is the total current leaking in the steady-state. The $\boldsymbol{V_d}$ provided by the voltage regulator may be intentionally changed to provide power-saving, high-performance, etc. modes for a given IC, or it might unintentionally face minute shifts due to temperature changes and process variations.

Temperature Effect. The difference in temperature results in subtle variations in the physical characteristics of IC. Table 4.1 represents some of the models in the literature which can capture the relationship between $\boldsymbol{P_{leak}}$ and temperature change. Although each one of these models is a viable representation for the particular fabrication technology that the authors have considered, further research is required to craft a model for a new design and fabrication technology. Hence, even if a run-time HT detection mechanism uses the internal temperature as a complementary side-channel to power, it might fail to adopt to a given new IC since power and temperature relation might be significantly different from the mechanism's expectation.

Table 4.1: Models to capture the temperature and currents leakage relationship. ($\boldsymbol{a_*}$: scalars, $\boldsymbol{T}$: temperature)

Authors	Models
Su [130]	$\boldsymbol{P_{leak} \propto a_2 T^2 + a_1 T + a_0}$
Liao [83]	$\boldsymbol{P_{leak} \propto a_2 e^{-a0/T - a_1}}$
Liu [84]	$\boldsymbol{P_{leak} \propto a_2 + a_2(T - a_1)}$
Skadron [151]	$\boldsymbol{P_{leak} \propto a_2(1 - e^{a_1/T})e^{a_0/T}}$

Aging Effect The continuous usage of IC's internal components results in change of their physical characteristic. In particular, the switching voltage of transistors, $\boldsymbol{V_{th}}$, starts to slowly increase over time. As it is stated in [90], although an increase in $\boldsymbol{V_{th}}$ does not affect $\boldsymbol{P_{dyn}}$, it has a strong impact on $\boldsymbol{P_{leak}}$ since $\boldsymbol{I_{static}}$ has an exponential relationship to $\boldsymbol{V_{th}}$:

$$\boldsymbol{I_{static} \propto e^{V_{gs} - V_{th}}},$$

where $\boldsymbol{V_{gs}}$ is the gate-source voltage and remains mostly the same as long as $\boldsymbol{V_d}$ is constant. Furthermore, [90] shows that the increase in $\boldsymbol{V_{th}}$ results in small decrease in $\boldsymbol{P_{short}}$ as well.

4.3.3 Related Works

A detailed taxonomy of HTs and their detection mechanisms are presented in [136, 79, 65]. Here, we review the works in literature which are most closely related to the work presented in this chapter to acknowledge their similarities and discriminative features from our proposed methodology.

Side-channel based HT detection. A circuit connected to a power supply emits unintended information in various side-channels. Each circuit has unique side-channel fingerprints and any modification, such as HT, will disrupt it. This variation caused by HT can be leveraged to detect HT presence. Power consumption rate [122, 80, 101, 88, 87, 66], electromagnetic emissions [57, 67, 134], heat generation [44, 103], and timing information [68, 145, 76] have been extensively studied as a medium for HT detection. In this chapter, we use power consumption since it has been the most popular choice for side-channel based HT detection in the literature.

Machine learning models for side-channel analysis. HT detection is quite challenging since the effect of HT is often insignificant and may lie in the process variation margin.

Thus, various statistical models and Machine Learning (ML) techniques for studying side-channel information have been used to identify the presence of HT in the circuit. ELM [140], BPNN [80, 101], KNN/DT/Navie Bayse/Deep Learning [88, 87], clustering [28, 144, 103], SVM [66], OC-SVM [67, 86] are among the most popular choices for inspecting side-channel information for HT detection. In this chapter, we propose using HTM which to the best of our knowledge has never been used in literature for hardware Trojan detection.

Golden chip free HT detection. While the earlier approaches for HT detection have frequently assumed access to a golden chip for acquiring training samples, recent works have suggested various ideas to overcome this limitation. In [85], the authors use trusted simulations to generate the needed HT-free samples for initial training of their model. Then, in post-fabrication stage, they use process control monitor sensors data to eliminate the process variation effect. In [134], the authors suggest using thermal maps and its active area shape from the GDS II file which is not affected with process variation for training the model. In [57], an electromagnetic spectrum modeling method from the early stage of IC production is introduced which can be used as a golden reference. The authors of [61] propose a temporal self-referencing approach which compares the current signature of the chip at two different time windows to isolate the Trojan effect. Similar to [61], in this chapter, we propose a temporal self-referencing approach but without windowing requirement. HTM gets one sample point at a time and evaluates that point's relationship to other sample points.

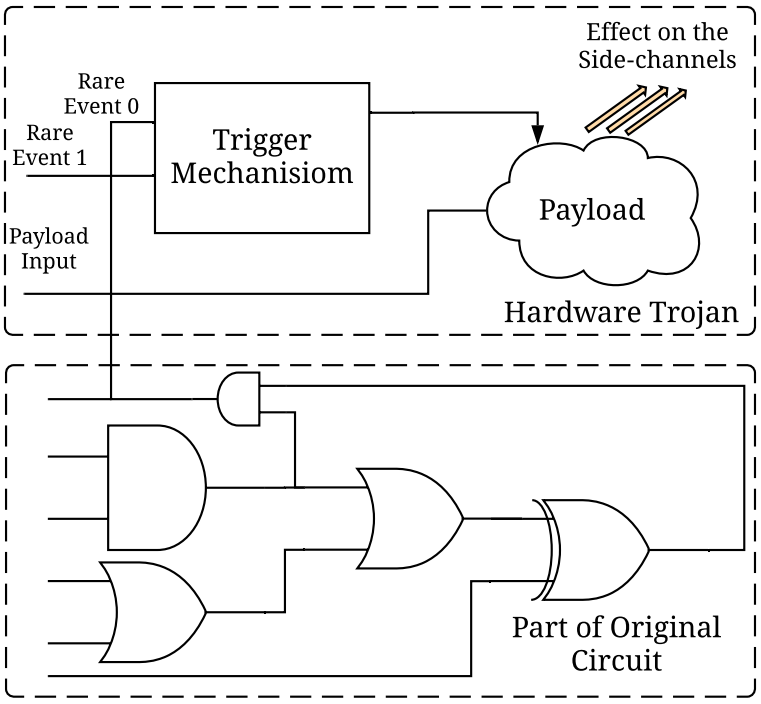


Figure 4.2: HT with a trigger mechanism and a payload.

4.4 Problem Definition

4.4.1 Threat Model

We consider any malicious modification to a circuit as HT. As it is depicted in Figure 4.1, we assume that, HT attacks occur under two scenarios (i) a third party IP used in the IC design phase contains unspecified malicious functionalities, or (ii) an attacker in the foundry modifies the layout of the chip before fabrication. Furthermore, we assume that the HT used in the chips is always composed of a trigger mechanism and a payload. Note that with this assumption, our proposed methodology will not cover all types of HTs. However, it is still highly desirable due to its golden-chip free characteristics for mission critical ICs as a last line of defense.

As shown in Figure 4.2, the trigger mechanism monitors the chip and activates the payload

under rare condition to evade possible HT detection solutions in the post-fabrication testing. In this chapter, we do not set any assumption on type of the trigger mechanism. Therefore, the HT may not have an explicit added hardware for monitoring [49]. The payload of hardware Trojan may result in change of functionality, denial of service, or data leakage from IC. Among these outcomes, this thesis focuses on the data leakage and subtle change of functionalities (such as power drain) that can happen over the life span of the chip without leaving any obvious immediate trace in the system. In particular, we test our defense mechanism against HT attacks which result in less than $\mathbf{3\sigma}$ variation in the side-channel footprints even when they are activated.

In this chapter, we assume that the EDA tools used for design, simulation, and verification are trusted; hence, the formal verification methods can be used in post-fabrication testing of semiconductor supply chain to assure correct functionality of the chip if the HT is not triggered.

4.4.2 Defense Evaluation Method

The statistical side-channel analysis methods used in the literature have various assumptions for training their statistical models. The two common assumptions are (i) access to golden chip and (ii) knowledge of possible HT types. While the first assumption is rarely feasible, since the majority of side-channel based HT detection methods employ this assumption, we include them as the best possible outcomes in our comparisons. Furthermore, although the research community has revealed various forms of HTs, there is always the possibility of new HT vulnerabilities that have not been discovered before. Hence, we evaluate defense mechanism in terms of their capability for detecting unknown HTs (zero-day) as well as known HTs. We assume that all the methods compared in this chapter are capable of monitoring the chip in run-time. With this assumption, once each detection method is

trained, we use it to infer whether or not the HT is triggered at each segment of the side-channel signal. We compare HT detection methods in terms of accuracy as follows,

$$\boldsymbol{Accuracy} = \frac{\boldsymbol{N_{TP}} + \boldsymbol{N_{TN}}}{\boldsymbol{N_{TP}} + \boldsymbol{N_{TN}} + \boldsymbol{N_{FP}} + \boldsymbol{N_{FN}}} \tag{4.3}$$

where $\boldsymbol{N_{TP}}$, $\boldsymbol{N_{TN}}$, $\boldsymbol{N_{FP}}$ and $\boldsymbol{N_{FN}}$ are number of true positive, true negative, false positive and false negative accordingly. A *positive* side-channel signal segment means that a HT is activated, and a *true* indication means that the detection method has correctly identified the type of that segment. Note that we use the same number of positive and negative samples in our comparisons; hence, Equation 4.3 is unbiased and it reflects possible high false positive rates.

4.5 Neuro-Inspired architecture

In this section, we discuss the structure of an HTM model. We first introduce various components of an HTM model and then explain how the temporal learning algorithm works, which makes it suitable for run-time HT detection.

4.5.1 Human Neocortex and HTM Model

The Neocortex in brain is responsible for life-long learning, cognitive processes, and perceptive functionalities. The Neocortex consists of billions of pyramidal neuron cells stacked beside and on top of each other to create various levels of abstraction inside the human brain. The functionalities of these biological neurons are partially modelled inside the smallest building block of the HTM architecture which are known as the HTM neuron cells[56]. One layer of HTM is made of *mini-columns* that have a fixed number of HTM neurons stacked upon one another. Multiple mini-columns are stacked side by side to form a *cortical*

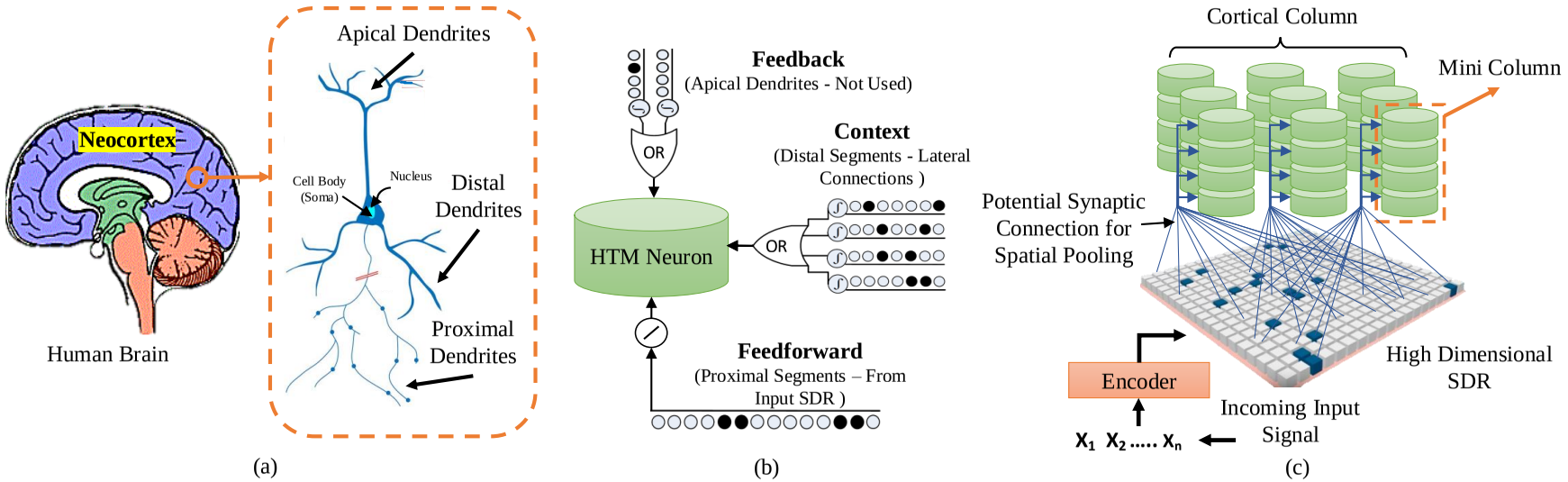


Figure 4.3: (a) Pyramidal neuron in human brain. (b) HTM neuron cell. (c) One level HTM with spatial pooling and input SDR.

column (Fig. 4.3) in a layer. For the rest of the chapter, column and mini-column are used interchangeably. Similar to biological neurons, HTM neuron cells can connect with other nearby neurons under two types of connective dendrite segments (i) proximal, to get feed-forward input from the input space and other cells at lowers levels, and (ii) distal, which provides the connections to the lateral cells in the same layer. The distal synapses are usually different for each cell while cells of the same column share the same proximal synapses. A biological cell has an apical dendrite segment connections from the layer above as well; however, in this chapter we consider only a single layer of HTM neurons which do not require connections to higher levels.

4.5.2 HTM components

Encoder

The human neocortex receives different sensory signals coming from different sensory organs and combines them together. HTM works similarly. HTM has an encoder which is the first stage of the HTM model. Different incoming signals ($\boldsymbol{X_1, X_2 .... X_n}$) are first encoded into separate high dimensional Sparse Distributed Regions (SDRs) by the encoder. Then all separate SDRs are combined together to form a single SDR which acts as an input to the next HTM block (Fig. 4.3). SDRs are binary sparse matrices, and these matrices are distributed representations of the input signals. SDRs have typically less than 2% active values (i.e., neuron set to 1) for every input. In SDRs only a few neurons overlap across different inputs which is different from the dense representations of the state-of-the-art Neural Networks (NN) where the activity of many neurons can overlap (i.e., fully connected NN). This sparse nature of the activations enables HTM to learn to make accurate HT detection, regardless of the noise in the collected signals. Currently, various types of encoders have been suggested in the literature [110]. In this chapter we use a random distributed scalar encoder which

we found to be the most suitable for the side-channel data analysis. This encoder maps a scalar value into a random SDR while preserving an overlapping representation of the input data. Concurrently, the encoder dynamically changes the min/max range of the input data without affecting the performance. It also ensures that similar inputs have high overlap and their representations do not change over the times.

Spatial Pooler

The next block after the encoder is the spatial pooler, which receives the SDRs generated by the encoder. For the spatial pooling mechanism, each column is connected to a subset of the input SDR by a potential synaptic connection. A distinct permanence value is assigned to each potential synapse. This permanence value can be incremented/decremented while learning/forgetting of features from the input SDRs. The increment of the permanence value means a synaptic connection is strengthened between the input SDR and the column, whereas a decrement weakens the synaptic connections to forget the input. The spatial pooler learns the spatial features from the input SDRs and creates a new SDR of the same size at every time step.

Temporal Pooler

The temporal pooler region consists of multiple columns. Each neuron of the columns can comprise two and sometimes up to a dozen distal dendrite segments. Each distal dendrite segment is connected to afferent synapses from multiple cells of the neighboring columns in the same layer. A neuron in a column become active if the sum of the active synapses connected to the proximal segment exceeds a certain threshold. We discuss the activation process in detail in the next section.

4.5.3 Activation

The activation and learning jointly happen in the Spatial Pooler and the Temporal Pooler regions. Similar to biological neurons, each HTM neuron cell can be in three possible states: inactive, predictive, and active, with inactive as the default state. The predictive state of a neuron is determined by the activity of the distal segments which in turn is determined by the activation state of the other neurons. A neuron becomes active only if it was in the predictive state at the previous instant, with an exception that will be described later.

To formulate the aforementioned statement, let us consider a $\boldsymbol{M} \times \boldsymbol{N}$ matrix where M is the number of cells per column and N is the number of columns in the layer. Let us denote this matrix by $\boldsymbol{A^t}$. The matrix $\boldsymbol{A^t}$ represents the current activation state of the cells in a particular layer at time $\boldsymbol{t}$ and $\boldsymbol{a^t_{i,j}}$ is the $\boldsymbol{i,j}$th element of $\boldsymbol{A^t}$. The term $\boldsymbol{a^t_{i,j}}$ denotes the activation state of cell $\boldsymbol{i}$ in column $\boldsymbol{j}$. Let us denote a distal dendrite segment of a column by $\boldsymbol{d}$. Each dendrite segment has a synaptic connection. Let us denote the weights of the synaptic connections of the $\boldsymbol{d}$th dendrite segment of the $\boldsymbol{i}$th cell and $\boldsymbol{j}$th column by $\boldsymbol{D^d_{i,j}}$. We note that the synaptic connections are considered to have been established when the weights are above a specific threshold. Let us denote a matrix $\boldsymbol{\tilde{D}^d_{i,j}}$, which has the weights of established connection. The weights below the specific threshold in $\boldsymbol{\tilde{D}^d_{i,j}}$ are considered to be zero.

In the spatial pooler, a large part ($\mathbf{50\%}$) of the input SDR is initially connected to the proximal synapses of each cell in each column. The initial connections are random and remain the same for the time onward. A proximal synapse becomes active when the corresponding element's value in the input becomes one. The total number of active proximal synapses is calculated for each column at every instant, and $\mathbf{2\%}$ of the columns that have the maximum number of active proximal synapses are activated. Let us denote the set of activated columns at every instant $\boldsymbol{t}$ by $\boldsymbol{C^t_a}$. The rest of the columns are inhibited and become inactive.

Every neuron has several distal segments. If the total activations in at least one of the distal segments of a neuron (i, j) exceeds a specific threshold of θ_d, that neuron enters a predictive state. If the predictive state of a neuron is denoted by $\pi_{i,j}$, then $\pi_{i,j}$ can be expressed in the following form,

$$\pi_{i,j} = \begin{cases} 1 & \text{if } \exists_d \left\| \tilde{D}_{ij}^{d} \circ A^{t} \right\|_1 > \theta_d \\ 0 & \text{otherwise} \end{cases} \tag{4.4}$$

where $\circ$ denotes the element-wise multiplication operation. In the current time step, the activation happens only in the cells of the columns, which are already active. If no cells in an active column are in a predictive state, all the cells in that column will be activated.

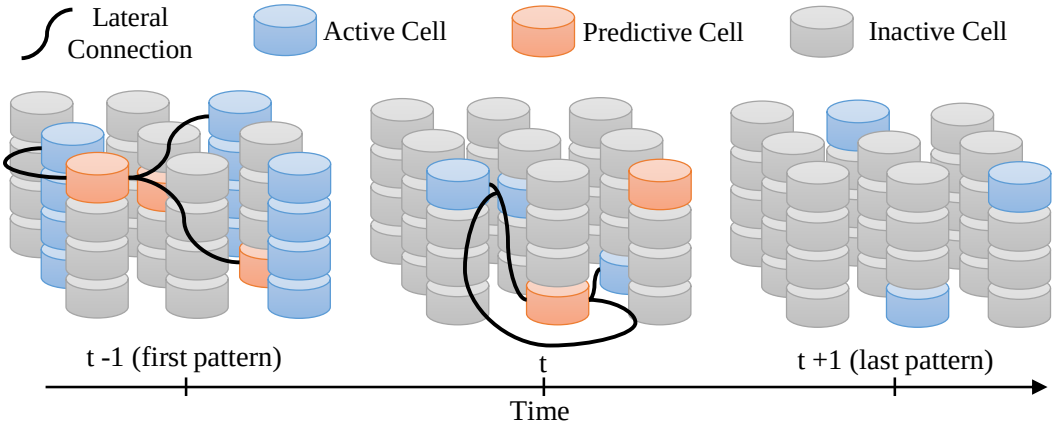


Figure 4.4: HTM cells activation process.

Figure 4.4 depicts HTM cells activation process. If any cell is in a predictive state within an active column, then this cell is preferred over the other cells for activation while the other cells in that active column are inhibited. The inhibition of the other cells within a column ensures that the activations are sparse which allows the HTM to capture multiple temporal contexts from the input data. Putting it all together, the activation of a cell (i, j) is given

by,

$$\boldsymbol{a}_{i,j}^{t} = \begin{cases} \mathbf{1} & \boldsymbol{j} \in \boldsymbol{C}_{a}^{t} \text{ and } \boldsymbol{\pi}_{i,j}^{t-1} = \mathbf{1} \\ \mathbf{1} & \boldsymbol{j} \in \boldsymbol{C}_{a}^{t} \text{ and } \sum_{i} \boldsymbol{\pi}_{i,j}^{t-1} = \mathbf{0} \\ \mathbf{0} & \text{otherwise.} \end{cases} \tag{4.5}$$

4.5.4 Learning

Learning happens following the Hebbian Learning [128] rules. This learning algorithm is designed to reinforce the synaptic connections of the segments which correctly predicted the activation at the next time step following three major acts: (i) The learning rule only updates the weights of the cells that were predicted to become active at the current time step in the previous step (they were in a predictive state); (ii) The weights of the synapses from the cells that were active in the previous step (which had an important role in taking this node to a predictive state), are increased; (iii) The weights of the synapses from the cells that were not active in the previous time step, are decreased. Formally, the fractional increase of the weights $\boldsymbol{D}_{i,j}^{d}$ of the active distal segment $\boldsymbol{d}$ of the active cell $\boldsymbol{(i,j)}$ at time $\boldsymbol{t}$ are determined by the adaptation rule,

$$\boldsymbol{\Delta D}_{i,j}^{d} = \boldsymbol{r}^{+} \hat{\boldsymbol{D}}_{i,j}^{d} \circ \boldsymbol{A}^{t-1} - \boldsymbol{r}^{-} \hat{\boldsymbol{D}}_{i,j}^{d} \circ (\mathbf{1} - \boldsymbol{A}^{t-1}) \tag{4.6}$$

where $\boldsymbol{r}^{+}$ and $\boldsymbol{r}^{-}$ are the increase and decrease rates of the permanence values of the synaptic connections. If a column becomes active and no cell in the active column is in the predictive state, then the cell with more activated dendrite segments is selected for the update as described above. Furthermore, if the weights $\boldsymbol{D}_{i,j}^{d}$ of the active distal segment $\boldsymbol{d}$ of the active cell $\boldsymbol{(i,j)}$ are wrongly predicted to become active, the weights are decreased by,

$$\boldsymbol{\Delta D}_{i,j}^{d} = -\boldsymbol{r}_{\boldsymbol{f}}^{-} \hat{\boldsymbol{D}}_{i,j}^{d} \text{ where } \boldsymbol{a}_{i,j}^{t} = \mathbf{0} \text{ and } \left\| \tilde{\boldsymbol{D}}^{d} \circ \boldsymbol{A}^{t-1} \right\|_{\mathbf{1}} > \boldsymbol{\theta}_{d} \tag{4.7}$$

where r_f^- is a decrease rate much smaller than r^-. In other words, forgetting (i.e., forgetting/dropping the temporal context that incorrectly predicted the next step) occurs at a much slower rate than learning (i.e., remembering the temporal context that correctly predicted the next step).

In summary, the learning algorithm explained here updates the lateral connections between the cells of columns by observing one point at a time (i.e., online). This update of the lateral connections between the cells occurs based on the temporal context provided by the earlier data point to the current data point. Due to contextual inhibition in the cells, the activations of the cells are sparse. Thus the temporal contexts are learned by a sparse distributed representation of the active cells and is continually updated as new data is received.

4.5.5 Hardware Trojan Detection

HTM provides a medium to build a representative model of the spatial temporal relationship between different points in a given signal. We initialize HTM by training it over collected power side-channel data for a short period of time (about 10 minutes), where we insure that HT is not activated by applying only basic test vectors that would not trigger the HT. Here, we assume that the triggered HT designer chooses a sequence of events and inputs that normally will not happen during the testing phase of the chip.

After initial training, we use Algorithm 2 for detecting HT activation in run time. At every time step $\boldsymbol{t}$, we first calculate the input $\boldsymbol{SDR_t}$. Then, spatial pooler generates the $\boldsymbol{SpatialSDR_t}$ which is equivalent to $\boldsymbol{C_a^t}$. After that, activation and learning happens in temporal pooler which gives us list of active columns and predictive columns. The difference between $\boldsymbol{ActiveCol_t}$ and $\boldsymbol{PredictiveCol_{t-1}}$ determines whether or not the input signal

at time step $\boldsymbol{t}$ is anomalous.

Algorithm 2: Using HTM for HT detection.

Input: Input side-channel data at time step t:$\{\boldsymbol{x_t}\}$

Output: Anomaly Score: $\boldsymbol{Score_t}$

$\boldsymbol{t} \leftarrow$: $\boldsymbol{Timesteps}$

$\boldsymbol{RandomScalarEncoder}$, $\boldsymbol{SpatialPooler} \leftarrow$ Initialize

$\boldsymbol{TemporalPooler} \leftarrow$ Initialize

for *each* $\boldsymbol{x_t}$ *from input data* **do**

- $\boldsymbol{InputSDR_t}$ = Encode $\boldsymbol{x_t}$ using $\boldsymbol{RandomScalarEncoder}$
- Feed the $\boldsymbol{InputSDR_t}$ to $\boldsymbol{SpatialPooler}$
- $\boldsymbol{SpatialSDR_t}$ = Column $\boldsymbol{Activation}$ in $\boldsymbol{SpatialPooler}$
- Feed the $\boldsymbol{SpatialSDR_t}$ to $\boldsymbol{TemporalPooler}$
- $\boldsymbol{TemporalSDR_t}$ = Activation and Learning
- $\{\boldsymbol{ActiveCol_t}, \boldsymbol{PredictiveCol_t}\} = \boldsymbol{TemporalSDR_t}$
- **if** $\boldsymbol{ActiveCol_t}$ - $\boldsymbol{PredictiveCol_{t-1}} >$ *Threshold* **then**
  - $\boldsymbol{Score_t} \leftarrow \boldsymbol{ActiveCol_t}$ - $\boldsymbol{PredictiveCol_{t-1}}$
  - **return** $\boldsymbol{Score_t}$
- **else**
  - $\boldsymbol{Score_t} \leftarrow \boldsymbol{0}$
  - **return** $\boldsymbol{Score_t}$

An illustrative example of how HTM works for HT detection is shown in Figure 4.5. Whenever HTM encounters a previously seen data $\boldsymbol{X_t}$ at time t, some columns enter the active state. Simultaneously, temporal memory at time step t predicts the next time step data and as a result, some of the columns enter the predictive state as well. Again, whenever HTM sees next data $\boldsymbol{x_{t+1}}$ at time step t+1, some columns enter the active state. If the difference between the number of active columns in time step t+1 and predictive columns in t is equal or less than a threshold value, then the data at time step t+1 is considered as a non-anomalous data (scenario 1). But if the difference between the number of the active

columns in the next time step and predictive columns in previous time step is greater than a threshold value, then the data at the next time step is considered as an anomalous data (scenario 2).

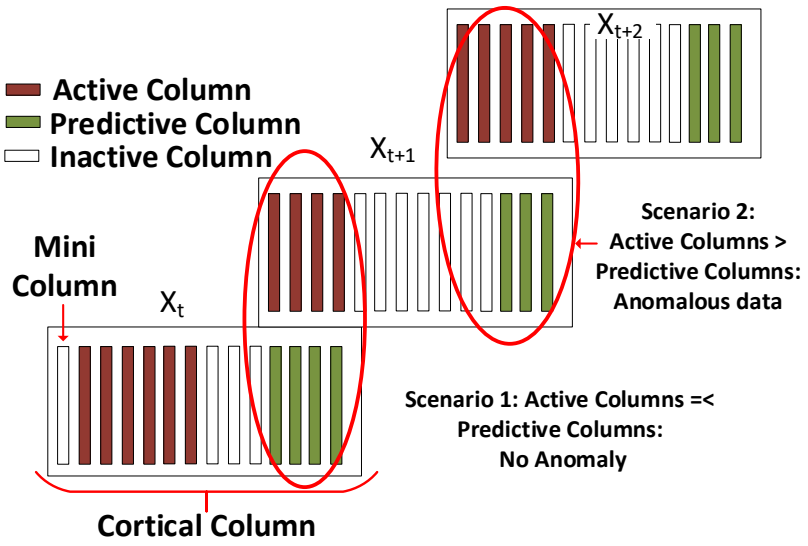


Figure 4.5: An example of anomaly detection with HTM.

4.5.6 Thresholding

In this thesis, we extended the thresholding concept following [3]. For a given current input $\boldsymbol{x_t}$, $\boldsymbol{a(x_t)}$ is a binary vector which represents the current active columns and $\boldsymbol{\pi(x_{t-1})}$ is a binary vector which represents the predictive columns. Let the prediction error, $\boldsymbol{s_t}$, be a scalar value inversely proportional to the number of bits common between $\boldsymbol{a(x_t)}$ and $\boldsymbol{\pi(x_{t-1})}$,

$$\boldsymbol{s_t} = 1 - \frac{\boldsymbol{a(x_t).\pi(x_{t-1})}}{\boldsymbol{|a(x_t)|}} \tag{4.8}$$

where $\boldsymbol{|a(x_t)|}$ is the scalar norm, i.e. the total number of 1 bits in $\boldsymbol{a(x_t)}$. If there is a shift in the behavior of the system, the prediction error will be high at the point of shift, but will automatically degrade to zero as the model adjusts itself to the "new normal". Furthermore,

Equation 4.8 results in an instantaneous measure of the predictability of current signal stream. However, in practice, the input signal is noisy and shifts in the system are common paradigm which we do not want to detect them as abnormalities. To address these issues, we model the distribution of error as an indirect metric to check for the likelihood that current state has abnormal behavior. We model the distribution of the prediction error values over a window of size $\boldsymbol{W}$ as a rolling normal distribution where the error mean, $\boldsymbol{\mu_t}$, and variance, $\boldsymbol{\sigma_t^2}$, are continuously updated as follows:

$$\boldsymbol{\mu_t = \frac{\sum_{i=0}^{i=W-1} s_{t-i}}{W}} \ , \ \boldsymbol{\sigma_t^2 = \frac{\sum_{i=0}^{i=W-1}(s_{t-i} - \mu_t)^2}{W-1}}. \tag{4.9}$$

Next, over a window of size $\boldsymbol{W'}$, we calculate the recent short term average prediction error rate ($\boldsymbol{W' \ll W}$), and the complement of the Gaussian tail probability (Q-function [70]) as our anomaly likelihood:

$$\boldsymbol{L_t = 1 - Q(\frac{\tilde{\mu}_t - \mu_t}{\sigma_t})} \tag{4.10}$$

where $\boldsymbol{\tilde{\mu}_t = \frac{\sum_{i=0}^{i=W'-1} s_{t-i}}{W'}}$. Then, we threshold $\boldsymbol{l_t}$ using a predefined parameter $\boldsymbol{\epsilon}$ to report an anomaly:

$$\boldsymbol{AnomalyDetected_t \equiv L_t \geq 1 - \epsilon.} \tag{4.11}$$

Since thresholding $\boldsymbol{L_t}$ involves thresholding a tail probability, there is an inherent upper bound limit on number alerts and false positive. If the value of $\boldsymbol{\epsilon}$ is very close to 0, it would be unlikely to get alerts with probability much higher than $\boldsymbol{\epsilon}$. Therefore, we directly follow the findings of [3] and set the value of $\boldsymbol{\epsilon}$ to $\boldsymbol{10^{-5}}$ which is claimed to work well across a large range of domain.

4.6 Evaluation

4.6.1 Experimental Setup

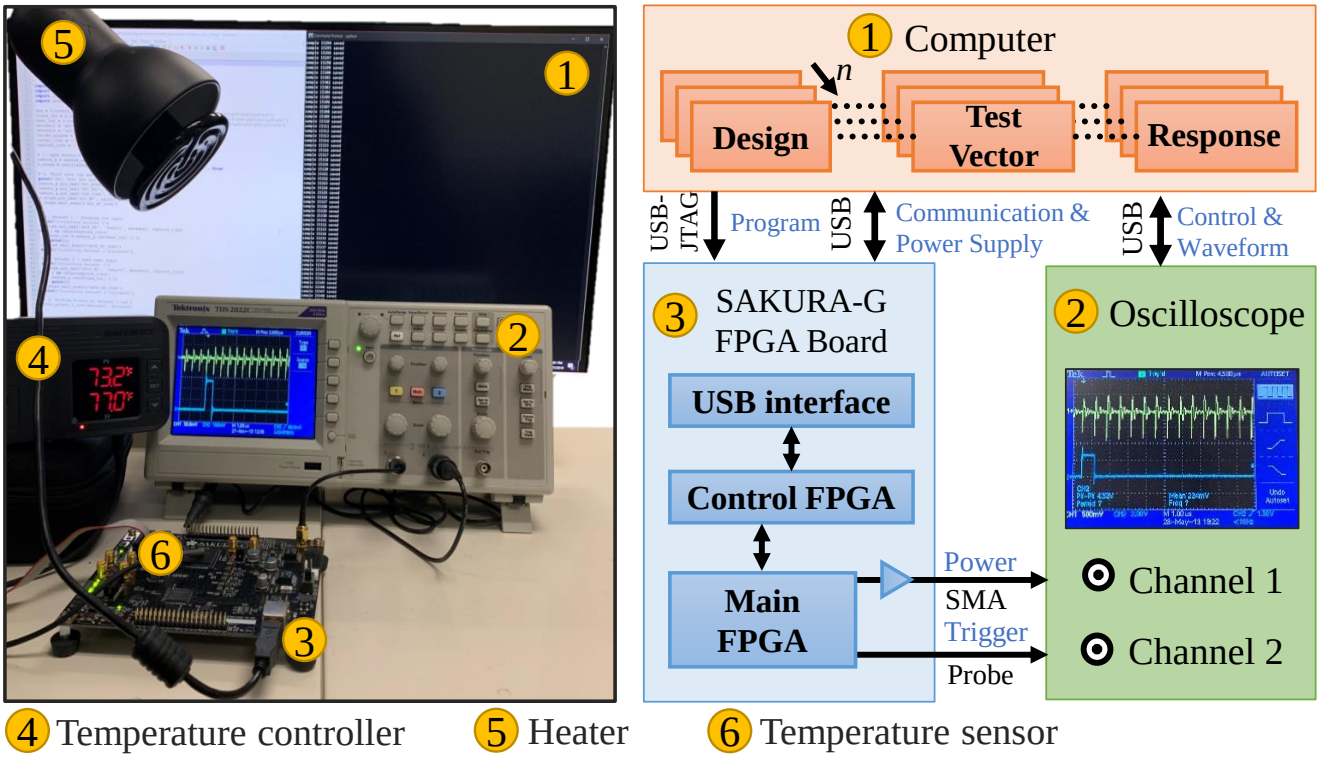


Figure 4.6: Test bed for power side-channel data collection.

To evaluate our model, we measure the power side-channel of HT benchmarks using our automated testbed. Our experimental setup is composed of a Sakura-G FPGA board, Tektronix TDS2022C oscilloscope, and computer. We implement the HT benchmarks from TrustHub [135] on the FPGA. The benchmarks include different HT added to a base circuit that is either an AES cryptography core with a 128 bits secret key or an RS232 UART serial communication circuit. The list of benchmarks is summarized in table 4.2. The computer programs the FPGA with the benchmark bitstream, sends the input test vectors, receives the output through the USB port communication, validates the output, and stores the measured power trace from the oscilloscope. For the AES circuit, the input vectors are the secret key and plain text, and the output is the encryption result (ciphertext), whereas the

RS232 circuit receives an arbitrary input vector and outputs the same. A trigger signal is generated by the FPGA that indicates the start of the operation, encryption in AES, or data transmission in RS232. The trigger signal is used by the oscilloscope to synchronize the data collection with circuit operation. In order to address the process variation, we collect the power data randomly using two Sakura-G boards. Since our approach is golden chip-free, the side-channel data is collected in the presence of HT under two scenarios: when the HT is inactive and when it is activated. Consequently, all the measured power data include the static power of HT in addition to base circuit power consumption, which makes the detection more challenging. The only difference between the two data collection scenarios is the dynamic power consumption of HT.

Table 4.2: Hardware Trojan Benchmarks. Number of LUTs for HT free AES circuit is 9707.

Benchmark	Trojan payload
AES-T500	Causes denial of service by draining the battery
AES-T600	Leaks the secret key through leakage current
AES-T700	Leaks the secret key through a power covart channel based on CDMA
AES-T800	Leaks the secret key through a power covart channel based on CDMA
AES-T1600	Leaks the secret key through RF transmission
RS232-T100	Causes denial of service by forcing the data ready signal value in receiver to 0

Benchmark	HT infected circuit size (# of LUT)	$\frac{\#HT\ LUT}{\#Base\ LUT}(\%)$
AES-T500	9763	0.58
AES-T600	9735	0.29
AES-T700	9779	0.74
AES-T800	9787	0.82
AES-T1600	10037	3.4
RS232-T100	117	4.46

4.6.2 Model Parameters

Since we propose a truly golden-chip free approach, we avoid fine-tuning the HTM model parameters to have the best performance for the given benchmarks. Instead, the results in this section are provided using the parameter values from [3]. In that work, the number

of columns is 2048 with 32 cells per column, the max number of segments per each cell is 128, and the max number of synopsis per segment is 32. In the spacial pooler, synaptic permanence values are increased by 0.003 and decreased by 0.0005 and the connections are made/abolished at the 0.2 thresholds. For the temporal pooling, the activation threshold is 13, initial permanence values are 0.21, and the value of $\boldsymbol{r}^+$ and $\boldsymbol{r}^-$ is set to 0.1.

4.6.3 Rival Detection Methods

Table 4.3 compares our approach proprieties to the most closely related works in the literature. HTM training does not involve power-hungry backpropagation or any other optimization mechanism. Hence, our proposed model is the only approach that is capable of online training to cope with physical changes such as aging and operating temperature variation that may affect the collected side-channel data. Also, we do not include any infected IC data for training the model. Hence, it is capable of detecting zero-day attacks by construct. Furthermore, our approach is a run-time detection method, which makes it suitable for detecting triggered HTs. On the downside, since our model always runs along with the target IC, it adds an overhead to the system, which is discussed in detail in Sec. 4.6.4. HTM is resilient to process variation noise because it is trained for each chip independently.

Table 4.3: Qualitative Comparison of HT detection methods.

Method	Side-Channel	Golden-chip Free	Online Learning	Zero-day Defense	Run-time	Noise Resilient
[85], [148]	Power	✓	✗	✗	✗	N/A
[57]	EM	✓	✗	✗	✗	N/A
[103]	Thermal	✗	✗	✗	✓	✓
MLP/CNN	Power	✗	✗	✗	✓	✗
[61]	Power	✓	✗	N/A	✓	✓
OC-SVM	Power	✓	✓	✓	✓	✓
HTM(Ours)	Power	✓	✓	✓	✓	✓

For quantitative comparisons, due to the lack of open-source HT detection projects, the reproduction of state-of-the-art HT detection approaches for our collected side-channel data

is not feasible. Instead, based on what has been reported in a recent survey on machine learning methods for hardware security [32], we compare our detection mechanism against One-Class Support Vector Machine (OC-SVM) [8, 75] and two powerful statistical models; A Multi-Layer Perceptron (MLP), and a Convolutional Neural Network (CNN). Here, we use a variation of OC-SVM which can be trained in an online fashion to be more comparable to HTM [47]. We report OC-SVM results with Radial Basis Function (RBF) kernel, which had the best performance among the other OC-SVM settings on the collected side-channel signals based on our preliminary experiments. The MLP consist of three fully connected layers, each with 500 nodes and ReLU activation, which is immediately followed by dropout layers with a rate of 0.2 to avoid overfitting. The CNN is composed of two convolutional layers. The numbers of filters for these two layers are 6 and 12 accordingly, while the kernel sizes are 7 and 10. These parameters are optimized with a heuristic Brute-force approach to assure the best performance of all methods that we compare our algorithm with.

The training data set for MLP and CNN includes both HT infected and HT free side-channel data for each benchmark. In this fashion, the models learn the *discriminative* features between the golden chip behavior and *known* HT infected data. Intuitively, the models that learn discriminative features will perform better for detecting known HTs. In contrast, our HTM detection approach and OC-SVM try to learn the *descriptive* features that describe the correct behavior of the system and detect the possible *anomalies*. Hence, they do not require any HT infected data for training. Although it is only fair if we compare our approach with other anomaly detection methods, we intentionally choose to compare our models with these two classifiers to demonstrate that it can achieve the best possible performance for the given given side-channel data.

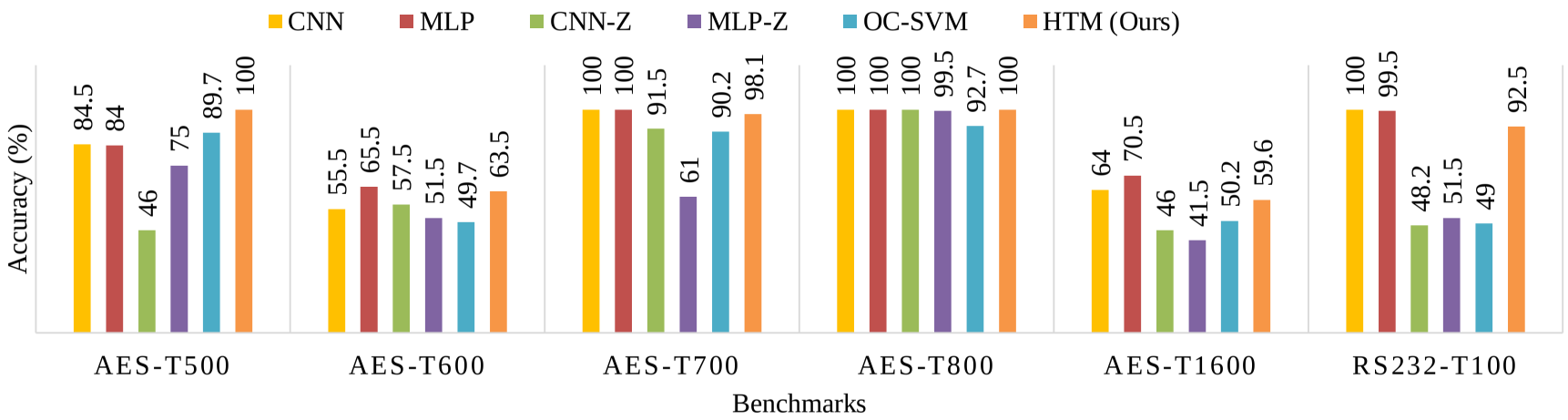


Figure 4.7: Accuracy comparison of different methods for triggered HT detection through power side channel.

4.6.4 Results

Accuracy

We use anomaly detection accuracy as a performance metric for evaluation and report our results for 6 HT benchmarks in Figure 4.7. During each benchmark operation -encryption in the AES benchmarks or data transmission in the RS232 benchmark- the circuit's power consumption is measured under two circumstances: inactive HT and triggered HT. Each model's goal is to mark the power signals as a positive sample when HT is triggered. In this figure, CNN-Z and MLP-Z have the same architecture as CNN and MLP; however, they are trained for zero-day HT attacks. The training dataset for zero-day attacks includes positive and negative samples from all the benchmarks except for the benchmark being tested. Figure 4.7 shows that the proposed HTM approach detects the activated Trojan with a high accuracy, comparable with state-of-the-art classification methods. Notably, HTM outperforms OC-SVM, which is one of the widely used unsupervised techniques for HT detection, by 15.61% accuracy margin.

Furthermore, as explained in Section 4.4.1, the high accuracy illustrates low false positive rate for all the compared models, which means that the models would not raise a false alarm when the chip operation is normal.

Case Study

We investigate the effect of HT on power side-channel in a case study on RS232-T100 benchmark that implements a pair of UART serial communication transmitter and receiver. In this study, we collect the power consumption of the circuit during one operation cycle of sending a test vector and receiving it in two cases of inactive HT and active HT. The normalized side-channel signal and its anomaly likelihood are depicted in Figure 4.8. Anomaly likeli-

hood plot is the output of HTM followed by the thresholding process discussed in Section 4.5.6. In the figure, the notable difference between the two signals demonstrates that the activation of HT has altered the power side-channel in multiple data points. Consequently, our model identifies several anomalous data points in the signal and marks this sample as HT triggered.

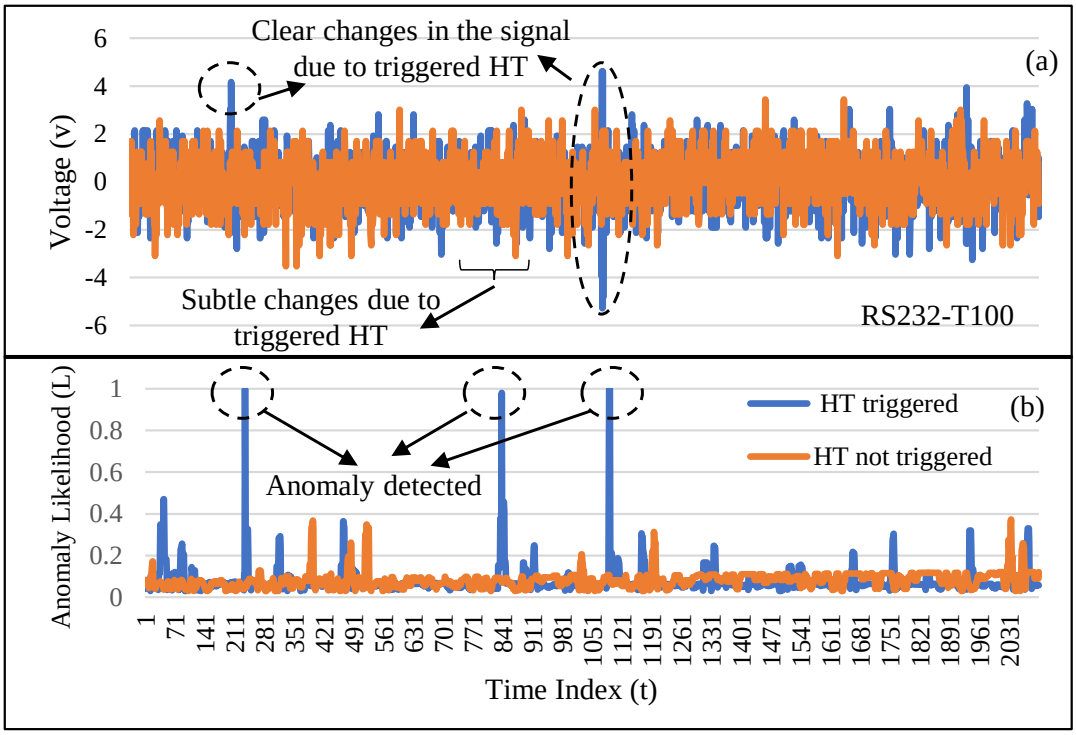


Figure 4.8: Anomaly likelihood in two side-channel signals. (a) Normalized side-channel power signals (b) Calculated anomaly likelihood based on HTM.

Temperature Effect

Figure 4.9 presents the effect of temperature change on the accuracy of different HT detection method for the AES-T800 benchmark. Using the heater in our testbed, we increase the temperature in the chip's environment. For each **10**°C rise in temperature, we collect power side-channel measurements in the cases of HT inactive and HT active. In this experiment, we keep the retraining feature of HTM and OC-SVM on to adapt to the temperature variation. Solely for comparison reasons, we also assume that it is possible to collect golden chip data

in different temperatures for training MLP and CNN. Therefore, the accuracy of MLP and CNN in this figure shows the best possible achievable accuracy by a statistical model. As Figure 4.9 demonstrates, the rise in temperature results in an immediate accuracy drop for OC-SVM while our proposed approach is designed to be immune to such changes in the system. Indeed, even in **85**°C, HTM has 80% accuracy, which is comparable to the best achievable HT detection accuracy.

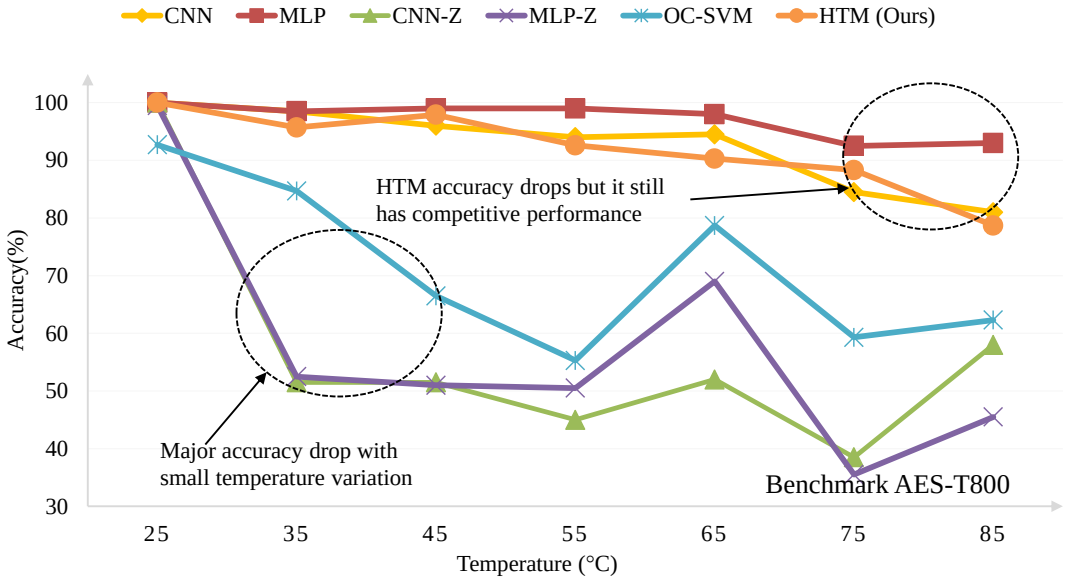


Figure 4.9: Effect of temperature change on different methods.

Noise Effect

In addition to temperature changes, other factors such as process variation, aging, and voltage level variation may cause changes in the power consumption patterns. Here, we model these changes by adding a synthetic white noise $\mathcal{N}(\mu, \sigma^2)$ to the test signals. Figure 4.10 shows the change in the accuracy of different HT detection methods when encountering varying levels of noise in the collected signal for the AES-T800 benchmark. Since continuous retraining of neural network-based detection algorithms is not feasible in practice, training data set of CNN, MLP, CNN-Z, and MLP-Z does not include the added noise. However, online learning is always active for OC-SVM and HTM. Therefore, they learn the effect of

the added noise while working on the test signals, and they report a consistent accuracy regardless of the added noise. As shown in Figure 4.10, the accuracy of CNN and MLP immediately drops because these methods infer all the noisy test signals as HT infected (high false-positive detection rate). CNN-Z and MLP-Z have never been trained with the collected training signal for the AES-T500 benchmark, and they show an arbitrary reaction to the added noise, as expected.

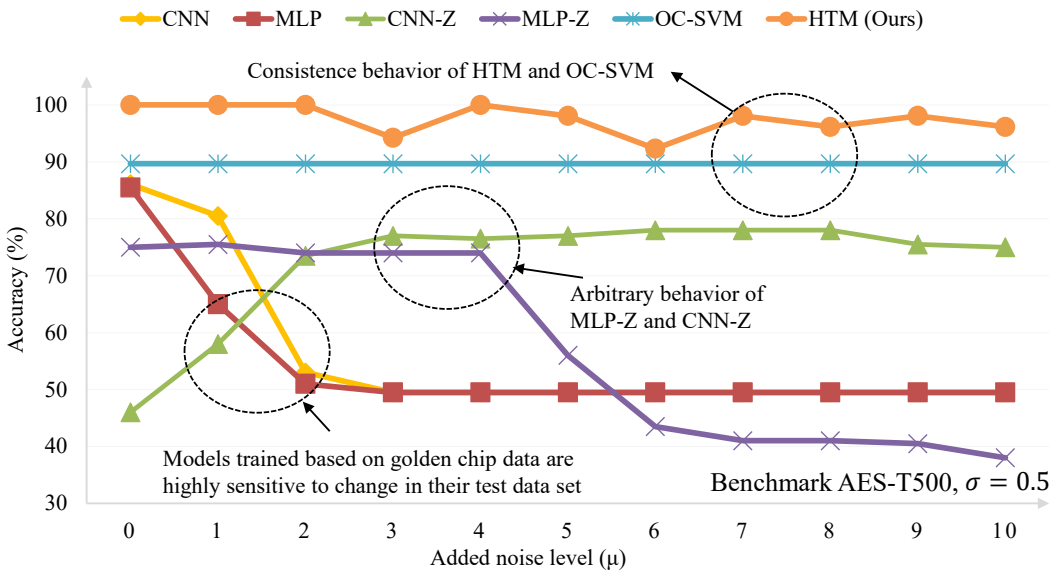


Figure 4.10: Noise addition to the test side-channel data.

Model Parameters Change

The majority of HTM advantages are because they perform simplistic tasks over high dimensional data points. To elaborate on this, we change the number of columns and cells per column to evaluate our model sensitivity to these factors. As shown in Table 4.4, the accuracy of HTM for AES-700 benchmarks drops when we use a small number of cells/columns. Note that, in this chapter, we use 2048 columns with 32 cells per column, as it is suggested by [3] to work for a wide range of anomaly detection applications. In other terms, we do not refer to this table to choose optimal values for our HTM model parameters.

Table 4.4: The effect of model parameters change for HT detection accuracy over AES-T700 benchmark.

# Cells \ # Columns	4	8	16	32	64
128	0.5	0.5	0.5	0.5	0.5
512	0.5	0.5	0.73	1	1
1024	0.5	0.5	0.61	1	1
2048	0.5	0.5	0.79	0.98	1

Model Overhead

This work focuses on proposing a methodology for golden-chip free zero-day HT detection as a proof of concept. As described in Section 4.6.1, we use an oscilloscope for data collection and process the data in a separate computer system. However, in practice, a hardware implementation of HTM should be utilized to provide an efficient run-time monitoring system. To this end, we include the information regarding the overhead of HTM usage compared to other available models.

For hardware implementation of HTM, we borrow the neuromorphic architecture proposed by Zyarah et al. [155] due to its competitive performance. In this architecture, each cell is a minor core within 2D columns that can establish lateral synaptic connections with other cells to learn sequences. In addition, a memory unit is associated with each column to syntactically establish the dynamic behavior of the interconnections rather than rigid wires among the cells, which allows us to implement HTM on an FPGA platform. For OC-SVM implementation with online learning, we use hardware implementation of the sequential minimal algorithm that can be found in [91]. Neural network acceleration through FPGA is a rapidly evolving research field. In order to be fair, rather than implementing CNN and MLP, we report the latest achievements for implementing MobileNet [82] over the FPGA platform, which has a middling number of parameters between CNN and MLP. Table 4.5 represents

the essential resource utilization and power consumption of hardware implementation of each model on FPGA platform. Among these three methods, HTM has minimum power consumption. Also, note that the reported power for MobileNet includes only inference phase of this model while the power for HTM and OC-SVM includes both training and inference.

Table 4.5: Resource utilization and power consumption.

Methods	MobileNet	OC-SVM	HTM (Ours)
Platform	Zynq7z045	Zynq7z020	Zynq7z045
Frequency(MHz)	100	100	100
LUT	9203	14529	6349
LUTRAM	746	591	9108
Power(W)*	2.15	0.524	0.417

* Does not include the ARM core power consumption.

In this chapter, we run our experiments on a 3.7 GHz Intel Core i7 windows platform. On average, our experiments show that a delay of $\mathbf{7.2}\boldsymbol{ms}$ for learning and making prediction of each sample should be expected. However, the recent advances for hardware implantation of HTM have shown that more 1000X speed up for HTM is achievable. For instance, the authors of [155] show that an FPGA implementation of HTM model will require only $\mathbf{5.75}\boldsymbol{\mu s}$ for spatial pooling and $\mathbf{5.052}\boldsymbol{\mu s}$ for performing temporal memory tasks.

4.7 Discussion

4.7.1 Run-time HT Detection

As the last line of defense, run-time HT detection approaches play a crucial role in detecting subtle HTs that are not detected in the verification phase. This chapter provides evidence that HTM can be used as a powerful run-time HT detection mechanism robust to the system

variation. Similar to other HT detection methodologies, our approach consumes a considerable amount of energy. Hence, it is not suitable for circuits that rely on limited power resources. In such cases, we suggest to decouple the detection mechanism from the main circuit and use it as a service in the cloud. Recent advances in communication technologies (5G) make it possible to establish a high speed, low power, and low latency connection between the cloud and the target chip to transfer raw measurements.

4.7.2 Using local and other side-channels

In addition to the power side-channel, various other modalities can be used to monitor the chip. Currently, many ICs are equipped with sensors that provide fine grid thermal and power consumption information. Multiple works in the literature have shown that the use of electromagnetic side-channel of the chip can result in high accuracy HT detection. In this work, we only use the power side-channel for evaluating the performance of our detection approach due to ease of access. However, incorporating other modalities can improve the results, which we leave for our future work. Furthermore, we have considered only one point for our power measurements which was sufficient for the tested benchmarks. In practice, we suggest to use a more fine grid points for side-channel signals collection.

4.7.3 HTM Characteristics for HT Detection

The high dimensional form of encoded input in SDRs followed by spatial and temporal pooling gives HTM the capacity to capture complex patterns in the signal with minimum required computations. The complexity of all the operations in the HTM is at most $\boldsymbol{n \log n}$, where $\boldsymbol{n}$ is the size of the encoder. Moreover, the high dimensionality of the data flow in HTM makes it highly resilient against noise in the input signal compared to neural networks and other ML techniques. Furthermore, the majority of unsupervised ML techniques cannot

adapt to changes in the signal since retraining the model in real-time is not feasible. However, HTM uses Hebbian Learning rules that are a light training methodology and can be applied in real-time. Last but not least, in our experiments, we notice that training HTM requires much less number of samples than other unsupervised ML techniques.

4.8 Conclusion

In this chapter, we proposed a novel idea of using an HTM for IC behavior monitoring to detect stealthy HTs which cannot be detected during chip verification process. Our model is solely trained based on DUT, hence it is golden chip free and does not require access to a library of known HTs. Furthermore, our proposed method can adopt to effects such as aging, process variation, and temperature change due to its low-overhead real-time retraining. The results show that our proposed detection mechanism can detect 92.20% of triggered HT in five benchmarks while consuming less power compared to state-of-the-art machine learning techniques.

Bibliography

[1] *Screening framework guidance for providers of synthetic double-stranded DNA*. US Department of Health and Human Services, 2010.

[2] S. Adee. The hunt for the kill switch. *iEEE SpEctrum*, 45(5):34–39, 2008.

[3] S. Ahmad, A. Lavin, S. Purdy, and Z. Agha. Unsupervised real-time anomaly detection for streaming data. *Neurocomputing*, 262:134–147, 2017.

[4] Y. Alkabani and F. Koushanfar. Consistency-based characterization for ic trojan detection. In *Proceedings of the 2009 International Conference on Computer-Aided Design*, pages 123–127. ACM, 2009.

[5] S. F. Altschul, T. L. Madden, A. A. Schäffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman. Gapped blast and psi-blast: a new generation of protein database search programs. *Nucleic acids research*, 25(17):3389–3402, 1997.

[6] Applied Biosystems. Applied biosystems 3400 DNA synthesizer: User guide. 2010.

[7] A. J. Aviv, B. Sapp, M. Blaze, and J. M. Smith. Practicality of accelerometer side channels on smartphones. In *Proceedings of the 28th Annual Computer Security Applications Conference*, pages 41–50. ACM, 2012.

[8] C. Bao, D. Forte, and A. Srivastava. On application of one-class svm to reverse engineering-based hardware trojan detection. In *Fifteenth International Symposium on Quality Electronic Design*, pages 47–54. IEEE, 2014.

[9] R. G. Baraniuk. Compressive sensing [lecture notes]. *IEEE signal processing magazine*, 24(4):118–121, 2007.

[10] C. Bayens, G. L. Le T, 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)*, pages 1181–1198. USENIX Association, 2017.

[11] Y. Benjamini and D. Yekutieli. The control of the false discovery rate in multiple testing under dependency. *Annals of statistics*, pages 1165–1188, 2001.

[12] Bill and Melinda Gates foundation. Collaboration of biotech, academia, and nonprofit pharma could significantly reduce cost, boost supplies of antimalarial drug, 2004.

[13] R. Bitter, T. Mohiuddin, and M. Nawrocki. *LabVIEW: Advanced programming techniques*. CRC press, 2017.

[14] G. Bloom, B. Narahari, R. Simha, and J. Zambreno. Providing secure execution environments with a last line of defense against trojan circuit attacks. *computers & security*, 28(7):660–669, 2009.

[15] C. Bock. Sysml and uml 2 support for activity modeling. *Systems Engineering*, 9(2):160–186, 2006.

[16] C. Bolton, S. Rampazzi, C. Li, A. Kwong, W. Xu, and K. Fu. Blue note: How intentional acoustic interference damages availability and integrity in hard disk drives and operating systems. In *Blue Note: How Intentional Acoustic Interference Damages Availability and Integrity in Hard Disk Drives and Operating Systems*, page 0. IEEE, 2018.

[17] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[18] H. Bügl, J. P. Danner, R. J. Molinari, J. T. Mulligan, H.-O. Park, B. Reichert, D. A. Roth, R. Wagner, B. Budowle, R. M. Scripp, et al. Dna synthesis and biological security. *Nature biotechnology*, 25(6):627, 2007.

[19] P. Calain, M. C. Monroe, and S. T. Nichol. Ebola virus defective interfering particles and persistent infection. *Virology*, 262(1):114–128, 1999.

[20] P. Chapman and D. Evans. Automated black-box detection of side-channel vulnerabilities in web applications. In *Proceedings of the 18th ACM conference on Computer and communications security*, pages 263–274. ACM, 2011.

[21] D. K. Chaturvedi. *Modeling and simulation of systems using MATLAB and Simulink*. CRC press, 2017.

[22] S. R. Chhetri, S. Faezi, and M. A. Al Faruque. Information leakage-aware computer-aided cyber-physical manufacturing. *IEEE Transactions on Information Forensics and Security*, 13(9):2333–2344, 2018.

[23] S. R. Chhetri, S. Faezi, and M. A. A. Faruque. Fix the leak!: an information leakage aware secured cyber-physical manufacturing system. In *Proceedings of the Conference on Design, Automation & Test in Europe*, pages 1412–1417. European Design and Automation Association, 2017.

[24] M. Christ, A. W. Kempa-Liehr, and M. Feindt. Distributed and parallel time series feature extraction for industrial big data applications. *arXiv preprint arXiv:1610.07717*, 2016.

[25] S. S. Clark, B. Ransford, A. Rahmati, S. Guineau, J. Sorber, W. Xu, K. Fu, A. Rahmati, M. Salajegheh, D. Holcomb, et al. Wattsupdoc: Power side channels to nonintrusively discover untargeted malware on embedded medical devices. In *HealthTech*, 2013.

[26] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[27] F. Courbon, P. Loubet-Moundi, J. J. Fournier, and A. Tria. A high efficiency hardware trojan detection technique based on fast sem imaging. In *2015 Design, Automation & Test in Europe Conference & Exhibition*, pages 788–793. IEEE, 2015.

[28] Q. Cui, K. Sun, S. Wang, L. Zhang, and D. Li. Hardware trojan detection based on cluster analysis of mahalanobis distance. In *2016 8th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)*, volume 1, pages 234–238. IEEE, 2016.

[29] T. G. Dietterich. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer, 2000.

[30] P. Du, W. A. Kibbe, and S. M. Lin. Improved peak detection in mass spectrum by incorporating continuous wavelet transform-based pattern matching. *Bioinformatics*, 22(17):2059–2065, 2006.

[31] S. R. Eddy. Hidden markov models. *Current opinion in structural biology*, 6(3):361–365, 1996.

[32] R. Elnaggar and K. Chakrabarty. Machine learning for hardware security: Opportunities and risks. *Journal of Electronic Testing*, 34(2), 2018.

[33] R. Elnaggar, K. Chakrabarty, and M. B. Tahoori. Run-time hardware trojan detection using performance counters. In *2017 IEEE International Test Conference (ITC)*, pages 1–10. IEEE, 2017.

[34] D. Eppestein. ***k***-best enumeration. *CoRR*, abs/1412.5075, 2014.

[35] D. Eppstein. Finding the k shortest paths. *SIAM Journal on computing*, 28(2):652–673, 1998.

[36] S. Faezi. Data-driven modeling for minimizing the side-channel information leakage in additive manufacturing. Master's thesis, UC Irvine, 2017.

[37] S. Faezi, S. R. Chhetri, A. V. Malawade, J. C. Chaput, W. Grover, P. Brisk, and M. A. Al Faruque. Oligo-snoop: A non-invasive side channel attack against dna synthesis machines. In *Network and Distributed Systems Security (NDSS) Symposium 2019*, 2019.

[38] S. Faezi, R. Yasaei, and M. A. Al Faruque. Htnet: Transfer learning for golden chip-free hardware trojan detection. In *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1484–1489. IEEE, 2021.

[39] S. Faezi, R. Yasaei, A. Barua, and M. A. Al Faruque. Brain-inspired golden chip free hardware trojan detection. *IEEE Transactions on Information Forensics and Security*, 16:2697–2708, 2021.

[40] F. J. Fahy. *Foundations of engineering acoustics*. Elsevier, 2000.

[41] A. Faruque, M. Abdullah, 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*, page 19. IEEE Press, 2016.

[42] Federal select agent program. Select agents and toxin list, 2017.

[43] G. D. Forney. The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, 1973.

[44] D. Forte, C. Bao, and A. Srivastava. Temperature tracking: An innovative run-time approach for hardware trojan detection. In *2013 IEEE/ACM International Conference on Computer-Aided Design*, pages 532–539. IEEE, 2013.

[45] P. Fritzson and V. Engelson. Modelica—a unified object-oriented language for system modeling and simulation. In *European Conference on Object-Oriented Programming*, pages 67–90. Springer, 1998.

[46] S. Galanie, K. Thodey, I. J. Trenchard, M. Filsinger Interrante, and C. D. Smolke. Complete biosynthesis of opioids in yeast. *Science*, 349(6252):1095–100, Sep 2015.

[47] C. Gautam, A. Tiwari, S. Suresh, and K. Ahuja. Adaptive online learning with regularized kernel for one-class classification. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2019.

[48] R. Gayatri, Y. Gayatri, C. Mitra, S. Mekala, and M. Priyatharishini. System level hardware trojan detection using side-channel power analysis and machine learning. In *2020 5th International Conference on Communication and Electronics Systems (ICCES)*, 2020.

[49] S. Ghandali, G. T. Becker, D. Holcomb, and C. Paar. A design methodology for stealthy parametric trojans and its application to bug attacks. In *International Conference on Cryptographic Hardware and Embedded Systems*, pages 625–647. Springer, 2016.

[50] D. G. Gibson, J. I. Glass, C. Lartigue, V. N. Noskov, R.-Y. Chuang, M. A. Algire, G. A. Benders, M. G. Montague, L. Ma, M. M. Moodie, C. Merryman, S. Vashee, R. Krishnakumar, N. Assad-Garcia, C. Andrews-Pfannkoch, E. A. Denisova, L. Young, Z.-Q. Qi, T. H. Segall-Shapiro, C. H. Calvey, P. P. Parmar, C. A. Hutchison, 3rd, H. O. Smith, and J. C. Venter. Creation of a bacterial cell controlled by a chemically synthesized genome. *Science*, 329(5987):52–6, Jul 2010.

[51] D. V. Goeddel, D. G. Kleid, F. Bolivar, H. L. Heyneker, D. G. Yansura, R. Crea, T. Hirose, A. Kraszewski, K. Itakura, and A. D. Riggs. Expression in escherichia coli of chemically synthesized genes for human insulin. *Proceedings of the National Academy of Sciences*, 76(1):106–110, 1979.

[52] I. Graja, S. Kallel, N. Guermouche, S. Cheikhrouhou, and A. Hadj Kacem. A comprehensive survey on modeling of cyber-physical systems. *Concurrency and Computation: Practice and Experience*, 32(15):e4850, 2020.

[53] A. Hagberg, P. Swart, and D. S Chult. Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.

[54] M. Hähnel, W. Cui, and M. Peinado. High-resolution side channels for untrusted operating systems. In *2017 USENIX Annual Technical Conference (USENIX ATC 17)*, pages 299–312, 2017.

[55] T. Hastie, S. Rosset, J. Zhu, and H. Zou. Multi-class adaboost. *Statistics and its Interface*, 2(3):349–360, 2009.

[56] J. Hawkins and S. Ahmad. Why neurons have thousands of synapses, a theory of sequence memory in neocortex. *Frontiers in neural circuits*, 10:23, 2016.

[57] J. He, Y. Zhao, X. Guo, and Y. Jin. Hardware trojan detection through chip-free electromagnetic side-channel statistical analysis. *IEEE Transactions on Very Large Scale Integration Systems*, 25(10).

[58] G. E. Hinton. Connectionist learning procedures. In *Machine Learning, Volume III*, pages 555–610. Elsevier, 1990.

[59] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[60] A. Hojjati, A. Adhikari, K. Struckmann, E. Chou, T. N. Tho Nguyen, K. Madan, M. S. Winslett, C. A. Gunter, and W. P. King. Leave your phone at the door: Side channels that reveal factory floor secrets. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 883–894. ACM, 2016.

[61] T. Hoque, S. Narasimhan, X. Wang, S. Mal-Sarkar, and S. Bhunia. Golden-free hardware trojan detection with high sensitivity under process noise. *Journal of Electronic Testing*, 33(1):107–124, 2017.

[62] G. Hospodar, B. Gierlichs, E. De Mulder, I. Verbauwhede, and J. Vandewalle. Machine learning in side-channel analysis: a first study. *Journal of Cryptographic Engineering*, 1(4):293, 2011.

[63] K. Hu, A. N. Nowroz, S. Reda, and F. Koushanfar. High-sensitivity hardware trojan detection using multimodal characterization. In *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 1271–1276. EDA Consortium, 2013.

[64] Y. Huang, S. Bhunia, and P. Mishra. Mers: statistical test generation for side-channel analysis based trojan detection. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 130–141, 2016.

[65] Z. Huang, Q. Wang, Y. Chen, and X. Jiang. A survey on machine learning against hardware trojan attacks: Recent advances and challenges. *IEEE Access*, 8:10796–10826, 2020.

[66] T. Iwase, Y. Nozaki, M. Yoshikawa, and T. Kumaki. Detection technique for hardware trojans using machine learning in frequency domain. In *2015 IEEE 4th Global Conference on Consumer Electronics (GCCE)*, pages 185–186. IEEE, 2015.

[67] D. Jap, W. He, and S. Bhasin. Supervised and unsupervised machine learning for side-channel based trojan detection. In *2016 IEEE 27th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, pages 17–24. IEEE, 2016.

[68] Y. Jin and Y. Makris. Hardware trojan detection using path delay fingerprint. In *2008 IEEE International workshop on hardware-oriented security and trust*, pages 51–57. IEEE, 2008.

[69] Y. Jin and D. Sullivan. Real-time trust evaluation in integrated circuits. In *2014 Design, Automation & Test in Europe Conference & Exhibition*, pages 1–6. IEEE, 2014.

[70] G. K. Karagiannidis and A. S. Lioumpas. An improved approximation for the gaussian q-function. *IEEE Communications Letters*, 11(8):644–646, 2007.

[71] N. Karimi, J.-L. Danger, and S. Guilley. On the effect of aging in detecting hardware trojan horses with template analysis. In *2018 IEEE 24th International Symposium on On-Line Testing And Robust System Design (IOLTS)*, pages 281–286. IEEE, 2018.

[72] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[73] R. Kohavi et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, volume 14, pages 1137–1145. Montreal, Canada, 1995.

[74] A. Kulkarni, Y. Pino, and T. Mohsenin. Adaptive real-time trojan detection framework through machine learning. In *2016 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 120–123. IEEE, 2016.

[75] A. Kulkarni, Y. Pino, and T. Mohsenin. Svm-based real-time hardware trojan detection for many-core platform. In *2016 17th International Symposium on Quality Electronic Design (ISQED)*, pages 362–367. IEEE, 2016.

[76] P. Kumar and R. Srinivasan. Detection of hardware trojan in sea using path delay. In *2014 IEEE Students' Conference on Electrical, Electronics and Computer Science*, pages 1–6. IEEE, 2014.

[77] A. C. Larsen, A. Gillig, P. Shah, S. P. Sau, K. E. Fenton, and J. C. Chaput. General approach for characterizing in vitro selected peptides with protein binding affinity. *Analytical chemistry*, 86(15):7219–7223, 2014.

[78] E. A. Lee. The past, present and future of cyber-physical systems: A focus on models. *Sensors*, 15(3):4837–4869, 2015.

[79] H. Li, Q. Liu, and J. Zhang. A survey of hardware trojan threat and defense. *Integration*, 55:426–437, 2016.

[80] J. Li, L. Ni, J. Chen, and E. Zhou. A novel hardware trojan detection based on bp neural network. In *2016 2nd IEEE International Conference on Computer and Communications*, pages 2790–2794. IEEE, 2016.

[81] S. Li, B. W. Karney, and G. Liu. Fsi research in pipeline systems–a review of the literature. *Journal of Fluids and Structures*, 57:277–297, 2015.

[82] J. Liao, L. Cai, Y. Xu, and M. He. Design of accelerator for mobilenet convolutional neural network based on fpga. In *2019 IEEE 4th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, volume 1, pages 1392–1396. IEEE, 2019.

[83] W. Liao, J. M. Basile, and L. He. Leakage power modeling and reduction with data retention. In *Proceedings of the 2002 IEEE/ACM international conference on Computer-aided design*, pages 714–719. ACM, 2002.

[84] Y. Liu, R. P. Dick, L. Shang, and H. Yang. Accurate temperature-dependent integrated circuit leakage power estimation is easy. In *2007 Design, Automation & Test in Europe Conference & Exhibition*, pages 1–6. IEEE, 2007.

[85] Y. Liu, K. Huang, and Y. Makris. Hardware trojan detection through golden chip-free statistical side-channel fingerprinting. In *Proceedings of the 51st Annual Design Automation Conference*. ACM, 2014.

[86] Y. Liu, Y. Jin, A. Nosratinia, and Y. Makris. Silicon demonstration of hardware trojan design and detection in wireless cryptographic ics. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 25(4):1506–1519, 2016.

[87] F. K. Lodhi, I. Abbasi, F. Khalid, O. Hasan, F. Awwad, and S. R. Hasan. A self-learning framework to detect the intruded integrated circuits. In *2016 IEEE International Symposium on Circuits and Systems*, pages 1702–1705. IEEE, 2016.

[88] F. K. Lodhi, S. R. Hasan, O. Hasan, and F. Awwadl. Power profiling of microcontroller's instruction set for runtime hardware trojans detection without golden circuit models. In *Design, Automation & Test in Europe Conference & Exhibition, 2017*, pages 294–297. IEEE, 2017.

[89] M. Loog, R. P. W. Duin, and R. Haeb-Umbach. Multiclass linear dimension reduction by weighted pairwise fisher criteria. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(7):762–766, 2001.

[90] D. Lorenz. *Aging analysis of digital integrated circuits*. PhD thesis, Technische Universität München, 2012.

[91] V. Mahadevan. Code base for svms smo algorithm and implementation in hardware. `https://github.com/venkatesh20/SVM`, 2015.

[92] I. Martinovic, D. Davies, M. Frank, D. Perito, T. Ros, and D. Song. On the feasibility of side-channel attacks with brain-computer interfaces. In *USENIX security symposium*, pages 143–158, 2012.

[93] L. Masure, C. Dumas, and E. Prouff. A comprehensive study of deep learning for side-channel analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 348–375, 2020.

[94] Mathworks Co. Matlab r2017b, 2017.

[95] L. McBride and M. Caruthers. An investigation of several deoxynucleoside phosphoramidites useful for synthesizing deoxyoligonucleotides. *Tetrahedron Letters*, 24(3):245 – 248, 1983.

[96] A. L. McGuire, R. Fisher, P. Cusenza, K. Hudson, M. A. Rothstein, D. McGraw, S. Matteson, J. Glaser, and D. E. Henley. Confidentiality, privacy, and security of genetic and genomic test information in electronic health records: points to consider. *Genetics in Medicine*, 10(7):495, 2008.

[97] D. H. McKnight and N. L. Chervany. The meanings of trust. 1996.

[98] G. Mohammadi. *Trustworthy Cyber-Physical Systems*. Springer, 2019.

[99] S. Narasimhan, X. Wang, D. Du, R. S. Chakraborty, and S. Bhunia. Tesr: A robust temporal self-referencing approach for hardware trojan detection. In *2011 IEEE International Symposium on Hardware-Oriented Security and Trust*, pages 71–74. IEEE, 2011.

[100] P. Ney, K. Koscher, L. Organick, L. Ceze, and T. Kohno. Computer security, privacy, and dna sequencing: Compromising computers with synthesized dna, privacy leaks, and more. In *26th USENIX Security Symposium.[October 25, 2017]*, 2017.

[101] L. Ni, J. Li, S. Lin, and D. Xin. A method of noise optimization for hardware trojans detection based on bp neural network. In *2016 2nd IEEE International Conference on Computer and Communications*, pages 2800–2804. IEEE, 2016.

[102] A. Nouri and C. F. Chyba. Dna synthesis security. In *Gene Synthesis*, pages 285–296. Springer, 2012.

[103] A. N. Nowroz, K. Hu, F. Koushanfar, and S. Reda. Novel techniques for high-sensitivity hardware trojan detection using thermal and power maps. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 33(12):1792–1805, 2014.

[104] J. Peccoud, J. E. Gallegos, R. Murch, W. G. Buchholz, and S. Raman. Cyberbiosecurity: From naive trust to risk awareness. *Trends in biotechnology*, 36(1):4–7, 2018.

[105] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[106] P. Perera and V. M. Patel. Learning deep features for one-class classification. *IEEE Transactions on Image Processing*, 28(11), 2019.

[107] R. Polikar. Ensemble based systems in decision making. *IEEE Circuits and systems magazine*, 6(3):21–45, 2006.

[108] M. Potkonjak, A. Nahapetian, M. Nelson, and T. Massey. Hardware trojan horse detection using gate-level characterization. In *Proceedings of the 46th Annual Design Automation Conference*, pages 688–693. ACM, 2009.

[109] C. Ptolemaeus. *System design, modeling, and simulation: using Ptolemy II*, volume 1. Ptolemy. org Berkeley, 2014.

[110] S. Purdy. Encoding data for htm systems. *arXiv preprint arXiv:1602.05925*, 2016.

[111] R. M. Rad, X. Wang, M. Tehranipoor, and J. Plusquellic. Power supply signal calibration techniques for improving detection resolution to hardware trojans. In *Proceedings of the 2008 IEEE/ACM International Conference on Computer-Aided Design*, pages 632–639. IEEE Press, 2008.

[112] K. Reshma, M. Priyatharishini, and M. N. Devi. Hardware trojan detection using deep learning technique. In *Soft Computing and Signal Processing*, pages 671–680. Springer, 2019.

[113] D.-K. Ro, E. M. Paradise, M. Ouellet, K. J. Fisher, K. L. Newman, J. M. Ndungu, K. A. Ho, R. A. Eachus, T. S. Ham, J. Kirby, M. C. Y. Chang, S. T. Withers, Y. Shiba, R. Sarpong, and J. D. Keasling. Production of the antimalarial drug precursor artemisinic acid in engineered yeast. *Nature*, 440(7086):940–3, Apr 2006.

[114] Robert F. Service. Dna could store all of the world's data in one room, 2017.

[115] S. J. Russell and P. Norvig. *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,, 2016.

[116] H. Salmani. Cotd: Reference-free hardware trojan detection and recovery based on controllability and observability in gate-level netlist. *IEEE Transactions on Information Forensics and Security*, 12(2), 2017.

[117] H. Salmani, M. Tehranipoor, and R. Karri. On design vulnerability analysis and trust benchmarks development. In *2013 IEEE 31st international conference on computer design (ICCD)*. IEEE, 2013.

[118] M. Schmidt and G. Giersch. Dna synthesis and security. *DNA microarrays, synthesis and synthetic DNA*, pages 285–300, 2011.

[119] sckit-learn developers. Multi-layer perceptron classifier, 2018.

[120] Sensaphone. Sensaphone 1400 environmental monitoring system, 2017.

[121] B. Shakya, T. He, H. Salmani, D. Forte, S. Bhunia, and M. Tehranipoor. Benchmarking of hardware trojans and maliciously affected circuits. *Journal of Hardware and Systems Security*, 1(1), 2017.

[122] R. Shende and D. D. Ambawade. A side channel based power analysis technique for hardware trojan detection using statistical learning approach. In *2016 thirteenth international conference on wireless and optical communications networks (WOCN)*, pages 1–4. IEEE, 2016.

[123] J.-J. Shu. A new integrated symmetrical table for genetic codes. *Biosystems*, 151:21–26, 2017.

[124] R. Singh. Synthetic biology market by products (dna synthesis, oligonucleotide synthesis, synthetic dna, synthetic genes, synthetic cells, xna) and technology (genome engineering, microfluidics technologies, dna synthesis and sequencing technologies) - global opportunity analysis and industry forecast, 2013 - 2020, 2014.

[125] L. Smith and C. Higgins. Scholars or spies?, 2017.

[126] A. Søe-Knudsen and S. Sorokin. Modelling of linear wave propagation in spatial fluid filled pipe systems consisting of elastic curved and straight elements. *Journal of Sound and Vibration*, 329(24):5116–5146, 2010.

[127] D. Solomatine, L. M. See, and R. Abrahart. Data-driven modelling: concepts, approaches and experiences. *Practical hydroinformatics*, pages 17–30, 2009.

[128] S. Song, K. D. Miller, and L. F. Abbott. Competitive hebbian learning through spike-timing-dependent synaptic plasticity. *Nature neuroscience*, 3(9):919, 2000.

[129] F. Stellari, P. Song, and H. A. Ainspan. Functional block extraction for hardware security detection using time-integrated and time-resolved emission measurements. In *2014 IEEE 32nd VLSI Test Symposium (VTS)*, pages 1–6. IEEE, 2014.

[130] H. Su, F. Liu, A. Devgan, E. Acar, and S. Nassif. Full chip leakage estimation considering power supply and temperature variations. In *Proceedings of the 2003 international symposium on Low power electronics and design*, pages 78–83. ACM, 2003.

[131] V. Subramanian, A. S. Uluagac, H. Cam, and R. A. Beyah. Examining the characteristics and implications of sensor side channels. In *ICC*, pages 2205–2210, 2013.

[132] T. Sugawara, D. Suzuki, R. Fujii, S. Tawa, R. Hori, M. Shiozaki, and T. Fujino. Reversing stealthy dopant-level circuits. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 112–126. Springer, 2014.

[133] J. Sztipanovits, T. Bapty, S. Neema, L. Howard, and E. Jackson. Openmeta: A model- and component-based design tool chain for cyber-physical systems. In *From programs to systems. The systems perspective in computing*, pages 235–248. Springer, 2014.

[134] Y. Tang, S. Li, L. Fang, X. Hu, and J. Chen. Golden-chip-free hardware trojan detection through quiescent thermal maps. *IEEE Transactions on Very Large Scale Integration Systems*, 27(12):2872–2883, 2019.

[135] M. Tehranipoor, R. Karri, F. Koushanfar, and M. Potkonjak. Trusthub. *Available on-line: https://www.trust-hub.org*, 2016.

[136] M. Tehranipoor and F. Koushanfar. A survey of hardware trojan taxonomy and detection. *IEEE design & test of computers*, 27(1), 2010.

[137] J. B. Tenenbaum, V. De Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500):2319–2323, 2000.

[138] C. Thèves, P. Biagini, and E. Crubézy. The rediscovery of smallpox. *Clinical Microbiology and Infection*, 20(3):210–218, 2014.

[139] A. Tijsseling. Fluid-structure interaction in liquid-filled pipe systems: a review. *Journal of Fluids and Structures*, 10(2):109–146, 1996.

[140] S. Wang, X. Dong, K. Sun, Q. Cui, D. Li, and C. He. Hardware trojan detection based on elm neural network. In *2016 First IEEE International Conference on Computer Communication and the Internet (ICCCI)*, pages 400–403. IEEE, 2016.

[141] S. Wei and M. Potkonjak. Scalable hardware trojan diagnosis. *IEEE Transactions on very large scale integration (VLSI) systems*, 20(6), 2012.

[142] K. Xiao, D. Forte, Y. Jin, R. Karri, S. Bhunia, and M. Tehranipoor. Hardware trojans: Lessons learned after one decade of research. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 22(1):6, 2016.

[143] K. Xiao, X. Zhang, and M. Tehranipoor. A clock sweeping technique for detecting hardware trojans impacting circuits delay. *IEEE Design & Test*, 30(2):26–34, 2013.

[144] M. Xue, R. Bian, W. Liu, and J. Wang. Defeating untrustworthy testing parties: a novel hybrid clustering ensemble based golden models-free hardware trojan detection method. *IEEE Access*, 7:5124–5140, 2018.

[145] N. Yoshimizu. Hardware trojan detection by symmetry breaking in path delays. In *2014 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, pages 107–111. IEEE, 2014.

[146] R. W. Young. Sabine reverberation equation and sound power calculations. *The Journal of the Acoustical Society of America*, 31(7):912–921, 1959.

[147] H. Zhang. The optimality of naive bayes. *AA*, 1(2):3, 2004.

[148] J. Zhang, G. Su, Y. Liu, L. Wei, F. Yuan, G. Bai, and Q. Xu. On trojan side channel design and identification. In *Proceedings of the 2014 IEEE/ACM International Conference on Computer-Aided Design*, pages 278–285. IEEE Press, 2014.

[149] Y. Zhang, A. Juels, A. Oprea, and M. K. Reiter. Homealone: Co-residency detection in the cloud via side-channel analysis. In *2011 IEEE symposium on security and privacy*, pages 313–328. IEEE, 2011.

[150] Y. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart. Cross-tenant side-channel attacks in paas clouds. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 990–1003. ACM, 2014.

[151] Y. Zhang, D. Parikh, K. Sankaranarayanan, K. Skadron, and M. Stan. Hotleakage: A temperature-aware model of subthreshold and gate leakage for architects. *University of Virginia Dept of Computer Science Tech Report CS-2003*, 5, 2003.

[152] B. Zhao, H. Lu, S. Chen, J. Liu, and D. Wu. Convolutional neural networks for time series classification. *Journal of Systems Engineering and Electronics*, 28(1):162–169, 2017.

[153] Y. Zhao, Z. Nasrullah, and Z. Li. Pyod: A python toolbox for scalable outlier detection. *Journal of Machine Learning Research*, 2019.

[154] B. Zhou, R. Adato, M. Zangeneh, T. Yang, A. Uyar, B. Goldberg, S. Unlu, and A. Joshi. Detecting hardware trojans using backside optical imaging of embedded watermarks. In *Proceedings of the 52nd Annual Design Automation Conference*, page 111. ACM, 2015.

[155] A. M. Zyarah and D. Kudithipudi. Neuromorphic architecture for the hierarchical temporal memory. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 3(1):4–14, 2019.