UNIVERSITY OF CALIFORNIA,
IRVINE

Golden chip-Free Hardware Trojan Detection through Side-Channel Analysis using Machine Learning

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

MASTER OF SCIENCE

in Computer Engineering

by

Rozhin Yasaei

Dissertation Committee:
Associate Professor Mohammad Abdullah Al Faruque, Chair
Professor Fadi Kurdahi
Assistant Professor Zhou Li

2021

DEDICATION

I dedicate this thesis to three beloved people, my family, who have meant and continue to mean so much to me. A special feeling of gratitude to my loving parents, Mino and Sasan whose words of encouragement and push for tenacity ring in my ears. My brother Farbod who has never left my side and is very special to me.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my advisor and committee chair, Professor Mohammad Al Faruque, for his consistent support and valuable insights through out this project.

I would also like to thank my committee members, Professor Fadi Kurdahi and Professor Zhou Li, for assigning their valuable time to review my work.

I like to thank my colleagues in Autonomous and Intelligent Cyber-Physical Systems (AICPS) laboratory, in particular Sina Faezi, senior PhD student, for all the lessons that I learned from him and the research related discussions we had.

I thank IEEE for giving me the permission to use the content of our original paper [6] published in 2021 Design, Automation Test in Europe Conference Exhibition (DATE). Some text of this thesis is a reprint of the material as it appears in DATE'21.

I would like to thank UCI's Department of Electrical Engineering and Computer Science, for giving me the initial fellowship funding which enabled me to thrive in this wonderful path.

Finally, I would also like to thank Office of Naval Research (ONR) for partially funding my work under award number N00014-17-1-2499. Any opinions, findings, and conclusions or recommendations expressed in this thesis are those of the author and do not necessarily reflect views of the funding agency.

VITA

Rozhin Yasaei

EDUCATION

Master of Science in Computer Engineering **2021**
University of California, Irvine *Irvine, California, USA*

Bachelor of Science in Electrical Engineering **2018**
Sharif University of Technology *Tehran, Iran*

RESEARCH EXPERIENCE

Graduate Research Assistant **2018–2021**
University of California, Irvine *Irvine, California*

TEACHING EXPERIENCE

Teaching Assistant **2016-2018**
Sharif University of Technology *Tehran, Iran*

Teaching Assistant and Reader **2018–2021**
University of California, Irvine *Irvine, California*

REFEREED JOURNAL PUBLICATIONS

Brain-Inspired Golden Chip Free Hardware Trojan Detection **2021**
IEEE Transaction on Information Forensics and Security (IEEE TIFS'21)

REFEREED CONFERENCE PUBLICATIONS

HTnet: Transfer Learning for Golden Chip-Free Hardware Trojan Detection **2021**
IEEE/ACM Design Automation and Test in Europe Conference (DATE'21)

DATASET

Hardware Trojan Power and EM Side-Channel Dataset `ieee-dataport.org/3599`
A dataset of power and Electromagnetic side-channel signals collected for hardware Trojan benchmarks

ABSTRACT OF THE DISSERTATION

Golden chip-Free Hardware Trojan Detection through Side-Channel Analysis using Machine Learning

By

Rozhin Yasaei

Master of Science in Computer Engineering

University of California, Irvine, 2021

Associate Professor Mohammad Abdullah Al Faruque, Chair

Design and fabrication outsourcing has made integrated circuits vulnerable to malicious modifications by third parties known as hardware Trojan (HT). Over the last decade, the use of side-channel measurements for detecting the malicious manipulation of the chip has been extensively studied. However, the suggested approaches mostly suffer from two major limitations: reliance on trusted identical chip (e.i. golden chip); untraceable footprints of subtle hardware Trojans, which remain inactive during the testing phase. To overcome these shortcomings, we propose a novel idea of maintaining a dynamic model of the integrated circuit throughout its life cycle to detect HT that might have been injected anywhere in the supply chain. In this thesis, we thoroughly investigate post-silicon HT detection through side-channel analysis using various machine learning models. In this regard, we gather a comprehensive dataset of power and Electromagnetic (EM) side-channel signals for hardware Trojan benchmarks from Trust Hub [20] benchmarks to develop a statistical model of the chip for HT detection. We release our collected power and EM side-channel signals for various HT benchmarks as a public dataset in [21]. Afterward, we explore many machine learning models and various techniques that eventually lead to three approaches for golden chip-free HT detection and HT detection models that outperform the existing methods. Our two recently published papers [7, 6] are also developed based on this dataset, and they provide

further ideas on how to use the dataset to construct an HT detection model.

Chapter 1

Introduction

Given the growing demand for low-cost integrated circuits (IC), 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). In this thesis, we want to construct a data-driven model for an IC that captures the normal behavior of IC and can detect the malicious affect of Trojans on the chip parameters. Initially, we target power model and investigate two top-down and bottom-up approaches to build it.

1.1 Bottom Up Approach

We build the power model of a hardwrae design on FPGA in a bottom-up manner, which is illustrated in Figure 1.1 First, we construct the model of the basic elements in an FPGA, which are, Configurable Logic Blocks (CLBs) and I/O pads. Then, based on the CLB model and IO pad model, we build the routing model and form the final power model. While the SPICE model of I/O pads are usually available on the vendor's website, the model of CLBs

and their building blocks, slices, are lacking. Therefore, we need to build models for slices. To characterize the power profile of the different type of slices, we implement a different design that is mapped to slices of a single type, apply different test patterns, measure the power. Then, we use the measured power traces together with the mapping information (extracted from the .xdl file) to train the power model. To limit the way a design is mapped to the FPGA elements, we identified two alternatives. The first is using Pblock during implementation. The Pblock constrains the resources on FPGA design can use. The second way is to implement the design directly using the .xdl file, which specified how the slices and interconnect are configured. Although it gives more flexibility, the difficulty is much higher due to the lower abstraction level. Thus, we use Pblock to create some small design that fits into a small number of slices and characterize them with different placement and input patterns.

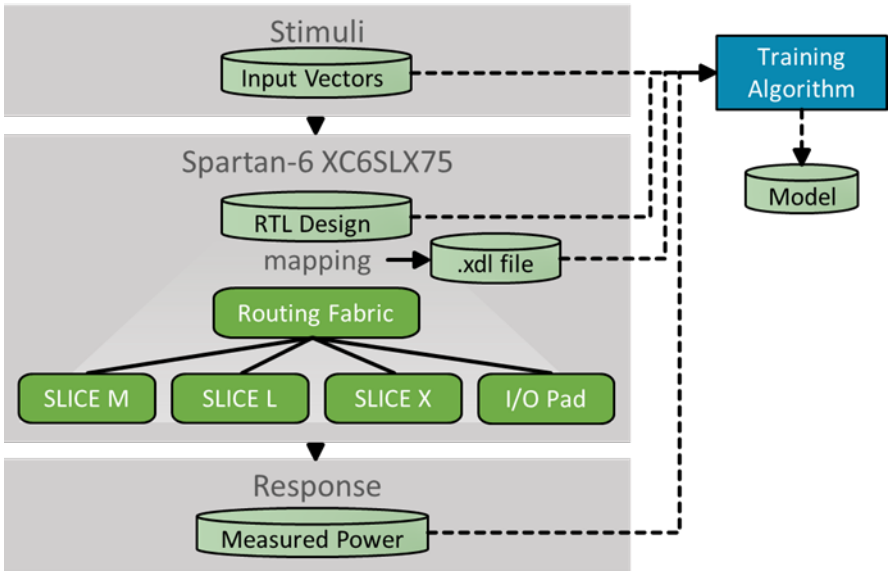

Figure 1.1: The bottom-up data-driven approach of building a power model for IC.

To build up an accurate power model of FPGA in a bottom-up approach, we list all the resources that the FPGA has and measure the power consumption of each resource. Table 1.1 represents the available resources in the main FPGA.

We implement a set of Verilog codes which use only one single resource of the FPGA. This will

Table 1.1: All resources in main FPGA (XC6SLX75)

Components	Number of Components
Logic Cells	74637
SLICEMs	2916
SLICELs	2915
SLICEXs	5831
LUTs	46648
Flip-Flops	93296

guarantee that the power consumption of each resource which we measured is exclusively for one single resource. At the same time, as shown in Figure 1.2, we have changed the location of the resource on the board and collected power data of it automatically. By doing so, we consider the effect of resource location on the power trace and we tried to create an accurate bottom-up power model which covers all the resources of the FPGA.

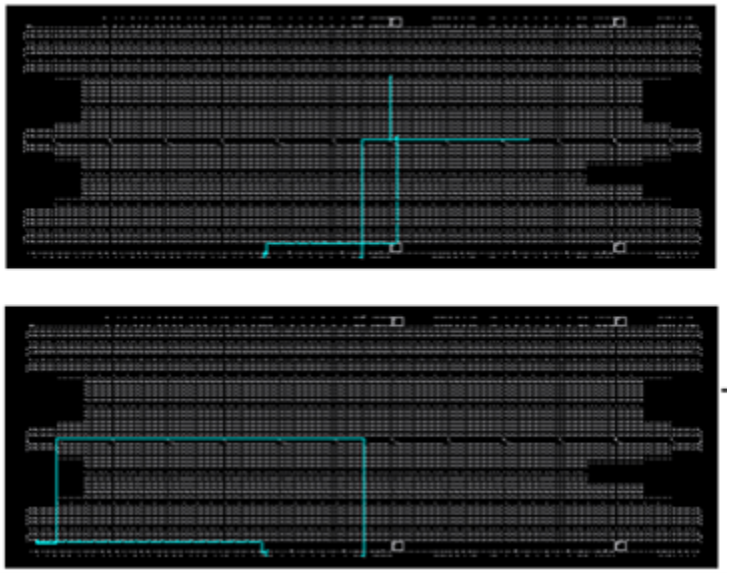

Figure 1.2: Different locations of SLICEM.

Take SLICEMs for example, the function of the Verilog codes is to force one single SLICEM to do an additional calculation. In the meanwhile, for each location of SLICEM, we have used NI USB-6229 to collect 1011712 power data in 5 seconds. Part of the power data is shown in Figure 1.3.

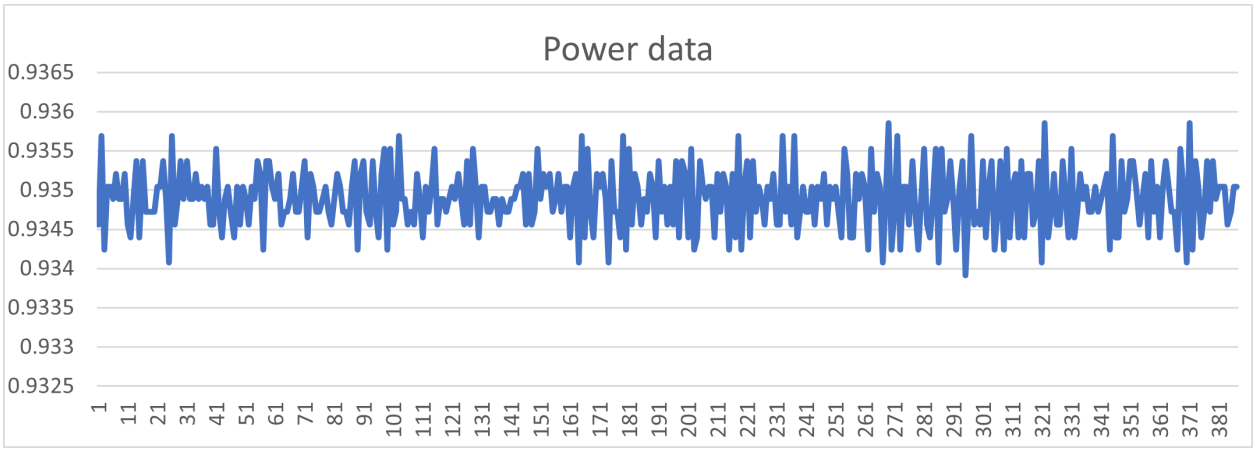

Figure 1.3: Power data of SLICEM.

We have used commercial tools for power estimation, which cannot provide the required accurate power model. The accuracy of the power model is critical as the HTs are designed to be stealthy and very small and their effect on power consumption may fall into error margin. Therefore, another approach for creating the power model is required which lead up to top-down approach.

1.2 Top-Down Approach

In the top-down approach, we change our direction to measure the actual power consumption of the chip as side-channel emission instead of power estimation using the CAD tools. Thus, we develop an automated testbed to measure the power consumption and electromagnetic emission of the chip and gather a dataset of time-series side-channel signals, as discussed in chapter 2. Afterward, we explore various machine learning techniques and construct golden chip-free HT detection models based on the side-channel through three different approaches; feature engineering-based model, feature agnostic end-to-end deep learning model, and self-referencing model based on transfer learning, as elaborated in chapter 3.

Chapter 2

Side-Channel Data Collection

Design and fabrication outsourcing has made integrated circuits vulnerable to malicious modifications by third parties known as hardware Trojan (HT). Over the last decade, the use of side-channel measurements for detecting the malicious manipulation of the chip has been extensively studied. However, the suggested approaches mostly suffer from two major limitations: reliance on trusted identical chip (e.i. golden chip); untraceable footprints of subtle hardware Trojans which remain inactive during the testing phase. To overcome these shortcomings, we propose a novel idea of maintaining a dynamic model of the integrated circuit throughout its life cycle for the purpose of detecting HT that might have been injected anywhere in the supply chain. In this work, we gather a comprehensive dataset of power and Electromagnetic (EM) side-channel signals for hardware Trojan benchmarks from Trust Hub [20] benchmarks to develop a statistical model of the chip for HT detection. Our two recently published papers [7, 6] are based on this dataset that provide further information on how to use the dataset to develop an HT detection model. **We release our collected power and EM side-channel signals for various HT benchmarks as a public dataset in [21]**

2.0.1 Side-Channel Analysis for HT Detection in Literature

Given the growing demand for low-cost integrated circuits (IC), companies tend to use third-party IP cores and outsource the chip fabrication process to the foundries over the globe (Figure 2.1. The globalization of semiconductor industry has raised security concerns about intentional malicious modification of the original design, referred to as *Hardware Trojan* (HT). HT insertion has serious consequences such as leakage of sensitive information, denial of service, change of functionality, or performance degradation. 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 [1]. Due to the intricate nature and potential consequence of HTs, many researchers has been working on the countermeasures through various approaches.

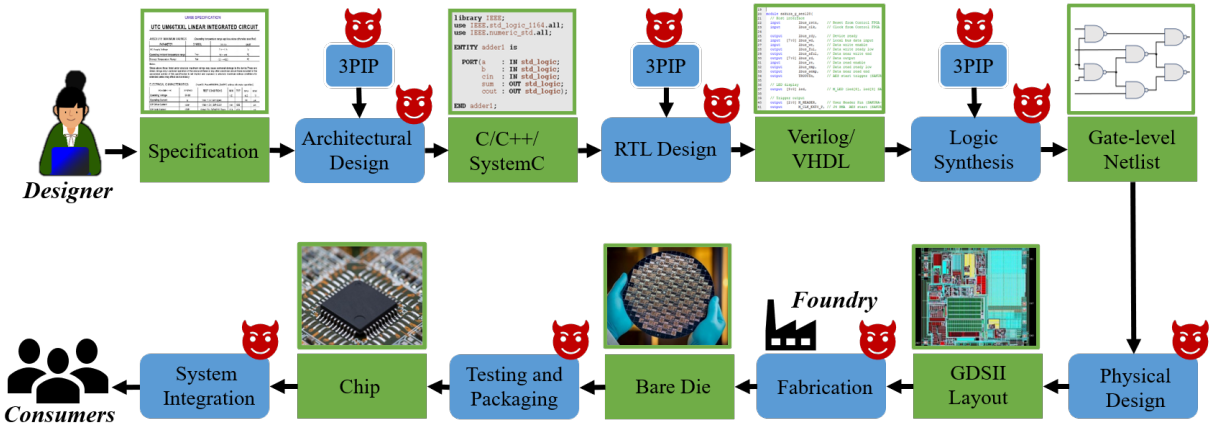

Figure 2.1: IC supply chain which is vulnerable to HT insertion in any stage.

Currently, in the literature, there has been various countermeasures proposed for HT detection, HT insertion prevention, or performing trustworthy computing on untrusted components. The first way to make sure the fabricated chip is genuine is reverse engineering. By delayering the IC, reconstructing the circuits structure and comparing it with the original

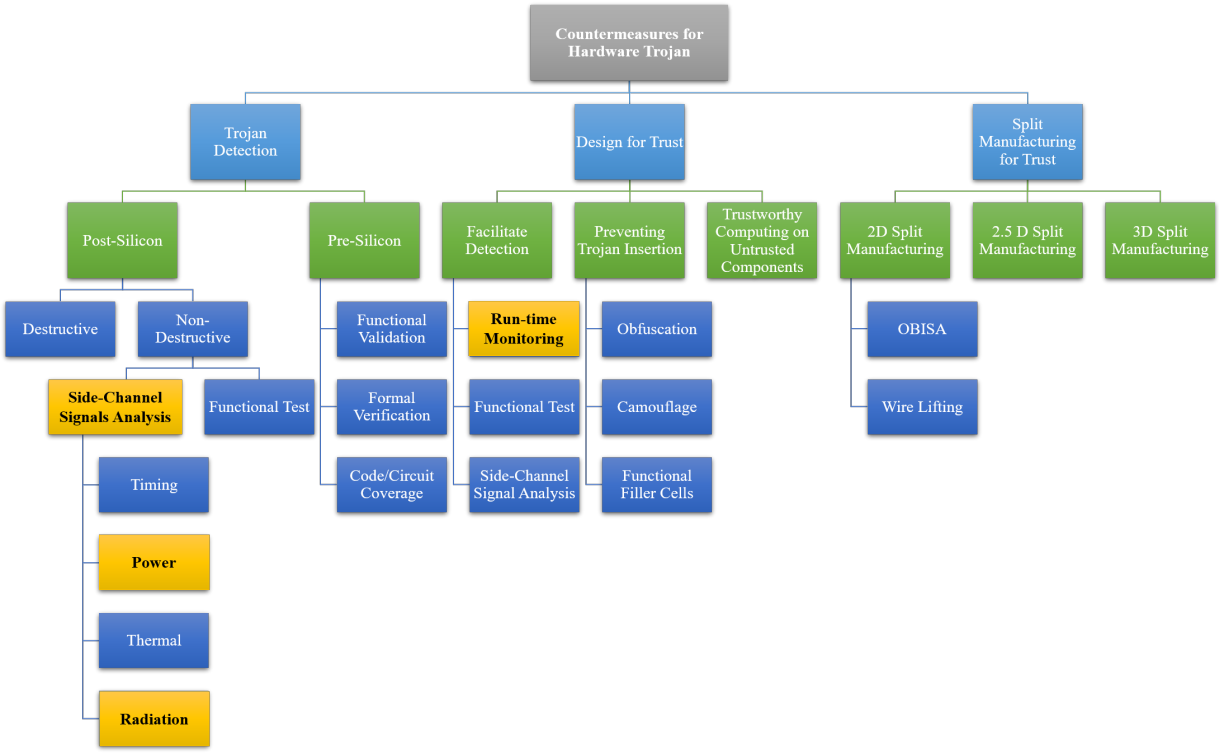

Figure 2.2: Summary of existing defense mechanism for hardware Trojan

design, one can make sure the IC is not modified by the foundry. However, it does not guarantee the detection of HTs inserted during design stage (pre-sillicon) and the cost to reverse engineering an IC is expensive and the chip will not be usable afterwards. To address this issue, lots works in the literature, including our approach, focus on observing side-channel signals for possible anomalous behavior due to existence of the HT in the system through life cycle of the chip.

Second, traditional non-destructive hardware Trojan detection techniques can be mainly divided into two categories, which are enhanced logic testing and side-channel analysis. The first approach tries to activate the HT by giving certain test vectors, capture the response of the device under test (DUT) and then determine if there is a HT. Side-channel analysis approaches control the operation condition of a DUT, measure the side-channel emission and then compare it with the one acquired from a golden chip. The need to a trusted golden chip for comparison is the main limitation of the side-channel analysis methods.

Since HTs can be designed to be dormant during post-manufacturing testing and only triggered by a rare condition during run time, they can easily escape these conventional HT countermeasures. First, during mass production, the usage of automatic test equipment is charged by the time. It would be prohibitively expensive to comprehensively test every condition on every IC in order to activate the HT when using logic testing approaches. Second, by using techniques like power gating [17], the circuit under test will not express any abnormal behavior and therefore cannot be detected using side-channel analysis. A possible solution would be randomly select some ICs from mass production and keep performing these HT detection techniques. However, an adversary in the foundry still can choose to insert the HT only in a small amount of chips therefore it has a little chance of being uncovered. As a result, a real-time monitoring mechanism becomes very important to ensure the integrity of an IC.

In this thesis, we measure the power and EM side-channel signals of a chip to develop a golden chip free HT detection model and data collection experiments are designed in a way to be used for run-time monitoring the chip and detecting the HT when it get triggered.

2.1 Data Collection Parameters

The data collection condition depends on several factors which are elaborated further. The factors and their possible values for each one is as follows:

1. Physical parameter (power consumption or EM radiation)
2. HT benchmarks (The 12 different benchmarks listed in Table 2.1)
3. Trojan condition (disabled, enabled, or triggered)
4. Circuit input vector (fixed value or "next input=current output" method)

5. Chip external temperature (25°C, 35°C, 45°C, 55°C, 65°C, 75°C, and 85°C)

In each experiments, all these factors are set and under these condition, 10000 time-series signal is collected and individually stored in .csv format. As the main circuit under test is AES, the data collection is synchronized with AES encryption cycles. Thus, each time-series data represents the emissions from ship in the time interval between passing the input vector to AES and receiving the encryption output.

2.1.1 Hardware Trojan Benchmarks

Hardware Trojan is a malicious circuit that is inserted intentionally into a target circuit in order to degrade performance, leak sensitive information, deny the service, or change the functionality of chip. The Trojan circuit consists of two major part; payload and trigger. Payload is the part of Trojan that define its functionality and performs the malicious activity. Due to stealthy nature of Trojans, the payload is often inactive to evade the detection during testing and verification stages. The trigger is the optional circuit that monitors various signals or events in the base circuit and activates the payload when a specific signal or event is observed.

In this dataset, we measure the power and EM side-channel signals of HT benchmarks derived from Trust Hub [20]. The list of our HT benchmarks with their payload and trigger types are summarized in the Table 2.1. All the HTs target an encryption core circuit, AES 128bits. The AES circuits receives a 128 bits input value (plain text), encrypts it using a secret key and generates a 128 bits output (cipher text). The key is the sensitive information in this design which the security of encryption depends on keeping the key secret.

Table 2.1: Summary of HT benchmarks.

Benchmark	Trigger Condition	Payload
AES-T400	A predefined number as input	Leaks the key through RF signal
AES-T500	A predefined sequence of numbers as input	Drains the battery with a continuously rotating shift register
AES-T600	A predefined number as input	Leaks the key by increasing the leakage current for each 0 bit in key
AES-T700	A predefined number as input	Leaks the key as CDMA sequence code through a covert channel
AES-T800	A predefined sequence of numbers as input	Leaks the key as CDMA sequence code through a covert channel
AES-T1000	A predefined number as input	Leaks the key as CDMA sequence code through a covert channel
AES-T1100	A predefined sequence of numbers as input	Leaks the key as CDMA sequence code through a covert channel
AES-T1300	A predefined number as input	Leaks the key by increasing the dynamic power consumption
AES-T1400	A predefined sequence of numbers as input	Leaks the key by increasing the dynamic power consumption
AES-T1600	A predefined sequence of numbers as input	Leak the key through RF signal
AES-T1800	A predefined number as input	Drains the battery with a continuously rotating shift register
AES-T2000	A predefined sequence of inputs	Leaks the key by increasing the leakage current for each 0 bit in key

2.1.2 Trojan condition

An inactive HT does not alter the chip's functionality and hardly affect the side-channel profile of chip. As a consequence, its detection is very challenging in the testing stage due to the low probability of activating it with the limited numbers of test vectors. Therefore, we push the detection process to the run-time when it is possible to detect the Trojan as soon as it gets triggered and injects anomalies to the side-channel measurements of chip as depicted in Figure 2.3. An example of power side-channel for AES-T500 HT benchmark in the cases of inactive and active Trojans is depicted in Figure 2.4 which illustrates the anomalous HT activation effect on side-channel signals.

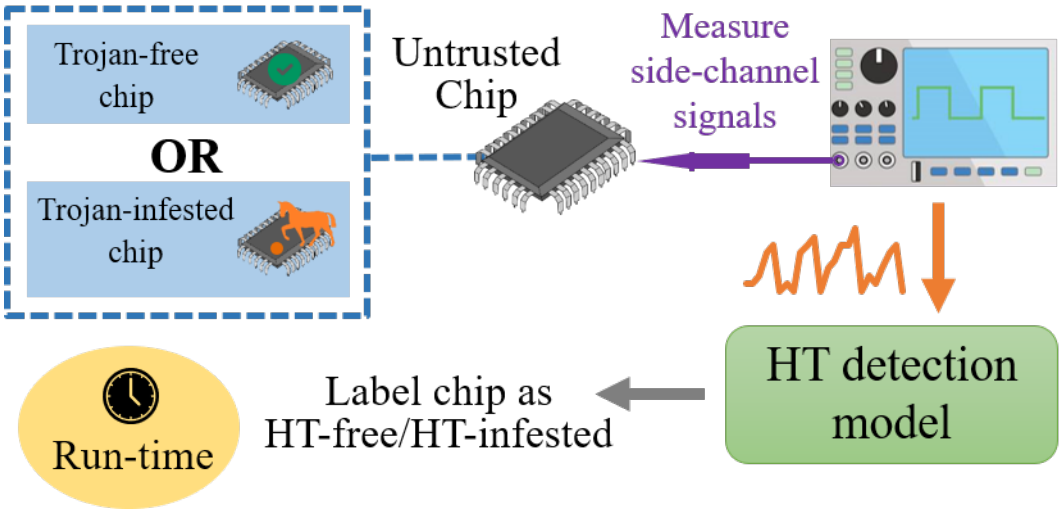

Figure 2.3: The proposed pipeline to detect HT when it gets triggered in run-time.

In order to develop this run-time monitoring model to detect Trojan, we gather a comprehensive dataset of power and EM radiation side-channel signals for hardware Trojan benchmarks. We measure the side-channel signals in three conditions:

- Trojan Disabled: The DUT includes both the main circuit and Trojan, but the trigger part of HT is disabled and it cannot be triggered. Thus, the collected side-channel signals includes the emission from both main circuit and the inactive HT.

- Trojan Enabled: The DUT includes both the main circuit and Trojan and the trigger part of HT is enabled and it can be triggered. Thus, the collected side-channel signals includes the emission from both main circuit and the HT which can be active or inactive.

- Trojan Triggered: The DUT includes both the main circuit and Trojan and the trigger part of HT is not only enabled but also it is activated by applying its trigger condition. Thus, the collected side-channel signals includes the emission from both main circuit and the active HT.

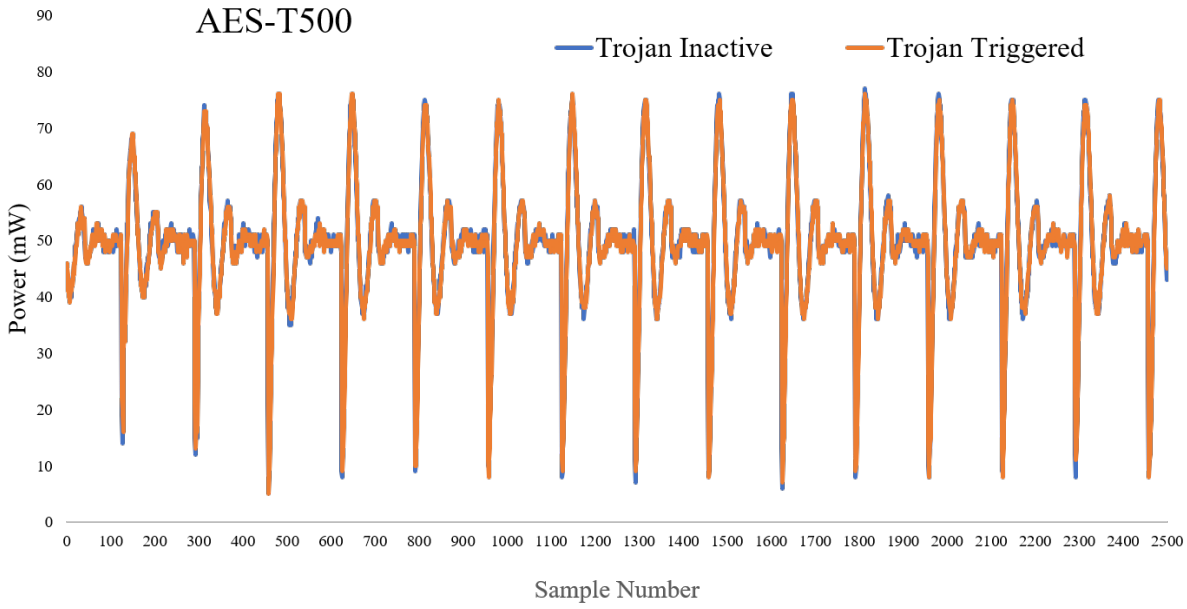

Figure 2.4: A example of the power side-channel time-series signal for AES-T500 benchmark.

2.1.3 Input Vector

The main circuit in our experiment is the AES core that performs encryption on a 128 bits input vector. The input vector affect the switching in the circuit and alter the side-channel

measurements. In each data collection experiment, the encryption process is repeated for 10000 times which leads to generating 10000 time-series signal. Therefore, we send 10000 input vectors to AES iteratively for each experiment which are generated in two way. The first method is to always send a fixed value as input to AES in which the measurement are not influenced by variation in switching activity. The other way is to send various sequences to the AES which infuses some variation to the measurements. In order to generate different input vector, we use the "next input=current output" method in which set the initial input vector as zero and the input for the next encryption cycle is the output of current cycle. Thus, instead of a fixed input vector, we generate a fixed sequence of varied input vectors. The directory of data samples collected using the "next input=current output" method have "$_1$" suffix and for fixed value method, it has "$_2$" suffix.

2.1.4 Chip External Temperature

Another variable in this data collection is the temperature because it is high affect the power and EM emission of the chip. In order to analyze the temperature effect which mimics the circuit againg phenomena as well, our experimental setup embeds a heater, temperature sensor, and temperature controller and we measure the signals under 7 different temperatures including 25°C, 35°C, 45°C, 55°C, 65°C, 75°C, and 85°C.

2.2 Automated Experimental Setup

In this section, we describe our automated testbeds for power and EM side-channel data collection.

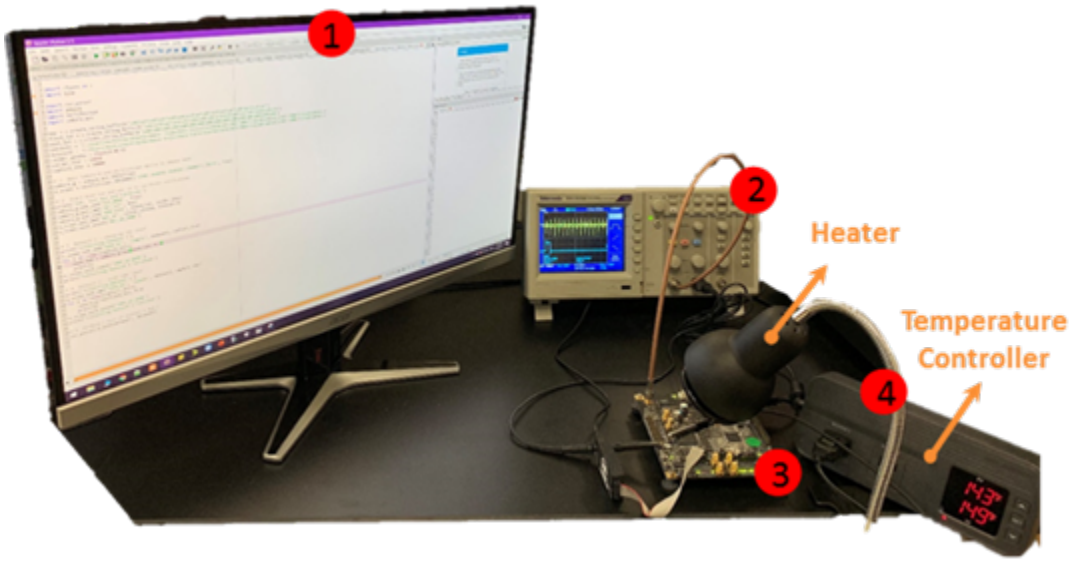

Figure 2.5: Our experimental setup for power side-channel data collection.

2.2.1 Automated Power Side-Channel data collection

To speed up the data collection process, we have implemented an automated FPGA testbed. Since variables of the data collection are numerous, it is impossible for people to repeatedly program different design, apply corresponding test vectors, tune the oscilloscope to have steady waveform-of-interest displayed, transfer the measured power data to PC and post-process the data. The automated testbed, as shown in Figure 2.6 and Figure 2.5, comprises four major components:

1. Test control program on computer
2. Oscilloscope
3. SAKURA-G FPGA board
4. Temperature Controller

The test control program runs on a desktop computer and consists of three modules and a job queue. The job queue is a series of test to be conducted, which includes the design-under-test

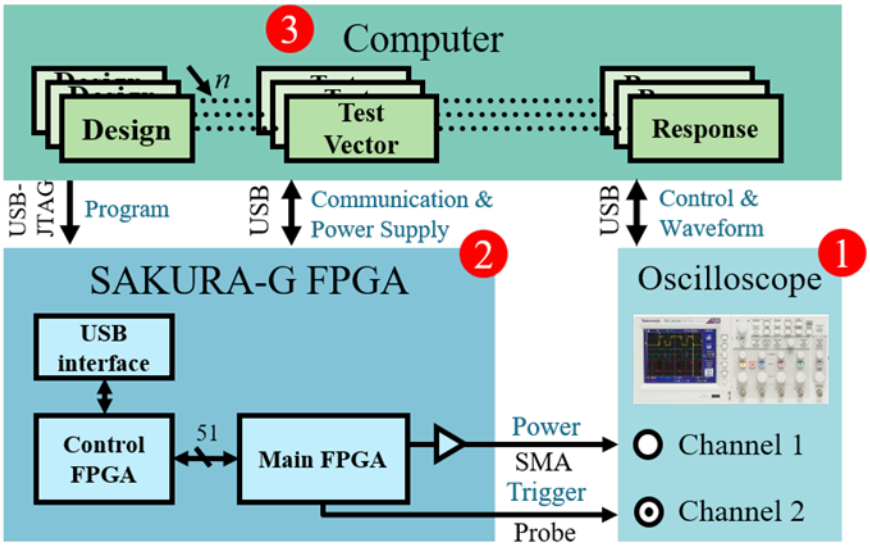

Figure 2.6: The automated power analysis testbed architecture.

(DUT) in bitstream format, a test configuration, and the name of the response waveform file, which will later be captured and saved. The test control program dispatches a test job from the job queue and uses the three modules to complete the test. First, it uses the programmer module to program the design bitstream into the main FPGA (XC6SLX75) via a USB-JTAG cable. At the same time, it uses the test configurator to transfer the description file to the secondary FPGA (XC6SLX45), which serves as the test pattern generator (TPG), which is shown in Figure 2.7. The test description file specifies the connection of the two FPGA and the test vector the TPG will generate. After the test start, the TPG on the secondary FPGA will send a start signal, which will be used by the oscilloscope as a trigger to capture the waveform. The oscilloscope controller uses the test duration in the test description file to adjust the oscilloscope to have steady waveform displayed, and then save the waveform to the controlling PC. The temperature controller consists of a temperature sensor which is attached to the chip, a heater, and a controller to keep the temperature in the desired range.

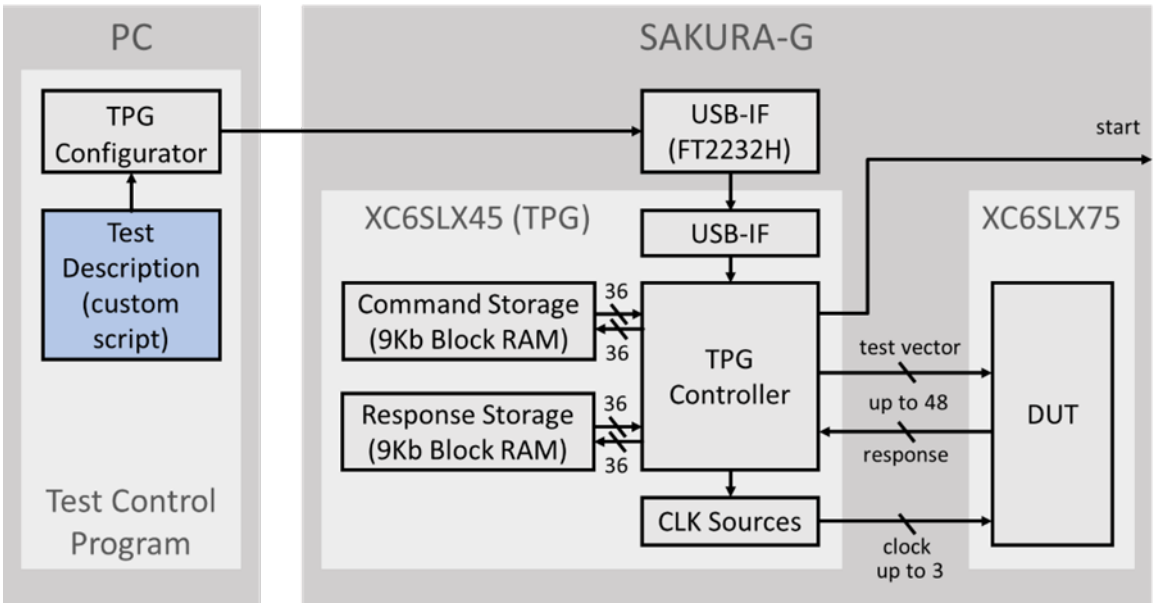

Figure 2.7: Test Program Generator (TPG) architecture.

2.2.2 Automated EM Side-Channel data collection

Similar to automated power analysis testbed, we create an automated testbed to measure the Electromagnetic side-channel of the circuit. In this testbed, the test program generator architecture is the same as the power testbed, but the data collection equipment differs. The automated testbed, as shown in Figure 2.8, comprises five major components:

1. Oscilloscope
2. Spectrum Analyzer
3. SAKURA-G FPGA board
4. EM probe
5. Test control program on the computer
6. Temperature controller

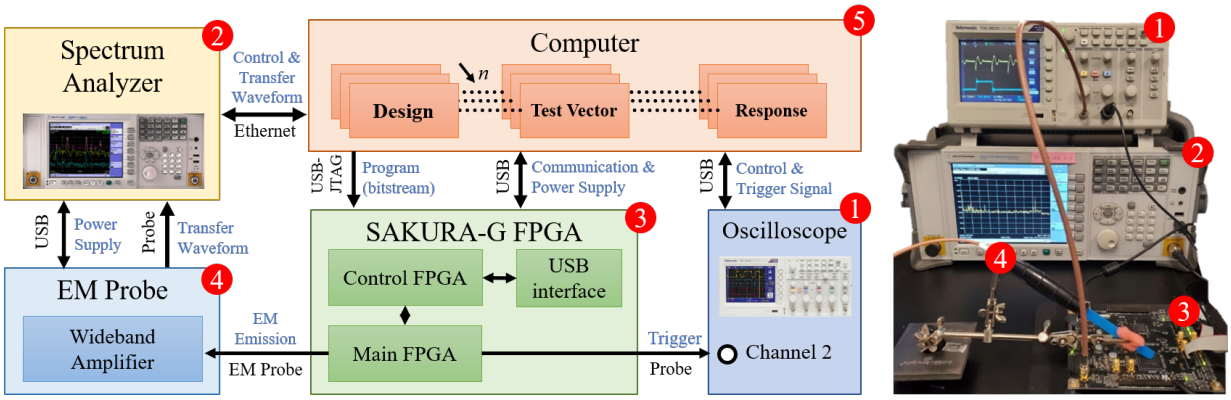

Figure 2.8: Our experimental setup for EM side-channel data collection.

The desktop computer is connected to the Sakura-G FPGA board through JATG cable for programming, as well as a USB cable for communication and supplying the power of the board. The test program the FPGA and implement the target circuit design automatically. Then, it starts the data collection process by sending the input waveforms to the circuit input and receiving the output. The target circuit for our current benchmarks is the AES circuit, which outputs the encrypted version of the input signal. When on encrypted starts, the oscilloscope captures a trigger signal which is used for synchronization purposes. The computer is connected to the oscilloscope through a USB cable. When the oscilloscope captures the trigger signal, it means the encryption is started, and it transmits it to the computer program to start the EM data collection. EM emissions of FPGA are measured by an EM probe, which includes an amplifier. It sends the amplified signal to a spectrum analyzer for further analysis. The spectrum analyzer is connected to the computer through an Ethernet cable, and it starts the data collection when the encryption process starts, and the computer receives the trigger signal. The spectrum analyzer captures the EM signal, saves the waveform in the frequency domain, and sends it to the computer.

Chapter 3

Machine Learning for Hardware Trojan Detection

This chapter discusses the machine learning methodologies for HT detection based on the side-channel dataset that we gathered using our automated testbed, mentioned in chapter 2. The ultimate goal of the model is distinguishing between the normal behavior of the circuit versus the scenario where an HT is triggered in the system. Hence, we craft statistical models to classify EM and power consumption traces and label the circuit as HT-free or HT-triggered. To evade the need for a trusted golden chip for HT detection, we propose defense mechanisms in the run-time that can alert Trojan's presence when it gets activated. In the following sections, we first review the existing HT detection methods and current research challenges. Then, we discuss three golden chip-free HT detection approaches based on machine learning, which are trained and tested on our power and EM side-channel, to pinpoint the HT activation in the run-time.

3.1 HT detection methods in the literature

Given the growing demand for low-cost IC, companies tend to use third-party IP cores and outsource the chip fabrication process to the foundries over the globe. This trend has raised security concerns about 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 [1].

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 [19, 4, 5], 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 method-

ologies 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 [15]. A few works in the literature have tried to resolve this issue by using self-referencing techniques [10], or using accurate trusted simulations combined with embedded sensors in the chip to analyze the side-channel emissions [16]. 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[11]. Despite the efforts for triggering the HT during the testing of the chip [11], the HT may remain inactive. Therefore, complimentary run-time monitoring and validation mechanisms such as [2, 12, 8, 14] 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 [13, 14]. Variety of machine learning models have been used for HT detection and as the Figure 3.1 demonstrates, the majority of them are supervised model which perform classification and require two set of labeled data.

3.1.1 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.

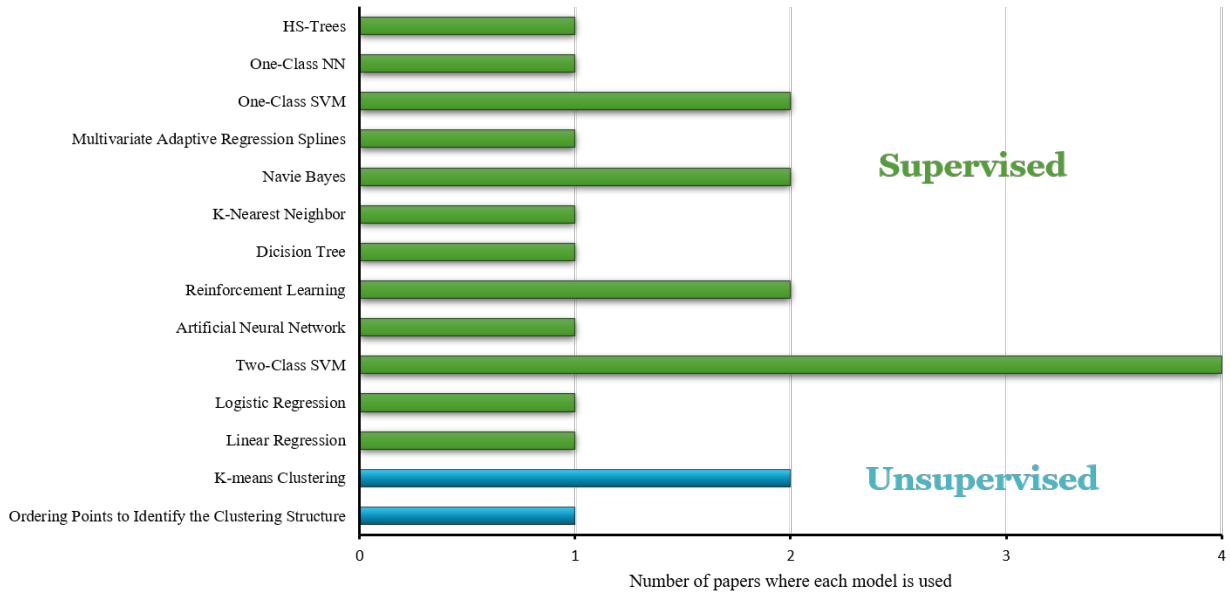

Figure 3.1: A survey of machine learning models used for HT detection.

- Detection of HTs often requires activating them while the probability of triggering an HT during testing is very low due to their stealthy nature.
- Process variation in the IC production supply chain will result in the HT detection mechanisms' failure if proper measures are not taken.
- 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 Feature Engineering Approach

The majority of the solutions proposed in the literature for HT detection through side-channels build a statistical model based on the specific features that the researchers had found in the side-channel data under examination. Following this trend, in our first attempt, we create the pipeline represented in Figure 3.2 for HT detection. This figure shows the three stages of hardware design, fabrication, and Run-time and how the model fits into different

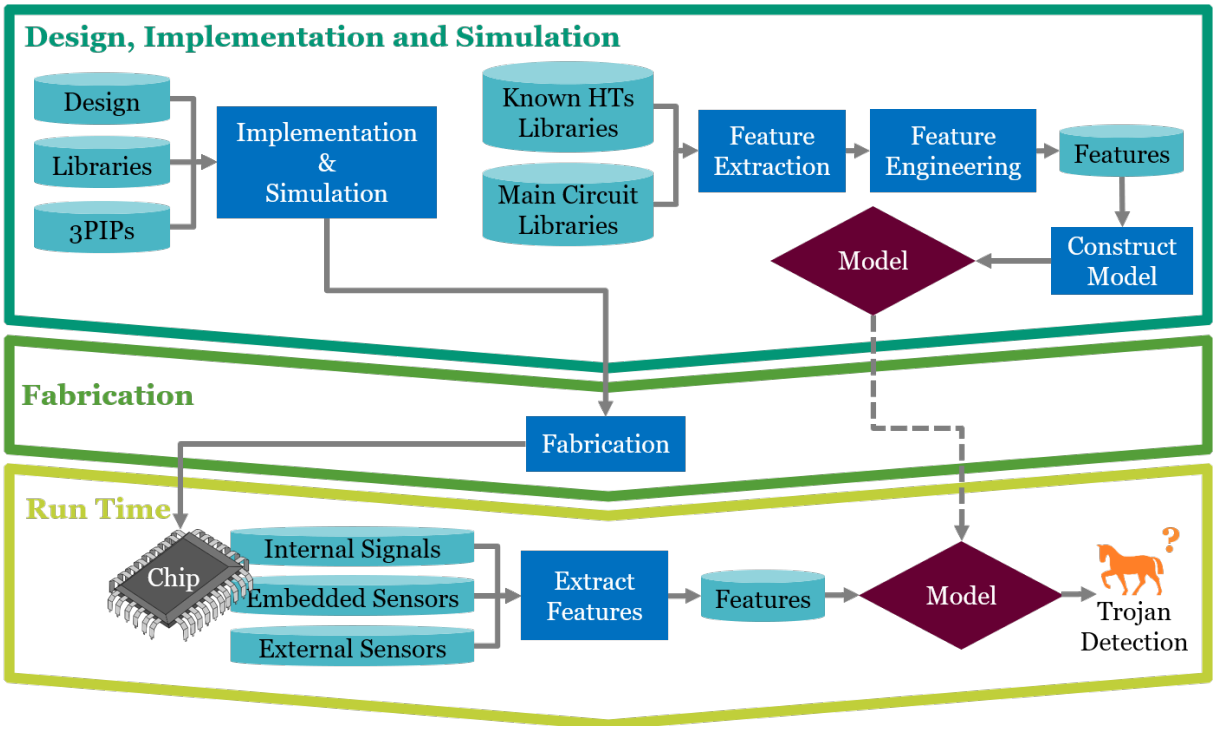

Figure 3.2: HT detection approach based on feature engineering.

stages. In the design and implementation stage, this pipeline consists of three major modules. The raw side-channel signal from libraries of known Trojans and circuits is fed to the feature extraction module. This module converts the input signal to a set of predetermined features. Next, the feature Engineering module reduces the number of features extracted and selects the most effective ones. A model is developed based on the initial dataset's selected features. It is passed to the run-time stage to detect HT from the extracted features from the fabricated chip side-channel signals. To evaluate this approach, we develop various classifier models and train and test them on out power consumption side-channel dataset. We elaborate each module in the pipeline and the evaluation results below.

3.2.1 Feature Extraction

We implement all the feature extractors that have returned useful discriminative features in the literature for time series data and concatenated those features. As it is shown in

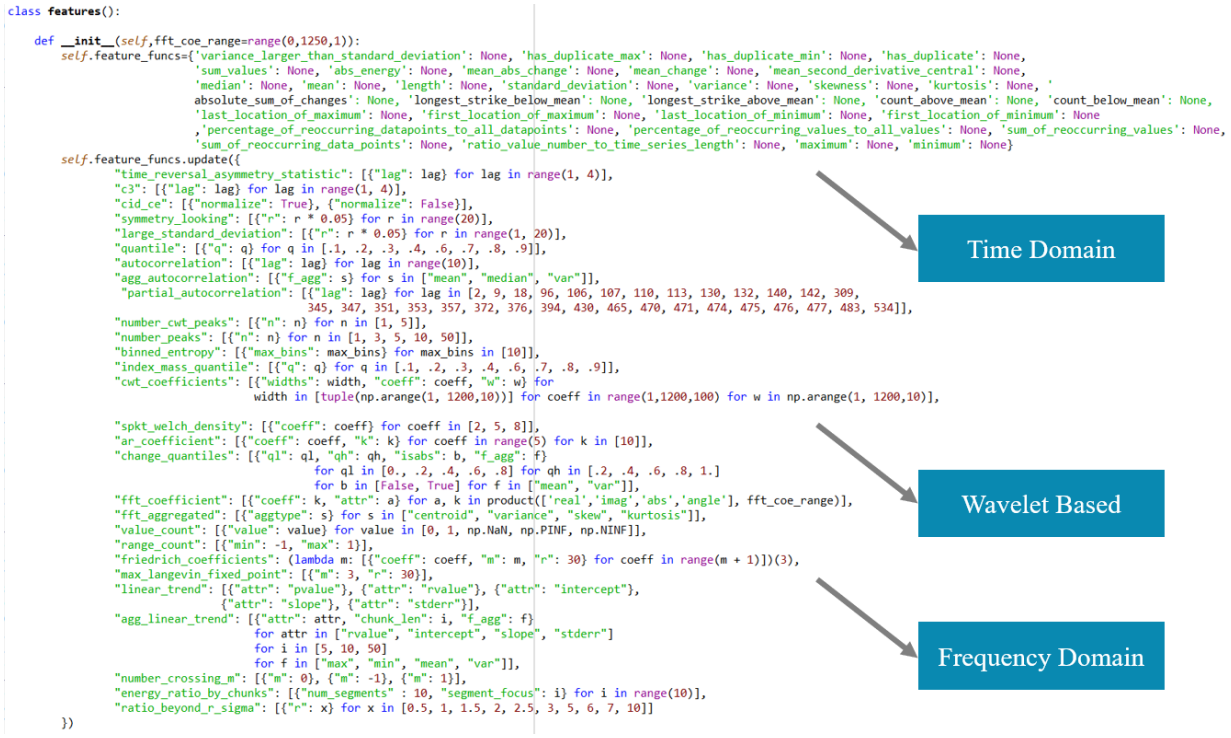

Figure 3.3: Snapshot of the feature extraction module.

Figure 3.3, we extract various types of time domain features, use coefficients of Fast Fourier Transform (FFT) as frequency domain features, and calculate Continuous Wavelet Transform (CWT) coefficients for different windows sizes as wavelet-based features.

3.2.2 Feature Engineering

Given that our oscilloscope stores 2500 points in the signal at a time, the total number features extracted in the feature extractor module adds up to around 7000 features. Since the number of features is enormous, in the training phase, the classifiers might not converge to the optimal statistical model representing the input-output relationship. Hence, decreasing the number of features can simplify the model behavior. Here, we chose to use the method described in [3] for features dimensionality reduction.

Table 3.1: HT detection accuracy using feature extraction before feature selection.

Classifier	AdaBoost	L-SVM	NB	KNN	NN	QDA	RF	VC
AES-T400	71.88%	66.46%	74.00%	79.38%	81.29%	67.54%	70.12%	80.79%
AES-T500	99.71%	99.67%	99.71%	88.38%	99.71%	99.71%	99.00%	99.71%
AES-T600	80.50%	67.38%	73.21%	79.42%	82.67%	57.00%	71.75%	82.17%
AES-T700	100%	100%	100%	98.00%	100%	100%	100%	100%
AES-T800	100%	100%	100%	98.71%	100%	100%	100%	100%
AES-T1000	71.75%	68.00%	72.13%	77.54%	82.08%	68.92%	69.25%	82.33%
AES-T1100	82.25%	67.83%	73.71%	78.46%	82.08%	59.21%	69.29%	76.58%
AES-T1300	90.92%	67.46%	98.33%	79.00%	95.75%	59.79%	97.21%	98.33%
AES-T1400	76.13%	67.38%	73.92%	78.42%	81.38%	59.96%	67.04%	80.58%
AES-T1600	73.79%	66.29%	74.21%	78.21%	81.71%	73.54%	70.08%	81.63%
AES-T1800	73.37%	65.96%	73.79%	78.96%	81.96%	63.38%	72.04%	82.08%
AES-T2000	72.29%	65.75%	72.67%	77.37%	81.33%	68.83%	72.25%	82.25%
Average Accuracy	**82.72%**	**75.18%**	**82.14%**	**82.65%**	**87.50%**	**73.16%**	**79.84%**	**87.20%**

Table 3.2: HT detection accuracy using feature extraction after feature selection.

Classifier	AdaBoost	L-SVM	NB	KNN	NN	QDA	RF	VC
Average Accuracy	86.44%	85.28%	82.01%	82.48%	87.43%	81.06%	79.80%	87.29%

3.2.3 Classifier Models

We have implemented eight different classifiers that are widely used in the literature for the classification task and HT detection in particular. Here is the list of classifier we implement and test:

- Adaptive Boosting (AdaBoost)
- Linear Support Vector Machine (L-SVM)
- Naive Bayes (NB)
- K Nearest Neighbor (KNN)
- Neural Network (NN)
- Quadratic Discriminant Analysis (QDA)
- Random Forest (RF)
- Voting Classifier (VC)

3.2.4 Evaluation

We have implemented eight different classifiers that are widely used in the literature for the classification task and HT detection in particular. Table 3.1 represents the accuracy of each classifier for detecting the given HT benchmark before feature selection while Table 3.2 shows the average accuracy of benchmarks for the different classifiers. Comparing the results indicated that the neural network classifier has the best performance. It also demonstrates that the classifier's accuracy is maintained or even improved after the feature selection process. It means that we can utilize the feature selection to reduce the model overhead without deterring its performance.

3.3 Feature Agnostic Deep Learning Approach

Although the pipeline described in the previous section seems to return acceptable prediction accuracy in most of the benchmarks, we believe that it cannot be used as a live representation of the circuit power consumption behavior because of the following reasons:

1. Once the model is designed and trained for a given benchmark, it will not be flexible to add new feature extractors to the feature extraction module. Also, if the feature selection module determines to eliminate some of the features, retrieving them would mean a completely different implementation and retraining of the classifiers.

2. We want to keep the model alive along with the main circuit and update it to address the process variation and temperature effect on the distortion of the side-channel profile on the chip.

3. Using a model in run-time and keeping it alive requires an efficient and high-performance hardware implementation of the model. Hardware implementation of the model will be significantly less complicated and efficient if the operations used are repetitive. However, the algorithms' architecture extracting the features is usually completely different from the classifiers. For instance, if we decide to use frequency-domain features (FFT) and random forest as a classifier, we will have numerous multiplication operations on one side of the pipeline. In contrast, the classifier part would be mostly if-else statements.

4. Based on our experience, choosing features for each circuit is very time-consuming, and often, the knowledge learned for one circuit cannot be easily transferred to the next. Hence, if we decide to use the general feature extraction-based approach described in the previous section, huge engineering effort will be required for each given new hardware circuit.

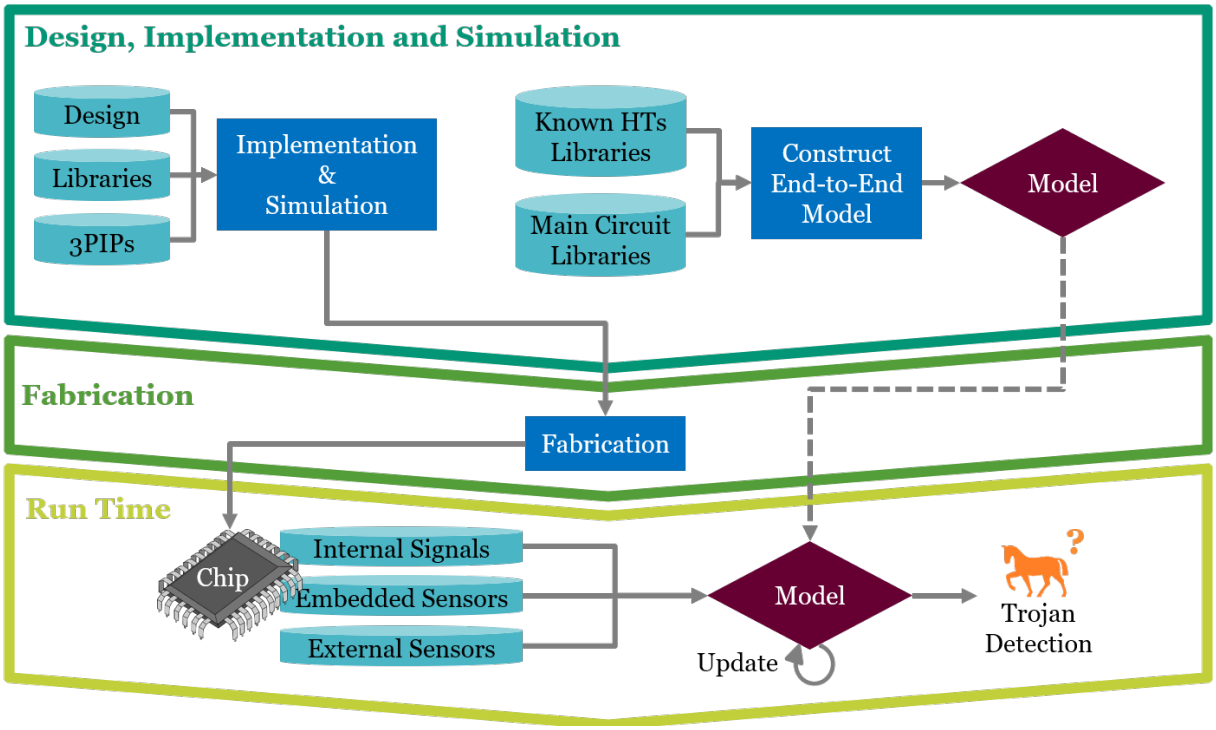

Figure 3.4: HT detection approach based on end-to-end deep learning model.

To address these issues, we explore a deep learning approach by combining the feature extraction module with the classifier to develop an end-to-end model. The overview of this new approach is depicted in Figure 3.4. In this approach, the feature extraction and selection modules are removed and included inside our customized deep learning model. Thus, the model automatically extract and select features and this dynamic feature extraction make it possible to update the model in run-time and keep it alive.

3.3.1 Our Customized CNN Architecture

We develop a customized Convolutional Neural network (CNN) for golden chip-free HT detection in run-time. As shown in Figure 3.5, this model has two convolutional layers which have a dynamic feature extraction behavior. We embed an attention mechanism into it, which works similarly to a feature selection algorithm. The attention mechanism essentially determines the importance of half of the features based on the other half of the

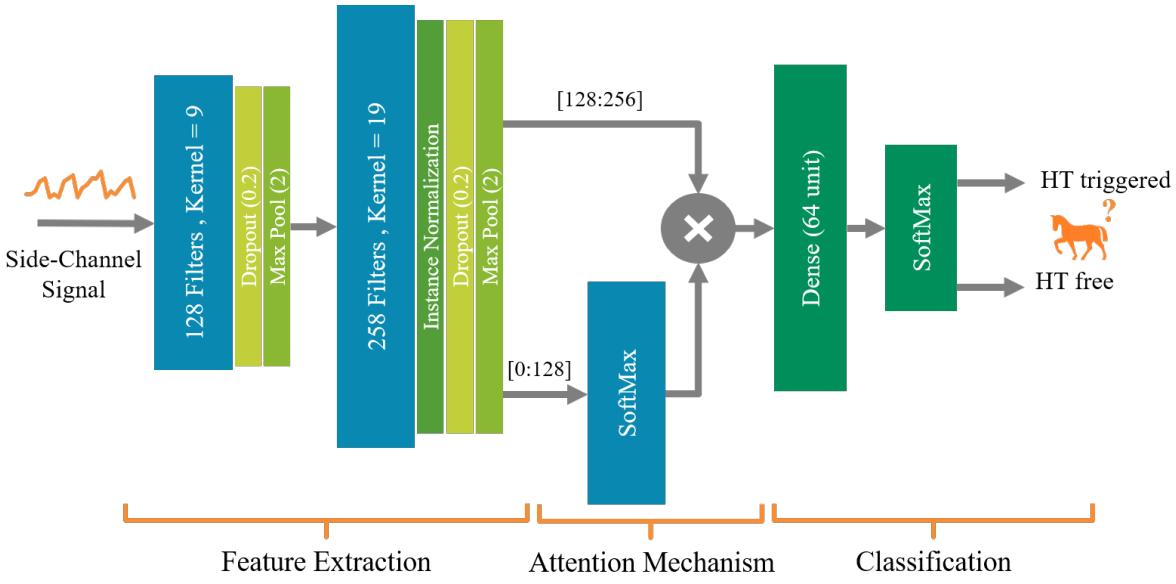

Figure 3.5: Our customized CNN architecture.

features, which results in dynamic feature selection behavior. The last part of our customized neural network architecture is a fully connected dense layer followed by two softmax nodes for the classification. It is an end-to-end model, which means that while training, the optimizer update the weights in all the layers to minimize prediction error. The model suggested in this section eliminates the need for hand-crafted engineered features for each circuit. Hence we call this approach a feature agnostic model.

3.3.2 Evaluation

We compare our customized convolutional neural network's performance with the existing deep learning models in the literature in terms of accuracy. For this evaluation, we implement various models and train and test them on our power consumption side-channel dataset for different HT benchmarks. The machine learning models implemented are as follows:

- Our customized convolutional neural network
- Encoder (En)

Table 3.3: HT detection accuracy using the end-to-end classifiers.

Classifier	Ours	En	FCN	MCD	MLP	ResNet	TWIESN	CNN
AES-T400	**89.77%**	89.23%	59.85%	83.47%	84.22%	71.15%	56.73%	80.93%
AES-T500	**98.00%**	98.13%	98.03%	98.33%	98.05%	98.10%	95.20%	98.58%
AES-T600	**93.98%**	94.18%	68.48%	92.46%	93.20%	87.10%	56.98%	90.93%
AES-T700	**100%**	100%	100%	100%	100%	100%	99.93%	100%
AES-T800	**100%**	100%	100%	100%	100%	100%	99.93%	100%
AES-T1000	**71.35%**	70.85%	54.13%	66.93%	66.13%	54.68%	54.40%	62.65%
AES-T1100	**77.61%**	78.25%	53.63%	73.50%	75.30%	53.93%	58.40%	72.13%
AES-T1300	**100%**	100%	98.90%	99.98%	100%	97.55%	60.80%	99.98%
AES-T1400	**85.39%**	84.60%	61.25%	80.36%	83.23%	68.45%	56.33%	78.40%
AES-T1600	**75.90%**	74.20%	57.40%	69.19%	71.53%	60.28%	55.88%	67.58%
AES-T1800	**93.45%**	93.13%	61.68%	90.63%	91.63%	69.05%	55.98%	86.75%
AES-T2000	**92.98%**	93.00%	57.45%	84.34%	90.13%	73.30%	50.43%	87.48%
Average Accuracy	**89.87%**	89.63%	72.56%	86.60%	87.78%	77.80%	66.75%	85.45%

- Fully Convolutional Network (FCN)
- Minimum Covariance Determinant (MCD)
- Multi-Layer Perceptron (MLP)
- Residual neural network (ResNet)
- Time Warping Invariant Echo State Network (TWIESN)
- Convolutional Neural Network (CNN)
- Best Feature-Based (BFB)

The evaluation results in Table 3.3 demonstrates that our customized network outperforms all the current deep learning methods as well as the best feature approach.

3.4 Transfer Learning Approach

Any model, which performs classification between an HT-infected circuit and its healthy version will need samples from each one of those datasets for training. However, in reality, access to HT free circuit, or in other terms, the golden chip may not be possible due to fabrication cost.

In order to avoid the need for a golden chip, the concept of transfer learning is explored. The overview of this approach is as illustrated in Figure 3.6. We construct a base model from the side-channel dataset of HT benchmarks, and this base model learns fundamental knowledge about the effect of Trojan activation on side-channel parameters. Despite the previous approaches, the base model is retrained in the chip testing phase before being used for HT detection in the run-time. This retraining process makes the model capture the normal side-channel profile of the chip under test used along with the base knowledge of

Trojans to recognize the anomalous disruption caused by HT activation in run-time. We call this approach transfer learning because a base knowledge is derived from an available dataset for known circuits, and the knowledge is transferred for a new circuit for which we do not have both classes of labeled data.

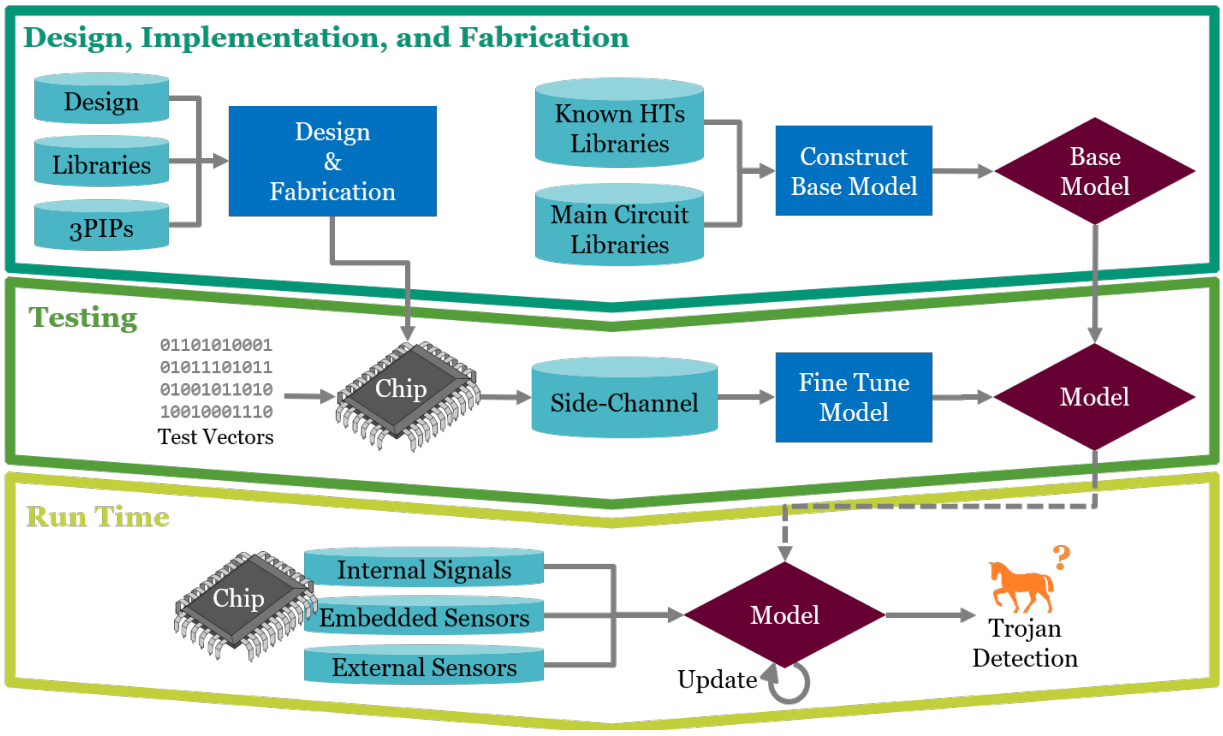

Figure 3.6: Transfer learning HT detection approach overview.

3.4.1 Training Process

In this setting, we create an initial training dataset that includes both HT-inactive and HT-active side-channel traces based on known hardware Trojans. Inspired by [18], we first train the model to learn our previous knowledge about the existing HTs. Although the initial training is not directly used, it will help the feature extraction and classification parts of our suggested model to know what type of features they should look for when looking at similar power traces. As it is presented in Figure 3.7, the initial model is trained. We take this model and retrain, or in better terms, fine-tune it for the new IC which we have fabricated.

We assume that in the functional testing phase, the HT does not get triggered. Hence, the IC provides trusted power traces. (If it does not, we assume that other simpler HT-detection mechanisms will be able to capture it). We use these traces to fine-tune the model by setting the learning rate very low.

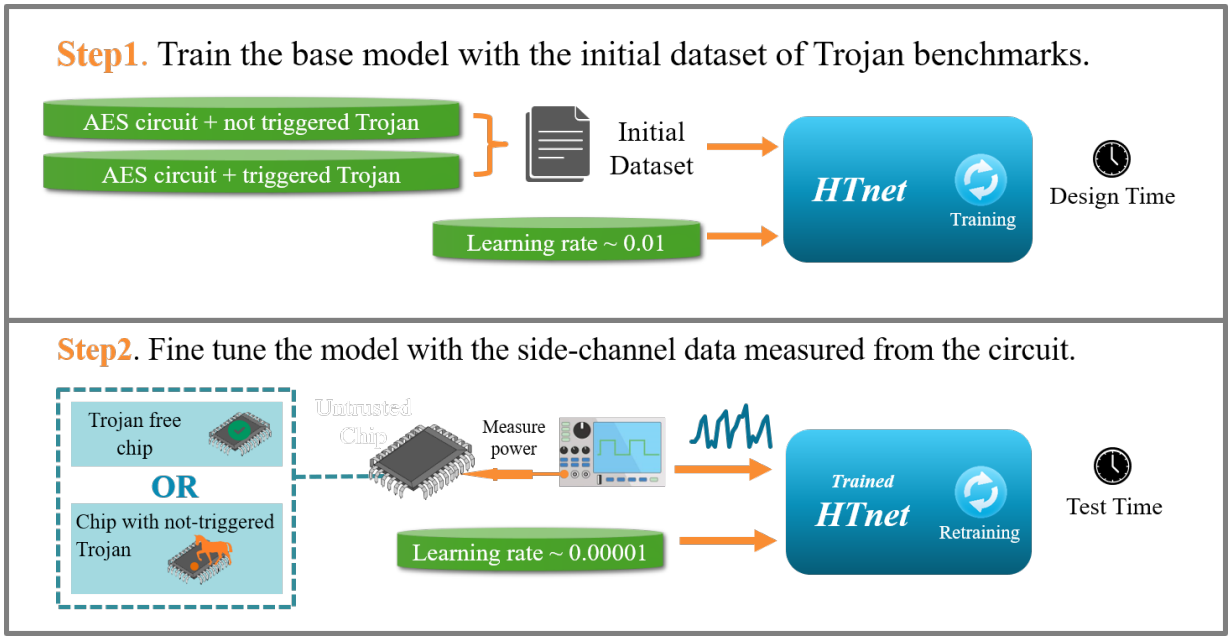

Figure 3.7: Two steps of training and fine tuning the model based on transfer learning.

3.4.2 HTnet Model

The transfer learning approach with our side-channel dataset is deeply investigated in our paper [6] and as a result, a golden chip free HT detection model called HTnet is generated which leverages the concept of self-referencing and transfer learning to eliminate the need for a golden chip.

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 3.8 represents HTnet that is the result of this exhaustive search with fine-tuned

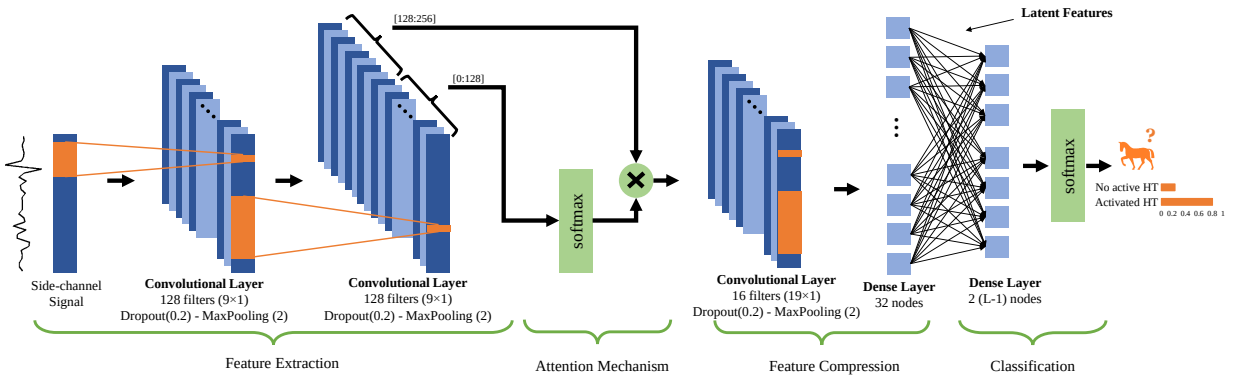

Figure 3.8: HTnet architecture used as side-channel signals classifier.

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, an attention mechanism is implemented 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 paper, 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 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

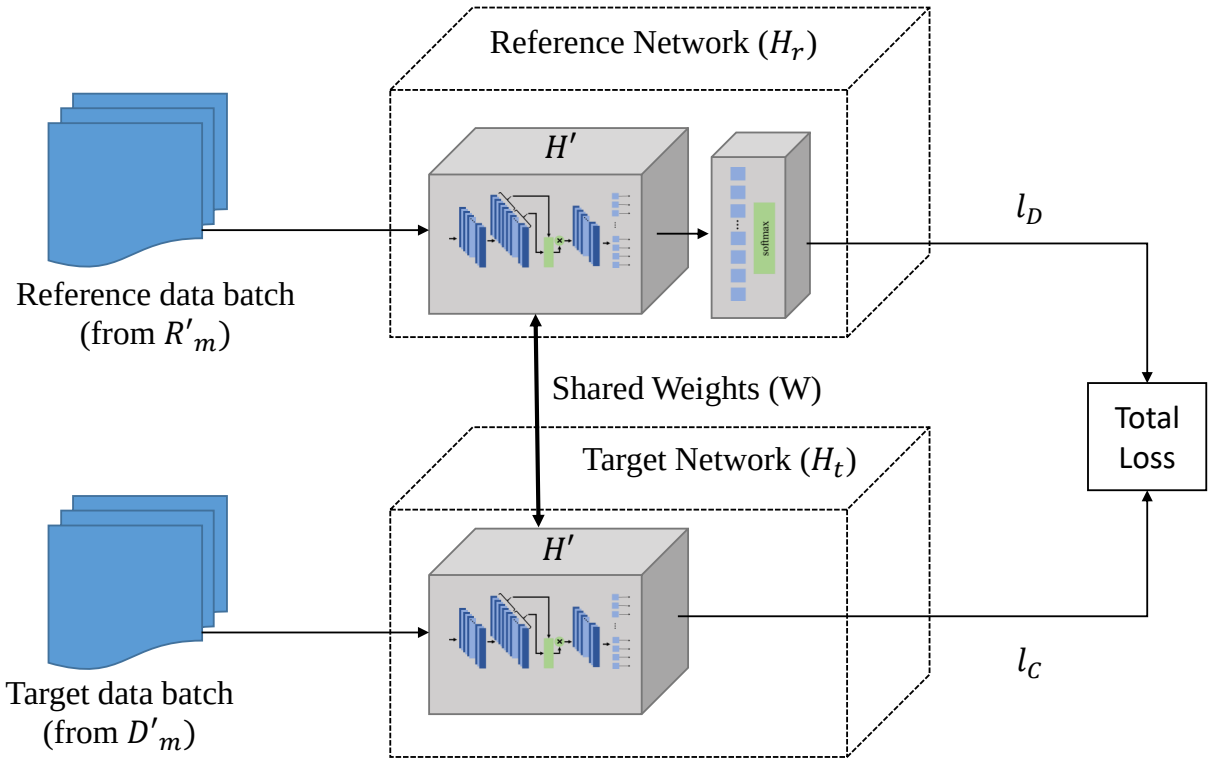

Figure 3.9: Training reference and target networks.

the chip testing phase.

As shown in Figure 3.9, we discard the last dense layer in H, call it H', and create two new models: a reference model H_r and a target model H_t. H_r consists of H' followed by a new softmax dense layer. H_t consists of only H' which shares the same weights with H' part of the reference model and is responsible for one-class feature extraction. To train H_r and H_t we use a reference dataset, and a target dataset.

3.4.4 Feature Extraction Evaluation

HTnet uses Stochastic Gradient Descent (SGD) optimizer with learning rate of 0.001 for training H and learning rate of 0.000001 for H_r and H_t. Figure 3.10 shows the outputs of H_t for some of the HT-inactive and the HT-triggered samples. As shown in this figure,

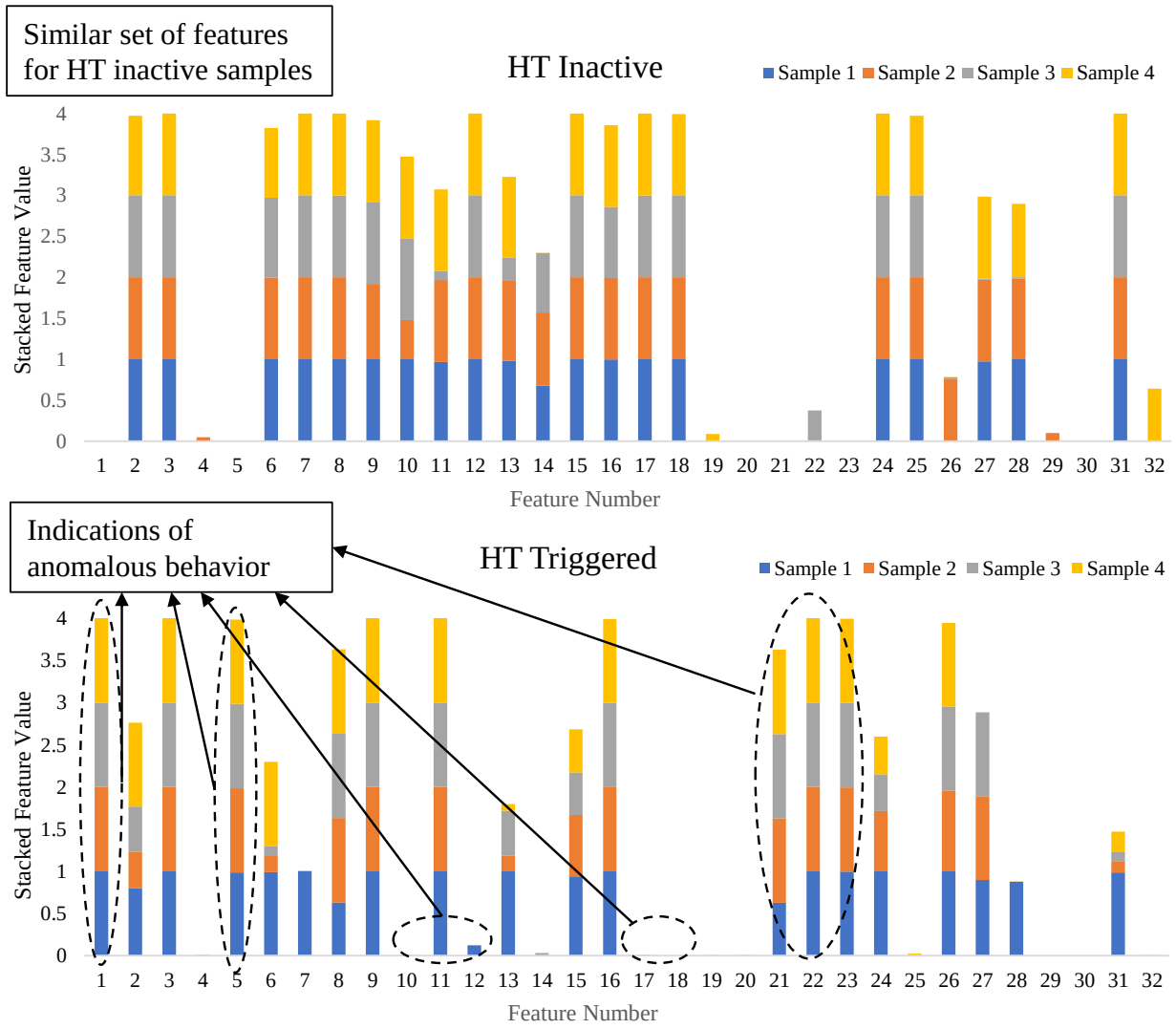

Figure 3.10: Extracted features for four HT-inactive and HT-triggered samples from AES-500 benchmark.

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.

Table 3.4: HTnet comparison with the methods in [9] 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

3.4.5 Golden Chip-Dependent HT Detection Evaluation

Given that our side-channel dataset includes both HT-inactive and HT active samples for each benchmark, [6] 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 3.4 compares the performance of HTnet to the methods proposed in one of the latest works in the literature [9] (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, HTnet first extracts and selects the best set of features using [3] 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.

Table 3.5: 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

3.4.6 Golden Chip-Free HT Detection Evaluation

Various off-the-shelf anomaly detection mechanisms from [22] are incorporateed in the paper to evaluate the effectiveness of extracted set of features from EM and power side-channel signals for HT detection. As shown in Table 3.5, the 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, evaluations is only provided for benchmarks with reliable EM traces. As shown in 3.5, EM and power side-channel signals fusion can boost the accuracy of our golden chip-free approach to pass the results presented in Table 3.4 in some cases.

Bibliography

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

[2] 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.

[3] 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.

[4] 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.

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

[6] S. Faezi, R. Yasaei, and M. Al Faruque. Htnet: Transfer learning for golden chip-free hardware trojan detection. *IEEE/ACM Design Automation and Test in Europe Conference (DATE'21)*, 2021.

[7] S. Faezi, R. Yasaei, A. Barua, and M. Al Faruque. Brain-inspired golden chip free hardware trojan detection. *IEEE Transaction on Information Forensics and Security (IEEE TIFS'21)*, 2021.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

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

[16] 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.

[17] S. Narasimhan, D. Du, R. S. Chakraborty, S. Paul, F. G. Wolff, C. A. Papachristou, K. Roy, and S. Bhunia. Hardware trojan detection by multiple-parameter side-channel analysis. *IEEE Transactions on computers*, 62(11):2183–2195, 2013.

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

[19] 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.

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

[21] R. Yasaei, S. Faezi, and M. Al Faruque. Hardware trojan power and em side-channel dataset. *Available in IEEE DataPort: https://dx.doi.org/10.21227/9fwb-8978*, 2021.

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

Appendix A

Hardware Trojan Benchmarks

I used the hardware Trojan benchmarks from Trust Hub [20] and implemented them on the FPGA for collecting side-channel signals. In the following, each benchmarks description is provided.

A.0.1 AES-T1000 Benchmark

Whenever a predefined input plaintext is observed, the Trojan leaks the secret key from a cryptographic chip running the AES algorithm through a covert channel. The channel adapts the concepts from spread spectrum communications (also known as Code-Division Multiple Access (CDMA)) to distribute the leakage of single bits over many clock cycles. The Trojan employs this method by using a pseudo-random number generator (PRNG) to create a CDMA code sequence; the PRNG initialized to the input plaintext. The code sequence is then used to XOR modulate the secret information bits. The modulated sequence is forwarded to a leakage circuit (LC) to set up a covert CDMA channel in the power side-channel. The LC is realized by connecting eight identical flip-flop elements to the single output of the XOR gate to mimic a large capacitance.

Table A.1: Trojan Taxonomy of AES-T1000 Benchmark.

Hardware Trojan Taxonomy	
Insertion Phase	Design
Abstraction Level	Register Transfer Level
Activation Mechanism	Triggered Internally
Malicious Effect	Leak Information
Location	Processor
Physical Characteristics	Functional

A.0.2 AES-T1100 Benchmark

Once a predefined sequence of input plaintext is observed, the Trojan leaks the secret key from a cryptographic chip running the AES algorithm through a covert channel. The channel adapts the concepts from spread spectrum communications (also known as Code-Division Multiple Access (CDMA)) to distribute the leakage of single bits over many clock cycles. The Trojan employs this method by using a pseudo-random number generator (PRNG) to create a CDMA code sequence, the PRNG initialized to the input plaintext. The code sequence is then used to XOR modulate the secret information bits. The modulated sequence is forwarded to a leakage circuit (LC) to set up a covert CDMA channel in the power side-channel. The LC is realized by connecting eight identical flip-flop elements to the single output of the XOR gate to mimic a large capacitance.

Table A.2: Trojan Taxonomy of AES-T1100 Benchmark

Hardware Trojan Taxonomy	
Insertion Phase	Design
Abstraction Level	Register Transfer Level
Activation Mechanism	Triggered Internally
Malicious Effect	Leak Information
Location	Processor
Physical Characteristics	Functional

A.0.3 AES-T700 Benchmark

Whenever a predefined input plaintext is observed, the Trojan leaks the secret key from a cryptographic chip running the AES algorithm through a covert channel. The channel adapts the concepts from spread spectrum communications (also known as Code-Division Multiple Access (CDMA)) to distribute the leakage of single bits over many clock cycles. The Trojan employs this method by using a pseudo-random number generator (PRNG) to create a CDMA code sequence; the PRNG initialized to a predefined value. The code sequence is then used to XOR modulate the secret information bits. The modulated sequence is forwarded to a leakage circuit (LC) to set up a covert CDMA channel in the power side-channel. The LC is realized by connecting eight identical flip-flop elements to the single output of the XOR gate to mimic a large capacitance.

Table A.3: Trojan Taxonomy of AES-T700 Benchmark

Hardware Trojan Taxonomy	
Insertion Phase	Design
Abstraction Level	Register Transfer Level
Activation Mechanism	Triggered Internally
Malicious Effect	Leak Information
Location	Processor
Physical Characteristics	Functional

A.0.4 AES-T800 Benchmark

Once a predefined sequence of input plaintext is observed, the Trojan leaks the secret key from a cryptographic chip running the AES algorithm through a covert channel. The channel adopts the concepts from spread spectrum communications (also known as Code-Division Multiple Access (CDMA)) to distribute the leakage of single bits over many clock cycles. The Trojan employs this method by using a pseudo-random number generator (PRNG) to create a CDMA code sequence; the PRNG initialized to a predefined value. The code sequence

is then used to XOR modulate the secret information bits. The modulated sequence is forwarded to a leakage circuit (LC) to set up a covert CDMA channel in the power side-channel. The LC is realized by connecting eight identical flip-flop elements to the single output of the XOR gate to mimic a large capacitance.

Table A.4: Trojan Taxonomy of AES-T800 Benchmark

Hardware Trojan Taxonomy	
Insertion Phase	Design
Abstraction Level	Register Transfer Level
Activation Mechanism	Triggered Internally
Malicious Effect	Leak Information
Location	Processor
Physical Characteristics	Functional

A.0.5 AES-T1300 Benchmark

Whenever a predefined input plaintext is observed, the Trojan demonstrates an attack on the AES-128 block-cipher and its corresponding key schedule. The idea is to artificially introduce leaking intermediate states in the key schedule that depend on known input bits and key bits, but that naturally would not occur during regular processing of the cipher. The Trojan uses AND conjunctions to pairwise combine each key bit with another input bit. The output of the AND gates are then combined to the leaked intermediate value by XORing all of them. The Trojan leaks one byte of the AES round key for each round of the key schedule. The leakage circuit (LC) is a 16-bit shift register and loaded it with an initial alternating sequence of zeros and ones. The shift register is only enabled in case the input to the leakage circuit is one, which results in additional dynamic power consumption.

Table A.5: Trojan Taxonomy of AES-T1300 Benchmark

Hardware Trojan Taxonomy	
Insertion Phase	Design
Abstraction Level	Register Transfer Level
Activation Mechanism	Triggered Internally
Malicious Effect	Leak Information
Location	Processor
Physical Characteristics	Functional

A.0.6 AES-T1400 Benchmark

Once a predefined sequence of input plaintext is observed, the Trojan demonstrates an attack on the AES-128 block-cipher and its corresponding key schedule. The idea is to artificially introduce leaking intermediate states in the key schedule that depend on known input bits and key bits, but that naturally would not occur during regular processing of the cipher. The Trojan uses AND conjunctions to pairwise combine each key bit with another input bit. The output of the AND gates are then combined to the leaked intermediate value by XORing all of them. The Trojan leaks one byte of the AES round key for each round of the key schedule. The leakage circuit (LC) is a 16-bit shift register and loaded it with an initial alternating sequence of zeros and ones. The shift register is only enabled in case the input to the leakage circuit is one, which results in additional dynamic power consumption.

Table A.6: Trojan Taxonomy of AES-T1400 Benchmark

Hardware Trojan Taxonomy	
Insertion Phase	Design
Abstraction Level	Register Transfer Level
Activation Mechanism	Triggered Internally
Malicious Effect	Leak Information
Location	Processor
Physical Characteristics	Functional

A.0.7 AES-T400 Benchmark

Modulating an (unused) pin on a chip generates an RF signal. This signal can be used to transmit the key bits. This attack is performed at 1560KHz and can be received with an ordinary AM radio. The data carried by the AM signal needs to be easily interpreted by a human. A beep scheme is utilized where a single beep followed by a pause represents a '0' and a double beep followed by a pause represents a '1'. A description of the detail implementation of AM transmission can be found at [6]. In this implementation, the Trojan gets activated when a predefined input plaintext is observed.

Table A.7: Trojan Taxonomy of AES-T400 Benchmark

Hardware Trojan Taxonomy	
Insertion Phase	Design
Abstraction Level	Register Transfer Level
Activation Mechanism	Triggered Internally
Malicious Effect	Leak Information
Location	Processor
Physical Characteristics	Functional

A.0.8 AES-T1600 Benchmark

Modulating an (unused) pin on a chip generates an RF signal. This signal can be used to transmit the key bits. This attack is performed at 1560KHz and can be received with an ordinary AM radio. The data carried by the AM signal needs to be easily interpreted by a human. A beep scheme is utilized where a single beep followed by a pause represents a '0' and a double beep followed by a pause represents a '1'. A description on detail implementation of AM transmission can be found at [6]. In this implementation, the Trojan gets activated when a predefined sequence of input plaintext is observed.

Table A.8: Trojan Taxonomy of AES-T1600 Benchmark

Hardware Trojan Taxonomy	
Insertion Phase	Design
Abstraction Level	Register Transfer Level
Activation Mechanism	Triggered Internally
Malicious Effect	Leak Information
Location	Processor
Physical Characteristics	Functional

A.0.9 AES-T1800 Benchmark

At the core of lightweight applications, such as medical implant devices, are the batteries that power them, and the success of the device rests heavily on them. This Trojan drains the battery once it gets activated. The Trojan gets activated after observing a predefined input plaintext. The Trojan payload is a shift register which continuously rotates after Trojan activation. The Trojan increases the power consumption and hence decreases the expected lifetime of the battery.

Table A.9: Trojan Taxonomy of AES-T1800 Benchmark

Hardware Trojan Taxonomy	
Insertion Phase	Design
Abstraction Level	Register Transfer Level
Activation Mechanism	Triggered Internally
Malicious Effect	Denial of Service
Location	Processor
Physical Characteristics	Functional

A.0.10 AES-T500 Benchmark

At the core of lightweight applications, such as medical implant devices, are the batteries that power them, and the success of the device rests heavily on them. This Trojan drains the battery once it gets activated. The Trojan gets activated after observing a specific sequence of the input plaintext. The Trojan payload is a shift register which continuously rotates

after Trojan activation. The Trojan increases the power consumption and hence decreases the expected lifetime of the battery.

Table A.10: Trojan Taxonomy of AES-T500 Benchmark

Hardware Trojan Taxonomy	
Insertion Phase	Design
Abstraction Level	Register Transfer Level
Activation Mechanism	Triggered Internally
Malicious Effect	Denial of Service
Location	Processor
Physical Characteristics	Functional

A.0.11 AES-T600 Benchmark

After detecting a specific input plaintext, the Trojan leaks the secret key of AES-128 through the leakage current. The leakage circuit (LC) consists of a shift register holding the secret key and two inverters. The least significant bit is connected to one inverter whose output connected to the input of the other inverter. Whenever the least significant bit of the shift register is '0', a direct path between power and ground composed by the PMOS of the first inverter and the NMOS of the second inverter is created for a limited time. Therefore, the secret key can be retrieved by measuring the leakage current.

Table A.11: Trojan Taxonomy of AES-T600 Benchmark

Hardware Trojan Taxonomy	
Insertion Phase	Design
Abstraction Level	Register Transfer Level
Activation Mechanism	Triggered Internally
Malicious Effect	Leak Information
Location	Processor
Physical Characteristics	Functional

A.0.12 AES-T2000 Benchmark

After detecting a specific sequence of input plaintext, the Trojan leaks the secret key of AES-128 through the leakage current. The leakage circuit (LC) consists of a shift register holding the secret key and two inverters. The least significant bit is connected to one inverter whose output connected to the input of the other inverter. Whenever the least significant bit of the shift register is '0', a direct path between power and ground composed by the PMOS of the first inverter and the NMOS of the second inverter is created for a limited time. Therefore, the secret key can be retrieved by measuring the leakage current.

Table A.12: Trojan taxonomy of AES-T2000 benchmark.

Hardware Trojan Taxonomy	
Insertion Phase	Design
Abstraction Level	Register Transfer Level
Activation Mechanism	Triggered Internally
Malicious Effect	Leak Information
Location	Processor
Physical Characteristics	Functional