Systematic Analysis and Methodologies for Hardware Security

by

Samer Moein
B.Sc., Kuwait University, 2004
M.Sc., Kuwait University, 2011

A Dissertation Submitted in Partial Fulfillment of the
Requirements for the Degree of

DOCTOR OF PHILOSOPHY

in the Department of Electrical and Computer Engineering

Systematic Analysis and Methodologies for Hardware Security

by

Samer Moein
B.Sc., Kuwait University, 2004
M.Sc., Kuwait University, 2011

Supervisory Committee

Dr. Fayez Gebali, Co-Supervisor
(Department of Electrical and Computer Engineering)

Dr. T. Aaron Gulliver, Co-Supervisor
(Department of Electrical and Computer Engineering)

Dr. Alex Thomo, Outside Member
(Department of Computer Science)

Supervisory Committee

Dr. Fayez Gebali, Co-Supervisor
(Department of Electrical and Computer Engineering)

Dr. T. Aaron Gulliver, Co-Supervisor
(Department of Electrical and Computer Engineering)

Dr. Alex Thomo, Outside Member
(Department of Computer Science)

ABSTRACT

With the increase in globalization of Integrated Circuit (IC) design and production, hardware trojans have become a serious threat to manufacturers as well as consumers. These trojans could be intensionally or accidentally embedded in ICs to make a system vulnerable to hardware attacks. The implementation of critical applications using ICs makes the effect of trojans an even more serious problem. Moreover, the presence of untrusted foundries and designs cannot be eliminated since the need for ICs is growing exponentially and the use of third party software tools to design the circuits is now common. In addition if a trusted foundry for fabrication has to be developed, it involves a huge investment. Therefore, hardware trojan detection techniques are essential. Very Large Scale Integration (VLSI) system designers must now consider the security of a system against internal and external hardware attacks. Many hardware attacks rely on system vulnerabilities. Moreover, an attacker may rely on deprocessing and reverse engineering to study the internal structure of a system to reveal the system functionality in order to steal secret keys or copy the system. Thus hardware security is a major challenge for the hardware industry. Many hardware attack mitigation techniques have been proposed to help system designers build secure systems that can resist hardware attacks during the design stage, while others protect the system against attacks during operation.

In this dissertation, the idea of quantifying hardware attacks, hardware trojans, and hardware trojan detection techniques is introduced. We analyze and classify hardware attacks into risk levels based on three dimensions Accessibility/Resources/Time (ART). We propose a methodology and algorithms to aid the attacker/defender to select/predict the hardware attacks that could use/threaten the system based on the attacker/defender capabilities. Because many of these attacks depends on hardware trojans embedded in the system, we propose a comprehensive hardware trojan classification based on hardware trojan attributes divided into eight categories. An adjacency matrix is generated based on the internal relationship between the attributes within a category and external relationship between attributes in different categories. We propose a methodology to generate a trojan life-cycle based on attributes determined by an attacker/defender to build/investigate a trojan. Trojan identification and severity are studied to provide a systematic way to compare trojans. Trojan detection identification and coverage is also studied to provide a systematic way to compare detection techniques and measure their effectiveness related to trojan severity. We classify hardware attack mitigation techniques based on the hardware attack risk levels. Finally, we match these techniques to the attacks the could countermeasure to help defenders select appropriate techniques to protect their systems against potential hardware attacks.

Contents

List of Tables

List of Figures

ACKNOWLEDGEMENTS

In the name of Allah, the Most Gracious and the Most Merciful

Alhamdulillah, all praises belongs to Allah the merciful for his blessing and guidance. He gave me the strength to reach what I desire. I would like to thank:

My parents, my family, for supporting me at all stages of my education and their unconditional love.

My Supervisor, Dr. Fayez Gebali, for all the support, and encouragement he provided to me during my work under his supervision. It would not have been possible to finish my research without his invaluable help of constructive comments and suggestions.

My Supervisor, Dr. T. Aaron Gulliver, for his precious time and valuable suggestions for the work done in this dissertation. It would not have been possible to finish my research without his invaluable help of constructive comments and suggestions.

You will not live long, try to live forever
Samer Moein

DEDICATION

To my parents, **Moein Moein and Amal Ahmed** for their love, prayers, and encouragement.

To my lovely wife, **Mai Fawzy** for always standing by me, and believing in me.

To my beautiful daughter **Mayrin**.

List of Abbreviations

AC	Alternating Current
ACA	Acoustic Attack
AES	Advanced Encryption Standard
AIT	Advanced Imaging Techniques Attack
ALU	Arithmetic and Logic Unit
ART	Accessibility/Resources/Time
ASIC	Application Specific Integrated Circuit
C-JTAG	Covert JTAG Port Attack
CAD	Computer-Aided Design
CMOS	Complementary Metal Oxide Semiconductor
CPT	Chip Partition Technique
CRT	Chinese Remainder Theorem
CUA	Chip Under Authentication
DARPA	Defence Advanced Research Projects Agency
DC	Direct Current
DEMA	Differential Electro-Magnetic Attack
DEMFA	Differential Electro-Magnetic Frequency Analysis
DEP	Deprocessing Attack
DFA	Deferential Fault Analysis
DoS	Denial of Service

DPA	Differential Power Analysis Attack
DPFA	Differential Power Frequency Analysis
DRA	Data Remanence Attack
DRAM	Dynamic Random Access Memory
DUA	Device Under Attack
DUT	Device Under Test
EEPROM	Electrically Erasable Programmable Read-Only Memory
EM	Electro-Magnetic
EMA	Electro-Magnetic Attack
FAT	Fault Analysis Attack
FBA	Frequency Based Analysis Attack
FIB	Focused-Ion Beam
FIT	Fault Injection Attack
FPGA	Field Programmable Gate Array
FSM	Finite State Machine
GC	Golden Chip
HRA	High Risk Attacks
I/O	Input/Output
IC	Integrated Circuit
IP	Intellectual Property
IR	Infrared Radiation
ISA	Instruction Set Architecture
JTAG	Joint Test Action Group
LED	Light Emitting Diode
LIVE	Light-Induced Voltage Alteration
LRA	Low Risk Attacks

MICRO	Microprobing Attack
MRA	Medium Risk Attacks
O-JTAG	Overt JTAG Port Attack
OBIC	Optical Beam Induced Current
OEA	Optical Emanation Attack
OPLP	Optically Enhanced Position-Locked Power Analysis Attack
PKI	Public Key Infrastructure
PSD	Power Spectral Density Signal
PV	Process Variation
RAM	Random Access Memory
RE	Reverse Engineering Attack
RL	Hardware Attack Risk Level
RNG	Random Number Generator
RO	Ring Oscillator
RSA	Rivest, Shamir, and Adleman
RTL	Register-Transfer Level
S-Box	Substitution Box
SAM	Scanning Acoustic Microscopy
SCADA	Supervisory Control And Data Acquisition
SEM	Scanning Electron Microscopy
SEMA	Simple Electro-Magnetic Attack
SFA	Simple Fault Analysis
SNR	Signal-to-Noise Ratio
SPA	Simple Power Analysis Attack
SRAM	Static Random Access Memory
SSH	Secure Shell

SSL	Secure Sockets Layer
TA	Timing Attack
TAP	Test Access Port
TCK	Test Clock
TMS	Test Mode Select
UV EPROM	Ultraviolet Erasable Programmable Read Only Memory
VLSI	Very Large Scale Integration
XRF	X-Ray Fluoroscopy

Chapter 1

Introduction

1.1 Background

Securing the hardware of computing and communications systems is now a primary concern of system designers. Significant research in this area is being done by both industry and academia. These systems often store private keys or other sensitive data, so a compromise of this data or the hardware that processes this data can lead to loss of privacy, forged access, or monetary theft. Even if an attacker fails to gain the secret information that is stored in the hardware, attackers may be able to disrupt the hardware functionality or deny service leading to other kinds of system failures. Hardware attacks aim at physically accessing the system to obtain stored information, study the internal structure of the hardware, or to inject a fault. Most hardware attack classification models proposed in the literature are based on the level at which an attacker accesses the system [1]. Another concern is side channel attacks, which can be classified based on the awareness of the attacks [2]. These classifications are overlapping and qualitative in nature. This leaves system designers and users unable to judge which attacks are more important to consider. A review of hardware attacks shows that four main classification criteria or factors should be considered for effective detection of these attacks. The four criteria are: accessibility, resources, time, and awareness. The classification model in [1] depends only on the attacker access to a system, while the classification model in [2] depends on awareness only.

There are many real hardware attack cases incident which have caused significant government, industry, and consumer concerns. In 2009, several countries participated

in a multinational air manoeuvre. One of the exercises stipulated that airplanes from country B were to launch missiles against airplanes from country A. The aircraft used by country B were manufactured in country A. The missiles failed to launch even after numerous attempts. A thorough inspection of the airplanes from country B was unsuccessful in determining the reason for the failure of the weapons control systems. After the manoeuvre was over, the airplanes from country B used the missile systems and no problems were encountered. A possible reason for the failure is that a hardware backdoor was used to disable the missile systems [47].

Between 2011 and 2012, many owners of a certain car make and model observed a particular error message displayed on their dashboards. It was suspicious that this error appeared after the expiration of the car warranties. The car dealers informed the owners that their computer system had to be reprogrammed and if the problem appeared again, the computers should be replaced. From the timing of this event, it could be concluded that a timed failure was inserted into the computer systems of these cars [48].

In September 2007, Israeli jets bombed a suspected nuclear installation in north-eastern Syria. Among the mysteries surrounding this airstrike is the failure of the Syrian radar systems. It has been suggested that the commercial off-the-shelf microprocessors used in these systems may have been fabricated with a hardware backdoor which was used to temporarily disable them [49].

Many types of hardware attacks have been identified. One type monitors and analyzes the execution time needed during cryptographic processing. This attack was first discussed in [5], and the first practical implementation was presented in [6]. A timing attack against the Rivest, Shamir, and Adleman (RSA) algorithm using the Chinese Remainder Theorem (CRT) was given in [7]. An attack against the Advanced Encryption Standard (AES) algorithm was presented in [8]. An attack against the Patterson algorithm within the McEliece public-key cryptosystem was given in [9], and against the secret permutation in the McEliece public-key cryptosystem in [10]. A detailed study of this type of attack was presented in [11]. Another type monitors the power consumption by measuring the electromagnetic power radiations [12–14]. The acoustic signals from an encryption coprocessor can be monitored to deduce its starting positions, which can reveal encryption key information [15–18]. Optically enhanced power analysis is an innovative technique that reveals the current in transistors [19–24]. Diffused reflections from computer displays can be used to reconstruct the data on the screen [25, 26]. Other examples of hardware attacks include data re-

manence in Static Random Access Memory (SRAM) [27], and in flash memory [28] and failure analysis [29–31]. Additional attacks are discussed in [1–4, 32–35].

The use of semiconductor devices in military, financial, economic, and other critical infrastructure has raised significant concerns regarding hardware security. Malicious modifications to ICs, commonly known as hardware trojans, which alter the parameters of ICs leading to abnormal behaviour of the system. Many hardware attacks rely on these trojans to attack systems. Moreover, an attacker may use a hardware attack to study the internal structure of a system to reveal the device functionality to steal secret keys or copy the device. Thus, hardware security is a major challenge for the hardware industry [95, 96].

In response to the threat of hardware attacks, the Defence Advanced Research Projects Agency (DARPA) initiated the trust in ICs program to develop techniques for trojan detection [49, 50]. This highlights the fact that hardware designers and researchers must be vigilant to the insertion of hardware trojans during all phases of the chip production life-cycle.

Several researchers have proposed taxonomies for hardware trojans based on their attributes [51–54]. In [53], hardware trojans were classified based on two categories: trigger and payload. These are in fact activation mechanisms for trojans. In [52] and [54], the classification was based on three categories: physical, activation, and action. Although this adds two categories to the previous taxonomy, the classification is not related to the chip life-cycle. In [51], a more detailed classification was developed based on five categories: insertion phase, abstraction level, activation mechanism, effect, and location. This classification considers the chip life-cycle and the targeted location, but not the physical characteristics of trojans. The taxonomy in [51] was tested using a set of trojans to verify the classification, and the attributes corresponding to each trojan was identified. However, the relationship between the attributes associated with a hardware trojan has not been investigated in the literature.

Hardware trojans can be implemented in microprocessors, microcontrollers, network and digital signal processors, Field Programmable Gate Array (FPGAs), Application Specific Integrated Circuits (ASICs), and other ICs. Therefore, research on hardware trojan detection techniques is becoming an essential. Figure 1.1 shows the classification of hardware trojan detection techniques proposed in [67]. It classifies the hardware trojan detection techniques into destructive and non-destructive techniques. Destructive techniques (i.e. reverse engineering) are used mainly to obtain a trojan free chip to be used later as a reference Golden Chip (GC). Such techniques are

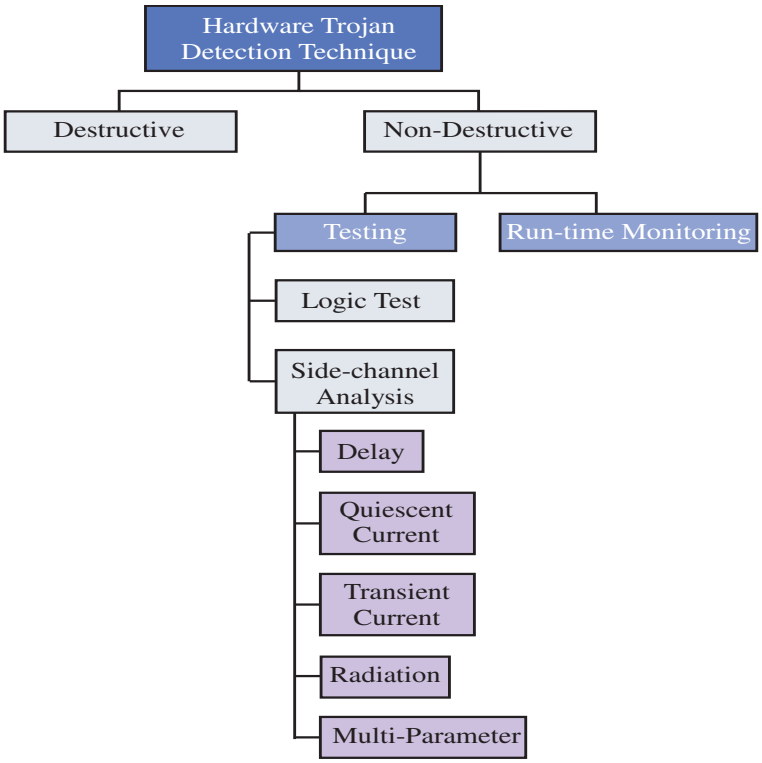


Figure 1.1: Hardware trojan detection technique classification [67].

extremely expensive and time consuming. Therefore, it is not practical to test chips using destructive techniques. Further, testing cannot rely on a set of sample chips because an adversary might insert a trojan in a small percentage of the chips. Process Variation (PV) also plays a critical factor to distinguish between false negative and trojan free chip based on the GC.

Non-destructive techniques can be classified into testing and run-time monitoring. Testing can be supported by design for security circuits (i.e. scan chains and self-test circuitry). These circuits can substantially enhance the sensitivity and/or coverage of trojan detection techniques. However, it must be ensured that the extra test circuitry is not compromised. Trojans could be designed to avoid triggering during the testing phase. Therefore, developing hardware trojan detection techniques that will not effect the chip layout or design flow and can be used to test a chip after manufacturing is extremely important.

Testing approaches are further classified into logic-testing and side-channel analysis approaches. Logic-testing approaches focus on generating a random test-vector for activating a trojan circuit and observing its effect at the outputs. The difficulty

with this approach is that all internal nodes should be tested with all possible logic values and the results observed. Chips become extremely dense in terms of gates so a comprehensive test and provide all candidate combination of the gates in the chips is an extremely challenging task.

Side-channel analysis approaches are based on the fact that any modification in a chip should be reflected in some side-channel parameter such as the dynamic power [68–72], leakage current [77, 78], path-delay [83–85], Electro-Magnetic (EM) emissions due to switching [76], or a combination of these [79, 82]. Side-channel approaches suffer from sensitivity to errors due to process variation and noise. Relating to reference measurements, noise is a crucial factor in determining if a suspect chip is infected chip. A good detection technique is one that has a high probability of detecting an infected chip and a low probability of false alarm. The advantage of side channel approaches over logic-testing approaches is not having to activate the trojan to be able to detect it. Many factors can affect this advantage such as the trojan properties, PVs, and noise.

Most existing hardware trojan detection techniques are based on side channel parameters. While they are effective, it is very hard to deal with the high complexity and density chips. High complexity chips make it difficult to detect small trojans and distributed trojans. Trojan circuits are typically small in size compared to the original design. Moreover, there may be inserted in blank spaces in the layout of chip during the fabrication phase and the circuit rewired to implement the malicious effects.

Run-time monitoring is used to continuously monitor a chip during operation to detect any malicious logic. This can be achieve by exploiting pre-existing redundancy in the circuit. With this approach, area and delay cost effects lead to performance overhead. Chip activity can continue while monitoring is done, but this result in performance overhead. Therefore, run-time monitoring improves the reliability of a chip that contains a trojan which has passed the test phase bypassing the trojan effects. Chips may also be equipped with a self destruct mechanism to disable it once a trojan is detected.

The knowledge, skill, sophistication and resources that a modern attacker possesses often enables them to introduce a modification into the design during the IC life-cycle. Many of these modifications go undetected during the testing and deployment phases [49, 96, 97]. Developing hardware attack mitigation techniques against these attacks begins with identifying them. Hardware attack mitigation techniques

can be divided into two categories. First, countermeasures have been proposed for types of attacks. Hiding techniques are based on reducing the signal strength or increasing the noise level [22]. Masking techniques reduce the correlation between the side channel emissions and the input data [99–102]. Generating random noise to decrease the Signal-to-Noise Ratio (SNR) and increase the noise in the emitted signals [128], or make the emissions independent of the chip operation [22,129]. The chip EM emissions can also be reduced by asynchronous logic gates [130,131], or reducing the overall chip emissions by using low power designs [22]. Moreover, emissions to regions on the chip can be reduced by partitioning the design [12,103]. Hardened the accessibility to the chip to prevent the attacker from collective the chip emissions due anti-tampering technique [35, 104]. Moreover, chips noise can be filtered to reduce the number of sample signals needed for information leakage [105]. These techniques can be used to counter most covert attacks. Sensor meshes implemented above the IC to sense all paths for interruptions, and robust low frequency sensors could be used to trigger if the clock edge is not sensed longer than a specified duration [113] could be used to counter overt attacks. Second, countermeasures have been proposed for specific hardware attacks. Algorithmic resistance, restricting physical access, randomizing computation time [111], and duplicate encryption [106] are used to counter fault attacks. Time/branch equalization, adding random delays, and constant time hardware [2] are used to counter timing attacks. Keyed hash functions, message authentication codes, public key infrastructure, and stream ciphers, are used to design protocols for increased security in Joint Test Action Group (JTAG) devices and implementing encryption [38]. Test Access Port (TAP) design allows enforcement of digital rights management, which works on hashing and challenge/response protocols to access JTAG infrastructure, or use public/private key pairs for authentication [108], are used to counter JTAG attacks. Cycling memory with random data can be used to mitigate data remanence attacks [28]. Partitioned cache can be used to prevent information leakage since objects are in secluded or locked mode in the cache [121]. Sensitive cache lines can be locked in a secluded secure partition [122]. Dynamic mapping memory-to-cache can be used to map the memory block to the desired cache line at run time [123]. Non-deterministic processor can be used to run instructions in random order [124]. These mitigation techniques are used to counter cache attacks. Encrypted busses to enforce the attacker to run out of time and resources to analyze the data which it may ciphered using random number generator [114] is used to mitigate microprobing attack.

1.2 Motivation

New techniques are constantly being developed to attack a system from the physical perspective, and countermeasures for these attacks are also being designed. What is required is a comprehensive survey of attacks which can be expanded as new attacks arise. A methodology that can be used to update this list should be based on the properties of each attack. This can help security designers to test their systems against emerging threats. From the attacker perspective, this methodology can be used to determine attacks which match their ability and awareness. From the defender perspective, it can be used to identify system vulnerabilities and develop countermeasures. The proposed methodology should be flexible so that new attacks can be incorporated. Obsolete attacks can also be removed. This methodology should be based on a set of attack criteria, and allowing weights to be specified for the criteria. Thus, as technology changes the weights can be adjusted. Further, the weights can differ between attackers and defenders depending on their capabilities.

Many hardware attacks depend on hardware trojans embedded in systems. Different views exist in the literature to describe hardware trojans, but they are all qualitative and not comprehensive in terms of the attributes of hardware trojans. Moreover, the relationships among the attributes associated with hardware trojans has not been investigated in the literature. Therefore, a comprehensive hardware classification is needed. A flexible methodology is needed that can be used with any hardware trojan classification, so that new attributes based on technology or chip manufacturing developments can easily be accommodated. As with any circuit, a hardware trojan goes through several production phases after it is embedded in the target system. Therefore, studying the production life cycle along with other attributes will provide a better insight into the insertion phase, functionality, logic type, physical characteristics, and location of a trojan.

Hardware trojan detection techniques are used to detect if there is a trojan embedded in a system. Each detection technique proposed based on a trojan designed by the authors of the technique to test the effective of the proposed technique. Most of the published result based on proposed trojan circuit, any modification to this circuit it affects the result. Studying the trojan properties is a more effective way to build a general detection technique than building a detection technique based on specific trojan. Identifying hardware trojan techniques and its coverage provides an idea to measure the effectiveness of any detection technique regarding other techniques.

This will also help security engineers compare different techniques to select the most appropriate ones.

System designers and system security engineers need to understand the threats that may affect their systems, and employ appropriate hardware attack mitigation techniques to protect their systems. A comprehensive survey of mitigation techniques for hardware attacks is needed. In addition, these techniques should be matched to the hardware attacks they can mitigate. Moreover, these techniques should be divided based on hardware attack risk levels to help the designers and security engineers choose appropriate techniques to protect their systems against hardware attacks.

1.3 Contributions

The goal of this work is to study and quantify the major hardware security areas, which are hardware attacks, hardware trojans, hardware trojan detection techniques, and hardware attack mitigation techniques. Methodologies are proposed to study the internal relationships within each area, and the external relationships with other areas. The contributions of this dissertation are summarized below:

1. Systematic analysis of hardware attacks:
    - An analysis of hardware attacks based on four criteria: awareness, accessibility, resources, and time.
    - An assessment of hardware attack risk levels.
    - An algebraic methodology is developed for investigating hardware attacks.
    - Algorithms for an attacker are presented to aid in designing attacks.
    - Algorithms for a defender are presented which can be used to predict and quantify system vulnerabilities.
2. Systematic analysis of hardware trojans:
    - A comprehensive hardware trojan taxonomy is developed based on eight categories: insertion, abstraction, effect, logic type, functionality, activation, physical layout, and location.
    - An algebraic approach is developed to investigate hardware trojans based on these categories.

3. Methodology for hardware trojan severity and detection coverage:
    - Generating hardware trojan identification and severity.
    - Hardware trojan detection classification, identification and coverage.
4. Hardware attack mitigation techniques based on hardware attack levels:
    - A comprehensive survey of hardware attack mitigation techniques is given based on hardware risk levels.
    - The mitigation techniques are with the matched hardware attacks that they can counter.

1.4 Agenda

This section presents an overviews of the dissertation and a short description of each chapter.

Chapter 1 presents the problem considered and the contributions of the dissertation.

Chapter 2 introduces new Accessibility/Resources/Time (ART) schema and new hardware attacks classification. A review of hardware attack properties and proposes categories for these properties. The L_1-norm is used to determine the attack risks. An algebraic approach to attack examination is presented as well as algorithms based on this approach is developed.

Chapter 3 reviews the chip life-cycle, then develop a comprehensive hardware trojan taxonomy based on eight categories: insertion, abstraction, effect, logic type, functionality, activation, physical layout, and location. The relationship between hardware trojan attributes is studied using an adjacency matrix. An algebraic approach is developed to investigating hardware trojans based on these categories. Also, two case studies are presented.

Chapter 4 proposes hardware trojan detection classification with supported published techniques. The values for hardware trojan attributes are discussed, calculated, and assigned. Moreover, hardware trojans identification and severity are proposed. In addition, hardware trojan detection identification and coverage are proposed.

Chapter 5 classifies hardware mitigation techniques based on hardware risk levels. Each mitigation technique is matched to the hardware attacks that they can counter.

Chapter 6 summarizes the dissertation contributions and presents some topics for future work.

Chapter 2

Systematic Analysis of Hardware Attacks

Many VLSI chips now contain cryptographic processors to secure their data and external communications. Attackers target the hardware to imitate or understand the system design, to gain access to the system or to obtain encryption keys. They may also try to initiate attacks such as denial of service to disable the services supported by a chip, or reduce system reliability.

In this chapter, quantifying hardware attacks based on *ART schema* is proposed. According to the proposed classification, hardware attacks could be covert or overt based on awareness of the targeted system. An algebraic methodology is proposed to examine hardware attacks based on the attack properties and associated risks. This methodology is employed to construct algorithms to develop hardware attack and defence strategies. It can also be used to predict system vulnerabilities and assess the security of a system.

The remainder of this chapter is organized as follows. Section 2.1 reviews the hardware attack properties and proposes categories for these properties. The L_1-norm is used in Section 2.2 to determine the attack risks. Section 2.3 discusses the algebraic approach to attack examination. Finally, Section 2.4 presents algorithms based on this approach.

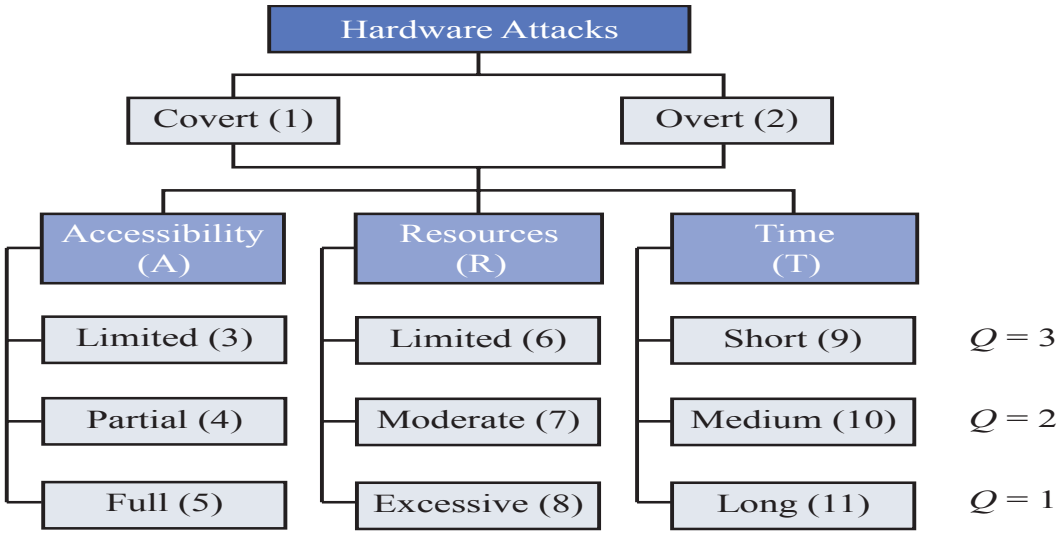


Figure 2.1: A hardware attack classification.

2.1 Hardware Attack Classification

Hardware attacks aim at physically accessing the system to obtain stored information, determine the internal structure of the hardware, or inject a fault. A quantified hardware attack classification based on four properties was proposed in [3, 4]. These properties are Awareness (W), Accessibility (A), Resources (R), and Time (T), as shown in Figure 2.1.

The awareness property (W) divides hardware attacks based on the evidence left of an attack on a system. Thus, the two categories are covert and overt. An attack is covert when the victim is not aware that it is taking place. Conversely, an attack is overt when the victim is aware that it is occurring.

The accessibility property (A) classifies hardware attacks based on the required level of access to a system. This property is divided into three categories: limited, partial, and full access. Limited access refers to no physical connection to the hardware, while with partial access an attacker can connect to the hardware or scan it. Full access means that the attacker can reach the gate level of a chip. The A levels are then {Full Access, Partial Access, Limited Access} $\equiv$ {1, 2, 3}.

The resources property (R) refers to the equipment and manpower needed to successfully launch an attack. This property is divided into three categories: limited, moderate, and excessive resources. Limited resources (R $<$ \$10,000) includes equipment such as an IC soldering/desoldering station, digital multimeter, universal chip programmer, prototyping boards, power supply, oscilloscope, logical analyzer, and

signal generator. Moderate resources ($10,000 ≤ R ≤$100,000) includes equipment such as a laser microscope, laser interferometer navigation, infrared imaging, and photomultiplier tube. Excessive resources (R > $100,000) includes equipment such as a laser cutter, Focused-Ion Beam (FIB), and Scanning Electron Microscope (SEM). The R levels are then {Excessive Resources, Moderate Resources, Limited Resources} ≡ {1, 2, 3}.

The time property (T) refers to the amount of time, effort, and experience required to execute an attack. This property is divided into three categories: short, medium, and long time. Short time refers to an attack that takes less than few days to succeed, while medium time refers to an attack that succeeds within weeks, and long time refers to an attack that succeeds within months. The T levels are then {Long Time, Medium Time, Short Time} ≡ {1, 2, 3}.

2.1.1 ART Schema

Figure 2.2 shows a three-dimensional (3D) model where each axis represents one of the properties Accessibility (A), Resources (R), and Time (T). This is based on the approach to quantifying covert hardware attacks in [4], and overt hardware attacks in [3]. With this model, an attack is represented as a point in 3D space whose coordinates are $\mathbf{p} = (a, r, t)$, where $1 \leq$ a, r, t ≤ 3. Each point may map to multiple hardware attacks, while an attack maps to a unique point. The focus in [3,4] was on placing attacks within the 3D ART model based on the requirements to be successful.

2.2 Hardware Attack Risk Levels (RL)

In this section, three attack levels, high, medium and low, are considered based on the results in [3, 4]. Regardless of its awareness, a hardware attack requires certain levels of accessibility, resources, and time, a, r, and t, respectively, to succeed. Based on these values, a risk level can be assigned to this attack with respect to the target system. The L_1-norm of the attack point $\mathbf{p}$ in the 3D ART space is given by

$$L_1 = a + r + t. \tag{2.1}$$

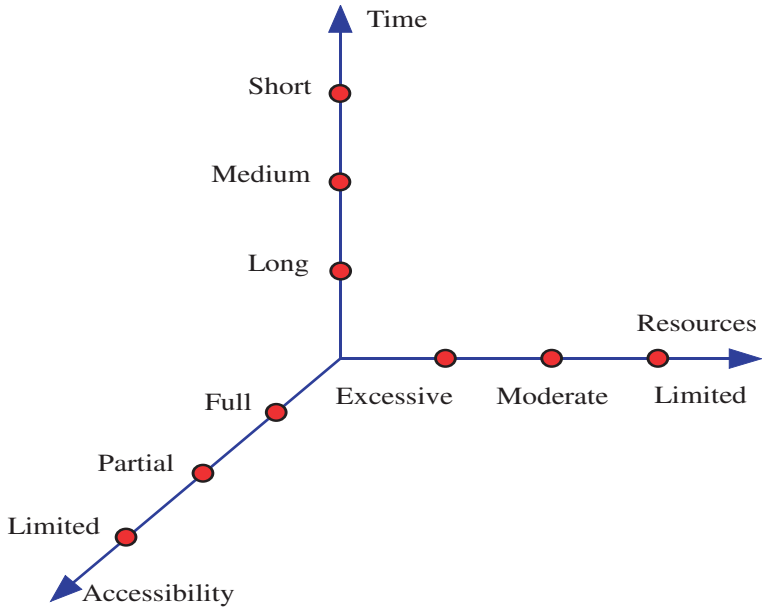


Figure 2.2: A 3D representation of the Accessibility (A), Resources (R), and Time (T) hardware attack properties.

Based on (2.1), attacks can be quantized into three levels: high risk, medium risk, and low risk.

2.2.1 High Risk Attacks (HRA)

High risk attacks are hardware attacks that require limited capabilities for execution. These attacks require limited resources and little time, so there is typically no evidence left and thus are often covert. Attacks belonging to this level are simple and so many attackers have the necessary resources and experience to execute them. Therefore, this attack level is the most dangerous. Examples of HRA are:

Electro-Magnetic (EMA) Attack

Electro-Magnetic (EM) emissions originate from the flow of electric current, which carried also other information like power and time that can be measured by using other signals as carriers. So, the attacker aims to measure this information to get the secret key. Also, the clock signal and power supply rails are particularly good carriers due to their high EM emissions. Attacks that exploit EM can be classified as either Simple Electro-Magnetic Attack (SEMA) (3, 3, 3) or Differential Electro-Magnetic Attack (DEMA) (3, 3, 2). SEMA studies a single EM signal to determine the internal

operation, while DEMA uses statical analysis for multiple EM signals to obtain the secret key [12].

The measures for EMA are [Covert, (3, 3, X), L_1= 8–9]. A = 3, since the attacker will not connect to DUA physically. R = 3, because limited resources are needed. T = 3, for SEMA since in this type the attacker study single EM. But in DEMA T = 2, because the attacker aims to study and analysis multiple EM, so more time and effort is expected.

Frequency Based Analysis (FBA) Attack

This type of attack is effective when EMA fails. This attack depends on frequency domain [2]. Instead of computing the differential signals in the time domain, this technique is performed in the frequency domain by calculating the differential Power Spectral Density (PSD) signal. The reasoning of analyzing signals in the frequency domain is that sometimes EM or power traces captured are temporally misaligned. As a result, Differential Electro-Magnetic Analysis (DEMA) or Differential Power Analysis (DPA) fails. This type of attack can be applied on both EM and power traces. For EM analysis, the attack is called the Differential EM Frequency Analysis (DEMFA). As for power analysis, the attack is called the Differential Power Frequency Analysis (DPFA) [36].

The measures for FBA are [Covert, (3, 3, 2), L_1= 8]. A = 3, since the attacker will not connect to Device Under Attack (DUA) physically. R = 3, because limited resources are needed and available for no price. T = 2, because this type of attack needs efforts to correlate the frequently spectrum.

Simple Power Analysis (SPA) Attack

This attack directly gives information about attacked device and can give also the secret key through direct analysis of the the power consumption got from visual representation, by measuring power consumption during computing occurred by cryptographic devices to get the operation used within the device and with knowledge of cryptanalysis techniques the attacker can get the secret key of encryption [37]. A computing device's power consumption depends on its current activity. The consumption depends on changes of the state of its components. So, by connecting a small resistor in series with the power, and from the voltage difference across that resistor we get the current [21].

The measures for SPA are [Covert, (2, 3, 3), L_1= 8]. A = 2, since the attacker needs to establish a communication with the device. R = 3, because limited resources are needed and available for no price. T = 3, for SPA since in this type the attacker gain the information directly with a minimum effort.

Fault Injection (FIT) Attack

Fault injection (glitch) attack inserts unexpected signals into the device to affect its normal operation. Usually this change targets the power supply or clock signals [35]. The injected signal could be implemented using EM pulse.

The measures for FIT are [Overt, (2, 3, 3), L_1= 8]. A = 2, since the attacker needs to communicate with the device; R = 3, because limited resources are needed; T = 3, because this type of attack needs only basic knowledge of hardware devices.

A HRA has an L_1-norm that satisfies the following inequality

$$8 \leq L_1 \leq 9. \tag{2.2}$$

2.2.2 Medium Risk Attacks (MRA)

Medium risk attacks require capabilities beyond those for a HRA, but less than for a LRA. Attacks belonging to this level typically require physical access to the system or higher permission to access the system than for a high risk attack. For example, the attacker has access to the chip surface but not to the internal circuitry. The attacker may need more time (e.g. to collect data and analysis it), and more resources compared to that for high risk attacks. Attacks belonging to this level cannot be accomplished without sufficient time, resources, and accessibility, which makes them harder to execute than high risk attacks. Therefore, the number of attackers with the required resources and experience will be smaller than for high risk attacks. Examples of MRA are:

Differential Power Analysis (DPA) Attack

This attack uses a statical model over a large number of power analyses to find the correct secret key. By measuring power consumption during operation of cryptographic devices, and knowing the cryptanalysis techniques, secret key information can be gathered [37]. A computing device's activity can be inferred by measuring the power supply to a chip [21].

The measures for DPA are [Covert, (2, 3, 2), L_1= 7]. A = 2, since the attacker needs to establish a communication with the device. R = 3, because limited resources are needed and available for no price. T = 2, because this type of attack requires complex analysis.

Timing (TA) Attack

Operations in a semiconductor chip can take a different time to complete and so the security operation. This time depend on the values of the secret key and input data. Carefully measuring and analyzing this time allows us to get back the secret key. This idea was first introduced in [5]. This type of attack was successfully performed on an actual smart card implementation of the RSA signature [6]. This type of attack depends on the performance of the system during some operation. Simply, the attacker collects a set of different messages with their processing times. Some encryption algorithms were attacked by this technique because of the software implementation of the algorithm.

The measures for TA are [Covert, (2, 3, 2), L_1= 7]. A = 2, since the attacker needs to establish a communication with the device. R = 3, because limited resources are needed. T = 2, because the attacker needs to study and analyze the system and its operations.

Acoustic (ACA) Attack

It is one of the oldest types of attacks. In the 1950s, British intelligence officer listened to the resetting key wheels of Egyptian encryption machines to deduce their starting positions, which enabled them to crack the encryption [18]. Recently, this type of attack has been studied to determine the text printed by dot matrix printers [15], and to determine which key on a keyboard was struck [16]. It was also used in microelectronic systems [17]. A microphone was placed close to a PC to determine when RSA encryption is running, and if it is running with different keys. Using ceramic capacitors on motherboard for power supply filtering and Alternating Current (AC) to Direct Current (DC) conversion, acoustic emissions are produced. Power supply current draw can be deduced from these emissions. Power analysis can then be performed but in low-pass filtered form.

The measures for ACA are [Covert, (2, 3, 2), L_1= 7]. A = 2, since the attacker needs to establish a communication with the device and add some equipments close to

attacked device. R = 3, because limited resources are needed. T = 2, because attacker needs to study and analysis the system signals, operations, and convert signals.

Optically Enhanced Position-Locked Power Analysis (OPLP) Attack

This is an innovative technique that allows the current through an individual transistor to become visible in the IC power trace [19]. Power consumption is measured for the entire chip, so the power for any part in the chip is also effected by other parts. (i.e. measuring the power for the processing data part will affect the power trace). Also, the power of fluctuations are affected by the number of bits changing their values, rather than the actual value of the manipulated data [35]. Its possible to observe the logic state of any transistor and activity of any part of memory cell, by targeting a laser on the area of interest in chip surface.

The measures for OPLP are [Covert, (2, 3, 2), L_1= 7]. A = 2, since the attacker needs to connect to the device physically. R = 3, because the attacker needs limited resources. T = 2, because this type of attack needs more knowledge in hardware layout and takes more time to analysis.

Optical Emanation (OEA) Attack

The average luminosity of a cathode ray tube diffuse reflection can be used to reconstruct the signal displayed on the CRT [25]. Also, Light Emitting Diode (LED) status indicators on data communication equipments, under a certain conditions, emit an optical signal that is significantly correlated with information being processed. The attacker gains access to all data going through the device, including plaintext in the case of data encryption systems [26].

The measures for OEA are [Covert, (3, 2, 2), L_1= 7]. A = 3, since the attacker will not connect to the DUA physically. R = 2, because specialized resources are needed. T = 2, since the attacker should have some experience to deal with and analyze the data from the emanations.

Covert JTAG Port (C-JTAG) Attack

JTAG attacks are classified under covert attack when the port is used to sniff secret data, and read-out secret. Sniff secret data means that the attacker's goal is to learn a secret that is being sent to a victim chip via JTAG, while, in read-out secret the attacker's goal is to learn a secret that is contained in the victim chip [38].

The measures for C-JTAG are [Covert, (2, 3, 2), L_1= 7]. A = 2, since the attacker needs to establish a communication with the device. R = 3, because this attack needs limited resources to accomplish. T = 2, because this type of attack needs more knowledge in hardware layout and takes more time and effort.

Data Remanence (DRA) Attack

Secret key is usually stored in SRAM in security processor. Once a device is tampered with, power is removed. At temperatures below -20° C, the contents of SRAM can be preserved even when power is removed for a certain period of time. Tampering event is triggered when the temperature falls below the low temperature threshold or rises above the high temperatures threshold [27].
There is a period of time in which the SRAM device will retain data once the power has been removed, which is a security problem. Data remanence attack also affects other types of memory like Dynamic Random Access Memory (DRAM), Ultraviolet Erasable Programmable Read Only Memory (UV EPROM), Electrically Erasable Programmable Read-Only Memory (EEPROM), and Flash [28]. This means that we can extract some information from memory that has been erased.

The measures for DRA are [Covert, (2, 3, 2), L_1= 5]. A = 2, since the attacker needs to connect to hardware physically. R = 3, since limited resources are needed. T = 2, since attacker should have enough knowledge to work with memories and that analysis takes time.

Fault Analysis (FAT) Attack

This type of attack feeds faulty inputs to a cryptographic device, then analyzes the resulting faulty outputs. The faulty inputs can be accidental or intentional to discover the key [2]. The response of intentional fault is studied using Simple Fault Analysis (SFA), or Deferential Fault Analysis (DFA). In SFA, one can use few faults to recover the secret key, while in DFA one will get the key bit by bit during the analysis for each fault [40]. This technique is practical for small circuits but is less practical for complicated circuits. Fault analysis attack is typically involved since the targeted system is sequential.

The measures for FAT are [Overt, (2, 3, 2). L_1= 7]. A = 2, since the attacker needs to communicate with the device; R = 3, because limited resources are needed; T = 2, because the attacker needs to do a lot of calculation and be experienced in

cryptographic algorithms and hardware devices.

Overt JTAG Port (O-JTAG) Attack

A JTAG attack is classified as overt when it is used to obtain test vectors and responses, modify state of authentic part, return false responses to test, or forcing Test Mode Select (TMS) and Test Clock (TCK) signals. The attack to obtain test vectors and responses requires two attack chips. One attack chip is placed upstream of the victim chip in order to send test vectors to the victim chip. The other attack chip is placed downstream of the victim chip in order to collect the responses as they propagate from the device under test back to the tester. In the authentic part state modification attack, the attacker's goal is to modify the victim chip state. The attacker controls the TMS and TCK lines of the victim chip. The attack chip is placed upstream of the victim chip on the same JTAG chain. In return false responses to test attack, the attacker's goal is to deceive the tester about the victim chips true state. In forcing TMS and TCK attack, the attacker must hijack the TMS and TCK lines to change the voltage seen by the victim's TMS and TCK input pins [38].

The measures for O-JTAG are [Overt, (2, 3, 2), L_1= 7]. A = 2, since the attacker needs to communicate with the device; R = 3, because this attack needs limited resources to accomplish; T = 2, because this type of attack needs more knowledge in hardware layout and takes a lot of time and effort to accomplish.

Advanced Imaging Techniques (AIT) Attack

Silicon become transparent to reflected or transmitted Infrared Radiation (IR) with $\lambda > 1100$ nm. Backside imaging is used to locate failed transistors or interconnects to navigation during FIB work [35]. Also, Laser radiation can ionize semiconductor regions if its photon energy exceeds the semiconductor band gab (> 1.1 eV). Laser radiation with $\lambda = 1.06\,\mu$m (1.17 eV) has a penetration depth of about $700\,\mu$m and provides good spatial ionization uniformity for silicon devices. In active photon probing, a scanning beam interacts with an IC [96].
There are two major laser scanned techniques which can be used for hardware security analysis. One is called Optical Beam Induced Current (OBIC) and is applied to an unbiased chip to find the active-doped areas on its surface [29]. The other, is called Light-Induced Voltage Alteration (LIVA), is applied to chip under operation [39].

The measures for AIT are [Covert, (2, 2, 2), L_1= 6]. A = 2, since the attacker

needs to scan the device. R = 2, because the attacker will use advanced imaging scanning device to perform this attack. T = 2, because this type of attack needs more knowledge in hardware layout and takes more time and effort to accomplish.

A MRA has an L_1-norm that satisfies the following inequality

$$5 \leq L_1 \leq 7. \tag{2.3}$$

2.2.3 Low Risk Attacks (LRA)

Low risk attacks require significant knowledge, equipment and/or time to succeed. Modern chips are multilayer and complex, so an attack that requires decapsulating a chip to access its internal components is very difficult. This type of attack requires full access to the chip, so they are typically not covert. Thus, the number of attackers with these capabilities will be much lower than for the other levels. Attacks belonging to this level can only be executed by research agencies, governments, organizations, or universities. Examples of LRA are:

Microprobing (MICRO) Attack

This type of attack is done by using a microprobe station. It consists of five elements: microscope, stage, device test, socket, micro-manipulators, and probe tips. The chip is normally placed in a test socket that provides all the necessary signals and is controlled by a computer. A probe tip can be either passive or active. A passive tip can be used for both eavesdropping and injecting signals, but it has low impedance and capacitance because it is normally connected directly to an oscilloscope. For that reason, it cannot be used for probing internal signals on the chip, but can be used for bus lines which are usually buffered. Also, a passive tip can be used to make connections to the bonding pads on a fully decapsulated chip. On the other hand, active tips offer high bandwidth with low loading capacitance and high input resistance [35].

The measures for MICRO are [Overt, (1, 2, 1), L_1= 4]. A = 1, since the attacker needs to get full access over the chip; R = 2, because here the attacker needs only micro probing station and limited man-power; T = 1, because this type of attack needs more experience in hardware layout and takes more time and huge effort to accomplish.

Reverse Engineering (RE) Attack

Reverse engineering for ASIC or custom IC means extracting information about the location of all transistors and interconnections. To gain this information, the attacker progressively removes the layers that formed during fabrication. Finally, by processing all the acquired information, a standard netlist file can be created and used to simulate the device. The same idea is used with FPGA, if the attacker manages to extract the configuration bitstream file from the attacked device, he will need to spend more time and effort to convert it into the logic equations and primitive blocks for further simulation and analysis, but there are some companies which do such work as a standard service [41].

The measures for RE are [Overt, (1, 1, 1), L_1= 3]. A = 1, since the attacker needs to get full access over the chip; R = 1, because here the attacker needs to reach the lowest level in hardware architecture; T = 1, because this attack needs more experience in hardware layout and takes a lot more time and huge effort to accomplish.

Deprocessing (DEP) Attack

Deprocessing refers to the opposite process of chip fabrication. There are many layers in standard Complementary Metal Oxide Semiconductor (CMOS) chip, the upper layers are usually made from aluminum and they are connected with each other through vias.

There are three methods of deprocessing: wet chemical etching, dry chemical etching, and mechanical polishing [42]. In wet chemical etching, each layer is removed by using specific chemicals. In contrast, dry chemical etching uses free radicals created from gas inside a vacuum chamber. But mechanical polishing is performed with the use of abrasive materials, the advantage of this process over dry and wet etching is that it is non selective. Thus, it is able to remove layer by layer and view features in the area of interest within the same plane [35].

The measures for DEP are [Overt, (1, 1, 1), L_1= 3]. A = 1, since the attacker needs to get full access over the chip; R = 1, because a lot of equipments and manpower are needed in this type of attack; T = 1, because this type of attack needs more experience in hardware layout and takes a lot more time and huge effort to accomplish. A LRA

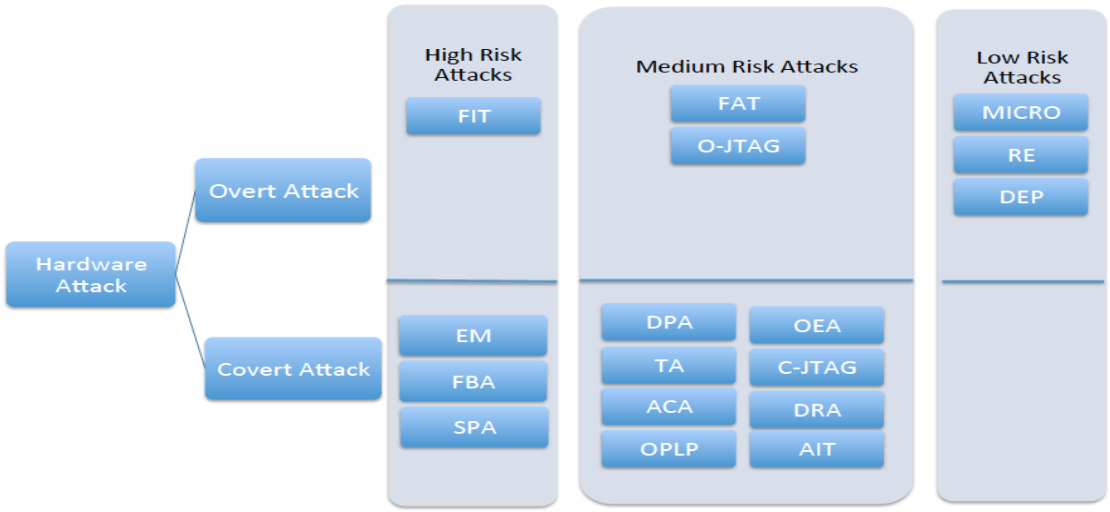


Figure 2.3: Hardware attacks

has an L_1-norm that satisfies the following inequality

$$3 \leq L_1 \leq 4. \tag{2.4}$$

This inequality can be expressed in the general form:

$$\text{RL} = \begin{cases} \text{HRA} & \text{when } L_1 + \Delta \geq 8 \\ \text{MRA} & \text{when } 5 \leq L_1 + \Delta \leq 7 \\ \text{LRA} & \text{when } L_1 + \Delta \leq 4 \end{cases} \tag{2.5}$$

where $\Delta \in \{\text{-1, 0, 1}\}$.

2.3 Algebraic Approach to Hardware Attacks

The proposed algebraic approach to hardware attacks is based on the hardware attack classification shown in Figure 2.1. The numbers in parentheses next to each criterion is the corresponding index, and Q is the associated risk. There are several steps in the methodology for both an attacker and a defender. These steps are described in this section.

Table 2.1: Hardware Attack Table

		Criteria											
		Awareness		Accessibility			Resources			Time			
		Covert	Overt	Limited Access	Partial Access	Full Access	Limited Resources	Moderate Resources	Excessive Resources	Short Time	Medium Time	Long Time	
	index i	1	2	3	4	5	6	7	8	9	10	11	
	risk Q_i	–	–	3	2	1	3	2	1	3	2	1	W_R
Hardware Attacks	SEMA	x		W			W			W			
	DEMA	x		W			W				W		
	FBA	x		W			W				W		
	SPA	x			W		W			W			
	FIT		x		W		W			W			
	DPA	x			W		W				W		
	TA	x			W		W				W		
	ACA	x			W		W				W		
	OPLP	x			W		W				W		
	OEA	x		W				W			W		
	C-JTAG	x			W		W				W		
	DRA	x			W		W				W		
	FAT		x		W		W				W		
	O-JTAG		x		W		W				W		
	AIT	x			W			W			W		
	MICRO		x			W		W				W	
	RE		x			W			W			W	
	DEP		x			W			W			W	
	W_C	–	–										
	N^*	–	–										

$*$ N is the number of attacks a criterion is involved in.

2.3.1 Hardware Attack Table

Developing a hardware attack table is the first step in the proposed approach. This table is updated by an attacker or defender whenever there is a new attack or a new criteria, or if there are changes in capabilities. It contains weights based on the attacks and associated criteria. Table 2.1 includes examples of hardware attacks that have been proposed in the literature.

Criteria Weights

Consider a system that may be vulnerable to the attacks given in Section 2.2 as shown in Figure 2.3. The risk levels in this figure (based on Figure 2.2), are employed with a weight W_i for each criteria in Table 2.1. For a given attack, the weight assigned by an attacker or defender satisfies

$$0 \leq W_i(Attack) \leq 1, \tag{2.6}$$

where i is the criterion index. In Table 2.1, an empty element corresponds to a weight of 0, which indicates that the criterion for the given attack is impossible or

secure. A weight $W_i = 1$ indicates that the criterion for the given attack is available or unsecured. For simplicity, $W_i = 1$ is assumed for all criteria that can affect the system to demonstrate the methodology.

The weighted risk for an attack is based on the risk for the criteria (Q_i) shown in Figure 2.1 and the corresponding weights W_i

$$W_R(Attack) = \sum_{i=1}^{n} W_i(Attack) \times Q_i, \tag{2.7}$$

where n is the number of criteria. If an attacker cannot satisfy one of the criteria for an attack (weight is zero), the weighted risk is set to 0. The range of W_R is

$$0 \leq W_R(Attack) \leq L_1. \tag{2.8}$$

Weighted Criteria

The weighted criterion is given by

$$W_C(criterion) = \sum_{j=1}^{m} W_j(Attack) \times Q_j, \tag{2.9}$$

where m is the number of attacks considered.

Definition 1. *The criteria with the largest value of W_C based on (2.9) are called the critical weighted criteria*

$$\widehat{W_C} = \max_{1 \leq i \leq n} W_C(i), \tag{2.10}$$

where i is the criterion index.

Attacker Table

An attacker determines the attack weights W_i based on their capabilities and the target system. These weights reflect the ability to satisfy a criterion for a given attack. Using (2.7), the weighted risk is obtained and entered in the W_R column. It is important for an attacker to know for which attacks $W_R \neq 0$, as these can be used against the target system. The total weight for each criterion from (2.9) is listed in the W_C row. A goal of an attacker is to increase the criteria weights, particularly the weight of the critical weighted criteria. The best attacks can be considered to

be those which have the largest value of W_R and include a critical weighted criterion $\widehat{W_C}$.

Defender Table

A defender determines the attack weights W_i based on their system and capabilities. These weights reflect the capacity to defend against an attack which requires a given criterion. Using (2.7), the weighted risk can be obtained and this is entered in the W_R column. It is important for a defender to know for which attacks $W_R \neq 0$, as these can be used against their system. A goal of a defender is $W_R = 0$ for all attacks to guarantee the security of the system (which is typically not achievable). The total weight for each criterion from (2.9) is listed in the W_C row. From a defender perspective, countermeasures should be developed to reduce the criteria weights W_C, particularly the weights for the critical weighted criteria $\widehat{W_C}$.

Attack Subsets

In Figure 2.1, hardware attacks are classified according to four properties. Each attack then has a combination of four risk values based on these properties. For simplicity, here we do not assign weights for the awareness property. An attacker may be able to undertake multiple attacks depending on their capabilities. For example, if an attacker can launch attacks that require partial access to a system, then they can also launch attacks that need only limited access. Conversely, if a security designer succeeds in protecting a system from partial access attacks, it can still be vulnerable to limited access attacks.

Definition 2. *The ability of an attacker or defender is a point in the 3D ART space which defines their capability to attack or defend a system, respectively, and is given by*

$$\mathbf{p_0} = (a_0, r_0, t_0). \tag{2.11}$$

The ability is now used to generate subsets of hardware attacks.

Definition 3. *Attacker coverage* $\mathbf{p_A}$*: the set of criteria levels that an attacker satisfies, defined as*

$$\mathbf{p_A} = \{a_A, r_A, t_A\}, \tag{2.12}$$

where

$$\begin{aligned} a_0 &\leq a_A \leq 3, \\ r_0 &\leq r_A \leq 3, \\ t_0 &\leq t_A \leq 3. \end{aligned}$$

Definition 4. *Defender coverage* $\mathbf{p_D}$*: the set of criteria levels that a defender has protection against, defined as*

$$\mathbf{p_D} = \{a_D, r_D, t_D\}, \tag{2.13}$$

where

$$\begin{aligned} 1 &\leq a_D \leq a_o, \\ 1 &\leq r_D \leq r_o, \\ 1 &\leq t_D \leq t_o. \end{aligned}$$

2.3.2 Adjacency Matrix for Attack Properties

We now examine the relationships between the attack criteria using an adjacency matrix. This matrix characterizes the connections between pairs of criteria, and thus shows the sets of attacks that have a pair of criteria in common. It will be used to determine the collective criteria and critical criteria, which are important for an attacker (resp. defender) to attack (resp. protect) a system. We begin with the following definitions.

Definition 5. *One weight criterion set* $X(i)$*: the subset of hardware attacks which contain criterion* i*, given by*

$$X(i) = \{Attack | W_i(Attack) > 0\}. \tag{2.14}$$

Definition 6. *Two weight criteria set* $X(i, j)$*: the subset of hardware attacks that contain criteria* i *and* j*, given by*

$$X(i, j) = \{Attack | W_i(Attack) \cdot W_j(Attack) > 0\}. \tag{2.15}$$

Definition 7. *Three weight criteria set* $X(i, j, k)$*: the subset of hardware attacks that*

contain criteria i, j, and k, given by

$$X(i,j,k) = \{Attack | W_i(Attack) \cdot W_j(Attack) \cdot W_k(Attack) > 0\}. \tag{2.16}$$

Assuming there are n criteria, the *adjacency matrix* $\mathbf{R}$ is a binary $(0-1)$ square, symmetric $n \times n$ matrix where $r(i,j) = r(j,i) = 1$ indicates that there is a subset $X(i,j)$ of hardware attacks that contain criteria i and j.

$$\mathbf{R} = \left[\begin{array}{c|ccccccc} A & 1 & 2 & 3 & 4 & 5 & \cdots & n \\ \hline 1 & 0 & 0 & 1 & 1 & 0 & \cdots & 0 \\ 2 & 0 & 0 & 0 & 1 & 1 & \cdots & 1 \\ 3 & 1 & 0 & 0 & 0 & 0 & \cdots & \vdots \\ 4 & 1 & 1 & 0 & 0 & 0 & \cdots & \vdots \\ 5 & 0 & 1 & 0 & 0 & 0 & \cdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ n & 0 & 1 & \cdots & \cdots & \cdots & \cdots & 0 \end{array}\right]$$

The adjacency matrix corresponding to the hardware attacks in Table 2.1 is

$$\mathbf{R_1} = \left[\begin{array}{c|ccccccccccc} A & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \\ \hline 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 2 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 3 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 4 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 5 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 6 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 7 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 8 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 9 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 10 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 11 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \end{array}\right]$$

where

$$r(i,j) = r(j,i) = 1 \Rightarrow X(i) \cap X(j) \neq \emptyset. \tag{2.17}$$

In $\mathbf{R_1}, r(5,8) \equiv r(\text{Full Access, Excessive Resources}) = 1$ indicates that a subset of hardware attacks require the criteria full access and excessive resources. From Table 2.1, this subset is $X(5,8) = \{\text{RE, DEP}\}$.

Row Entries in R

Assume there are v entries $r(i,j) = 1$ in row i. The relationship among the subsets of criteria in a single row is described by

$$\forall i : \ r(i,j) = 1 \ \Rightarrow X(i) \cap X(j) \neq \emptyset \tag{2.18}$$

Combining any $0 < k \leq v$ combinations of these entries will generate an attack subset. Thus, the number of subsets is

$$\sum_{k=1}^{v} \binom{v}{k} = 2^v - 1. \tag{2.19}$$

For example, consider row 3 in $\mathbf{R_1}$. There are $v = 5$ non zero entries corresponding to $j = 1, 6, 7, 9, 10$, which gives the subsets

$$\begin{aligned} X(3,1) &= \{SEMA, DEMA, FBA\} \\ X(3,6) &= \{SEMA, DEMA, FBA\} \\ X(3,7) &= \{OEA\} \\ X(3,9) &= \{SEMA\} \\ X(3,10) &= \{DEMA, FBA, OEA\} \end{aligned}$$

From (2.19), there are 31 possible subsets.

The subset of hardware attacks that satisfy at least one criteria in addition to criterion i is

$$X^{\cup}(i) = \bigcup_{r(i,j)\neq 0} X(i,j). \tag{2.20}$$

As an example, suppose that the subset of hardware attacks is required that satisfies one or more of criteria 6 and 7 as well as criterion 3. Using (2.20) gives

$$\begin{aligned} X^{\cup}(3) &= \bigcup_{j\in\{6,7\}} X(3,j) \\ &= X(3,6) \cup X(3,7) \\ &= \{SEMA, DEMA, FBA\} \cup \{OEA\} \\ &= \{SEMA, DEMA, FBA, OEA\} \end{aligned}$$

Conversely, the subset of hardware attacks that have all of a set of criteria including criterion i is given by

$$X^{\cap}(i) = \bigcap_{r(i,j)\neq 0} X(i,j). \tag{2.21}$$

As an example, suppose the subset of hardware attacks is required that satisfies both criteria 6 and 9 as well as criterion 3. Using (2.21) gives

$$\begin{aligned} X^{\cap}(3) &= \bigcap_{j\in\{6,9\}} X(3,j) \\ &= X(3,6) \cap X(3,9) \\ &= \{SEMA, DEMA, FBA\} \cap \{SEMA\} \\ &= \{SEMA\} \end{aligned}$$

Since **R** is a square, symmetric matrix, the same relationships between the criteria subsets can be obtained using the columns instead of the rows.

Definition 8. *Collective Criteria ($C(i)$): the number of criteria that can be combined with criterion i to produce a subset of hardware attacks, which is given by*

$$C(i) = \sum_{j=1}^{n} r(i,j). \tag{2.22}$$

Definition 9. *Critical criterion ($\hat{i}$): a criterion that can be combined with the maximum number of criteria to produce subsets of hardware attacks, which is given by*

$$\hat{i} = \max_{1\leq i\leq n} C(i). \tag{2.23}$$

The values of (2.22) for the example are given in Table 2.2, and show that the range of $C(i)$ is

$$3 \leq C(i) \leq 7. \tag{2.24}$$

From (2.24), $\hat{i} = 7$, so that moderate resources (criterion 7) and medium time (criterion 10) are the critical criteria.

Table 2.2: Collective Criteria

i	$C(i)$
1	6
2	6
3	5
4	6
5	5
6	5
7	7
8	3
9	5
10	7
11	3

2.4 Algorithms

The purpose of this section is to present algorithms to identify sets of candidate attacks based on the attacker/defender table. Three attack algorithms are proposed. These algorithms have the same steps from line 1 to line 3 and from line 5 to line 16.

For a given target system, on line 2 the ability point $\mathbf{p_0} = (a_0, r_0, t_0)$ and awareness are inputs. For example, suppose $\mathbf{p_0} = (2, 2, 2)$ and covert are inputs. Then on line 3, Table 2.1 is updated with new attacks or changes since the table was last modified. The weights W are obtained according to (2.6). For simplicity, $W = 1$ is assumed in all cases which indicates that the attacker is able to provide all criteria needed for the attacks. For the defender, this would indicate that the system is vulnerable to numerous attacks. Lines 5 to 16 calculate the attack coverage using (2.12) based on the corresponding ability. For the example, the attacker coverage is {(2, 2, 2), (2, 2, 3), (2, 3, 3), (3, 3, 3)}. Each point corresponds to a set of hardware attacks, namely $\mathbf{p} = (2, 2, 2) = \{\text{AIT}\}$, $\mathbf{p} = (2, 2, 3) = \emptyset$, $\mathbf{p} = (2, 3, 3) = \{\text{SPA}\}$, and $\mathbf{p} = (3, 3, 3) = \{\text{SEMA}\}$. Note that FIT is not included as it is an overt attack. The algorithms then determine a set of attacks based on the ability and requirements.

2.4.1 Attacks based on Criteria Relationships

Algorithm 1 is based on the relations between all criteria involved in the hardware attacks. It is used to generate attacks based on a set of one to three preferred criteria, i.e. criteria the attacker is considering to launch an attack. These criteria

Table 2.3: Attacker/Defender Table

		Criteria											
		Awareness		Accessibility			Resources			Time			
		Covert	Overt	Limited Access	Partial Access	Full Access	Limited Resources	Moderate Resources	Excessive Resources	Short Time	Medium Time	Long Time	
	index i	1	2	3	4	5	6	7	8	9	10	11	
	Q_i	–	–	3	2	1	3	2	1	3	2	1	W_R
Hardware Attacks	SEMA	x		1			1			1			9
	DEMA	x		1			1				1		8
	FBA	x		1			1				1		8
	SPA	x			1		1			1			8
	FIT		x		1		1			1			8
	DPA	x			1		1				1		7
	TA	x			1		1				1		7
	ACA	x			1		1				1		7
	OPLP	x			1		1				1		7
	OEA	x		1				1			1		7
	C-JTAG	x			1		1				1		7
	DRA	x			1		1				1		7
	FAT		x		1		1				1		7
	O-JTAG		x		1		1				1		7
	AIT	x			1			1			1		6
	MICRO		x			1		1				1	4
	RE		x			1			1			1	3
	DEP		x			1			1			1	3
	W_C	–	–	12	22	3	39	6	2	9	24	3	
	N^*	–	–	4	11	3	13	3	2	3	12	3	

$*$ N is the number of attacks a criterion is involved in.

should belong to different categories. On line 17 in Algorithm 1, (2.22) is used to solve for the collective criteria, and (2.23) to obtain the critical criteria. From $\mathbf{R_1}$, the critical criteria are moderate resources (criteria 7) and medium time (criteria 10). On line 18, a subset of hardware attacks is obtained based on a critical criterion using (2.14). On line 19, two critical criteria are considered (if more than one exists), using (2.15), The subset for three criteria are obtained using (2.16) on line 20.

2.4.2 Attacks based on Selected Attack Criteria

Algorithm 2 is based on the preferred criteria and provides attacks which have combinations of these criteria. On line 17 in Algorithm 2, the hardware attack table (Table 2.3) is used to construct $\mathbf{R}$. For the example, $\mathbf{R_1}$ is obtained. This matrix provides the relationships between each pair of criteria. On line 18, the number of preferred criteria to launch an attack is selected, i.e. the value of v in (2.19). On line 19, the number of combinations of preferred criteria is selected, i.e. the value of k in (2.19). Then on line 20, (2.19) is used to determine the number of combinations X

Algorithm 1 Attacks based on Criteria Relations

1: Given: target system
2: Input: $\mathbf{p_0} = (a_0, r_0, t_0)$, Awareness
3: Update the hardware attack table
4: Initialize: $\mathbf{S_T} = \emptyset$
5: Initialize: $a = a_0, r = r_0, t = t_0$;
6: **while** $a \leq 3$ **do**
7: **while** $r \leq 3$ **do**
8: **while** $t \leq 3$ **do**
9: $\mathbf{S_P} = (a_A, r_A, t_A)$;
10: $\mathbf{S_T} \leftarrow \mathbf{S_T} \cup \mathbf{S_P}$;
11: $t = t + 1$;
12: **end while**
13: $r = r + 1$;
14: **end while**
15: $a = a + 1$;
16: **end while**
17: Solve for $C(i)$ and $\hat{i}$ using (2.22) and (2.23)
18: Output: one criterion attack set from (2.14)
19: Output: two criteria attack set from (2.15)
20: Output: three criteria attack set from (2.16)

based on v and k. On line 22, a subset of hardware attacks is generated for each value in $\{1, \ldots, X\}$. On line 23, the common attacks between the sets of attacks are obtained using (2.21), or they are combined using (2.20). Finally, line 26 generates the hardware attack set that matches the preferred criteria. This algorithm is used when hardware attacks are required based on one criterion, or when criteria are combined according to specific criteria.

2.4.3 Attacks based on Criteria Occurrence

Algorithm 3 selects attacks based on the criteria involved and their weights. Then the best attacks to use against a system are chosen. The highest value of $\hat{N}$ is selected on line 17 in Algorithm 3, the highest value of $\widehat{W_R}$ is selected on line 18, and the highest value of $\widehat{W_C}$ is selected on line 19. One or more of the equations on lines 20 to line 23 is used to generate a hardware attack set based on the critical criteria $\widehat{W_C}$, $\hat{N}$, or both. An attack set can also be chosen that contains $\widehat{W_R}$.

For example, consider a target system with $\mathbf{p_0} = (3, 2, 2)$ and covert as inputs, which means the attacker can have limited access, moderate resources, and moderate

Algorithm 2 Attacks based on Selected Attack Criteria

1: Given: target system
2: Input: $\mathbf{p_0} = (a_0, r_0, t_0)$, Awareness
3: Update the hardware attack table
4: Initialize: $\mathbf{S_T}, \mathbf{S_R}, \mathbf{S_X} = \emptyset, m, k, X = 0$;
5: Initialize: $a = a_0, r = r_0, t = t_0$;
6: **while** $a \leq 3$ **do**
7: **while** $r \leq 3$ **do**
8: **while** $t \leq 3$ **do**
9: $\mathbf{S_P} = (a_A, r_A, t_A)$;
10: $\mathbf{S_T} \leftarrow \mathbf{S_T} \cup \mathbf{S_P}$;
11: $t = t + 1$;
12: **end while**
13: $r = r + 1$;
14: **end while**
15: $a = a + 1$;
16: **end while**
17: Construct $\mathbf{R}$
18: v= number of favourable criteria (2.19);
19: k = number of combination criteria (2.19) ;
20: Solve for X using (2.19)
21: **while** $X \geq 1$ **do**
22: $\mathbf{S_R} = (i, j)$;
23: $\mathbf{S_X} = \mathbf{S_X} \cap / \cup \mathbf{S_R}$ [(2.20) or (2.21)]
24: $X = X - 1$;
25: **end while**
26: Output: Attack set $\mathbf{S_X}$

time. On line 3, Table 2.4 is generated. Note that some of the attack criteria differ from those in Table 2.3 because advanced measuring techniques are available, i.e. the accessibility for SPA is limited access. For illustration purposes, $W = 1$ is assumed to indicate that the attacker meets the criterion needed for an attack, and $W = 0$ to indicate that the criterion is not met. Lines 5 to 16 calculate the attack coverage using (2.12) based on the corresponding ability. The attacker coverage is {(3, 2, 2), (3, 2, 3), (3, 3, 2), (3, 3, 3)}. Each point corresponds to a set of hardware attacks, namely $\mathbf{p}$ = (3, 2, 2)= {OEA}, $\mathbf{p}$ = (3, 2, 3)= $\emptyset$, $\mathbf{p}$ = (3, 3, 2)= {DEMA, DPA, FBA, TA}, and $\mathbf{p}$ = (3, 3, 3)= {SEMA, SPA}. On line 17, the criteria involved in the greatest number of attacks is calculated, which is {limited resources, medium time} with a value of 10. On line 18, the highest weighted risk among the attacks is determined, which is 9 corresponding to {SEMA, SPA}. On line 19, the highest weighted criteria

Algorithm 3 Attacks based on Criteria Occurrence

1: Given: target system
2: Input: $\mathbf{p_0} = (a_0, r_0, t_0)$, Awareness
3: Update the hardware attack table
4: Initialize: $\mathbf{S_T}, \mathbf{S_X}, N, WR, WC = \emptyset, \hat{N}, \widehat{WR}, \widehat{WC} = 0$;
5: Initialize: $a = a_0, r = r_0, t = t_0$;
6: **while** $a \leq 3$ **do**
7: **while** $r \leq 3$ **do**
8: **while** $t \leq 3$ **do**
9: $\mathbf{S_P} = (a_A, r_A, t_A)$;
10: $\mathbf{S_T} \leftarrow \mathbf{S_T} \cup \mathbf{S_P}$;
11: $t = t + 1$;
12: **end while**
13: $r = r + 1$;
14: **end while**
15: $a = a + 1$;
16: **end while**
17: $\hat{N} = \max N$ from Table 2.1;
18: $\widehat{W_R} = \max W_R$ from Table 2.1;
19: Obtain $\widehat{W_C}$ using (2.10);
20: $\mathbf{S_X} = \widehat{W_R}$ from Table 2.1; OR
21: $\mathbf{S_X} = \hat{N} \cap \widehat{W_R}$ using (2.21) OR
22: $\mathbf{S_X} = \hat{N} \cup \widehat{W_C}$ using (2.20) OR
23: $\mathbf{S_X} = \hat{N} \cup \widehat{W_C} \cap \widehat{W_R}$ using (2.20) and (2.21)
24: Output: Attack set $\mathbf{S_X}$

is calculated, which is 30 corresponding to {limited resources}. Lines 20 to 23 provide the hardware attack set. If line 20 is chosen, the set is {SEMA, SPA}, if line 21 is chosen, the set is {SEMA, SPA}, if line 22 is chosen, the set is {SEMA, DEMA, FBA, SPA, DPA, TA}, and if line 23 is chosen, the set is {SEMA, SPA}. The algorithm can be executed multiple times to obtain different attack sets and also whenever criteria change.

2.4.4 Defence Algorithms

The three attack algorithms can also be used by a defender. An attacker uses the output attack sets to launch an attack against a system, while a defender uses the output sets to develop countermeasures to protect their system against these attacks. The algorithms can also be used to examine the system by modifying lines 5 to 16,

Table 2.4: Attacker Table

		Criteria											
		Awareness		Accessibility			Resources			Time			
		Covert	Overt	Limited Access	Partial Access	Full Access	Limited Resources	Moderate Resources	Excessive Resources	Short Time	Medium Time	Long Time	
	index i	1	2	3	4	5	6	7	8	9	10	11	
	Q_i	–	–	3	2	1	3	2	1	3	2	1	W_R
Hardware Attacks	SEMA	x		1			1			1			9
	DEMA	x		1			1				1		8
	FBA	x		1			1				1		8
	SPA	x		1			1			1			9
	DPA	x		1			1				1		8
	TA	x		1			1				1		8
	OEA	x		1				1			1		7
	ACA	x			0		1				1		0
	OPLP	x			0		1				1		0
	C-JTAG	x			0		1				1		0
	DRA	x			0		1				1		0
	AIT	x			0			1			1		0
	W_C	–	–	21	0	0	30	4	0	6	20	0	
	N^*	–	–	7	0	0	10	2	0	2	10	0	

$*$ N is the number of attacks a criterion is involved in.

as shown in Algorithm 4. The modified algorithms allow the defender to determine the hardware attacks that their capabilities can protect against. Further, they aid the defender in examining their system against new attacks or changes in attack criteria. The defender should consider all possible approaches an attacker may use to launch an attack, so variations of the same attack may exist in the defender table. For example, there could be two DEP attacks, say DEP-1 and DEP-2, where DEP-1 assumes that the attacker uses in-house resources, while DEP-2 assumes the attacker using outsourcing and so requires fewer resources.

For example, consider a target system with $\mathbf{p_0} = (2, 2, 2)$ and covert as inputs. This indicates the security of the system prevents against attacks with partial access, moderate resources, and medium time. The defender employs Algorithm 4 to retrieve the set of attacks that can threaten their system. The defence table obtained is given in Table 2.5. This shows that some attacks can be executed at different accessibility levels. Thus some attacks are duplicated with different criteria, i.e. SPA-1 with limited access and SPA-2 with partial access depending on the measuring technique employed by an attacker. For illustration purposes, $W = 1$ is assumed to indicate that a criterion is not secure, and $W = 0$ to indicate that a criterion is secure. Lines 5 to 16 provide the defender coverage which is {(2, 2, 2), (2, 2, 1), (2, 1, 2), (2, 1, 1), (1, 2, 2), (1, 2, 1), (1, 1, 2), (1, 1, 1)}. Each point corresponds to a set of hardware attacks

Algorithm 4 Defence based on Criteria Occurrence

1: Given: target system
2: Input: $\mathbf{p_0} = (a_0, r_0, t_0)$, Awareness
3: Update the hardware attack table
4: Initialize: $\mathbf{S_T}, \mathbf{S_X}, N, WR, WC = \emptyset, \hat{N}, \widehat{WR}, \widehat{WC} = 0$;
5: Initialize: $a = a_0, r = r_0, t = t_0$;
6: **while** $a \geq 1$ **do**
7: **while** $r \geq 1$ **do**
8: **while** $t \geq 1$ **do**
9: $\mathbf{S_P} = (a_A, r_A, t_A)$;
10: $\mathbf{S_T} \leftarrow \mathbf{S_T} \cup \mathbf{S_P}$;
11: $t = t - 1$;
12: **end while**
13: $r = r - 1$;
14: **end while**
15: $a = a - 1$;
16: **end while**
17: $\hat{N} = \max N$ from Table 2.1;
18: $\widehat{W_R} = \max WR$ from Table 2.1;
19: Obtain $\widehat{W_C}$ using (2.10);
20: $\mathbf{S_X} = \widehat{W_R}$ from Table 2.1; OR
21: $\mathbf{S_X} = \hat{N} \cap \widehat{W_R}$ using (2.21) OR
22: $\mathbf{S_X} = \hat{N} \cup \widehat{W_C}$ using (2.20) OR
23: $\mathbf{S_X} = \hat{N} \cup \widehat{W_C} \cap \widehat{W_R}$ using (2.20) and (2.21)
24: Output: Attack set $\mathbf{S_X}$

for which the system is protected, namely $\mathbf{p}$ = (2, 2, 2)= {AIT}, $\mathbf{p}$ = (2, 2, 1)= $\emptyset$, $\mathbf{p}$ = (2, 1, 2)= $\emptyset$, $\mathbf{p}$ = (2, 1, 1)= $\emptyset$, $\mathbf{p}$ = (1, 2, 2)= $\emptyset$, $\mathbf{p}$ = (1, 2, 1)= {MICRO, DEP-2}, $\mathbf{p}$ = (1, 1, 2)= $\emptyset$, and $\mathbf{p}$ = (1, 1, 1)= {RE, DEP-1}. On line 17, the criteria involved in the greatest number of attacks is calculated, which is {limited resources} with a value of 16. On line 18, the highest weighted risk among the attacks is determined, which is 9 corresponding to {SEMA, SPA-1, TA-1}. On line 19, the highest weighted criteria is calculated, which is 48 corresponding to {limited resources}. Lines 20 to 23 provide the hardware attack set. If line 20 is chosen, the set is {SEMA, SPA-1, TA-1}, if line 21 is chosen, the set is {SEMA, SPA-1, TA-1}, if line 22 is chosen, the set {SEMA, SPA-1, TA-1}, and if line 23 is chosen, the set is {SEMA, SPA-1, TA-1}. This indicates that the system is vulnerable to these three attacks. The defender must consider developing countermeasures to these attacks, and the common criterion can be considered as the best approach to achieving this goal. The defender can also

Table 2.5: Defender Table

		Criteria											
		Awareness		Accessibility			Resources			Time			
		Covert	Overt	Limited Access	Partial Access	Full Access	Limited Resources	Moderate Resources	Excessive Resources	Short Time	Medium Time	Long Time	
	index i	1	2	3	4	5	6	7	8	9	10	11	
	Q_i	–	–	3	2	1	3	2	1	3	2	1	W_R
Hardware Attacks	SEMA	x		1			1			1			9
	DEMA	x		1			1				0		0
	FBA	x		1			1				0		0
	SPA-1	x		1			1			1			9
	SPA-2	x			0		1			1			0
	FIT		x		0		1			1			0
	DPA-1	x		1			1				0		0
	DPA-2	x			0		1				0		0
	TA-1	x		1			1			1			9
	TA-2	x		1			1				0		0
	TA-3	x			0		1			1			0
	ACA	x			0		1				0		0
	OPLP	x			0		1				0		0
	OEA	x		1				0			0		0
	C-JTAG	x			0		1				0		0
	DRA	x			0		1				0		0
	FAT		x		0		1				0		0
	O-JTAG		x		0		1				0		0
	AIT	x			0			0			0		0
	MICRO		x			0		0				0	0
	RE		x			0			0			0	0
	DEP-1		x			0			0			0	0
	DEP-2		x			0		0				0	0
	W_C	–	–	24	0	0	48	0	0	12	0	0	
	N^*	–	–	8	0	0	16	0	0	4	0	0	

$*$ N is the number of attacks a criterion is involved in.

execute the other algorithms to examine the system from different perspectives.

Chapter 3

Systematic Analysis of Hardware Trojans

This chapter examines hardware trojan threats for semiconductor chips intended for critical infrastructure and applications. The phases of the chip production life-cycle are reviewed and opportunities for trojan insertion are examined. Trojans are classified using a comprehensive attribute taxonomy based on eight categories. A matrix identifying the causal relationships between these attributes is defined. This matrix is used to characterize hardware trojans from both the attacker and defender perspectives.

The proposed methodology is flexible and can be used with any hardware trojan classification. Further, new attributes based on technology or chip manufacturing developments can easily be accommodated. As with any circuit, a hardware trojan goes through several production phases as it becomes embedded into the target system. Therefore, studying the life cycle along with other attributes will provide a better insight into the insertion phase, functionality, logic type, physical characteristics, and location of a trojan. The approach presented in this chapter has two major advantages over existing results. First, the relationships between the hardware trojan attributes are clearly defined based on a comprehensive taxonomy with eight categories. Second, the production life-cycle of a chip is used to determine how and where a trojan can be inserted into a chip. The relationships between the attributes can be used to identify missing trojan attributes. This information can be employed to improve chip security during manufacturing, develop trojan detection techniques, and assess chips when covert attacks occur.

The remainder of this chapter is organized as follows. Section 3.1 reviews the chip production life-cycle. Section 3.2 provides a comprehensive hardware trojan taxonomy based on eight categories: insertion, abstraction, effect, logic type, functionality, activation, physical layout, and location. An algebraic approach to investigating hardware trojans is given in Section 3.3. Finally, two case studies are presented in Section 3.4.

3.1 Chip Life-cycle

The vulnerability of ICs to hardware trojan attacks is increasing due to design outsourcing, overseas fabrication and increasing reliance on third-party Intellectual Property (IP). In addition, hardware attacks can originate from the use of unverified design automation tools. Figure 3.1 shows the modern IC production life-cycle phases: ***specification***, ***design***, ***fabrication*** (interfacing, masking, wafer fabrication and assembly, and packaging), ***testing*** and ***deployment*** [55–57, 96].

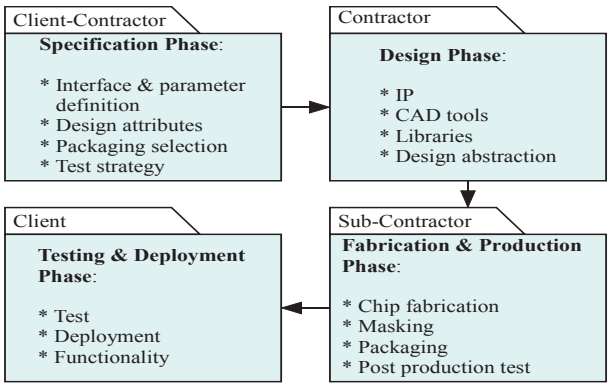


Figure 3.1: The integrated circuit (IC) design life-cycle phases.

Complete protection of a chip during its life-cycle is only possible if it is produced by a trusted source. The stages in the production process can be classified as ***trusted***, ***untrusted***, or ***either*** [55]. This is based on the level of control over a given stage. A ***trusted*** stage indicates that the designer has complete control, whereas an ***untrusted*** stage means that the designer has only limited control. ***Either*** indicates that control may be complete or limited. This classification assumes that the specification, testing and assembly phases are trusted. However, the increased complexity of chip design and fabrication requires state-of-the-art infrastructure and very

specialized processes at the production facilities. Often the *design* and *fabrication* must be carried out simultaneously, requiring support for multiple capabilities and mixed technologies. This creates significant economic challenges in meeting production quality and demand which has caused increased outsourcing to contractors and subcontractors during the design and manufacturing phases. Unfortunately, outsourcing significantly reduces trust [58]. For example, outsourcing the design phase reduces trust in the fabrication phase. Further, test vectors can be modified during the design phase, which leads to untrusted testing. Even if the testing phase is trusted, it does not imply that the assembly phase is trusted. This can only be ensured if it is done under client control.

There is typically a high level of control over the ***specification phase*** since the client and contractor must work closely to define the system blocks, design specifications, packaging, and test requirements. These are reviewed and validated to confirm the target die size and fabrication process [59]. However, deficiencies in the specification or incomplete validation can lead to vulnerabilities.

There is less control over the ***design phase*** primarily because many manufacturers outsource their designs. This makes standard cell design abstractions, design tools, design kits and design libraries untrusted [97]. The ***fabrication and production phase*** of an IC typically involves hundreds of steps. Computer-Aided Design (CAD) tools are used to define the doping, metallization and glass region masks. These masks are then used to transfer the design onto a silicon wafer. This process is repeated to embed the masks for the IC layers on the silicon. Each layer is tested for functionality and defective chips are discarded after the wafer is cut into individual ICs. The chips are then bonded to a mounting package and contact leads are attached. The mounting package is encapsulated with a plastic coating for protection and identifying part numbers and other data are added. This is followed by the ***testing and deployment phase*** in which ICs undergo final testing before they are distributed [97].

If the production life-cycle is not completely controlled by the designer, reverse engineering, malicious circuit insertion/modification, and IP piracy are possible. Of particular concern is an untrusted IC production facility where hardware trojans can be inserted into chips unknown to the designer.

3.2 Trojan Taxonomy

The growing application of semiconductor chips in critical applications such as military infrastructure, industrial automation, Supervisory Control And Data Acquisition (SCADA) systems, and the smart grid, has made security a serious concern. Tampering during the IC life-cycle can cause it to respond in an unexpected manner or worse in a manner determined by an attacker. A *hardware trojan* is a malicious component embedded in an integrated circuit which causes abnormal behaviour [95]. Hardware trojans can be implemented in microprocessors, microcontrollers, network and digital signal processors, Field Programmable Gate Array (FPGAs), Application Specific Integrated Circuits (ASICs), and other integrated circuits.

Any attempt to address the concern over hardware trojans should begin with an examination based on the processes involved in the IC production life-cycle. However, it is extremely challenging to characterize, classify, and detect hardware trojans. Trojans have previously been classified by their *physical* characteristics, by the *effects* they cause to a system, and by their *activation* mechanisms [35,54]. These approaches are based on the assumption that the trojans are inserted only during *fabrication* [51]. Typically the specification and design phases are considered trusted [55, 58], but here the vulnerability of these phases is also considered. In addition, a comprehensive model for trojan classification is presented which is based on eight key categories: insertion (*when was it inserted?*), abstraction (*where was it inserted?*), effect (*what can it do?*), logic type (*what logic type does it employ?*), functionality (*what functionality does it have?*), activation (*what activates it?*), physical layout (*how does it appear?*), and location (*where is it located?*). This classification is illustrated in Figure 3.2.

The **insertion** category represents the hardware manufacturing process. This process has many levels and each level is vulnerable to malicious insertions that can cause abnormal functionality [49, 95]. Modifications can be as early as the *specification phase* where parameters such as the size, structure, type, intended function, power, timing and delay can be targeted. Design constraints and deficiencies can be exploited by an attacker. Protocols and functions can also be modified to insert trojans [49, 95]. Alterations to the specification lead to changes in the design which result in the production of a modified IC. In the *design phase*, factors concerning the logical, functional, timing and physical constraints of realizing the design on the target hardware technology are considered. In this phase, designers typically use different IP blocks, design tools, and standard cell libraries and templates. These can

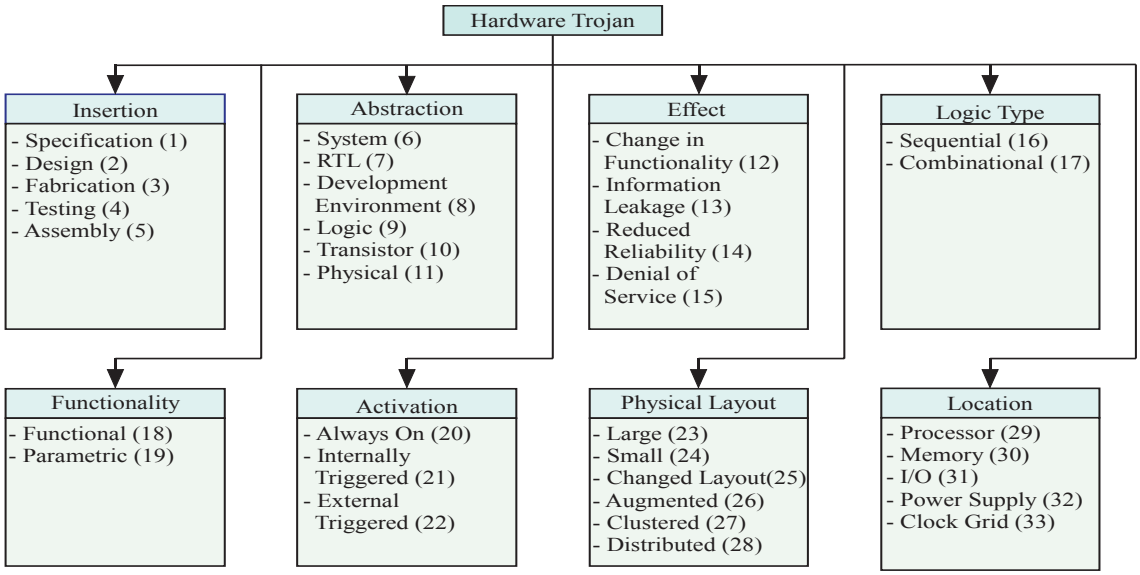


Figure 3.2: The hardware trojan taxonomy.

contain malicious components if they originate from untrusted or commercial off-the-shelf sources. The *fabrication phase* involves preparation of the silicon wafers and processes such as masking, photolithography, doping, etching and interconnections to complete the manufacturing and packaging. An attacker can introduce trojans by altering parameters in these processes, modifying the mask geometry and layout, or changing chemical compositions.

In the *testing phase*, hardware trojans can be inserted, or trojans inserted earlier can be concealed from detection. Editing during the design and prototype stages can result in large blocks of unused circuitry from previous iterations. This latent circuitry can be exploited by attackers to reroute signals and trigger trojans [49,60,97]. Testing is a critical phase since it is the easiest phase to detect trojans. However, testing at an untrusted facility can lead to test vectors being revealed and detection deficiencies [35]. An attacker can exploit this information to omit or mask test vectors that will reveal hardware trojans. Trojans can also be inserted which will return acceptable values for test vectors [51].

In the *assembly phase*, hardware components are interfaced on the chip. Every interface between components is a possible trojan insertion point. At an interface, improper termination or improper shielding against phenomena such as electromagnetic coupling makes the chip susceptible to exploitation by an attacker. For example, deficiencies can be identified to inject faults or gather sensitive information.

The **abstraction** category represents the level at which a hardware trojan is introduced [49,95]. The sophistication required to inject a trojan at a given abstraction level varies depending on the level.

A trojan can be introduced in the *system level* by altering interconnections, hardware modules, or communication protocols. This does not require a high level of sophistication. At the *Register Transfer Level (RTL)*, signals, registers and boolean functions are described in the function modules. Numerous trojan insertion opportunities exist in this level since an attacker can gain control over the hardware functionality [49, 51]. The *development environment* of an IC involves the simulation, verification, validation and synthesis. Software can be inserted into this abstraction level through CAD tools and scripts to hide the effects of hardware trojans [61]. The *logic (gate level)* is also prone to trojan insertion, but it is considered relatively secure since tampering requires a high level of sophistication. Logic design alteration is possible at the gate level or in the netlist [60].

The *transistor* level provides attackers with the opportunity to control circuit parameters such as power and timing. Trojans can be inserted by resizing or deleting existing transistors, or inserting new transistors, to modify the circuit functionality and characteristics. Modifications at the *physical* level involve the transistors and/or layout and are typically achieved by changing circuit parameters that affect the reliability and/or functionality.

The **effect** category describes the disruption or effect a trojan can have on a chip. These effects are: *change in functionality, information leakage, reduced reliability, and Denial of Service (DoS)*. A *change in functionality* is caused by trojans that introduce new logic or bypass existing logic to produce unexpected results. This can also be achieved by deleting logic [35]. A trojan can cause *information leakage* through a covert or existing channel. These channels may be radio frequency based or via JTAG or RS232 interfaces, and provide backdoor access to assist in compromising the chip. Information such as encryption keys can also be leaked through thermal or optical patterns created by the hardware [3, 60]. Hardware trojans can also cause *reduced reliability* by altering interface, functional or circuit characteristics such as path delay and power consumption. Increased power consumption may cause the ambient temperature of the circuit to rise above normal operating levels and/or cause faster battery depletion. Finally, a *DoS* trojan modifies device parameters to exhaust on-board resources such as power or memory, or introduces computational delays to degrade performance or create malfunctions.

The **logic type** category is based on the circuit logic that triggers the trojan. A *combinational* trojan uses a particular logic value at one or more circuit locations as the trigger. A *sequential* trojan is triggered by a sequence of conditions after a given period of operation [62, 82].

The **functionality** category is based on whether hardware trojans are *functional* or *parametric*. A *functional* trojan introduces a change in the functionality of the device [63]. A *parametric* trojan exploits the parametric effects of the device circuitry such as power consumption and thermal and delay profiles [64]. This is achieved by weakening transistors, modifying the length and/or thickness of wires, or changing physical geometries [81].

The **activation** category is based on whether a trojan is always on or triggered. An *always on* trojan is always active. A triggered trojan will cause some unintended or malicious effect only when activated [60]. The trigger can be either *external* or *internal*. An *externally triggered* trojan is activated via an external signal received by an antenna or sensor that interacts with the outside world. An attacker can then be at a geographically distant location. An *internally triggered* trojan waits for an internal condition which can be a sequence of one or more events that occur in the system. This condition is typically an internal logic state or a pattern of input/output signals [35].

The **physical layout** category is based on the physical characteristics of hardware trojans. Hardware trojans have been developed with various sizes and layouts to evade detection. Some trojans are very *small* and thus virtually undetectable by inspection of the power consumption or heat dissipation [65]. Other trojans are *large* and often employ unused circuitry in the device to avoid detection. The size of trojans is determined by the number of deleted, added or compromised components in the chip. The distribution of a trojan is based on the layout of the trojan on the chip. A *clustered* trojan has a topology in which the components are close to each other. A *distributed* trojan has a sporadic topology and can be dispersed throughout the chip. Some trojans employ a *changed layout* structure where an existing layout is modified. If the layout is added to it is known as an *augmented* circuit [81].

The **location** category is based on where the trojan is located. As mentioned above, trojans can be *distributed* or *clustered* among the IC components. They can be located in components such as the *processor, memory, input/output, power supply, or clock grid*. A trojan injected into the *processor* can be located in the logical units so that it can change instruction or execution cycles. Trojans in the *memory* units or

interfaces can create incorrect addresses, modify memory contents, or enable/disable read/write instructions. The *input/output* peripherals interface with external devices through communication and data buses such as serial ports. Trojans located here can modify the data or alter the way external devices communicate with IC components. Trojans in the *power supply* can create effects such as denial of service or reduced reliability. This is achieved by varying the current and/or voltage supply to the chip to cause malfunctions or abnormal behaviour. Finally, a trojan in the *clock grid* can cause variations in the clock frequency, or skip or freeze the clock signals supplied to chip modules [51].

Table 3.1 presents the hardware trojan attributes considered in existing taxonomies and those in the proposed taxonomy. This shows that the proposed classification illustrated in Figure 3.2 is more comprehensive that those given in [51–54].

3.3 Algebraic Approach to Hardware Trojan Attributes

In this section, an algebraic approach to hardware trojan attributes is presented. Figure 3.2 provides a comprehensive classification of these attributes. There are many connections between these attributes which indicate their relationships. The attributes belong to one of four trojan levels: chip life-cycle, abstraction, properties, and location, as shown in Figure 3.3.

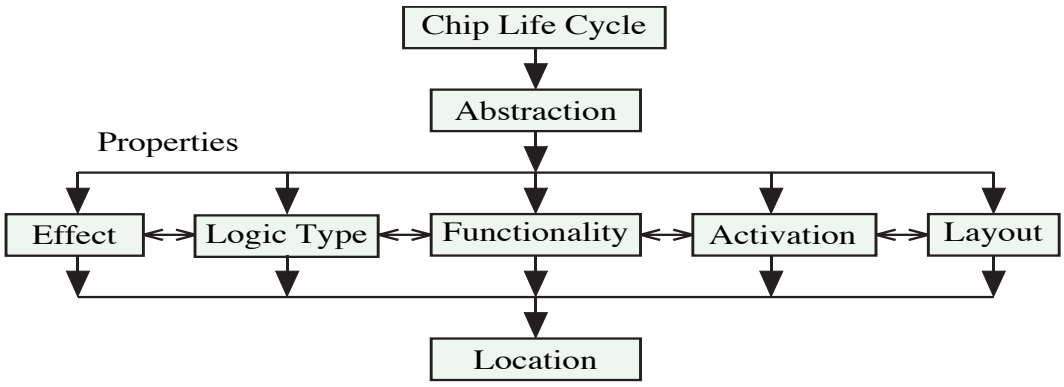


Figure 3.3: The four hardware trojan levels.

The chip life-cycle level in Figure 3.3 is equivalent to the insertion category in Figure 3.2. In this level, the attacker decides at which phase of the chip life-cycle

Table 3.1: Comparison of Hardware Trojan Taxonomies

k	Attribute	[53]	[52]	[54]	[51]	Proposed
1	Specification				✓	✓
2	Design				✓	✓
3	Fabrication				✓	✓
4	Testing				✓	✓
5	Assembly				✓	✓
6	System				✓	✓
7	RTL				✓	✓
8	Development Environment				✓	✓
9	Logic				✓	✓
10	Transistor				✓	✓
11	Physical				✓	✓
12	Change in Functionality		✓	✓	✓	✓
13	Information Leakage			✓	✓	✓
14	Reduced Reliability		✓	✓	✓	✓
15	Denial of Service				✓	✓
16	Sequential	✓				✓
17	Combinational	✓				✓
18	Functional		✓	✓		✓
19	Parametric		✓	✓		✓
20	Always On	✓	✓		✓	✓
21	Internally Triggered	✓	✓	✓	✓	✓
22	Externally Triggered	✓	✓	✓	✓	✓
23	Large		✓	✓		✓
24	Small		✓	✓		✓
25	Changed Layout			✓		✓
26	Augmented			✓		✓
27	Clustered		✓	✓		✓
28	Distributed		✓	✓		✓
29	Processor				✓	✓
30	Memory				✓	✓
31	I/O				✓	✓
32	Power Supply				✓	✓
33	Clock Grid				✓	✓

the trojan will be inserted. According to this decision, the abstraction of the trojan insertion will be determined. Therefore, the abstraction category in Figure 3.2 is the same as the abstraction level in Figure 3.3. The trojan properties level contains the properties a trojan may have according to the chip life-cycle and abstraction levels. The possible location for a trojan is based on the chip life-cycle, abstraction, and properties levels. Each trojan has a particular combination of attributes which define its characteristics. These are represented using an $n \times n$ matrix defined as

$$\mathbf{R} = \begin{bmatrix} R_1 & R_{12} & 0 & 0 \\ 0 & R_2 & R_{23} & 0 \\ 0 & 0 & R_3 & R_{34} \\ 0 & 0 & 0 & R_4 \end{bmatrix}$$

where n is the number of attributes.

3.3.1 Hardware Trojan Matrix

The hardware trojan matrix $\mathbf{R}$ has four square submatrices: $\mathbf{R_1}$, $\mathbf{R_2}$, $\mathbf{R_3}$, and $\mathbf{R_4}$. These matrices represent the levels in Figure 3.3. The transfer matrices $\mathbf{R_{12}}$, $\mathbf{R_{23}}$, and $\mathbf{R_{34}}$ represent the connections between the levels.

Single Element

In the matrix $\mathbf{R}$, $r(i,j) = 1$ indicates that attribute i can lead to attribute j

$$\forall i,j: \quad r(i,j) = 1 \ \Rightarrow \text{attribute}(i) \rightarrow \text{attribute}(j). \tag{3.1}$$

By definition $r(i,i) = 0$.

Row

In the matrix $\mathbf{R}$, $r(i,j) = 1$ in row i indicates that attribute i can lead to these attributes in column j

$$\forall i: \quad r(i,j) = 1 \ \Rightarrow \text{attribute}(i) \rightarrow \text{attribute}(j) \tag{3.2}$$

Column

In the matrix $\mathbf{R}$, $r(i,j) = 1$ in column j indicates the attributes that can lead to attribute j

$$\forall j: \quad r(i,j) = 1 \Rightarrow \text{attribute}(i) \rightarrow \text{attribute}(j) \tag{3.3}$$

Fan In

The fan in is the number of connections to an attribute from other attributes. This indicates how many other attributes can lead to this attribute. The fan in can be expressed as

$$F_{in}(j) = \sum_{i=1}^{n} r(i,j). \tag{3.4}$$

Fan Out

The fan out is the number of connections from an attribute to other attributes. This indicates how many attributes this attribute can lead to. The fan out can be expressed as

$$F_{out}(i) = \sum_{j=i}^{n} r(i,j). \tag{3.5}$$

Root Attribute

An attribute with $F_{in} = 0$ is called a root attribute. It is an attribute that is not the result of any other attribute so that

$$F_{in}(j) = \sum_{i=1}^{n} r(i,j) = 0. \tag{3.6}$$

Terminal Attribute

A terminal attribute is an attribute with $F_{out} = 0$. It is the location of a trojan inserted in a system. It does not lead to any attributes so that

$$F_{out}(i) = \sum_{j=i}^{n} r(i,j) = 0. \tag{3.7}$$

Intermediate Attribute

An intermediate attribute has $F_{in} \neq 0$ and $F_{out} \neq 0$. Thus it is an attribute that can be a consequence of other attributes, and also lead to other attributes.

3.3.2 Submatrix $\mathbf{R_1}$

Submatrix $\mathbf{R_1}$ represents the trojan chip life-cycle level shown in Figure 3.3. It also shows the relationships between attributes in the insertion phase in Figure 3.2. These start with specification (attribute 1), and end with assembly (attribute 5). For example, $\mathbf{R_1}(1,2) \equiv \mathbf{R_1}(\text{Specification, Design}) = 1$ indicates that if there is an incomplete or incorrect specification, it can lead to an improper or defective design.

$$\mathbf{R_1} = \left[\begin{array}{c|ccccc} A & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0 & 1 & 0 & 0 & 0 \\ 2 & 0 & 0 & 1 & 0 & 0 \\ 3 & 0 & 0 & 0 & 1 & 0 \\ 4 & 0 & 0 & 0 & 0 & 1 \\ 5 & 0 & 0 & 0 & 0 & 0 \end{array}\right]$$

3.3.3 Submatrix $\mathbf{R_2}$

Submatrix $\mathbf{R_2}$ represents the trojan design level in Figure 3.3. It also shows the relationships between the attributes in the abstraction level in Figure 3.2. The abstraction level has six sublevels beginning with system (attribute 6) and ending with physical (attribute 11). A modification in a sublevel can propagate to the following sublevels. For example, $\mathbf{R_2}(6,7) \equiv \mathbf{R_2}(\text{System, RTL}) = 1$ indicates that if there is an alteration to the interconnections, hardware modules, or communication protocols, it can lead to the RTL signals, registers, or boolean functions being compromised.

$$\mathbf{R_2} = \left[\begin{array}{c|cccccc} A & 6 & 7 & 8 & 9 & 10 & 11 \\ \hline 6 & 0 & 1 & 0 & 0 & 0 & 0 \\ 7 & 0 & 0 & 1 & 0 & 0 & 0 \\ 8 & 0 & 0 & 0 & 1 & 0 & 0 \\ 9 & 0 & 0 & 0 & 0 & 1 & 0 \\ 10 & 0 & 0 & 0 & 0 & 0 & 1 \\ 11 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right]$$

3.3.4 Submatrix $\mathbf{R_{12}}$

Submatrix $\mathbf{R_{12}}$ describes the connections between $\mathbf{R_1}$ and $\mathbf{R_2}$. This shows the relationships between the insertion attributes and the abstraction attributes.
For example, $\mathbf{R_{12}}(2,7) \equiv \mathbf{R_{12}}(\text{Design, RTL}) = 1$ indicates that if there is an attempt to insert a trojan in the design level, a modification of the RTL can be used to accommodate this alteration.

It is clear that the development environment (attribute 8) and transistor (attribute 10) are not directly connected to the insertion phase, but both are indirectly connected through paths in $\mathbf{R}$. The system level (attribute 6) has the highest *fan in* as it is connected to specification (attribute 1), testing (attribute 4), and assembly (attribute 5) from the insertion category. These three attributes are connected to the system level as control over these is required at the abstraction category.

$$\mathbf{R_{12}} = \left[\begin{array}{c|cccccc} A & 6 & 7 & 8 & 9 & 10 & 11 \\ \hline 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 1 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 & 0 & 1 \\ 4 & 1 & 0 & 0 & 1 & 0 & 0 \\ 5 & 1 & 0 & 0 & 0 & 0 & 0 \end{array}\right]$$

3.3.5 Submatrix $\mathbf{R_3}$

Submatrix $\mathbf{R_3}$ represents the properties of a trojan given in Figure 3.3 and contains attributes 12 to 28. This shows the relationships between the effect, logic type, functionality, activation, and physical layout attributes. For example, $\mathbf{R_3}(14,19) \equiv \mathbf{R_3}(\text{Reduced Reliability, Parametric}) = 1$ indicates that if a trojan reduces chip reliability (i.e. quicker battery drain), it can lead to changes in the power consumption,

thermal and delay profiles.

$$\mathbf{R_3} = \left[\begin{array}{c|cccccccccccccccc} A & 12 & 13 & 14 & 15 & 16 & 17 & 18 & 19 & 20 & 21 & 22 & 23 & 24 & 25 & 26 & 27 & 28 \\ \hline 12 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 13 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\ 14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 15 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 16 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 17 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 18 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 19 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 20 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 21 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 22 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 23 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 24 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 25 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 26 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ 27 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 28 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \end{array}\right]$$

3.3.6 Submatrix $\mathbf{R_{23}}$

Submatrix $\mathbf{R_{23}}$ is the transfer matrix that describes the connections between $\mathbf{R_2}$ and $\mathbf{R_3}$. Thus it shows the connections between the abstraction attributes and the trojan property attributes.

$$\mathbf{R_{23}} = \left[\begin{array}{c|cccccccccccccccc} A & 12 & 13 & 14 & 15 & 16 & 17 & 18 & 19 & 20 & 21 & 22 & 23 & 24 & 25 & 26 & 27 & 28 \\ \hline 6 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 7 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 8 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 9 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 10 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 11 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \end{array}\right]$$

Single Element Example

$\mathbf{R_{23}}(7, 12) \equiv \mathbf{R_{23}}$(RTL, Change Function) = 1 indicates that if a trojan has been inserted in the RTL abstraction level, which changes the function description, it can also change the function behaviour.

Row Example

For row = 7, $\mathbf{R_{23}}(7, j) \equiv \mathbf{R_{23}}(\text{RTL}, j) = 1$: $\mathbf{R_{23}}(7, 12) \equiv \mathbf{R_{23}}$(RTL, Change Function), $\mathbf{R_{23}}(7, 15) \equiv \mathbf{R_{23}}$(RTL, DoS), $\mathbf{R_{23}}(7, 16) \equiv \mathbf{R_{23}}$(RTL, Sequential), $\mathbf{R_{23}}(7, 17) \equiv \mathbf{R_{23}}$(RTL, Combinational), $\mathbf{R_{23}}(7, 18) \equiv \mathbf{R_{23}}$(RTL, Functional),
$\mathbf{R_{23}}(7, 20) \equiv \mathbf{R_{23}}$(RTL, Always On), $\mathbf{R_{23}}(7, 21) \equiv \mathbf{R_{23}}$(RTL, Internally Triggered), $\mathbf{R_{23}}(7, 22) \equiv \mathbf{R_{23}}$(RTL, Externally Triggered), $\mathbf{R_{23}}(7, 23) \equiv \mathbf{R_{23}}$(RTL, Big), and

Table 3.2: Matrix $\mathbf{R_{23}}$ Attributes Fan In and Fan Out

Attribute	F_{in}	F_{out}	Attribute	F_{in}	F_{out}
6	-	6	18	6	-
7	-	10	19	3	-
8	-	14	20	6	-
9	-	8	21	3	-
10	-	9	22	3	-
11	-	12	23	3	-
12	6	-	24	4	-
13	2	-	25	2	-
14	2	-	26	3	-
15	4	-	27	3	-
16	3	-	28	2	-
17	4	-	-	-	-

$\mathbf{R_{23}}(7, 24) \equiv \mathbf{R_{23}}$(RTL, Small) = 1, gives the trojan property attributes that may result if a trojan is inserted in the RTL abstraction level.

Column Example

For column = 14, $\mathbf{R_{23}}(i, 14) \equiv \mathbf{R_{23}}$(i, Reduced Reliability) = 1: $\mathbf{R_{23}}(10, 14) \equiv \mathbf{R_{23}}$(Transistor, Reduce Reliability), and $\mathbf{R_{23}}(11, 14) \equiv \mathbf{R_{23}}$(Physical, Reduce Reliability) = 1 indicates that if a trojan reduces the reliability of a chip, it may be because of a modification in the transistor and/or physical abstraction levels.

Fan In and Fan Out Example

Table 3.2 shows the fan in and fan out for each attribute in $\mathbf{R_{23}}$. The maximum fan out is 14 for development environment (attribute 8). Thus if a CAD tool is exploited by an attacker, there can be many chip vulnerabilities. The maximum fan in is 6 for change functionality, functional, and always on (attributes 12, 18, and 20, respectively). This means that many trojans will have these attributes.

3.3.7 Submatrix $\mathbf{R_{34}}$

Submatrix $\mathbf{R_{34}}$ describes the connections between $\mathbf{R_3}$ and the trojan location category. Thus it shows the connections between the trojan properties and the hardware

locations. For example, $\mathbf{R_{34}}(15, 29) \equiv \mathbf{R_{34}}(\text{DoS, Processor}) = 1$ indicates that if a DoS trojan has been inserted, it may be located in the processor to prevent the chip from executing properly. It is clear from this matrix that all locations are almost equally vulnerable to trojans.

$$\mathbf{R_{34}} = \left[\begin{array}{c|ccccc} A & 29 & 30 & 31 & 32 & 33 \\ \hline 12 & 1 & 1 & 1 & 1 & 1 \\ 13 & 1 & 1 & 1 & 1 & 1 \\ 14 & 1 & 1 & 1 & 1 & 1 \\ 15 & 1 & 0 & 1 & 1 & 1 \\ 16 & 1 & 1 & 1 & 1 & 1 \\ 17 & 1 & 1 & 1 & 1 & 1 \\ 18 & 1 & 1 & 1 & 1 & 1 \\ 19 & 0 & 0 & 1 & 1 & 1 \\ 20 & 1 & 1 & 1 & 1 & 1 \\ 21 & 1 & 1 & 1 & 1 & 1 \\ 22 & 1 & 1 & 1 & 1 & 1 \\ 23 & 1 & 1 & 0 & 0 & 0 \\ 24 & 1 & 1 & 1 & 1 & 1 \\ 25 & 1 & 1 & 1 & 1 & 1 \\ 26 & 1 & 1 & 1 & 1 & 1 \\ 27 & 1 & 1 & 1 & 1 & 1 \\ 28 & 1 & 1 & 1 & 1 & 1 \end{array}\right]$$

3.3.8 Matrix R

Matrix $\mathbf{R}$ represents the complete set of attributes in Figure 3.2, and the hardware trojan levels in Figure 3.3. The characteristics of a trojan can be inferred from the paths in $\mathbf{R}$. Note that an empty submatrix indicates that all entries are zero. Submatrix $\mathbf{R_4} = 0$ since the location attributes are terminal attributes and are not interconnected.

Single Element Example

$\mathbf{R}(17, 29) \equiv \mathbf{R}(\text{Combinational, Processor}) = 1$ indicates that if there is a trojan inserted in a processor, it may be a combinational circuit.

Row Example

For row = 6, $\mathbf{R}(6, j) \equiv \mathbf{R}$(System, j) = 1: $\mathbf{R}(6, 7) \equiv \mathbf{R}$(System, RTL), $\mathbf{R}(6, 12) \equiv \mathbf{R}$(System, Change Function), $\mathbf{R}(6, 13) \equiv \mathbf{R}$(System, Leak Information), $\mathbf{R}(6, 15) \equiv \mathbf{R}$(System, DoS), $\mathbf{R}(6, 18) \equiv \mathbf{R}$(System, Functional), $\mathbf{R}(6, 19) \equiv \mathbf{R}$(System, Parametric), and $\mathbf{R}(6, 20) \equiv \mathbf{R}$(System, Always On) = 1, which implies that if a trojan is introduced at the system abstraction level, it may lead to a combination of RTL, changed function, leaked information, DoS, functional, parametric, or always on.

Column Example

For column = 6, $\mathbf{R}(i, 6) \equiv \mathbf{R}$(i, System) = 1: $\mathbf{R}(1, 6) \equiv \mathbf{R}$(Specification, System), $\mathbf{R}(4, 6) \equiv \mathbf{R}$(Testing, System), and $\mathbf{R}(5, 6) \equiv \mathbf{R}$(Assembly, System) = 1, which implies that if a trojan is inserted at the system abstraction level, it may be because of an incomplete or modified specification, control of the testing to evade trojan detection, or modification in the assembly phase.

Root Attribute Example

From (3.6), specification is a root attribute of any trojan as it is at the start of the chip life-cycle. Thus it does not depend on any other attributes.

Terminal Attribute Example

From (3.7), the location attributes are terminal attributes since they do not lead to other attributes.

Fan In and Fan Out Example

Table 3.3 shows the fan in and fan out for the attributes in $\mathbf{R}$. The maximum fan in of 19 occurs for functional, always on, and augmented (attributes 18, 20, and 26, respectively). This means that these three attributes occur most frequently due to other attributes. Adding an inverter gate to a chip to insert a trojan is a simple example which combines all of these attributes. The maximum fan out of 21 occurs only for augmented (attribute 26). This means that this attribute most frequently leads to other attributes.

Table 3.3, shows the sums of the fan in and fan out for the attributes in $\mathbf{R}$. These values reflect the connectivity of an attribute with other attributes. A critical

Table 3.3: Sum of the Fan In and Fan Out for the Attributes in **R**

Attribute	F_{in}	F_{out}	$F_{in} + F_{out}$	Attribute	F_{in}	F_{out}	$F_{in} + F_{out}$	Attribute	F_{in}	F_{out}	$F_{in} + F_{out}$
1	0	3	3	12	18	17	35	23	14	13	27
2	1	2	3	13	10	13	23	24	16	17	33
3	1	2	3	14	8	11	19	25	11	14	25
4	1	3	4	15	16	16	32	26	19	21	40
5	1	1	2	16	13	15	28	27	17	19	36
6	3	7	10	17	17	18	35	28	14	17	31
7	2	11	13	18	19	18	37	29	16	0	16
8	1	15	16	19	9	9	18	30	15	0	15
9	2	9	11	20	19	18	37	31	16	0	16
10	1	10	11	21	13	15	28	32	16	0	16
11	2	12	14	22	12	14	26	33	16	0	16

attribute is an attribute with the highest sum, since it has the greatest number of connections with other attributes. For the analysis presented here, augmented (attribute 26) is the critical attribute. Thus inserting a trojan without modifying the chip layout makes detecting it a very difficult task.

3.4 Case Studies

In this section, two examples of hardware trojan classification are presented based on the algebraic approach in Section 3.3.

3.4.1 Combinational Trojan

Assume that an attacker works as a design engineer in a chip manufacturing facility. He decides to use the trojan shown in Figure 3.4 to change the functionality of a chip when a particular event occurs. This is a combinational logic triggered trojan where $A = B = 0$ causes the output C to have an incorrect value (C modified). This circuit reflects an attacker inserting a trojan based on a rare event which is unlikely to be detected during the testing phase. From the figure, the trojan characteristics are change in functionality, combinational, functional, internally triggered, small, clustered, and augmented (attributes 12, 17, 18, 21, 24, 26, and 27 in $\mathbf{R_3}$). Examining the corresponding columns in $\mathbf{R_{23}}$, these attributes are all directly connected to the development environment in the abstraction category (attribute 8).

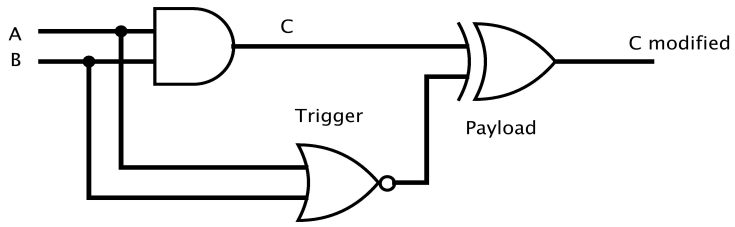


Figure 3.4: A combinational logic triggered trojan.

From $\mathbf{R_{12}}$, there is no direct connection between attribute 8 and an attribute in the insertion category (attributes 1 to 5). However, $\mathbf{R_2}$ provides a connection between the development environment (attribute 8) and RTL (attribute 7). Then attribute 7 is connected to design (attribute 2) and system (attribute 6). Attribute 6 is connected to specification, testing, and assembly (attributes 1, 4, and 5). In addition, $\mathbf{R_1}$ provides connections between the attributes in the insertion category. **R1** shows all candidates paths that can be used to embed this trojan. In **R1**, the light blue colour shows the horizontal connections, the light red colour shows the vertical connections, and the attribute connections are shown in dark red. Now assume that attributes 12, 17, 18, 21, 24, 26, and 27 have been identified for the trojan. These attributes all lie in $\mathbf{R_3}$. Moving forward (increasing attribute numbers), the expected trojan locations can be identified, and moving backward (decreasing attribute numbers), the untrusted attributes in the insertion and abstraction phases can be identified. It is clear that the row attributes (i.e. 12, 17, 18, 21, 24, 26, and 27), share can lead to all trojan locations (columns 29, 30, 31, 32, and 33). In **R1**, attributes 12, 17, 18, 21, 24, 26, and 27 all have value 1 in row 8 only. Thus a trojan with these seven attributes should be inserted using attribute 8 (development environment) in the abstraction phase. $\mathbf{R_{12}}$ can then be used to reach the life-cycle phase. Then using $\mathbf{R_1}$, the complete paths from the insertion phase to the location phase. These paths are represented by the directed graph in Figure 3.5, which provides a complete characterization of the hardware trojan in Figure 3.4. In this figure, S indicates a possible trojan insertion point.

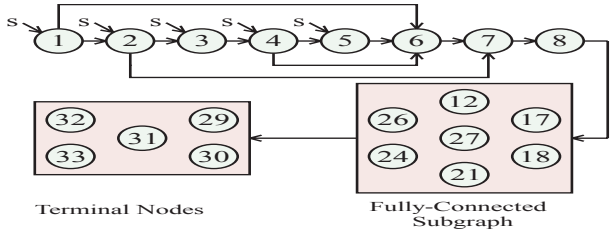


Figure 3.5: A directed graph characterization of the hardware trojan in Figure 3.4.

An attacker will choose a particular point in the chip life-cycle for insertion. Assuming the design phase is selected, the graph shown in Figure 3.6 represents the inserted trojan. Conversely, a defender has no information about the attacker decisions, so it necessary to consider all paths in Figure 3.5. However, if the specification and assembly phases are trusted, then the graph in Figure 3.5 will reduce to the graph in Figure 3.7. Then if a defender is able to verify that attribute 6 is not a trojan characteristic, the trojan in Figure 3.6 is identified exactly.

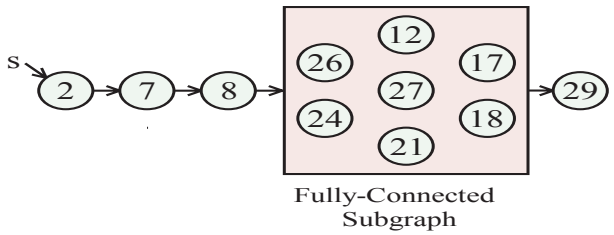


Figure 3.6: The trojan graph assuming it is inserted in the design phase.

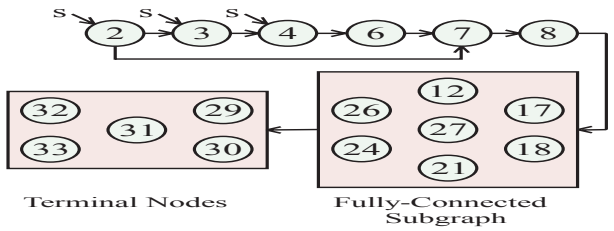
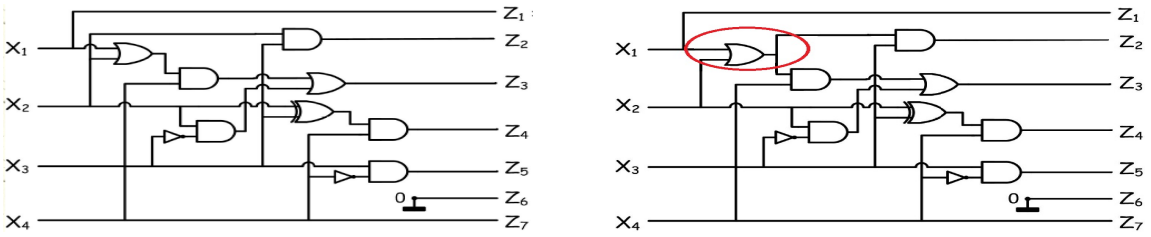


Figure 3.7: The defender graph assuming the trojan is inserted by the attacker during the chip life cycle.

According to this discussion, a defender can detect such a trojan using hardware trojan detection techniques [69, 70, 78].

3.4.2 Backdoor Trojan

Consider a chip manufacturer that designs a circuit to compute a function $F(X)$ for a system to authenticate user-password pairs X and $F(X)$. The design specifies ten users I_0 to I_9 with $F(X) = X^2$. Since there are ten users, the designer uses four bits, X_1 to X_4, to encode them. Since $F(X) = X^2$, the largest function output is 81, so seven bits are required, Z_1 to Z_7. The resulting circuit is shown in Figure 3.8a. Testing using these ten input values shows that the outputs are correct and so the function works properly.



(a) Trojan free circuit

(b) Circuit with a backdoor trojan

Figure 3.8: The circuits with and without the trojan.

With four bits, there are 16 input combinations, but in this case the six inputs 10 to 15 are not used so they are don't care conditions. With this incomplete specification, the designer asserts that the circuit performs the function $F(X) = X^2$ correctly. If an attacker makes the small modification indicated in Figure 3.8b, the output for the ten valid inputs is the same as with the original circuit. However, it also provides two additional correct outputs for inputs 10 and 11. Therefore, the modified circuit contains a backdoor trojan.

Table 3.4 shows the outputs from both circuits for all possible inputs. With an input of 10, circuit (a) produces 68, which is not the square of 10, while an input of 11 produces 89. Thus, circuit (a) provides outputs for the don't care conditions which are not the correct values for $F(X)$. However, circuit (b) outputs the correct values of $F(X)$ for inputs 10 and 11, so the user-password pairs $(10, 100)$ and $(11, 121)$ are valid. Thus, circuit (b) contains a back door trojan. It can be used to allow users who do not have permission to access the system. Note that both circuits have the same type and number of gates, the only difference is a wire connection. Further, the layouts are almost identical.

Table 3.4: The Outputs for the Circuits in Figure 3.8

		Inputs					Circuit (a)								Circuit (b)							
		X_1	X_2	X_3	X_4	X	Z_1	Z_2	Z_3	Z_4	Z_5	Z_6	Z_7	$F(X)$	Z_1	Z_2	Z_3	Z_4	Z_5	Z_6	Z_7	$F(X)$
Inputs	I_0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	I_1	0	0	0	1	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	1
	I_2	0	0	1	0	2	0	0	0	0	1	0	0	4	0	0	0	0	1	0	0	4
	I_3	0	0	1	1	3	0	0	0	1	0	0	1	9	0	0	0	1	0	0	1	9
	I_4	0	1	0	0	4	0	0	1	0	0	0	0	16	0	0	1	0	0	0	0	16
	I_5	0	1	0	1	5	0	0	1	1	0	0	1	25	0	0	1	1	0	0	1	25
	I_6	0	1	1	0	6	0	1	0	0	1	0	0	36	0	1	0	0	1	0	0	36
	I_7	0	1	1	1	7	0	1	1	0	0	0	1	49	0	1	1	0	0	0	1	49
	I_8	1	0	0	0	8	1	0	0	0	0	0	0	64	1	0	0	0	0	0	0	64
	I_9	1	0	0	1	9	1	0	1	0	0	0	1	81	1	0	1	0	0	0	1	81
Undefined	I_{10}	1	0	1	0	10	1	0	0	0	1	0	0	68	1	1	0	0	1	0	0	100
	I_{11}	1	0	1	1	11	1	0	1	1	0	0	1	89	1	1	1	1	0	0	1	121
	I_{12}	1	1	0	0	12	1	1	1	0	0	0	0	112	1	0	1	0	0	0	0	80
	I_{13}	1	1	0	1	13	1	1	1	1	0	0	1	121	1	0	1	1	0	0	1	89
	I_{14}	1	1	1	0	14	1	1	0	0	1	0	0	100	1	1	0	0	1	0	0	100
	I_{15}	1	1	1	1	15	1	1	1	0	0	0	1	113	1	1	1	0	0	0	1	113

Assuming the trojan in Figure 3.8b is used to gain control of the system to create a DoS attack, the characteristics are DoS, combinational logic, functional, and externally triggered. (attributes 15, 17, 18, and 22 in $\mathbf{R_3}$). Examining the corresponding columns in $\mathbf{R_{23}}$, this combination of attributes can occur if the trojan is inserted at the RTL, development environment, or logic type in the abstraction phase (attributes 7, 8, or 9). If this attack occurred due to an incomplete specification, the attacker needs to control the testing so that the trojan is not discovered. $\mathbf{R_2}$ provides a connection between logic type (attribute 9) and development environment (attribute 8), and between development environment (attribute 8) and RTL (attribute 7).

There are two steps the attacker takes. First, the trojan is injected during the design phase, and second, the testing phase is controlled to evade detection during testing. From $\mathbf{R_{12}}$, attribute 7 is connected to design (attribute 2) system (attribute 6), and attribute 6 is connected to specification, testing, and assembly (attributes 1, 4, and 5). Attribute 9 is directly connected to testing (attribute 4), as the trojan must evade detection. Incomplete specification as attribute 1 leads to this trojan, so $\mathbf{R_1}$ provides the connection between design (attribute 2) and specification (attribute 1). These paths are represented by the directed graph in Figure 3.9, which is a complete characterization of the hardware trojan in Figure 3.8b. In this figure, S indicates a

possible trojan insertion point.

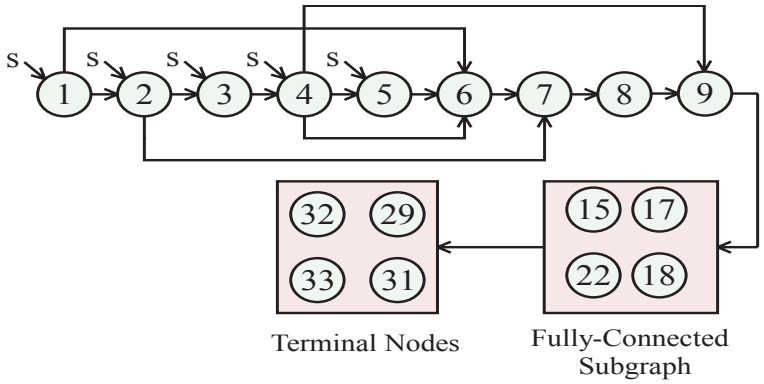


Figure 3.9: A directed graph characterization of the hardware trojan in Figure 3.8b.

From an attacker respective, he will choose a particular point in the chip life-cycle for insertion. Assuming the design or testing phase is chosen, the graph shown in Figure 3.10 represents the trojan. Conversely, a defender has no information about the attacker decisions, so it necessary to check all possible paths in Figure 3.9.

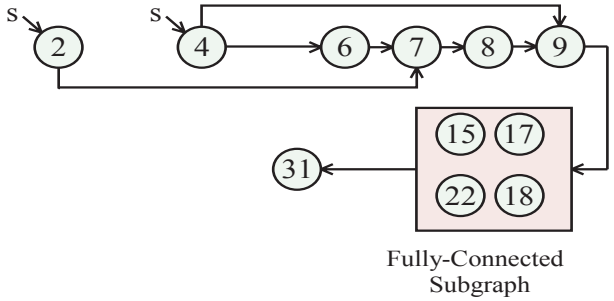


Figure 3.10: The trojan graph assuming it is inserted by the attacker during the design or testing phase.

According to this discussion, a defender can detect such a trojan using hardware trojan detection techniques [86].

$\mathbf{R} =$

A	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
1	0	1	0	0	0	1	0	0	0	0	0																						
2	0	0	1	0	0	0	1	0	0	0	0																						
3	0	0	0	1	0	0	0	0	0	0	1																						
4	0	0	0	0	1	1	0	0	1	0	0																						
5	0	0	0	0	0	1	0	0	0	0	0																						
6						0	1	0	0	0	0	1	1	0	1	0	0	1	1	1	0	0	0	0	0	0	0	0					
7						0	0	1	0	0	0	1	0	0	1	1	1	1	0	1	1	1	1	1	0	0	0	0					
8						0	0	0	1	0	0	1	0	0	1	1	1	1	0	1	1	1	1	1	1	1	1	1					
9						0	0	0	0	1	0	1	0	0	1	1	1	1	0	1	1	1	0	0	0	0	0	0					
10						0	0	0	0	0	1	1	0	1	0	0	1	1	1	1	0	0	0	1	0	1	1	0					
11						0	0	0	0	0	0	1	1	1	0	0	0	1	1	1	0	0	1	1	1	1	1	1					
12												0	0	0	0	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
13												0	0	0	0	0	1	0	1	1	0	1	0	1	0	1	1	1	1	1	1	1	1
14												0	0	0	0	0	0	0	1	1	0	0	0	1	0	1	1	1	1	1	1	1	1
15												0	0	0	0	1	1	1	0	1	1	1	1	1	1	1	1	1	1	0	1	1	1
16												1	0	0	1	0	0	1	0	0	1	1	1	0	1	1	1	1	1	1	1	1	1
17												1	1	0	1	0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
18												1	0	0	1	1	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
19												0	1	1	0	0	0	0	0	1	0	0	0	1	0	1	0	1	0	0	1	1	1
20												1	1	1	1	0	1	1	1	0	0	0	1	1	1	1	1	1	1	1	1	1	1
21												1	0	0	1	1	1	1	0	0	0	0	1	1	0	1	1	1	1	1	1	1	1
22												1	1	0	1	1	1	1	0	0	0	0	0	1	0	1	1	0	1	1	1	1	1
23												1	0	0	1	1	1	1	0	1	1	0	0	0	1	1	1	1	1	1	0	0	0
24												1	1	1	1	0	1	1	1	1	1	0	0	0	0	1	1	0	1	1	1	1	1
25												1	0	0	1	1	1	1	0	1	0	0	1	0	0	1	1	0	1	1	1	1	1
26												1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1
27												1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	0	0	1	1	1	1	1
28												1	1	1	1	1	1	1	1	1	1	0	1	0	0	1	0	0	1	1	1	1	1
29																																	
30																																	
31																																	
32																																	
33																																	

$\mathbf{R1} =$

A	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
1	0	1	0	0	0	1	0	0	0	0	0																						
2	0	0	1	0	0	0	1	0	0	0	0																						
3	0	0	0	1	0	0	0	0	0	0	1																						
4	0	0	0	0	1	1	0	0	1	0	0																						
5	0	0	0	0	0	1	0	0	0	0	0																						
6						0	1	0	0	0	0	1	1	0	1	0	0	1	1	1	0	0	0	0	0	0	0	0					
7						0	0	1	0	0	0	1	0	0	1	1	1	1	0	1	1	1	1	1	0	0	0	0					
8						0	0	0	1	0	0	1	0	0	1	1	1	1	0	1	1	1	1	1	1	1	1	1					
9						0	0	0	0	1	0	1	0	0	1	1	1	1	0	1	1	1	0	0	0	0	0	0					
10						0	0	0	0	0	1	1	0	1	0	0	1	1	1	1	0	0	0	1	0	1	1	0					
11						0	0	0	0	0	0	1	1	1	0	0	0	1	1	1	0	0	1	1	1	1	1	1					
12												0	0	0	0	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
13												0	0	0	0	0	1	0	1	1	0	1	0	1	0	1	1	1	1	1	1	1	1
14												0	0	0	0	0	0	0	1	1	0	0	0	1	0	1	1	1	1	1	1	1	1
15												0	0	0	0	1	1	1	0	1	1	1	1	1	1	1	1	1	1	0	1	1	1
16												1	0	0	1	0	0	1	0	0	1	1	1	0	1	1	1	1	1	1	1	1	1
17												1	1	0	1	0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
18												1	0	0	1	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
19												0	1	1	0	0	0	0	0	1	0	0	0	1	0	1	0	1	0	0	1	1	1
20												1	1	1	1	0	1	1	1	0	0	0	1	1	1	1	1	1	1	1	1	1	1
21												1	0	0	1	1	1	1	0	0	0	0	1	1	0	1	1	1	1	1	1	1	1
22												1	1	0	1	1	1	1	0	0	0	0	0	1	0	1	1	0	1	1	1	1	1
23												1	0	0	1	1	1	1	0	1	1	0	0	0	1	1	1	1	1	1	0	0	0
24												1	1	1	1	0	1	1	1	1	1	1	0	0	0	1	1	0	1	1	1	1	1
25												1	0	0	1	1	1	1	0	1	0	0	1	0	0	1	1	0	1	1	1	1	1
26												1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1
27												1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	0	0	1	1	1	1	1
28												1	1	1	1	1	1	1	1	1	1	0	1	0	0	1	0	0	1	1	1	1	1
29																																	
30																																	
31																																	
32																																	
33																																	

Chapter 4

Hardware Trojan Identification and Detection

The majority of techniques that have been developed to detect hardware trojans are based on specific attributes. Further, the ad hoc approaches that have been employed to design methods for trojan detection are largely ineffective. These trojans have a number of attributes which can be used to systematically develop detection techniques. Based on this concept, a detailed examination of current trojan detection techniques and the characteristics of existing hardware trojans is presented. This is used to develop a new approach to hardware trojan identification and classification. This identification can be used to compare trojan risk or severity and trojan detection effectiveness. For this purpose, identification vectors are generated for each trojan and trojan detection technique based on the corresponding attributes. Vector are also given which represent trojan risk or severity and trojan detection effectiveness.

Any attempt to address hardware security concerns should begin with a classification of the threats based on the processes involved in the IC production life-cycle. A comprehensive model for trojan classification was presented in Chapter 3 which is based on eight categories: insertion, abstraction, effect, logic type, functionality, activation, physical layout, and location. A classification of attributes based on this model is illustrated in Figure 3.2.

The relationships between these attributes presented in Chapter 3 identifies the attributes that can be detected using a given technique, i.e. a technique that can detect a sequential trojan circuit can also detect a combinational trojan circuit, or a technique that can detect a small trojan can also detect a large trojan. Methods used

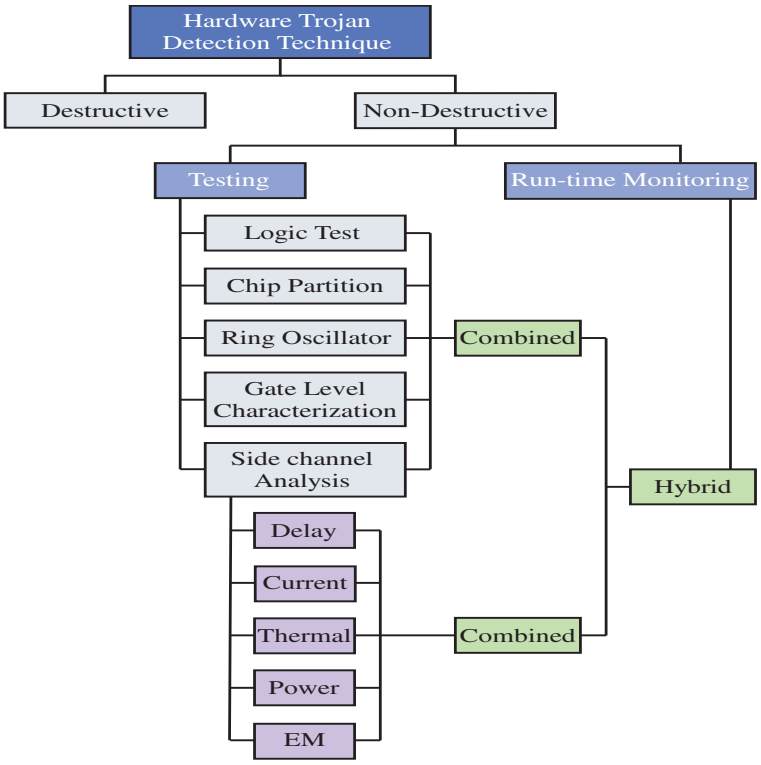


Figure 4.1: The proposed hardware trojan detection classification.

to detect a trojan in one category may also be useful in detecting trojans in other categories. For example, a trojan inserted in the insertion phase may affect attributes in other categories, i.e. 99.99% of all trojans are inserted in the specification phase, so to be effective a technique should detect if a specification attribute has been compromised. In this paper, hardware trojans and hardware trojan detection techniques are examined based on the corresponding trojan attributes and their relationships.

The remainder of this chapter is organized as follows. Section 4.1 reviews the hardware trojan attributes and hardware trojan detection techniques that will be used to illustrate the proposed identification approaches. The trojan attributes are studied and values assigned in Section 4.2. Section 4.3 presents an example of hardware trojan vectors, and Section 4.4 an example of hardware trojan detection vectors.

4.1 Hardware Trojan Detection Techniques

Most hardware trojan detection techniques are based on side-channel analysis. However, noise and PVs can affect the accuracy of side-channel information. Thus multiple

parameters are typically measured to improve the detection performance [79]. For example, in [72] power consumption and delay were measured and combined with gate level characteristics. The use of multiple techniques is shown as combined and hybrid blocks in Figure 4.1. This is a new approach to trojan detection. Compared to Figure 1.1, the proposed approach combines multiple side channel analysis techniques and/or other techniques to improve the effectiveness of trojan detection. Because of the possibility of a trojan evading detection during testing, run-time monitoring is very important. For example, non-destructive technique can be combined with destructive packaging or discarding the output when a trojan is detected.

4.1.1 Side-channel Techniques

Delay

Path Delay Sensors A sensor to detect a hardware trojan in an IC was developed in [84]. This sensor predicts the delay characteristics of specific sequences of operations in the circuit and compares these with reference values. A significant difference indicates that a trojan may be present. Numerous experiments were conducted with different trojan delays and circuit paths. This approach was able to identify trojans with reasonable accuracy compared to non-sensor based techniques. However, the sensors increase the chip area unless they are inserted in empty spaces. This also requires that delay measurements be made within the chip which may be difficult to obtain.

The trojan presented in [84] is inserted during the design stage in the logic abstraction level and causes changes in the circuit parameters to reduce reliability. It is composed of only gates and is always on.

Path Delay Fingerprint A trojan detection technique based on the path delay introduced by trojans was developed in [83, 85]. This method differentiates between the delay for a GC and the IC being tested. The effects of PVs are considered to limit the false positive rate.

The trojan presented in [83] causes changes in circuit function or information leakage and is inserted during the fabrication stage in the logic abstraction level. This trojan can be implemented using combinational or sequential logic and is internally triggered by a specific sequence or a counter. The trojan in [85] also causes changes in circuit function or information leakage and is inserted during the manufacturing

or design stages in the layout or logic abstraction levels. It is implemented using combinational logic and so is always on.

Current

self-referencing A self-referencing approach was employed in [80] to detect hardware trojans. A test vector is used to create a leakage current signature for a GC. The leakage current for an IC under test is compared with the GC signature to determine if a trojan may exist. Different size trojans were inserted in the IC to determine the detection sensitivity.

The trojan presented in [80] causes changes in the circuit function or leaks information, and is inserted during the design stage in the logic abstraction level. This trojan is implemented using sequential logic and is internally triggered by a counter.

Current Integration In [81], current integration method was used to detect hardware trojans. Typically, the leakage current and current consumed by a chip is measured and compared with that of a GC. Instead of using the current directly, it is integrated to convert current into charge which magnifies any differences in the measured current. Therefore, this approach can be useful in identifying small trojans where the effect on the current is masked by PVs and noise.

The trojan presented in [81] causes changes in circuit function and is inserted during the design phase in the layout abstraction level. This trojan can be implemented using combinational or sequential logic and can be internally triggered by a counter or a specific sequence.

Power Consumption

Power measurements for random test patterns can be used to generate a power signature using a GC. This signature can be based on the power consumption as well as the noise generated from surrounded circuits and PV effects. By applying the same random test patterns to all chips, they can be divided into groups according to their power signatures. Reverse engineering can then be applied to a chip from each set to determine which groups are trojan free. The results obtained can be used to choose chips to be used to generate the GC signature, as well as determine suspicious ships which may contain a trojan.

Power Consumption Trace In [68], trojans implemented at the layout level in FPGAs were considered with standard evaluation boards. A side-channel technique using power consumption was developed for trojan detection.

The trojan presented in [68] can cause Denial of Service (DoS), changes in circuit functions and information leakage. It is inserted during the design stage in the layout abstraction level. This trojan can be implemented using either sequential or combinational logic and is internally triggered by a counter or a specific condition.

Process control monitor The detection of hardware trojans which leak information was considered in [69] using a statistical analysis of the output power consumption to classify ICs as infected or trojan free. The advantage of this approach is that a golden chip is not required. Instead, Monte-Carlo simulation is used to determine the classification threshold. A hardware trojan was designed within the PVs of a chip to determine the effectiveness of this approach. It was shown that good trojan detection accuracy can be achieved, but a detailed statistical model is required to achieve suitable accuracy, so this approach can require significant time and resources.

The trojan presented in [69] leaks information and is inserted during the fabrication stage in the system abstraction level, so it is parametric and the trojan is always on.

Power Consumption In [70] a side-channel technique based on power consumption was considered to detect hardware trojans that create information leakage. A hardware trojan was designed within PV limits, and the difference in power consumption between an infected and a trojan free IC was evaluated. It was found that comparing the power consumption does not aid in trojan detection, but a statistical analysis provided better performance. This shows the usefulness of statistical analysis in hardware trojan detection.

The trojan presented in [70] leaks information and was inserted during the fabrication stage in the system abstraction level. Since it leaks information, this trojan is parametric and is always on.

Electromagnetic (EM) Radiation

In [76], hardware trojan detection was considered based on Electromagnetic (EM) radiation. Trojans were inserted next to I/O pins an their effect on the EM emissions was measured [88]. It was shown that significant differences can exist in the emissions

at different chip locations due to the presence of hardware trojans. In particular, it is easier to detect a trojan at the corners of chips because of the proximity to the power lines.

The trojan presented in [76] causes DoS and is inserted during the design stage in the layout or by development environment abstraction level. The trojan is sequential and is externally triggered.

4.1.2 Multi-Parameter

Most side-channel techniques monitor just the leakage current to detect hardware trojans. This requires a GC to determine current anomalies which indicate that a hardware trojan may be present. However, this method has several disadvantages such as PV effects and noisy measurements. Therefore, techniques which consider multiple parameters have been developed based on different chip emissions to increase the detection rate and decrease the number of false positives.

Thermal and Power

The approach in [79] uses thermal and power maps to detect hardware trojans. To improve detection accuracy, statistical analysis as well as thresholding and clustering are used. Further, the effects of both PVs and noise are included in the model.

The trojan presented in [79] causes changes in circuit function and is inserted during the manufacturing stage in the RTL abstraction level. This trojan can be implemented using either sequential or combinational logic and is internally triggered by a counter or a specific condition.

4.1.3 Hybrid Techniques

Logic Testing

Current and Operating Frequency In [82], hardware trojan detection was considered using random test vectors to generate different circuit emissions. The leakage current, transient supply current and maximum operating frequency were used in the trojan detection process. Techniques to improve the detection sensitivity were employed, and power gating was used to prevent undesirable circuit switching. A disadvantage of this approach is that it is not as effective as logic testing for small trojans.

The trojan presented in [82] causes changes in circuit function and is inserted during the fabrication stage in the logic abstraction level. This trojan can be implemented using either combinational or sequential logic and is internally triggered by a specific sequence or a counter.

Gate Level Characterization (GLC)

Power Consumption The use of gate level characterization (GLC) in the detection of hardware trojans was examined in [71]. GLC considers the timing delay, switching power and leakage power of each gate in the circuit. Equations are developed for each gate and a trojan variable is used to determine the presence of a trojan. Experiments were performed using several benchmarks to obtain the GLC of simple gates such as AND, OR, NOT, NOR and NAND. The disadvantage of GLC is that the characterizing all gates in a circuit is a very time consuming process and the complexity can make it intractable to include all gates in the circuit.

The trojan presented in [71] causes changes in the function of the circuit and is inserted during the fabrication stage in the logic abstraction level. This trojan can be implemented using a small number of gates and is internally triggered.

Power Consumption and Delay In [72], hardware trojans were also detected using GLC. The delay and switching power characteristics of each gate was determined using a number of measurements.

The trojan presented in [72] causes changes in the function of the circuit or parametric changes to reduce reliability, or leaks information. It was inserted during the fabrication stage in the logic/layout abstraction levels. This trojan can be implemented using combinational or sequential and is internally triggered by a specific sequence or a counter.

4.1.4 Ring Oscillator (RO)

Length Optimized Ring Oscillators

In [73], a Ring Oscillator (RO) was employed to detect hardware trojans. This is based on differences in the RO frequency due to the presence of a trojan. Both sequential and combinational logic trojans were considered and different RO lengths examined to determine the effect of this length on trojan detection. Trojans were introduced without changing the layout of the circuit. It was shown that a RO of

length 7 provides adequate detection performance. The disadvantage of this scheme is that it is difficult to extend the results to other conditions and circuit configurations.

The trojan presented in [73] causes DoS and is inserted during the specification or design stages in the RTL abstraction level. This trojan can be implemented using either sequential or combinational logic and is internally triggered by a counter or a specific condition.

Ring Oscillator Network

In [74], the detection of hardware trojans was examined using a RO network. The oscillator frequency can change if a hardware trojan is present in the circuit, and ROs are distributed throughout the circuit can thus be used for monitoring purposes. To limit the effects of PVs and noise, statistical and outlier analysis are used.

The trojan presented in [74] causes changes in the function of the circuit and is inserted during the fabrication stage of the logic abstraction level. This trojan can be implemented using gates or combinational trojans and is either always on or internally triggered by a specific condition.

In [75], another RO technique was introduced to detect hardware trojans. A number of trojans and different RO locations were considered to determine the best location for the ROs. Rather than considering the oscillator frequency, principal component analysis was used to extract relevant information from the measurements. The average of a number of measurements was used to reduce the effects on noise.

The trojan presented in [75] causes changes in the function of the circuit and was inserted during the logic abstraction level. This trojan can be implemented using combinational logic and is internally triggered by a given sequence or specific condition.

4.1.5 Chip Partition Technique (CPT)

Current

In [77], time and power signatures were used to compare a Chip Under Test (CUT) with a GC. Current sensors are used to obtain a signature. This sensor consists of a current mirror, a current comparator and a scan register. The circuit is partitioned into regions and each region is tested separately using a corresponding sensor. If the current signature differs significantly from that of the GC, a trojan may be present.

The trojan presented in [77] causes changes in the function of the circuit and is inserted during the design stage in the logic/layout abstraction levels. This trojan can be implemented using sequential logic and is internally triggered based on a counter.

Power

In [78], hardware trojans were detected by partitioning the chip into regions and using the power ports in each region to detect abnormal activity. Test patterns were used in each region to examine the behavior and the effects of noise were reduced by appropriate placement of the ports.

The trojan presented in [78] causes changes in the function of the circuit and was inserted during the manufacturing/design stage in the developmental abstraction level. This trojan can be implemented using either combinational or sequential logic and is internally triggered by a specific condition or a counter.

4.1.6 Run Time Monitoring

Temperature Tracking

In [86], hardware trojans were detected during run-time using thermal maps. A framework for temperature based detection was proposed which consists of design, test and run-time detection. A statistical characterization of the ICs is determined using thermal sensors placed in the design. During testing, the noise levels are determined, and these results are used at run-time along with thermal measurements to detect trojans.

The trojan presented in [86] causes changes in the function of the circuit or DoS, and is inserted during the design stage in the logic abstraction level. This trojan can be implemented using either combinational or sequential logic and is internally or externally triggered.

Redundancy

In [87], design rules were developed to aid in trojan detection during run-time and provide fast recovery from attacks by deactivating the trojan. Eight design rules were proposed for hardware trojan detection based on the test results for IP codes from several vendors. These rules were optimized based on constraints such as latency,

Table 4.1: Classification of Hardware Trojan Detection Techniques

	Attributes																														Chip	
	Insertion					Abstraction						Effect				Type		Functionality		Activation			Physical Layout						Location					Attribute	
Technique	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	GC	PV
[68]		C									C	C	C		C	C	C	C	C	C	C		C	C		C	C		C	C	C	M	M		C
[69]				C		C							C				C		C	C				C		C	C		C	M	C	M	M		C
[70]	M		C			M					C		C				C		C	C				C		C	C		C	C	C	M	M	R	C
[71]			C						C			C					C	C		C	C		C	C		C	C		C	M	M	M	M	R	
[72]			C						C	C	C	C	C	C			C		C	C	C		C	C		C	C		C	M	M	M	M		
[73]	C	C					C								C	C	C	C			C		C		C	C	C		C	M	M	M	M	R	C
[74]		C	C				C					C					C	C		C			C	C	C	C	C	C	C	M	M	M	M	R	C
[75]		C	C						C	C		C			C		C	C		C	C		C	C	C	C	C	C	C	M	M	M	M	R	C
[76]		C						C	C						C	C		C				C	C	C	C	C	C	C	M	M	C	M	M	R	
[77]		C							C			C				C		C			C		C	C	C	C	C	C	C	M	C	M	M	R	C
[78]		C	C				C	C	C			C				C	C	C	C	C	C		C	C		C	C	C	C	M	M	M	M	R	C
[79]			C				C		C			C				C	C	C			C		C	C	C	C	C	C	C	M	M	M	M	R	C
[80]		C	C				C	C	C			C	C			C		C			C		C	C	C	C	C	C	C	M	M	M	M		C
[81]		C							C		C	M	M			C	C	C	M	C	C		C	C		C	C	C	C	M	M	M	M	R	C
[82]			C						C			C				C	C	C		C	C		C	C		C	C		C	M	M	M	M	R	C
[83]			C					C				C	C			C	C	C	C	C	C		C	C	C		C		C	M	C	M	M	R	C
[84]		C	C						C	C				C	C		C		C	C			C	C	C	C	C	M	C	M	M	M	M		C
[85]		C	C					C	C		C	C	C				C	C	C	C			C	C	C	C	C	C	C	M	M	M	M	R	
[86]		C	C				C	C	C				C		C	C	C	C		C	C	C	C		C		M	M	C	M	C	M	M	R	
[87]		C	C					C	C			C				C	C	C			C		C	C		C	C		C	M	M	M	M		

area, number of operations, and cost. This approach was used mainly to determine the trustworthiness of IP cores from third party vendors.

The trojan presented in [87] causes changes in the function of the circuit and was inserted during the design stage in the development abstraction level. This trojan can be implemented using combinational or sequential logic and is internally triggered by a specific sequence or a counter.

Table 4.1 provides a summary of the attributes for the detection techniques considered in this section. Each technique can detect hardware trojans with certain attributes. The letter C indicates that a technique can protect against the attribute, M means the technique may provide protection, while an empty cell means that the technique cannot protect against the attribute. In addition, a technique may require results from a golden chip to compare with measurements from a CUT. This is indicated by an R in the GC column. Moreover, if a technique considers PVs, this is indicated by a C in the PV column.

4.2 Hardware Trojan Attributes

A comprehensive investigation of hardware trojan attributes was presented in [91]. These attributes provide a complete characterization of trojans. In [94], the relationships between the attributes were studied and used to describe the trojan life-cycle from the insertion phase to the location phase. Assigning weights to these attributes

was also considered. The goal was to identify critical combinations of attributes and use them for detection purposes. Each trojan has a number of attributes, and hardware trojan detection techniques have been developed to detect some or all of these attributes. However, the number and combinations of attributes needed for detection has not been considered. In this chapter, the trojan attributes are examined in detail, and they are ranked within each category.

The attributes in each category are used to identify trojans and evaluate their risk or severity. They can also be used to identify trojan detection techniques and evaluate their effectiveness. Each trojan has two vectors I_T and C_T. The first vector identifies the attributes involved in the trojan, while the other presents the attributes in terms of their risk or severity. Each trojan detection technique also has two vectors I_D and C_D. I_D identifies the attributes that can be detected, while C_D presents the attributes in terms of the effectiveness of the technique. Therefore, trojan identification I_T, trojan detection identification I_D, trojan risk or severity C_T, and trojan detection effectiveness C_D, are defined.

The vector I_T is used to identify hardware trojans based on the attributes involved in the trojan

$$I_T = I_R I_A I_E I_L I_F I_C I_P I_O, \tag{4.1}$$

where $\{I_R, I_A, I_E, I_L, I_F, I_C, I_P, I_O\}$ represent the {Insertion, Abstraction, Effect, Logic type, Functionality, Activation, Physical layout, Location} category values, respectively. Each value specifies the attributes involved in the trojan within the category.

I_D is a vector that contains the attributes that identify a trojan detection technique

$$I_D = I_T I_G, \tag{4.2}$$

where $\{I_T, I_G\}$ represent the {Trojan parameters, Chip attribute} category values, respectively. I_G is used to specify if the technique requires a golden chip and/or considers process variations. Each value specifies the attributes that the detection technique can be used against.

C_T is a vector that represents the risk level of a trojan

$$C_T = C_R C_A C_E C_L C_F C_C C_P C_O, \tag{4.3}$$

where $\{C_R, C_A, C_E, C_L, C_F, C_C, C_P, C_O\}$ represent the {Insertion, Abstraction, Effect, Logic type, Functionality, Activation, Physical layout, Location} category risk values,

respectively.

C_D is a vector that represents the strength or effectiveness of a trojan detection technique

$$C_D = C_T C_G, \tag{4.4}$$

where $\{C_T, C_G\}$ represent the {Trojan parameters, Chip attribute} category effectiveness values, respectively. C_G is used to specify if the trojan detection technique requires a golden chip and/or considers process variations.

The values for each category are determined as follows.

4.2.1 InseRtion (R) Category

The insertion category consists of five attributes: specification (1), design (2), fabrication (3), testing (4), and assembly (5).

Insertion Identification (I_R)

The identification parameters for the attributes in this category are based on the fact that starting from specification, the existence of a trojan in an attribute may affect subsequent attributes within the category. Thus, the number of attributes that may be affected is used for identification. Column I_R in Table 4.2 shows that the value 5 is assigned to the specification attribute, which means that a trojan inserted during the specification phase may affect all other attributes (all 5 attributes). Further, a trojan inserted in the design phase can affect the 3 subsequent attributes.

Insertion Risk/Effectiveness (C_R)

The risk and effectiveness values for the attributes in this category are also based on the effect on subsequent attributes. For example, a trojan inserted in the specification phase can affect the design and thus also the fabrication, so that specification $\rightarrow$ design $\rightarrow$ fabrication $\rightarrow$ testing $\rightarrow$ assembly. Thus, a trojan will propagate through the rest of the sequence. which means that fabrication, testing, and assembly may also be affected.

Table 4.2 shows insertion attributes with their Identification (I_R) and Risk/Effectiveness (I_R) values. For example, $I_R = C_R = 3$ to fabrication attribute means that:

From a trojan respective, $I_R = 3$, means that the trojan is inserted during fabrication (trojan inserted in fabrication attribute). $C_R = 3$, means that the fabrication

Table 4.2: Insertion Attribute Values

k	Attribute	I_R	C_R
1	Specification	5	5
2	Design	4	4
3	Fabrication	3	3
4	Testing	2	2
5	Assembly	1	1

attribute is infected, and both of testing and assembly attributes may be infected because of the fabrication infection.

From a trojan detection respective, $I_R = 3$, means that the detection technique can detect a trojan inserted during fabrication phase (fabrication attribute is infected). $C_R = 3$, means that detecting a trojan that infect the fabrication attribute would protect the chip from propagating its effect to the rest attributes within insertion category.

Therefore, the (I_R) and (C_R) ranges for the attributes within insertion category are:

$$\begin{aligned} 1 \;&\leq\; I_R \;\leq\; 5 \\ 1 \;&\leq\; C_R \;\leq\; 5 \end{aligned} \tag{4.5}$$

4.2.2 Abstraction (A) Category

Abstraction category consists of 6 attributes: system (6), RTL (7), development environment (8), logic (9), transistor (10), and physical (11).

Abstraction Identification (I_A)

Attributes within this category are related to each other. Starting from system attribute, Infection in this attribute it may infect all other subsequent attributes. So, to identify attribute within this category, the highest attribute infected is enough to use. Since, all other attributes will suspected because of infection of the identified attribute. Column I_R in Table 4.3, shows that value (5) assigned to RTL attribute, which means that if an attacker insert a trojan in RTL level, all other subsequent levels may effected by this modification.

Table 4.3: Abstraction Attribute Values

k	Attribute	I_A	C_A
6	System	6	6
7	RTL	5	5
8	Development Environment	4	4
9	Logic	3	3
10	transistor	2	2
11	Physical	1	1

Abstraction Risk/Effectiveness (C_A)

Modification in an abstraction level can propagate to the following levels. For example, any modification in system level it may reflect RTL level. Because if there is an alteration to the interconnections, hardware modules, or communication protocols, it can lead to the RTL signals, registers, or boolean functions being compromised. Also, infected system attribute may leads to RTL attribute, and so forth. So, system $\rightarrow$ RTL $\rightarrow$ development environment $\rightarrow$ logic $\rightarrow$ transistor $\rightarrow$ physical. Inserted a trojan in any point may lead to propagate it to the rest of the sequence. Therefore, infected system level may leads to a trojan propagation to the rest (5) attributes, while infection in RTL may leads to trojan propagation to the rest (4) attributes. Assigning severity value for each attribute in this category is based on the level of an attribute to the other attributes. Because infection occurred in any level will effect all subsequent levels.

Table 4.3 shows abstraction attributes with their assigned values. For example, $I_A = C_A = 5$ to RTL attribute means that:

From trojan respective, $I_A = 5$, means that the trojan is inserted in RTL level. $C_A = 5$, means that the RTL attribute is infected, and all subsequent levels (development environment, logic, transistor, and physical) may be infected because of the RTL infection.

From trojan detection respective, $I_A = 5$, means that the detection technique can detect a trojan inserted during RTL level (RTL attribute is infected). $C_A = 5$, means that detecting a trojan that infect the RTL attribute would protect the chip from propagating its effect to the rest attributes within abstraction category.

Therefore, the (I_A) and (C_A) ranges for the attributes within abstraction category

are:

$$\begin{aligned} 1 \ &\leq \ I_A \ \leq \ 6 \\ 1 \ &\leq \ C_A \ \leq \ 6 \end{aligned} \tag{4.6}$$

4.2.3 Effect (E) Category

Effect category consists of four attributes: change in functionality (12), information leakage (13), reduce reliability (14), and denial of service (15).

Effect Identification (I_E)

For an n different trojan effects, there is $2^n - 1$ different combinations. Based on [94], four different effects are declared. So, there is (15) different possible combinations. Different character is assigned for each combination to be uniquely identified ($(1) \rightarrow (F)$). Column I_E in Table 4.4, shows that value (5) identify a trojan that can change in a system functionality, and leak information from the system.

Effect Risk/Effectiveness (C_E)

The effect category attributes have different severity value based on the system, and it may different from system to other. So, these values are not fixed but we assign values to use it to illustrate the idea. Assume that our system is integrated in the ministry of defence, so information leakage is the most critical attribute in this category. Also, assumed that change in functionality and denial of service come after, then reduce reliability. Based on these assumption, $C_E = \{2, 4, 1, 2\}$ represent {Change in Functionality, Information Leakage, Reduce Reliability, Denial of Service} respectively, as shown in Table 4.4. As long as we assigned these value based on the severity of each attribute, so a trojan that combine attributes within this category has a severity value equivalent of the summation of severity value of combined attributes. For example, Column C_E in Table 4.4 = 5 twice. First one, when a trojan has effects: information leakage and reduce reliability (13&14). $C_E(13) + C_E(14) = 4 + 1 = 5$. Second, when a trojan has effects: change in functionality, reduce reliability, and denial of service (12&14&15). $C_E(12) + C_E(14) + C_E(15) = 2 + 1 + 2 = 5$. To determine which one inserted in a chip, I_E value is needed. For $I_E = 8$, it means that the first combination is the attributes combination of the trojan affection. On the other hand, if $I_E = D$, it means the second combination is the attributes combination of the trojan affection.

Table 4.4: Effect Attribute Values

k	Attribute	I_E	C_E
12	Change in Functionality	1	2
13	Information Leakage	2	4
14	Reduce Reliability	3	1
15	Denial of Service	4	2
12 & 13		5	6
12 & 14		6	3
12 & 15		7	4
13 & 14		8	5
13 & 15		9	6
14 & 15		A	3
12 & 13 & 14		B	7
12 & 13 & 15		C	8
12 & 14 & 15		D	5
13 & 14 & 15		E	7
12 &13 & 14 & 15		F	9

Table 4.4 shows effect attributes with their assigned values. For example, $I_E = C$ and $C_E = 8$ mean that:

From trojan respective, $I_E = C$, means that the inserted trojan has effects: change functionality, information leakage, and denial of service. $C_E = 8$, means that if a trojan has these effects combined will have a comprise value (8) of total (9).

From trojan detection respective, $I_E = C$, means that the detection technique designed to detect a trojan that embedded to change functionality, information leakage, and denial of service. $C_E = 8$, means that if a trojan detection technique can detect a trojan with effects: change functionality, information leakage, and denial of service, this detection gain coverage value (8) out of total coverage (9).

Therefore, the (I_E) and (C_E) ranges for the attributes within effect category are:

$$\begin{aligned} 1 \;&\leq\; I_E \;\leq\; F \\ 1 \;&\leq\; C_E \;\leq\; 9 \end{aligned} \tag{4.7}$$

4.2.4 Logic Type (L) Category

Logic type category consists of 2 attributes: sequential (16), and combinational (17).

Table 4.5: Logic Type Attribute Values

k	Attribute	I_L	C_L
16	Sequential	2	2
17	Combinational	1	1
16 & 17	Both	3	3

Logic Type Identification (I_L)

This category consists of two different logic types, and a trojan can be designed to be belong one of these types or both together. Therefore, to identify attribute within this category, three different parameters are needed. As shown in Table 4.5, $I_L = 2$ is used to identify a sequential trojan, $I_L = 1$ is used to identify a combinational trojan, and $I_L = 3$ is used to identify a trojan that designed based on combined of combinational and sequential logic type.

Logic Type Risk/Effectiveness (C_L)

Sequential logic consists of combinational logic and memory unit. This leads to a combination logic may be a state in a sequential logic. Therefore, sequential trojan logic is more dangerous than combinational one because there are many factors that effect its activation. These factors are unknown and unexpected, while the combinational trojan always running which make detecting it more easy compared to the sequential one. Therefore, assigning severity value for sequential attribute ideally is higher than combinational attribute, as shown in Table 4.5. We proposed the value of sequential attribute is double the value of combinational attribute. But these values can be modified if it is needed with keeping the higher value for sequential attribute is a must. As shown in Table 4.5, $C_L = 1$ is used as severity value for inserted combinational trojan, $C_L = 2$ is used as severity value for inserted sequential trojan, and $C_L = 3$ is used as severity value for inserted trojan that designed of both logic type.

Table 4.5 shows logic type attributes with their assigned values. For example, $I_L = C_L = 1$ to the combinational attribute means that:

From trojan respective, $I_L = 1$, means that a combinational trojan is inserted. $C_L = 1$, means that if a combinational trojan is inserted, then it will have a comprise value (1) of total (3).

From trojan detection respective, $I_L = 1$, means that the trojan detection tech-

nique is designed to detect a combinational trojan. $C_L = 1$, means that if a detection technique designed to detect only combinational trojan, this detection gain coverage value (1) out of total coverage (3).

Therefore, the (I_L) and (C_L) ranges for the attributes within logic type category are:

$$\begin{aligned} 1 \ &\leq \ I_L \ \leq 3 \\ 1 \ &\leq \ C_L \ \leq 3 \end{aligned} \tag{4.8}$$

4.2.5 Functionality (F) Category

Functionality category consists of 2 attributes: functional (18), and parametric (19).

Functionality Identification (I_F)

This category consists of two different logic types, and a trojan can be designed to be belong one of these types or both together. Therefore, to identify attribute within this category, three different parameters are needed. As shown in Table 4.6, $I_F = 2$ is used to identify a parametric trojan, $I_F = 1$ is used to identify a functional trojan, and $I_F = 3$ is used to identify a trojan that designed based on combined of parametric and functional logic type.

Functionality Risk/Effectiveness (C_F)

Both attributes have different capabilities, but parametric trojan can affect system operations, which means that parametric trojan covers functional trojan effects. Also, some parametric trojan is designed to leak sensitive information with out effecting the system functionality, which makes it more dangerous than functional trojan. In fact, a trojan designed to have both features will be more serious. Therefore, assigning severity value for parametric attribute ideally is higher than functional attribute, and a trojan has both attributes will have the highest severity value, as shown in Table 4.6. These values can be assigned differently if it is needed to raise certain attribute over other attribute based on different system.

Table 4.6: Functionality Attribute Values

k	Attribute	I_F	C_F
18	Functional	1	1
19	Parametric	2	2
18 &19	Both	3	3

Functionality Severity (C_F)

Table 4.6 shows functionality attributes with their assigned values. For example, $I_F = C_F = 2$ to parametric attribute means that:

From trojan respective, $I_F = 2$, means that a parametric trojan is inserted. $C_F = 2$, means that if a parametric trojan is inserted, then it will have a comprise value of (2) out of (3).

From trojan detection respective, $I_F = 2$, means that the trojan detection technique can detect a parametric trojan if inserted. $C_F = 2$, means that if a detection technique designed to detect only parametric trojan, this detection gain coverage value (2) out of total coverage (3).

Therefore, the (I_F) and (C_F) ranges for the attributes within functionality category are:

$$\begin{aligned} 1 \;&\leq\; I_F \;\leq\; 3 \\ 1 \;&\leq\; C_F \;\leq\; 3 \end{aligned} \tag{4.9}$$

4.2.6 Activation (C) Category

Activation category consists of 3 attributes: always on (20), internally triggered (21), and externally triggered (22).

Activation Identification (I_C)

For an n different trojan activation mechanisms, there is $2^n - 1$ different combinations. Based on [94], three different activation mechanisms are declared. So, there is (7) different possible combinations. Different character is assigned for each combination to be uniquely identified ((1) $\rightarrow$ (7)). Column I_C in Table 4.7, shows that value (6) identify a trojan that can internally or externally activate.

Table 4.7: Activation Attribute Values

k	Attribute	I_C	C_C
20	Always On	1	1
21	Internally Triggered	2	2
22	Externally Triggered	3	4
20 & 21		4	3
20 & 22		5	5
21 & 22		6	6
20 & 21 & 22		7	7

Activation Risk/Effectiveness (C_C)

The activation category attributes have different severity value based on the system, and it may different from system to other. So, these values are not fixed but we assign values to use it to illustrate the idea. Assume that the trojan that externally triggered is the hardest trojan to detect, because the attack will not activate this trojan during testing to avoid detection. The internally triggered trojan may be detected during testing accidentally. The always on trojan is always running and can be detected with adequate trojan detection technique. Based on these assumption, C_C = {4, 2, 1} represent {Externally triggered, Internally triggered, Always on} respectively, as shown in Table 4.7. We assumed that $C_C = 4$ for externally triggered, which higher than internally triggered combined with always on. This assumption reflects the difficulty of detecting this attributes over other attributes. As long as we assigned these value based on the severity of each attribute, so a trojan that combine attributes within this category has a severity value equivalent of the summation of severity value of combined attributes. For example, Column C_C in Table 4.7 = 5 ,when a trojan is always on and can be activated externally (20 & 22). $C_C(20) + C_C(22) = 1 + 4 = 5$.

Table 4.7 shows activation attributes with their assigned values. For example, $I_C = 4$ and $C_C = 3$ mean that:

From trojan respective, $I_C = 4$, means that the inserted trojan is always on and can be activated internally. $C_C = 3$, means that if the trojan inserted is always active and can be activated internally, then it will have a comprise value of (3) out of (7).

From trojan detection respective, the trojan detection technique can detect a parametric trojan if inserted $I_C = 4$, means that the trojan detection technique can detect a trojan if it is activation mechanism is always on or activated internally or

both. $C_C = 3$, means that if a detection technique designed to detect always on and internally triggered trojan only, this detection gain coverage value (3) out of total coverage (7).

Therefore, the (I_C) and (C_C) ranges for the attributes within activation category are:

$$\begin{aligned} 1 \ &\leq \ I_C \ \leq \ 7 \\ 1 \ &\leq \ C_C \ \leq \ 7 \end{aligned} \tag{4.10}$$

4.2.7 Physical Layout (P) Category

Physical layout category consists of 6 attributes: large (23), small (24), changed layout (25), augmented (26), clustered (27), and distributed (28).

Physical Layout Identification (I_P)

It is clear that attributes within this category is divided internally. For example, small and large attributes are related, which means if a detection technique able to detect small trojan it will detect large one. Also, if a trojan is classified as small trojan or large trojan but not both. Same concept with changed layout and augmented, also clustered and distributed. Identifying attributes separately in this category does not produce enough information, so we left them cells blank as shown in Table 4.8. A trojan should has one attribute from each division is a must. Therefore, eight different combination of these attributes are needed to identified. Assigning character ((1) $\rightarrow$ (8)) to identify these different combination, as shown in Table 4.8. Column I_P in Table 4.8, shows that value (4) identify a large trojan inserted without changing the chip layout and it is distributed across the chip.

Physical Layout Risk/Effectiveness (C_P)

We argued that each trojan has three different attributes related to three different sub-group within this category. First sub-group contains small and large. It is clear that small trojan is difficult to detect comparing with large one. Second sub-group contains trojan that changed the chip layout or not (augmented). It is also clear that if the layout of chip is changed, means that chip is infected without spending efforts to prove that. On the other hand, if the layout of a chip does not change still maybe this chip is

Table 4.8: Physical Layout Attribute Values

k	Attribute	I_P	C_P
23	Large	-	1
24	Small	-	2
25	Changed Layout	-	1
26	Augmented	-	2
27	Clustered	-	1
28	Distributed	-	2
23 & 25 & 27		1	3
23 & 25 & 28		2	4
23 & 26 & 27		3	4
23 & 26 & 28		4	5
24 & 25 & 27		5	4
24 & 25 & 28		6	5
24 & 26 & 27		7	5
24 & 26 & 28		8	6

infected and needs more effort to decide if the it is trojan free or infected. Third sub-group contains clustered and distributed. Distributed trojan is more dangerous than clustered one. For example, if the inserted trojan is large after distributed it become small trojan, and each sub circuit has different effect and activation mechanism. So detecting portion of the trojan may other parts continue working. Therefor, small, augmented, and distributed attributes have more value than large, changed layout, and clustered attributes. Based on these assumption, $C_P = \{1, 2, 1, 2, 1, 2\}$ represent {large, small, changed layout, augmented, clustered, distributed} respectively, as shown in Table 4.8. Every trojan should hold a combination three attributes belong three different sub-groups. So, the severity value for a trojan within this category is summation of severity value of combined attributes. For example, Column C_P in Table 4.8 = 6, when inserted small trojan without changed the layout and distribute it across the chip (24 & 26 & 28). $C_P(24) + C_P(26) + C_P(28) = 2 + 2 + 2 = 6$. It is clear that small, augmented, and distributed trojan are the worst case for this category.

Table 4.8 shows physical layout attributes with their assigned values. For example, $I_P = C_P = 3$ mean that:

From trojan respective, $I_P = 3$, means that a large and clustered trojan is inserted

without changing the chip layout. $C_P = 3$, means that a large and clustered trojan is inserted but it changed the chip layout.

From trojan detection respective, $I_P = 3$, means that the trojan detection technique can detect a large and clustered trojan even if it does not changing the chip layout. $C_P = 3$, means that if a detection technique designed to detect a large and clustered trojan that changed the chip layout, this detection gain coverage value (3) out of total coverage (6).

Therefore, the (I_P) and (C_P) ranges for the attributes within physical layout category are:

$$\begin{aligned} 1 &\leq I_P \leq 8 \\ 3 &\leq C_P \leq 6 \end{aligned} \tag{4.11}$$

4.2.8 Location (O) Category

Location category consists of 5 attributes: processor (29), Memory (30), I/O (31), power supply (32), and clock grid (33).

Location Identification (I_O)

For an n different trojan location, there is $2^n - 1$ different combinations. From trojan perspective, the trojan is located in one of these locations. From trojan detection perspective, the detection technique may have the ability to detect a trojan in different locations. Based on [94], five different locations are declared. So, there is (31) different possible combinations. Different character is assigned for each combination to be uniquely identified ($(1) \rightarrow (V)$). The first five parameters (1→5) are used to identify the trojan and trojan detection technique. The rest are used only to identify the detection technique that has the ability to detect a trojan in different locations in a chip.

Column I_O in Table 4.9, shows that value (2) identify a trojan inserted in memory and a detection technique that can detect if there a trojan in memory. While $I_O = F$, identifies a detection technique that can detect if there a trojan located in power supply, or clock grid.

Location Risk/Effectiveness (C_O)

The location category attributes ideally have same value of severity, because if there a trojan in any attribute it will effect the whole system. Therefore, assume that all attributes in this category has $C_O = 1$. From trojan perspective, the trojan will implemented in one of these attribute. According to this view, the severity for any trojan within this category is (1). From detection technique perspective, the detection technique may designed to detect if there a trojan inserted in one or more of these attributes. Based on the severity of each attribute, so a trojan that combine attributes within this category has a severity value equivalent of the summation of severity value of combined attributes. For example, column C_O in Table 4.9 = 2, when a detection technique is design to detect a trojan inserted in two different locations, which higher than a detection technique designed to detect a trojan in one location only. To determine which which location that this detection technique can work in a chip, I_O value is needed. For $I_O = 8$, it means this detection technique can be used to detect if there is a trojan in processor or power supply.

Table 4.9 shows location attributes with their assigned values. For example, $I_O = 3$ and $C_O = 5$:

From trojan respective, $I_O = 3$, means that a trojan inserted in I/O. $C_O = 5$, means that the trojan can be inserted in processor, memory, I/O, power supply, or clock grid.

From trojan detection respective, $I_O = 3$, means that the trojan detection technique designed to detect a trojan inserted in I/O only. $C_O = 5$, means that if a detection technique designed to detect trojan inserted in processor, memory, I/O, power supply, or clock grid, this detection gain coverage value (5) out of total coverage (5).

Therefore, the (I_O) and (C_O) ranges for the attributes within physical layout category are:

$$\begin{aligned} 1 \;&\leq\; I_O \;\leq\; V \\ 1 \;&\leq\; C_O \;\leq\; 5 \end{aligned} \tag{4.12}$$

4.2.9 Chip Attribute (G) Category

Chip attribute category consists of 2 attributes: golden chip (GC), and process variation (PV). This is an important category used only for the detection techniques.

Table 4.9: Location Attribute Values

k	Attribute	I_O	C_O
29	Processor	1	1
30	Memory	2	1
31	I/O	3	1
32	Power Supply	4	1
33	Clock Grid	5	1
29 & 30		6	2
29 & 31		7	2
29 & 32		8	2
29 & 33		9	2
30 & 31		A	2
30 & 32		B	2
30 & 33		C	2
31 & 32		D	2
31 & 33		E	2
32 & 33		F	2
29 & 30 & 31		G	3
29 & 30 & 32		H	3
29 & 30 & 33		I	3
29 & 31 & 32		J	3
29 & 31 & 33		K	3
29 & 32 & 33		L	3
30 & 31 & 32		M	3
30 & 31 & 33		N	3
30 & 32 & 33		O	3
31 & 32 & 33		P	3
29 & 30 & 31 & 32		Q	4
29 & 30 & 31 & 33		R	4
29 & 30 & 32 & 33		S	4
29 & 31 & 32 & 33		T	4
30 & 31 & 32 & 33		U	4
29 & 30 & 31 & 32 & 33		V	5

Table 4.10: Chip Attribute Values

Attribute	I_G	C_G
NONE	1	2
GC	2	1
PV	3	4
GC & PV	4	3

Chip Attribute Identification (I_G)

For an n different chip attributes, there is 2^n different combinations. Two different chip attributes are declared. So, there is (4) different possible combinations. Different character is assigned for each combination to be uniquely identified ((1) $\rightarrow$ (4)). Column I_G in Table 4.10, shows that value (4) identify a detection technique that requires a golden chip (GC) reference and it considers the process variation (PV).

Chip Attribute Effectiveness (C_G)

The chip attribute category attributes have different coverage value. Preparing a reference measurements using a golden chip is an expensive process that requires reverse engineering to make sure the chip is trojan free. Process variation is an important factor that may effects the detection credibility (false positive, false negative). Therefore, the detection technique that designed to detect a trojan without the need for the golden chip, and in the same time considers the process variation is the best detection technique ($C_G = 4$).

The detection technique that requires a golden chip, and does not consider process variation is the most expensive and least effective technique ($C_G = 1$). The detection technique that neither requires a golden chip, nor considers process variation is not an expensive but least effective technique ($C_G = 2$). The detection technique that requires a golden chip, but it considers process variation is an expensive and effective technique ($C_G = 3$). The detection technique that does not require a golden chip, but it considers process variation is less expensive and effective technique ($C_G = 4$).

For example, Column C_G in Table 4.10 = 4, when detection technique designed to detect a trojan without the need for reference measurements of a golden chip but in the same time consider the process variation factor during the detection, this detection gain coverage value (4) out of total coverage (4).

Table 4.11: Hardware Trojan Identification Example

Trojans	Parameters (I_T)								Severity (C_T)							
	I_R	I_A	I_E	I_L	I_F	I_C	I_P	I_O	C_R	C_A	C_E	C_L	C_F	C_C	C_P	C_O
Trojan A	2	6	2	1	2	1	7	7	2	6	4	1	2	1	5	2
Trojan B	3	3	1	2	1	2	8	1	3	3	2	2	1	3	6	1

Table 4.10 shows activation attributes with their assigned values. For example, $I_G = 3$ and $C_G = 3$ mean that:

$I_G = 3$, mean the trojan detection technique considers the process variation and does not required a golden chip during detecting. $C_G = 3$, means that if a detection technique designed to detect a trojan based on a reference measurements of a golden chip but in the same time consider the process variation factor during the detection, this detection gain coverage value (3) out of total coverage (4).

Therefore, the (I_G) and (C_G) ranges for the attributes within activation category are:

$$\begin{aligned} 1 \;&\leq\; I_G \;\leq\; 4 \\ 1 \;&\leq\; C_G \;\leq\; 4 \end{aligned} \tag{4.13}$$

4.3 Hardware Trojan Identification (T_F)

Each hardware trojan has two vectors $\{I_T, C_T\}$, which consists of eight characters each. The first vector (I_T) identifies the trojan and contains all attributes involved in the trojan. The last determines the severity of the trojan. Both vectors provide complete description of the trojan. Each character in associated trojan vectors represent the value associated for a trojan within a category. The severity for any crafted trojan can be measured, and compared with other trojan. Comparison between two hardware trojan is shown in Table 4.11.

Trojan A identifies as $\{2\ 6\ 2\ 1\ 2\ 1\ 7\ 7\}$. $I_R = 2$, means that the trojan is inserted during testing phase and assembly attribute may be involved. $I_A = 6$, means that the trojan modifies the system abstraction level and all sub-level (RTL, development environment, logic, transistor, and physical) may be involved. $I_E = 2$, this trojan is designed to leak information. $I_L = 1$, it is combinational logic trojan. $I_F = 2$, this trojan is parametric. $I_C = 1$, Always on trojan. $I_P = 7$, this is small, augmented,

and clustered trojan. $I_O = 7$, this trojan could be inserted in processor or I/O.

Trojan A severity is {2 6 4 1 2 1 5 2}. $C_R = 2$, means that the trojan is inserted during testing phase, which means this attribute is infected and assembly attribute may be infected. $C_A = 6$, means that the the trojan is inserted in system abstraction level, which means this attribute is infected and all sub-level (RTL, development environment, logic, transistor, and physical) may be infected. $C_E = 4$, this trojan has a severity of 4 out of 9, since this trojan only leaking the system information. $C_L = 1$, it has 1out of 2 severity, since it is a combinational trojan. $C_F = 2$, this trojan has 2 out of 2, since it is parametric trojan. $C_C = 1$, since it is always activated. $C_P = 5$, this trojan has 5 out 6, since its physical layout are small, augmented, and clustered. $C_O = 2$, this trojan has 2 out of 5 because it could be inserted in two different locations out of five.

Trojan B identifies as {3 3 1 2 1 2 8 1}. $I_R = 3$, means that the trojan is inserted during fabrication phase and all subsequent process (testing, and assembly) may be involved. $I_A = 3$, means that the trojan modifies the logic abstraction level and all sub-level (transistor, and physical) may be involved. $I_E = 1$, this trojan is designed to change the system functionality. $I_L = 2$, it is sequential logic trojan. $I_F = 1$, this trojan is functional. $I_C = 2$, internally triggered trojan. $I_P = 8$, this is small, augmented, and distributed trojan. $I_O = 1$, this trojan could be inserted in processor.

Trojan B severity is {3 3 2 2 1 3 6 1}. $C_R = 3$, means that the trojan is inserted in fabrication phase, which means this attribute is infected and assembly attribute may be infected. $C_A = 3$, means that the the trojan is inserted in logic abstraction level, which means this attribute is infected and all sub-level (RTL, development environment, logic, transistor, and physical) may be infected. $C_E = 2$, this trojan has a severity of 2 out of 9, since this trojan could only change the system functionality or denial of service. $C_L = 2$, it has 2 out of 2 severity, since it is a sequential trojan. $C_F = 1$, this trojan has 1 out of 2, since it is a functional trojan. $C_C = 3$, this trojan has 3 out of 7, since it could be always activated, and/or internally triggered. $C_P = 6$, this trojan has 5 out 6, since its physical layout are small, augmented, and distributed. $C_O = 1$, this trojan has 1 out of 5 because it could be inserted only in processor.

To compare these trojans, comparison between similar severity characters is the key. Parameters character is used when there are same severity characters found for the category. Comparing similar character between different trojans at a time. It is logically and naturally occurred that a character is higher in trojan A than trojan

B, and other character is lower in trojan A than trojan B. Attacker and defender should decide which parameters are more important and critical for them than other parameters. This decision based on them system and secured attributes on them system. Suppose that the testing phase is done on house and secured, so trojan A will never inserted in your system. Also, from the attacker respective, if the attacker not part of testing group, trojan A will never inserted. On the other hand, if it is assumed that fabrication stage is not trusted (not secured) as it is defined for trojan B, then any modification here could effect testing phase even if the attacker not working in testing group. Therefore, the comparison based on assumption made by the attacker and defender. Assume that the system dealing with sensitive information, so trojan A is more critical than trojan B. Because trojan A designed to leak information, while trojan B designed for change the functionality and denial of service. But if my system is implemented in aircraft, trojan B become more critical than trojan A. Because denial of service and change functionality are critical effect to the system than leaking system information. Therefore, the attacker can design the trojan based on any parameter he has power on, or trojan description he prefers. On the other hand, the defender need to determine the trojan severity that may effect his system to defend against it.

4.4 Trojan Detection Identification (D_F)

Each detection technique has two vectors $\{I_D, C_D\}$, which consist of nine characters each. The first vector (I_D) identifies the trojan attributes that a detection technique designed to cover. The last determines the detection technique coverage based on the severity of the trojan that has capable to detect. Both vectors provide complete description of the trojan detection technique. Each character in associated trojan detection vectors represents the value associated for it within a related category. Table 4.12 shows the hardware trojan detection techniques identifications and coverages discussed in Section 4.1. The coverage for any detection technique can be compared with other detection techniques to determine the best detection technique among several proposed techniques. Comparison between two detection techniques is shown in Table 4.13.

Technique [72] identifies as {3 3 B 1 2 4 7 V 1}. $I_R = 3$, means that the detection technique has the ability to detect a trojan during fabrication phase. $I_A = 3$, means that the detection technique is designed to detect if there an infection in logic

Table 4.12: Hardware Trojan Detection Techniques

	Parameters									Coverage								
Technique	I_R	I_A	I_E	I_L	I_F	I_C	I_P	I_O	I_G	C_R	C_A	C_E	C_L	C_F	C_C	C_P	C_O	C_G
[68]	4	1	C	3	3	4	7	V	3	4	1	8	3	3	3	5	5	4
[69]	2	6	2	1	2	1	7	V	3	2	6	4	1	2	1	5	5	4
[70]	5	6	2	1	2	1	7	V	4	5	6	4	1	2	1	5	5	3
[71]	3	3	1	1	1	4	7	V	2	3	3	2	1	1	3	5	5	1
[72]	3	3	B	1	2	4	7	V	1	3	3	7	1	2	3	5	5	2
[73]	5	5	4	3	1	2	3	V	4	5	5	2	3	1	2	4	5	3
[74]	4	5	1	1	1	1	8	V	4	4	5	2	1	1	1	6	5	3
[75]	4	3	1	1	1	4	8	V	4	4	3	2	1	1	3	6	5	3
[76]	4	4	4	2	1	3	8	V	2	4	4	2	2	1	4	6	5	1
[77]	4	3	1	2	1	2	8	V	4	4	3	2	2	1	2	6	5	3
[78]	4	5	1	3	3	4	8	V	4	4	5	2	3	3	3	6	5	3
[79]	3	5	1	3	1	2	8	V	4	3	5	2	3	1	2	6	5	3
[80]	4	5	5	2	1	2	8	V	3	4	5	6	2	1	2	6	5	4
[81]	4	3	5	3	3	4	8	V	4	4	3	6	3	3	3	6	5	3
[82]	3	3	1	3	1	4	7	V	4	3	3	2	3	1	3	5	5	3
[83]	3	4	5	3	3	4	5	V	4	3	4	6	3	3	3	4	5	3
[84]	4	4	A	1	2	1	8	V	3	4	4	3	1	2	1	6	5	4
[85]	4	4	5	1	3	1	8	V	2	4	4	6	1	3	1	6	5	1
[86]	4	5	9	3	1	7	2	V	2	4	5	6	3	1	7	4	5	1
[87]	4	4	1	3	1	2	7	V	1	4	4	2	3	1	2	5	5	2

abstract level. I_E = B, this technique designed to detect if the chip does not work as expected, leaking information, or/and reliability reduced. $I_L = 1$, detects if there a combinational logic trojan inserted. $I_F = 2$, detects if there a parametric trojan in the system that makes the chip does not works as expected. $I_C = 4$, detects if there is any permanent modification of the chip, and/or the trojan is activated based on internally triggered mechanism. $I_P = 7$, this technique designed to detect small, augmented, and clustered trojan. I_O = V, could be implemented in any chip. $I_G = 1$, does not require golden chip reference, and process variation not consider during detection.

Technique [72] coverage is {3 3 7 1 2 3 5 5 2}. $C_R = 3$, means that this detection technique designed to detect if there infection in the fabrication attribute, which helps to protect the subsequent (testing, and assembly)process stages. Therefore, it has value 3 out of 5, since this detection protects three process stages out of five. $C_A = 3$,

Table 4.13: Hardware Trojan Detection Example

Techniques	Parameters (I_D)									Coverage (C_D)								
	I_R	I_A	I_E	I_L	I_F	I_C	I_P	I_O	I_G	C_R	C_A	C_E	C_L	C_F	C_C	C_P	C_O	C_G
[72]	3	3	B	1	2	4	7	V	1	3	3	7	1	2	3	5	5	2
[82]	3	3	1	3	1	4	7	V	4	3	3	2	3	1	3	5	5	3

means that this detection technique designed to detect if there is a logic modification, which helps to protect the sub-levels (transistor, and physical). Therefore, it has value 3 out 6, since this technique protects three sub-levels out of six sub-levels. $C_E = 7$, this technique has 7 out of 9, it is designed to detect all trojan effects except the denial of service effect. $C_L = 1$, it has 1out of 3, since this technique is able to detect a combinational trojan only. $C_F = 2$, this technique has 2 out of 3, since it is designed to detect parametric trojan only. $C_C = 3$, it has 3 out of 7, because it only detect the trojan that always activated and/ or internally triggered. $C_P = 5$, this technique has 5 out 6, since it could detect small, augmented, and clustered trojan. $C_O = 5$, this technique has 5 out of 5, because it could detect described trojans inserted in any chip. $C_G = 2$, this technique has 2 out of 4, since it does not consider process variation but in the same time does not need reference measurements for detection.

Technique [82] identifies as {3 3 1 3 1 4 7 V 4 }. $I_R = 3$, means that the detection technique has the ability to detect a trojan during fabrication phase. $I_A = 3$, means that the detection technique is designed to detect if there an infection in logic abstract level. $I_E = 1$, this technique designed to detect if the chip does not work as expected. $I_L = 3$, detects if there a combinational or sequential logic trojan inserted. $I_F = 1$, detects if there a functional trojan in the system that makes the chip works as unexpected. $I_C = 4$, detects if there is any permanent modification of the chip, and/or could be internally triggered. $I_P = 7$, this technique designed to detect small, augmented, and clustered trojan. $I_O = \mathrm{V}$, could be implemented in any chip. $I_G = 4$, require golden chip reference, and process variation is considered during detection.

Technique [82] coverage is {3 3 2 3 1 3 5 5 3}. $C_R = 3$, means that this detection technique designed to detect if there infection in the fabrication attribute, which helps to protect the subsequent (testing, and assembly) process stages. Therefore, it has value 3 out of 5, since this detection protects three process stages out of five. $C_A = 3$, means that this detection technique designed to detect if there is a logic modification, which helps to protect the sub-levels (transistor, and physical). Therefore, it has value

3 out 6, since this technique protects three sub-levels out of six sub-levels. $C_E = 2$, this technique has 2 out of 9, it is designed to detect if there is the chip does not work as expected or if it stopped working. $C_L = 3$, it has 3 out of 3, since this technique is able to detect a combinational and sequential trojans. $C_F = 1$, this technique has 1 out of 3, since it is designed to detect only functional trojans. $C_C = 3$, it has 3 out of 7, because it only detect the trojan that always activated and/ or internally triggered. $C_P = 5$, this technique has 5 out 6, since it could detect small, augmented, and clustered trojan. $C_O = 5$, this technique has 5 out of 5, because it could detect described trojans inserted in any chip. $C_G = 3$, this technique has 3 out of 4, since it considers process variation but in the same time it requires reference measurements for detection.

To compare between different detection techniques, comparison between similar coverage characters is the key. Parameters character is used when there are same coverage characters found for the category. Comparing similar character between different detection technique at a time. It is logically and naturally occurred that a character is higher in detection technique A than detection technique B, and other character is lower in detection technique A than detection technique B. Defender should decide which parameters are more important and critical for them than other parameters. This decision based on them system and secured attributes on them system. Suppose that the testing phase is done on house and secured, so detection technique designed to detect infected testing phase is not enough.

From Table 4.13, both technique are similar in some category coverages but they are different in four character coverages. $C_E = 7$ for [72], which detect all effects of a trojan in a system except if it designed to stop the system from functioning. For [82], $C_E = 2$, which means that technique designed only to detect a trojan inserted to prevent the chip from working correctly or make it stop working. So, From these character only, the defender may choose technique [72] if his system deals with sensitive information, and trojan designed to leak information is the main threat to his system. For $C_G = 2$ in [72], which means this technique does not consider process variation but in the same time does not need reference measurements for detection. For [82], $C_G = 3$, which means this technique considers process variation but in the same time it requires reference measurements for detection. Based on these parameter, the defender may choose technique [82] because process variation is important factor to avoid false alarm, which not provided by other technique. The defender need to determine the trojan severity that may effect his system to defend against it.

Chapter 5

Hardware Attack Mitigation Techniques

Hardware attacks aim at physically accessing a digital system to obtain secret information or modify the system behavior. Many hardware attacks employ hardware trojans, and thus the insertion of these trojans into Integrated Circuits (ICs) has become a major threat. As a consequence, significant effort has been devoted to detecting hardware trojans. However, the increasing sophistication of trojans as well as the growing chip complexity makes detection increasingly difficult. Therefore, hardware attack mitigation techniques have been developed to improve system security. Of particular concern are covert attacks because of their stealth which makes them very dangerous. Some techniques focus on preventing chip piracy, while others concentrate on countering specific attacks. There are now a wide variety of mitigation techniques which can be used against hardware attacks. This provides security personnel with a choice of techniques to protect their systems. In this chapter, a comprehensive survey of hardware attack mitigation techniques is presented. These techniques are classified according to the risk associated with the attacks proposed in chapter 2. The mitigation techniques are matched to the hardware attacks that can be countered based on the resources required. The results presented can help security personnel choose appropriate mitigation techniques to protect their systems against hardware attacks.

The remainder of this chapter is organized as follows. Mitigation techniques for high risk attacks are presented in Section 5.1, while techniques for medium and low risk attacks are given in Sections 5.2 and Section 5.3, respectively.

5.1 High Risk Attack Mitigation Techniques

High risk attacks are very dangerous because most are covert. Many mitigation techniques have been proposed to counter these attacks. Most focus on making the chip emissions independent of the operations. Some of these techniques have been designed for a specific attack while others can counter multiple attacks. The high risk attack mitigation techniques are described below.

5.1.1 Hiding

Hiding is a powerful technique that can be used against an attacker attempting to gain information from chip emissions [21, 22]. The following techniques can be used to hide chip emissions.

Noise Generation

The Signal-to-Noise Ratio (SNR) can be reduced by either lowering the signal strength or increasing the noise level. For example, noise generators decrease the SNR which reduces the ability of an attacker to extract information from chip emissions [128]. This technique can be used to mitigate the high risk attacks: SPA, SEMA, DEMA, and FBA, as well as the medium risk attacks: DPA, TA, ACA, and Cache.

Balanced Logic

Balanced logic is a technique used to makes chip emissions independent of the data being processed. Dual-Rail Pre-charged (DRP) logic is used to create two outputs operating in different phases [22, 129]. This technique can be used to mitigate the high risk attacks: SPA, SEMA, and DEMA, as well as the medium risk attack: DPA.

Asynchronous Logic Gates

Asynchronous logic gates can be used to lower EM emission levels by reducing or eliminating the need for clock synchronization [130, 131]. This technique can be used to mitigate the high risk attacks: SPA, SEMA, and DEMA, and the medium risk attack: DPA.

Low Power Design

Low power design is a method used to reduce the SNR and hide chip emissions to reduce the ability of an attacker to obtain chip information [22]. This technique can be used to mitigate the high risk attacks: SPA, SEMA, DEMA, and FBA. It can also be used to mitigate the medium risk attacks: DPA, ACA, OPLP, OEA, and AIT.

5.1.2 Shielding

Shielding is an effective method to hide chip emissions. This can be achieved via physical shielding or filtering of chip emissions. For EM emissions, metal layers on the outside of a chip can be used for shielding. For FBA, a sensor mesh monitors the chip operations for interruptions or short circuits and raises an alarm if one of these events occur. Glass shielding, opaque material or black taping can be used to guard against optical attacks [26]. For ACA, acoustic shielding such as foam can be employed [98]. This technique can be used to mitigate the high risk attacks: SPA, SEMA, DEMA, and FBA, and the medium risk attacks: DPA, ACA, OPLP, OEA, AIT, and Cache.

5.1.3 Masking (Blinding)

Masking or blinding is a technique used to make it difficult for an attacker to determine the relationship between chip data and emissions. This can be accomplished on a per-gate basis using masking logic. It can also be achieved on a per-block basis by randomizing the input data and reversing this operation to obtain the output data [99–102]. The input data can also be masked with random data before any operations and the output obtained by removing the mask [2]. This technique can be used to mitigate the high risk attacks: SPA, SEMA, DEMA, and FBA, and the medium risk attacks: DPA, TA, ACA, OPLP, OEA, and AIT.

5.1.4 Design Partitioning

Design partitioning prevents information leakage between chip regions. For example, regions that operate on plaintext can be separated from those that operates on ciphertext [12,103]. This technique can be used to mitigate the high risk attacks: SPA, SEMA, DEMA, and FBA. It can also be used to mitigate the medium risk attacks: DPA, TA, ACA, OPLP, and OEA.

Table 5.1: Mitigation Techniques for High Risk Attacks

High Risk Attack	Hardware Mitigation Technique														
	Hiding														
	Noise Generation	Balanced Logic	Asynchronous Logic Gates	Low Power Design	Shielding	Masking (Blinding)	Design Portioning	Anti-tampering (Physical Security)	Noise Filter	Error Detection	Algorithmic Resistance	Restricting Physical Access	Duplicate Operations	Randomized Computation Time	Top Layer Sensor Meshes	Clock Frequency Sensor
SEMA	√	√	√	√	√	√	√	√	√							
DEMA	√	√	√	√	√	√	√	√	√							
FBA	√			√	√	√	√	√	√							
SPA	√	√	√	√	√	√	√	√	√							
FIT								√		√	√	√	√	√	√	√

5.1.5 Anti-tampering (Physical Security)

Anti-tampering or physical security is used to limit the time an attacker has to access a chip by creating a secure zone around the chip. This reduces the amount of emission data that can be collected [35, 104]. This technique can be used to mitigate the high risk attacks: SPA, SEMA, DEMA, FIT, and FBA, and the medium risk attacks: DPA, TA, ACA, OPLP, OEA, DRA, C-JTAG, O-JTAG, FAT, AIT, and Cache.

5.1.6 Emission Filtering

Hardware and/or software emission filters can be used to reduce the amount of data which is leaked [134]. This technique can be used to mitigate the high risk attacks: SPA, SEMA, DEMA, and FBA, and the medium risk attacks: DPA, ACA, OPLP, and OEA.

5.1.7 Error Detection

Error detection codes are used to generate check bits for input data and operations. If the check bits at the output are incorrect, a fault is detected and the output data is discarded [111]. This technique can be used to mitigate the high risk attack: FIT and the medium risk attack: FAT.

5.1.8 Algorithmic Resistance

Algorithmic resistance protects chip operations against fault attacks by making the implementation physically inaccessible. This requires encasing the device in a tamper-proof enclosure and adding sensors to detect any attempts at tampering [111]. This technique can be used to mitigate the high risk attack: FIT and the medium risk attack: DRA, C-JTAG, O-JTAG, AIT, and FAT.

5.1.9 Restricting Physical Access

Restricting access to a device is a simple countermeasure against fault attacks. Encapsulating a device in a tamper-resistant case is an effective means of restricting access [111], which has been successfully implemented [109]. This technique can be used to mitigate the high risk attack: FIT, and the medium risk attacks: DRA, C-JTAG, O-JTAG, AIT, and FAT.

5.1.10 Duplicate Operations

Chip operations can be executed multiple times and the outputs considered valid only when they are identical [110]. If the results differ, an alarm is raised. This is not the best solution to defend against fault based attacks since a fault may still go undetected. This technique increases the system complexity, but an attacker requires more resources and time to obtain sufficient data [106]. Further, while implementation

is simple the overhead is high. This technique can be used to mitigate the high risk attack: FIT and the medium risk attack: FAT.

5.1.11 Randomized Computation Time

Randomizing the computation time of chip operations provides protection against fault attacks [111]. This technique can be used to mitigate the high risk attack: FIT and the medium risk attacks: TA, and FAT.

5.1.12 Top Layer Sensor Meshes

Top layer sensor meshes are mainly used to protect against microprobing attacks. They are placed above the circuit to detect interruptions and short circuits. If procedures such as selective etching or laser cutting are sensed, an alarm will be raised and countermeasures taken such as erasing non-volatile memory [113]. These meshes can also protect against under-voltage or over-voltage analysis attacks. This technique can be used to mitigate the high risk attack: FIT and the low risk attacks: MICRO, RE, and DEP.

5.1.13 Clock Frequency Sensor

Robust low frequency sensors are used to detect tampering which slows the clock frequency [113]. If a sensor raises an alarm, countermeasures such as processor reset and bus line and register grounding can be taken. This technique can be used to mitigate the high risk attack: FIT and the low risk attacks: MICRO, RE, and DEP. Table 5.1 presents the mitigation techniques associated with high risk hardware attacks. Many of these techniques increase the noise level to hide the data signals, while others reduce the chip emission levels. The goal is to make it difficult for an attacker to extract the signals from a chip. Thus, chip access should be limited to only trusted personnel.

5.2 Medium Risk Attack Mitigation Techniques

Medium risk attacks require greater accessibility, resources, and time than high risk attacks. Therefore, some techniques used to mitigate medium risk attacks can also counter high risk attacks. Some techniques have been designed to protect against

specific attacks while others are effective against multiple attacks. The medium risk attack mitigation techniques are described below.

5.2.1 Deep Sub-micron Technology

Data can be protected using storage devices covered with a top metal layer or constructed with deep sub-micron technology which makes it difficult for an attacker to access the transistor level or recover data that has been erased [28]. This technique can be used to mitigate the medium risk attacks: DRA and AIT.

5.2.2 Cycling Memory with Random Data

Memory, i.e. ROM or flash, cells can be cycled $10 - 1000$ times with random data so that effects due to the fresh cells are eliminated. Further, programming memory cells before they are erased can prevent detectable residual data [28]. This technique can be used to mitigate the medium risk attack: DRA.

5.2.3 Time/Branch Equalization

The main countermeasure against timing attacks is to make all chip operations execute in the same amount of time, known as time equalization. For example, multiplication and exponential operations should have a same computation time. This prevents an attacker from determining the number of times each operations is executed, or even when an operation is run [106]. A variation of this technique is branch equalization, in which the time for two branches of a conditional statement are the same [2]. Note that time and branch equalization can have a negative impact on system performance. This technique can be used to mitigate the medium risk attack: TA.

5.2.4 Adding Random Delays

Adding random delays is another technique used to disguise the computation time for an operation. However, using fixed delays still enables an attacker to infer the system response and/or power consumption for specific operations. Therefore it is better to add random delays to counter timing attacks [106]. Unfortunately, adding delays has a negative impact on system performance. This technique can be used to mitigate the medium risk attack: TA.

5.2.5 Constant Time Hardware

Designing a constant time hardware is an easier approach, which uses up the same amount of time for each operation regardless of the input [2]. This technique can be used to mitigate the medium risk attack: TA.

5.2.6 Operation-Memory Access Prevention

This technique prevents memory access by using arithmetic and logic operations instead of look up tables [119]. Data is preloaded into the cache before it is used so attempts to access the cache always result in a hit, so no information is leaked [120]. This technique can be used to mitigate the medium risk attack: Cache.

5.2.7 Cache Partitioning

Cache partitioning prevents information leakage by placing the data in secluded or locked mode. A partitioned cache also separates the cache behaviour of one process from another, which prevents inter-process attacks. Although the cache is still shared, a process cannot access the partitions used by another process [121]. This technique can be used to mitigate the medium risk attack: Cache.

5.2.8 Cache Line Locking

Cache line locking is an extension of cache partitioning. A partitioned cache is static and prevents sharing of unused cache lines with other processes, which is inefficient. Cache line locking is more flexible since it only locks the sensitive cache lines in a secluded secure (private) partition, leaving the remaining cache lines for general use [122]. This technique can be used to mitigate the medium risk attack: Cache.

5.2.9 Direct Mapped Cache

Direct mapped cache maps a memory block to a cache line at run time. This can be achieved using the addresses with a remapping table to get the index of the cache lines. These indexes are hashed into a dedicated partition. The efficient version of direct mapped cache architecture is the An efficient implementation is dynamic memory-to-cache remapping [123]. This technique can be used to mitigate the medium risk attack: Cache.

Table 5.2: Mitigation Techniques for Medium Risk Attacks

Medium Risk Attack	Hardware Mitigation Technique: Hiding: Noise Generation	Hiding: Balanced Logic	Hiding: Asynchronous Logic Gates	Hiding: Low Power Design	Hiding: Shielding	Hiding: Masking/ Blinding	Hiding: Design Portioning	Hiding: Anti-tampering (Physical Security)	Hiding: Noise Filtering	Hiding: Deep Sub-micron Technology	Hiding: Cycling Memory with Random Data	Time/Branch Equalization	Adding Random Delays	Constant time Hardware	Secure JTAG Communication Protocol	Physically Unclonable Function (PUF)	TAP Design	Public/Private Key Pairs	Challenge/Response Protocol	Public Key Infrastructure (PKI)	Operation-Memory Access Prevention	Partitioned Cache	Cache Line Locking	Direct Mapped Cache	Non-deterministic Processor	Error Detection	Algorithmic Resistance	Restricting Physical Access	Duplicate Operations	Randomized Computation Time
DPA	√	√	√	√	√	√	√	√	√																					
TA	√					√	√	√				√	√	√																	√
ACA	√			√	√	√	√	√	√																					
OPLP				√	√	√	√	√	√																					
OEA				√	√	√	√	√	√																					
DRA								√		√	√																	√		
C-JTAG								√							√	√	√	√	√	√								√		
O-JTAG								√							√	√	√	√	√	√								√		
AIT	√			√	√	√		√		√																		√		
FAT								√																		√	√	√	√	√
Cache	√				√			√													√	√	√	√	√					

5.2.10 Non-deterministic Processor

A non-deterministic processor can be used to run instructions in a random order while maintaining data coherency and dependencies between instructions. The power profile in these processors makes it very difficult for an attacker to obtain information [124]. This technique can be used to mitigate the medium risk attack: Cache.

5.2.11 Secure JTAG Communication Protocol

A secure JTAG communication protocol can be achieved using hash functions, message authentication codes, and stream ciphers. These security primitives can be used to design secure protocols between any devices [38]. This technique can be used to mitigate the medium risk attacks: C-JTAG and O-JTAG.

5.2.12 Physically Unclonable Function (PUF)

Existing solutions for network communication such as SSH and SSL are computationally expensive. A simpler approach is to employ a PUF to provide a cryptographic key. The advantage of PUFs is that they are unique intrinsically and inherently random [107]. This technique can be used to mitigate the medium risk attacks: C-JTAG, and O-JTAG. It can also be used to mitigate the low risk attacks: RE, DEP, and Counterfeit.

5.2.13 Test Access Port (TAP) Design

JTAG TAP design employs hash functions and challenge/response protocols to securely access JTAG infrastructure. It allows hierarchy security for JTAG access [108]. This technique can be used to mitigate the medium risk attacks: C-JTAG and O-JTAG.

5.2.14 Public/Private Key Pairs

Public/private key pairs can be used for authentication to protect against JTAG attacks. Different keys for different JTAG access groups can be used to improve security [108]. This technique can be used to mitigate the medium risk attacks: C-JTAG and O-JTAG.

5.2.15 Challenge/Response Protocol

Random number generators can be used to generate challenge messages for use in authentication protocols [108]. This technique can be used to mitigate the medium risk attacks: C-JTAG and O-JTAG.

5.2.16 Public Key Infrastructure (PKI)

Most JTAG security techniques have been developed for data transfer from a JTAG interface to a device using a cryptographic protocol. The exchange of encryption keys can be implemented using a PKI for both testing and JTAG devices [38]. This technique can be used to mitigate the medium risk attacks: C-JTAG and O-JTAG. Table 5.2 presents the mitigation techniques associated with medium risk hardware attacks. Some of these techniques can also counter high risk attacks in Table 5.1. Thus, protecting a system against high risk hardware attacks also secures it against some medium risk hardware attacks. This is one of the advantages of the hardware attack risk assessment methodology proposed in [127]. Some mitigation techniques are only used against single hardware attacks such as JTAG, TA, and Cache.

5.3 Low Risk Attack Mitigation Techniques

Low risk attack mitigation techniques are primarily used to prevent an attacker from analyzing the inner structure of a chip. Often an attacker uses a low risk attack to understand the chip structure and then use this information in a high or medium risk attacks. This information can also be used to copy the chip. The low risk attack mitigation techniques are described below.

5.3.1 Randomized Clock Signal

This technique can be used to prevent an attacker from predicting the execution time of specific instructions. Processors with constant numbers of clock cycles makes this prediction an easy task. Random timing at the clock cycle level makes this task more difficult [112, 113]. This technique can be used to mitigate the low risk attack: MICRO.

5.3.2 Randomized Multi-threading

The unpredictability of execution cycles in a processor can be increased by implementing a multi-threaded architecture which randomly schedules execution on multiple threads [112]. Randomized combinational logic is used to determine the progression of thread execution in the processor [113]. This technique can be used to mitigate the low risk attack: MICRO.

5.3.3 Test Circuit Destruction

Chip testing is done after production, leaving residual test circuits which can be exploited by attackers to gain access to buses and control lines. Therefore, the destruction of test circuits is an important countermeasure, to achieve this, the test interface for one chip can be placed within the area of another chip on the wafer. Then when the wafer is cut into dies, the connections between the chip and test circuitry are destroyed [113]. This technique can be used to mitigate the low risk attack: MICRO.

5.3.4 Restricted Program Counter

The program counter can be used as an address pattern generator using microprobing and the watchdog counters that reset the processor can be disabled to reveal the address space. The program counter can be modified to cover the entire address space using offset counters. When the offset counter reaches its maximum value, it resets the processor instead of overflowing it. Each call, jump, or return instruction writes the address of the destination in *(r)* and resets *(c)* to zero. This approach makes the processor execute limited bytes of processor at a single type [113]. This technique can be used to mitigate the low risk attack: MICRO.

5.3.5 Encrypted Buses

Encrypted buses can be used to make it intractable for an attacker to obtain chip data. The encryption typically employs a Random Number Generator (RNG) which is initialized at the sender and receiver using a private key [114]. This technique can be used to mitigate the low risk attack: MICRO.

5.3.6 Light Sensor

Light sensors can be employed to prevent chip operation after it has been exposed [35]. This technique can be used to mitigate the low risk attacks: MICRO, RE, and DEP.

5.3.7 Glue Logic

Glue logic is used to transform block structures, i.e. ALU, I/O, registers, or CPU circuits, to similar application-specific integrated circuits (ASICs). This makes it very

Table 5.3: Mitigation Techniques for Low Risk Attacks

		Hardware Mitigation Technique														
		PUF	Randomized Clock Signal	Randomized Multi-threading	Test Circuitry Destruction	Restricted Program Counter	Top Layer Sensor Meshes	Clock Frequency Sensor	Encrypted Bus	Light Sensor	Glue Logic	Obfuscation	Verification Difference	IP Watermarking	IP Fingerprinting	IC Metering
Low Risk Attack	MICRO		√	√	√	√	√	√	√	√	√	√	√			
	RE	√					√	√		√	√	√	√	√	√	√
	DEP	√					√	√		√	√	√	√	√	√	√
	Counterfeit	√										√	√	√	√	√

difficult for an attacker to find specific signal or circuitry to attack physically. Glue logic design can be achieved using special design tools [35]. This technique can be used to mitigate the low risk attacks: MICRO, RE, and DEP.

5.3.8 Obfuscation

Hardware design obfuscation is an effective technique to counter low risk attacks. The structure of the hardware is modified to obtain a design that is essentially equivalent in functionality but conceals the chip functionality [115, 116]. Obfuscation makes it harder for an attacker to determine the chip functions so more resources are needed to be successful. Obfuscation can also be implemented using PUF or programmable logic. The logic is configurable to be functionally equivalent to numerous designs to conceal the signal paths [117]. This technique can be used to mitigate the low risk attacks: MICRO, RE, DEP, and Counterfeit.

5.3.9 Verification Difference

Verification difference is used to test chips by comparing measurements with signature values. This technique includes power and time delay analysis as well as Scanning Acoustic Microscopy (SAM), IR thermography and X-Ray Fluoroscopy (XRF) [132]. This technique can be used to mitigate the low risk attacks: MICRO, RE, DEP, and Counterfeit.

5.3.10 IP Watermarking

Intellectual Property (IP) watermarking is a technique similar to paper watermarking and is used to protect against counterfeiting. The key is to insert proprietary information into the IC design. The result is a unique hardware design that includes the watermark along with the chip functions. Watermarking can be embedded in different abstraction levels of the IC design making it difficult to counterfeit [118]. This technique can be used to mitigate the low risk attacks: RE, DEP, and Counterfeit.

5.3.11 IP Fingerprinting

IP fingerprinting assigns a unique and invisible ID into each instance of the IP [125]. This is employed to detect IP overbuilding by a factory. This technique can be used to mitigate the low risk attacks: RE, DEP, and Counterfeit.

5.3.12 IC Metering

IC metering is a set of security protocols that enable a designer to gain post-fabrication control of IC properties and use by passive or active counting of the ICs produced, or by remote runtime disabling [126]. This is achieved by adding new states and transitions to the original Finite State Machine (FSM) of the IC to create a Boosted Finite State Machine (BFSM). A unique and unpredictable ID generated by an IC is utilized to place an BFSM into the power-up state upon activation. The designer, knowing the transition table, is the only one who can generate input sequences required to bring the BFSM into the functional initial (reset) state [126].. Another IC metering protocol is based on PUFs [133]. This technique provides control over all hardware copies and allows counterfeit hardware to be disabled. This technique can be used to mitigate the low risk attacks: RE, DEP, and Counterfeit.

Table 5.3 presents the mitigation techniques associated with low risk hardware attacks. RE and DEP are closely related attacks and therefore can be mitigated using the same techniques. Several mitigation techniques have been designed to counter just MICRO attacks. Some sets of mitigation techniques can be used to counter all low risk attacks. A defender could use one of these sets, based on their system, to mitigate these attacks. For example, some of these techniques can be used to monitor chip overbuilding, while others prevent an attacker from determining the inner structure of a chip, so countermeasures may be required for one or both of these threats.

Chapter 6

Contributions and Future Work

The major hardware security areas, namely hardware attacks, hardware trojans, hardware trojan detection techniques, and hardware attack mitigation techniques have been examined in this dissertation. Methodologies were developed to study the internal relations within each area, and the external relations with other areas were covered.

6.1 Contributions

A level of security is required for each application according to its criticality. Unlike previous classifications, we proposed a model using hardware attack characteristics that provide a more complete classification covering the full span of existing hardware attacks. In addition, we proposed a methodology that can be used to assess the hardware security of a given chip or system. This methodology can assist system users and designers to be aware of the existing or potential threats and devise appropriate countermeasures. System designers can also use the methodology to modify their designs to make them more secure and immune hardware attacks. Hardware attacks were studied and quantified from both the attacker and defender perspective. From the attacker perspective, we defined the hardware attack requirements to succeed. From the defender perspective, we defined the risk assessment for hardware attacks to aid the defender in protecting their system.

A methodology was proposed to develop hardware attack and defence strategies. Algorithms were presented to reveal system vulnerabilities and assess the security of a system. This approach is flexible and can easily be adapted to system modifications

and changes in attacker and/or defender capabilities, as well as new hardware attacks. The attack criteria were categorized according to four properties: Awareness (W), Accessibility (A), Resources (R), and Time (T). For each attack, weights were assigned to the criteria depending on the capability of the attacker or defender to satisfy or protect against the criteria. A binary adjacency matrix was also defined to help classify hardware attacks.

Some hardware attacks (overt attacks) rely on studying a system to identify vulnerabilities (hardware trojans) that can be used later to attack the system (covert attacks). For this reason, hardware trojans are a growing concern in modern IC development and production. This is particularly true for chips intended for critical infrastructure and critical applications. Therefore, we considered the attributes of hardware trojans for every phase of the chip life-cycle. Previous results in the literature assume that some phases of this life-cycle can be trusted, while others may be untrusted. It was shown in this work that any phase can be vulnerable to hardware trojan insertion. The attributes of hardware trojans were characterized, and a hardware trojan matrix was developed to show their connections. This matrix provides many insights into the characteristics of trojans, and thus is a valuable tool for their detection. The results obtained can be used to study hardware trojans and their attributes, and determine untrusted phases in the chip manufacturing process.

Developing techniques to detect existing trojans in a system is essential. Although there have been many efforts move towards building effective detection techniques, the technology required for hardware trojan detection is becoming more complicated. Moreover, there is no standard to compare different detection techniques and measure their effectiveness. So, based on our study of hardware trojan attributes, we obtained identification vectors for a detection technique. We evaluated these attributes to determine which should be considered for effective detection. These detection technique identification vectors provide a systematic way that can be used to evaluate existing and new techniques. Our approach also provide a systematic way to compare trojans and their severity on a system. Also, provides a systematic way to compare detection techniques and their attributes coverage. Based on the system, the defender has the ability to choose the best and adequate detection technique that could be used to detect trojan insertion attempts.

Technology is changing more frequent, even a detection technique with acceptable coverage will not be enough to secure a system. Therefore, hardware attack mitigation techniques must be develop and studied to protect chips against hardware attack

attempts. A comprehensive survey of hardware attack mitigation techniques was presented. These techniques were classified based on the risk assessment for hardware attacks. Also, we matched each technique with hardware attacks that could countermeasure. So, based on hardware attacks that could threaten the system, the defender can apply one or more of these matched techniques to add more layer of security to his system in case of hardware trojan detection failure to detect embedded trojans in the system. Covert hardware attacks cover almost all high and medium risk attacks, which makes them critical to mitigate. Mitigation techniques which counter high risk hardware attacks have the capability to counter all medium risk hardware attacks because many techniques for medium risk attack are inherited from high risk mitigation techniques. Thus protecting a chip against high risk hardware attacks makes it mostly protected against medium risk hardware attacks. Mitigating counterfeiting in low risk attacks is the main concern because other are used to study the system to later launch a high or medium risk hardware attacks, or to make a copy of the system.

6.2 Future Work

The ideas and methodologies proposed in this dissertation could be expanded in the future in different directions:

1. We used the same weight for all capabilities defined by the attacker and defender in Chapter 2. We gave weight (1) for existence and (0) for absence of capability. We are planning to calculate these weights considering other capabilities. In this case, the weights will not be the same, even for the same attack. Therefore, based on the new weights a defender could change his protection strategy. Also, the attacker could depend on other attack with higher weight.

2. A vector was built identify trojan severity and trojan detection coverage in Chapter 4. Our future plan to convert each vector to a single value that could be used to describe the trojan severity and detection coverage. This modification makes the selection for appropriate detection technique for any user in the field an easy task, since the use of proposed idea requires more professional user to choose appropriate detection techniques based on his system status and trojan severity. So, by calculating equivalent single value, the comparison become more easy.

3. We discussed in this work the hardware attacks that could attack a system and the vulnerabilities that these attacks rely on. Also, we provided in Chapter 5 a comprehensive survey of hardware attack mitigation techniques exist in the literature. A powerful technique to protect data during operation is homomorphic encryption. Based on the comprehensive study we provided in this work, a secure design for homomorphic encryption could be developed.

Bibliography

[1] S. Skorobogatov. Semi-invasive attacks: A new approach to hardware security analysis. *Ph.D. Dissertation, University of Cambridge*, 2005.

[2] Y. Zhou and D. Feng. Side-channel attacks: Ten years after its publication and the impacts on cryptographic module security testing. *IACR Cryptology ePrint Archive*, 2005.

[3] S. Moein and F. Gebali. Quantifying overt hardware attacks: Using ART schema. In *Computer Science and its Applic.*, Lecture Notes in Electrical Engineering, vol. 330, Springer, pp. 511–516, 2015.

[4] S. Moein, F. Gebali and I. Traore. Analysis of covert hardware attacks. In *J. Convergence*, vol. 5, no. 3, pp. 26–30, 2014.

[5] P. C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In *Advances in Cryptology*, Lecture Notes in Computer Science, vol. 1109, Springer-Verlag, pp.104–113, 1996.

[6] J.-F. Dhem, F. Koeune, P.-A. Leroux, P. Mestré, J.-J. Quisquater and J.-L. Willems. A practical implementation of the timing attack. In *Smart Card Research and Applications*, Lecture Notes in Computer Science, vol. 1820, Springer-Verlag, pp.167–182, 2000.

[7] W. Schindler. A timing attack against RSA with the chinese remainder theorem. In *Cryptographic Hardware and Embedded Systems*, Lecture Notes in Computer Science, vol. 1965, Springer-Verlag, pp.109–124, 2000.

[8] F. Koeune and J.-J. Quisquater. A timing attack against Rijndael. *UCL Crypto Group Technical Report CG-1999/1*, 1999.

[9] A. Shoufan, F. Strenzke, H. G. Molter and M. Stöttinger. A timing attack against Patterson algorithm in the McEliece PKC. In *Information, Security and Cryptology*, Lecture Notes in Computer Science, vol. 5984, Springer-Verlag, pp. 161–175, 2009.

[10] F. Strenzke. A timing attack against the secret permutation in the McEliece PKC. In *Post-Quantum Cryptography*, Lecture Notes in Computer Science, vol. 6061, Springer-Verlag, pp. 95–107, 2010.

[11] C. Rebeiro, D. Mukhopadhyay and S. Bhattacharya. Timing Channels in Cryptography. *Springer*, 2015.

[12] J.-J. Quisquater and D. Samyde. Electromagnetic analysis (EMA): Measures and counter-measures for smart cards. In *Smart Card Programming and Security*, Lecture Notes in Computer Science, vol. 2140, Springer-Verlag, pp. 200–210, 2001.

[13] H. Kim, N. Bruce, H.-J. Lee, Y. Choi and D. Choi. Side channel attacks on cryptographic module: EM and PA attacks accuracy analysis. In *Information Science and Applications*, Lecture Notes in Electrical Engineering, vol. 339, Springer-Verlag, pp. 509–516, 2015.

[14] U. Hajime, E. Sho, N. Homma, Y. Hayashi and A Takafumi. Electromagnetic analysis against public-key cryptographic software on embedded OS. *IEICE Trans. Commun.*, E98-B(7), pp. 1242–1249, 2015.

[15] M. Backes, M. Dürmuth, S. Gerling, M. Pinkal and C. Sporleder. Acoustic side-channel attacks on printers. In *Proc. USENIX Conf. on Security*, pp. 307–322, 2010.

[16] Y. Berger, A. Wool and A. Yeredor. Dictionary attacks using keyboard acoustic emanations. In *Proc. ACM Conf. on Computer and Commun. Security*, pp. 245–254, 2006.

[17] A. Shamir and E. Tromer.(2004) Acoustic cryptanalysis on nosy people and noisy machines. [online]. Available: `http://www.tau.ac.il/~tromer/acoustic/ec04rump/`

[18] P. Wright and P. Greengrass. Spycatcher: The candid autobiography of a senior intelligence officer. *Bantam Doubleday Dell, New York, NY*, 1987.

[19] S. Skorobogatov. Optically enhanced position-locked power analysis. In *Cryptographic Hardware and Embedded Systems*, Lecture Notes in Computer Science, vol. 4249, Springer-Verlag, pp. 61–75, 2006.

[20] H. J. Mahanta, A. K. Azad and A. K. Khan. Differential power analysis: Attacks and resisting techniques. In *Proc. Inform. Sys. Design and Intelligent Applic.*, Advances in Intelligent Systems and Computing, vol. 340, Springer, pp. 349–358, 2015.

[21] P. C. Kocher, J. M. Jaffe and B. C. Jun. Differential power analysis. In *Advances in Cryptology*, Lecture Notes in Computer Science, vol. 1666, Springer Berlin, pp. 388–397, 1999.

[22] P. C. Kocher, J. M. Jaffe, B. C. Jun and P. Rohatgi. Introduction to differential power analysis. *J. Cryptographic Eng.*, vol. 1, no. 1, pp. 5–27, 2011.

[23] M. Joye, P. Paillier and B. Schoenmakers. On second-order differential power analysis. In *Cryptographic Hardware and Embedded Systems*, Lecture Notes in Computer Science, vol. 3659, Springer-Verlag, pp. 293–308, 2005.

[24] S. Mangard, E. Oswald and T. Popp. Power Analysis Attacks: Revealing the Secrets of Smart Cards. *Springer*, 2007.

[25] M. G. Kuhn. Optical time-domain eavesdropping risks of CRT displays. In *Proc. IEEE Symp. on Security and Privacy*, pp. 3–18, 2002.

[26] J. Loughry and D. Umphress. Information leakage from optical emanations. In *ACM Trans. Inform. and Sys. Security*, vol. 5, no. 3, pp. 262–289, 2002.

[27] S. Skorobogatov. Low temperature data remanence in static RAM. *University of Cambridge Computer Laboratory Technical Report 536*, 2002.

[28] S. Skorobogatov. Data remanence in flash memory devices. In *Cryptographic Hardware and Embedded Systems*, Lecture Notes in Computer Science, vol. 3659, Springer-Verlag, pp. 339–353, 2005.

[29] K. S. Wills, T. Lewis, G. Billus and H. Hoang. Optical beam induced current applications for failure analysis of VLSI devices. In *Proc. Int. Symp. for Testing and Failure Analysis*, pp. 21–26, 1990.

[30] R. Piscitelli, S. Bhasin and F. Regazzoni. Fault attacks, injection techniques and tools for simulation. In *Proc. Int. Conf. on Design & Technology of Integrated Systems in Nanoscale Era*, pp. 1–6, 2015.

[31] N. EL Mrabet, J. J. A. Fournier, L. Goubin and R. Lashermes. A survey of fault attacks in pairing based cryptography. *Cryptogr. Commun.*, vol. 7, no. 1, pp. 185–205, 2015.

[32] S. Al-Fedaghi and S. Moein. Modeling attacks. *Int. J. Safety and Security Eng.*, vol. 4, no. 2, pp. 97–115, 2014.

[33] S. Moein and F. Gebali. Quantifying covert hardware attacks: Using ART schema. In *Proc. Adv. in Inform. Science and Computer Eng.*, pp. 85–90, 2015.

[34] S. Moein and F. Gebali. A Formal methodology for quantifying overt hardware attacks. In *Proc. Adv. in Inform. Science and Computer Eng.*, pp. 63–69, 2015.

[35] M. Tehranipoor and C. Wang. Introduction to Hardware Security and Trust (Eds.). *Springer*, 2012.

[36] C. C. Tiu. A new frequency-based side channel attack for embedded systems. *Ph.D. Dissertation,University of Waterloo,Waterloo*, 2005.

[37] J. Zhang, D. Gu, Z. Guo and L. Zhang. Differential power cryptanalysis attacks against PRESENT implementation. In *Int. Conf. on Advanced Computer Theory and Engineering*, vol. 6, pp. V6-61–V6-65, 2010.

[38] K. Rosenfeld and R. Karri. Attacks and defenses for JTAG. In *IEEE Des. Test*, vol. PP, no. 99, pp. 1–18, 2013.

[39] C. Ajluni. Two new imaging techniques promise to improve IC defect identification. In *Electronic Design*, vol. 43, no. 14, pp. 37–38, 1995.

[40] R. Avanzi. Side channel attacks on implementations of curve-based cryptographic primitives. *IACR Cryptology ePrint Archive*, 2005.

[41] Chipworks. [online]. Available:`http://www.chipworks.com`

[42] L. Wagner. Failure Analysis of Integrated Circuits: Tools and Techniques (Eds.). *Springer*, 2012.

[43] S. Skorobogatov and R. Anderson. Optical fault induction attacks. In *Cryptographic Hardware and Embedded Systems*, Lecture Notes in Computer Science, vol. 2523, Springer-Verlag, pp. 2–12, 2003.

[44] S. Skorobogatov. Local heating attacks on flash memory devices. In *Proc. Int. Workshop on Hardware-Oriented Security and Trust*, pp. 1–6, 2009.

[45] S. Skorobogatov. Flash memory 'bumping' attacks. In *Cryptographic Hardware and Embedded Systems*, Lecture Notes in Computer Science, vol. 6225, Springer-Verlag, pp. 158–172, 2010

[46] S. Skorobogatov. Physical attacks and tamper resistance. In *Introduction to Hardware Security and Trust*, M. Tehranipoor and C. Wang (Eds.), Springer, New York, NY, pp. 143–174, 2012.

[47] Anonymous soldier from country B, personal communication.

[48] Author. Author experience and announced later related problem in CNN. `http://money.cnn.com/2015/02/04/autos/toyota-camry-lawsuit/`.

[49] S. Adee. The hunt for the kill switch. In *IEEE Spectrum*, vol. 45, no. 5, pp. 34–39, 2008.

[50] Defense Science Board. (2005) Task force on high performance microchip supply. [online]. Available: `http://www.acq.osd.mil/dsb/reports/ADA435563.pdf`

[51] R. Karri, J. Rajendra, K. Rosenfeld and M. Tehranipoor. Trustworthy hardware: Identifying and classifying hardware trojans. *IEEE Computer*, vol. 43, no. 10, pp. 39–46, 2010.

[52] R. M. Rad, X. Wang, M. Tehranipoor and J. Plusquellic. Power supply signal calibration techniques for improving detection resolution to hardware trojans. In *Proc. IEEE/ACM Int. Conf. on Computer-Aided Design*, pp. 632–639, 2008.

[53] F. Wolff, C. Papachristou, S. Bhunia, and R. S. Chakraborty. Towards trojan-free trusted ICs: Problem analysis and detection scheme. In *Proc. Conf. on Design, Automation and Test in Europe*, pp. 1362–1365, 2008.

[54] X. Wang, M. Tehranipoor and J. Plusquellic. Detecting malicious inclusions in secure hardware: Challenges and solutions. In *Proc. Int. Workshop on Hardware-Oriented Security and Trust*, pp. 15–19, 2008.

[55] B. Sharkey. (2007) TRUST in integrated circuit program - briefing to industry. [online]. Available: http://cryptocomb.org/DARPA-TrustinIntegratedCircuitsProgram.pdf

[56] T. Zhou and T. Tarim. An efficient and well-controlled IC system development flow: Design approved specification and design guided test plan. *Proc. Int. Symp on Circuits and Systems*, pp. 2775–2778, 2005.

[57] S. Padmanabhan. (2013) Discover a better way to go from C-level to synthesis for SoC designs. [online]. Available: http://electronicdesign.com/technologies/discover-better-way-go-c-level-synthesis-soc-designs

[58] D.R. Collins. Trust, a proposed plan for trusted integrated circuits. *Defense Advanced Research Projects Agency-Microsystem Technology Office*, 2006.

[59] StarChip. Product providers for semiconductor market. [online]. Available: http://www.starchip-ic.com/Products/Products.html

[60] A. Iqbal. (2013) Security threats in integrated circuits. [online]. Available: http://sdm.mit.edu/security-threats-in-integrated-circuits/

[61] J. A. Roy, F. Koushanfar, and I. L. Markov. Circuit CAD tools as a security threat. In *Proc. IEEE Int. Workshop on Hardware-Oriented Security and Trust*, pp. 65–66, 2008.

[62] N. Lin, S. Li, J. Chen, P. Wei and Z. Zhao. The influence on sensitivity of hardware trojans detection by test vector. In *Proc. Commun. Security Conf.*, pp. 1–6, 2014.

[63] G. T. Becker, A. Lakshminarasimhan, L. Lin, S. Srivathsa, V. B. Suresh, and W. Burelson. Implementing hardware trojans: Experiences from a hardware trojan challenge. In *Proc. IEEE Int. Conf. on Computer Design*, pp. 301–304, 2011.

[64] R. Kumar, P. Jovanovic, W. Burleson, and I. Polian. Parametric trojans for fault-injection attacks on cryptographic hardware. In *Proc. Workshop on Fault Diagnosis and Tolerance in Cryptography*, pp. 18–28, 2014.

[65] W. Danesh, J. Dofe and Q. Yu. Efficient hardware trojan detection with differential cascade voltage switch logic. In *Advanced VLSI Architecture Design for Emerging Digital Systems*, vol. 2014, no. 5, pp. 1–11, 2014.

[66] R. S. Chakraborty, S. Narasimhan and S. Bhunia. Hardware trojan: Threats and emerging solutions. In *IEEE International on High Level Design Validation and Test Workshop*, pp. 166–171, 2009.

[67] S. Narasimhan and S. Bhunia. Hardware trojan detection. In *Introduction to Hardware Security and Trust* , M. Tehranipoor and C. Wang (Eds.), Springer, New York, NY, pp. 339–364, 2012.

[68] C. Marchand and J. Francq. Low-level implementation and side-channel detection of stealthy hardware trojans on field programmable gate arrays. *IET Computers Digital Tech.*, vol. 8, no. 6, pp. 246–255, 2014.

[69] Y. Liu, K. Huang, and Y. Makris. Hardware trojan detection through golden chip-free statistical side-channel fingerprinting. In *Proc. ACM/EDAC/IEEE Design Automation Conf.*, pp. 1–6, 2014.

[70] Y. Liu, Y. Jin, and Y. Makris. Hardware trojans in wireless cryptographic ICs: Silicon demonstration & detection method evaluation. In *Proc. IEEE/ACM Int. Conf. on Computer-Aided Design*, pp. 399–404, 2013.

[71] D. Karunakaran and N. Mohankumar. Malicious combinational hardware trojan detection by gate level characterization in 90nm technology. In *Proc. Int. Conf. on Computing, Commun. and Networking Tech.*, pp. 1–7, 2014.

[72] M. Potkonjak, A. Nahapetian, M. Nelson, and T. Massey. Hardware trojan horse detection using gate-level characterization. In *Proc. IEEE/ACM Design Automation Conf.*, pp. 688–693, 2009.

[73] P. Kitsos and A. G. Voyiatzis. FPGA trojan detection using length-optimized ring oscillators. In *Proc. Euromicro Conf. on Digital System Design*, pp. 675–678, 2014.

[74] X. Zhang and M. Tehranipoor. RON: An on-chip ring oscillator network for hardware trojan detection. In *Proc. Design, Automation and Test in Europe Conf. and Exhibition*, pp. 1–6, 2011.

[75] A. Ferraiuolo, X. Zhang, and M. Tehranipoor. Experimental analysis of a ring oscillator network for hardware trojan detection in a 90nm ASIC. In *Proc. IEEE/ACM Int. Conf. on Computer-Aided Design*, pp. 37–42, 2012.

[76] O. Soll, T. Korak, M. Muehlberghuber, and M. Hutter. EM-based detection of hardware trojans on FPGAs. In *Proc. IEEE Int. Symp. on Hardware-Oriented Security and Trust*, pp. 84–87, 2014.

[77] Y. Cao, C. Chang, and S. Chen. A cluster-based distributed active current sensing circuit for hardware trojan detection. *IIEEE Trans. Inform. Forensics Security*, vol. 9, no. 12, pp. 2220–2231, 2014.

[78] X. Mingfu, H. Aiqun, and L. Guyue. Detecting hardware trojan through heuristic partition and activity driven test pattern generation. In *Proc. Commun. Security Conf.*, pp. 1–6, 2014.

[79] A. Nowroz, K. Hu, F. Koushanfar, and S. Reda. Novel techniques for high-sensitivity hardware trojan detection using thermal and power maps. *IEEE Trans. Comput.-Aided Design Integr. Circuits and Sys.*, vol. 33, no. 12, pp. 1792–1805, 2014.

[80] S. Narasimhan, X. Wang, D. Du, R. S. Chakraborty, and S. Bhunia. TeSR: A robust temporal self-referencing approach for hardware trojan detection. In *IEEE Int. Symp. on Hardware-Oriented Security and Trust*, pp. 71–74, 2011.

[81] X. Wang, H. Salmani, M. Tehranipoor, and J. Plusquellic. Hardware trojan detection and isolation using current integration and localized current analysis. In *Proc. IEEE Int. Symp. on Defect and Fault Tolerance of VLSI Systems*, pp. 87–95, 2008.

[82] S. Narasimhan, D. Du, R. Chakraborty, S. Paul, F. Wolff, C. Papachristou, K. Roy, and S. Bhunia. Hardware trojan detection by multiple-parameter side-channel analysis. *IEEE Trans. Comput.*, vol. 62, no. 11, pp. 2183–2195, 2013.

[83] Y. Jin and Y. Makris. Hardware trojan detection using path delay fingerprint. In *Proc. IEEE Int. Conf. on Hardware-Oriented Security and Trust*, pp. 51–57, 2008.

[84] M. Li, A. Davoodi, and M. Tehranipoor. A sensor-assisted self-authentication framework for hardware trojan detection. In *Proc. Design, Automation and Test in Europe Conf. and Exhibition*, pp. 1331–1336, 2012.

[85] P. Kumar and R. Srinivasan. Detection of hardware trojan in SEA using path delay. In *Proc. IEEE Students' Conf. on Elect., Electronics and Computer Sci.*, pp. 1–6, 2014.

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

[87] X. Cui, K. Ma, L. Shi, and K. Wu. High-level synthesis for run-time hardware trojan detection and recovery. In *Proc. ACM/EDAC/IEEE Design Automation Conf.*, pp. 1–6, 2014.

[88] C. Lavin, M. Padilla, J. Lamprecht, P. Lundrigan, B. Nelson and B. Hutchings. RapidSmith: Do-it-yourself CAD tools for Xilinx FPGAs. In *Proc. Int. Conf. on Field Programmable Logic and Applic.* pp. 349–355, 2011.

[89] R. Rad, J. Plusquellic and M. Tehranipoor. Sensitivity analysis to hardware trojans using power supply transient signals. In *Proc. Int. Workshop on Hardware-Oriented Security and Trust*, pp. 3–7, 2008.

[90] M. Banga and M. Hsiao. A region based approach for the identification of hardware trojans. In *Proc. IEEE Int. Workshop on Hardware-Oriented Security and Trust*, pp. 40–47, 2008.

[91] S. Moein, S. Khan, T. A. Gulliver, and F. Gebali, and M. W. El-Kharashi. An attribute based classification of hardware trojans. In *Proc. Int. Conf. on Computer Engineering and Sys.*, 2015.

[92] M. Abramovici and P. Bradley. Integrated circuit security: New threats and solutions. In *Proc. Workshop on Cyber Security and Inform. Intelligence Res.*, article no. 55, 2009.

[93] D. McIntyre, F. Wolff, C. Papachristou and S. Bhunia. Dynamic evaluation of hardware trust. In *Proc. IEEE Int. Workshop on Hardware-Oriented Security and Trust*, pp. 108–111, 2009.

[94] S. Moein, T. A. Gulliver, and F. Gebali. A new characterization of hardware trojan attributes. *IEEE Trans. Inform. Forensics Security*, submitted.

[95] M. Banga and M. S. Hsiao. A region based approach for the identification of hardware trojans. In *Proc. Int. Workshop on Hardware-Oriented Security and Trust*, pp.40–47, 2008.

[96] M. Tehranipoor and F. Koushanfar. A survey of hardware trojan taxonomy and detection. In *IEEE Design and Test of Computers*, vol. 27, no. 1, pp. 10–25, 2010.

[97] K. M. Goertzel and B. A. Hamilton. Integrated circuit security threats and hardware assurance countermeasures. In *Crosstalk - Real Time Information Assurance*, pp. 33–38. 2013.

[98] D. Genkin, A. Shamir and E. Tromer. RSA key extraction via low-bandwidth acoustic cryptanalysis. In *Advances in Cryptology*, Lecture Notes in Computer Science, Vol. 8616, Springer-Verlag, pp. 444–461, 2014.

[99] S. Chari, C. S. Jutla, J. R. Rao and P. Rohatgi. Towards sound approaches to counteract power-analysis attacks. In *Advances in Cryptology*, Lecture Notes in Computer Science, Vol. 1666, Springer-Berlin, pp. 398–412, 1999.

[100] L. Goubin and J. Patarin. DES and differential power analysis the "duplication" method. In *Cryptographic Hardware and Embedded Systems*, Lecture Notes in Computer Science, vol. 1717, Springer-Verlag, pp. 158–172, 1999.

[101] M.-L. Akkar and C. Giraud. An implementation of DES and AES, secure against some attacks. In *Cryptographic Hardware and Embedded Systems*, Lecture Notes in Computer Science, vol. 2162, Springer-Verlag, pp. 309–318, 2001.

[102] J. D. Golić and C. Tymen. Multiplicative masking and power analysis of AES. In *Cryptographic Hardware and Embedded Systems*, Lecture Notes in Computer Science, vol. 2523, Springer-Verlag, pp. 198–212, 2003.

[103] J. Friedman. Tempest: A signal problem. In *NSA Cryptologic Spectrum*. [online]. Available: `https://www.nsa.gov/public_info/_files/cryptologic_spectrum/tempest.pdf`.

[104] R. Anderson and M. Kuhn. Tamper resistance - a cautionary note. In *Proc. USENIX workshop on elec. commerce*, Vol. 2, pp. 1–11, 1996.

[105] W. Cilio, M. Linder, C. Porter, J. Di, S. Smith and D. Thompson. Side-channel attack mitigation using dual-spacer dual-rail delay-insensitive logic (D^3L). In *Proc. IEEE SoutheastCon*, pp.471–474, 2010.

[106] H. Bar-El. Introduction to side channel attacks. White Paper, Discretix Tech. Ltd. [online]. Available: `http://gauss.ececs.uc.edu/Courses/c653/lectures/SideC/intro.pdf`

[107] G. E. Suh and S. Devadas. Physical unclonable functions for device authentication and secret key generation. In *Proc. of ACM Design Automation Conf.*, pp. 9--14, 2007.

[108] C. J. Clark. Anti-tamper JTAG TAP design enables DRM to JTAG registers and P1687 on-chip instruments. In *Proc. Int. Workshop on Hardware-Oriented Security and Trust*, pp. 19–24, 2010.

[109] IBM. (2008) CCA basic services reference and guide for the IBM 4758 PCI and IBM 4764 PCI-X cryptographic coprocessors. [online]. Available: `http://www-03.ibm.com/security/cryptocards/pdfs/bs330.pdf`

[110] E. Biham and A. Shamir. Differential fault analysis of secret key cryptosystems. In *Advances in Cryptology*, Lecture Notes in Computer Science, Vol. 1294, Springer-Verlag, pp. 513–525, 1997.

[111] A. Barenghi, L. Breveglieri, I. Koren and D. Naccache. Fault injection attacks on cryptographic devices: Theory, practice, and countermeasures. *Proceedings of the IEEE*, vol. 100, no.11, pp. 3056–3076, 2012.

[112] S. W. Moore. Multithreaded Processor Design. *Kluwer Academic Publishers*, 1996.

[113] O. Kömmerling and M. G. Kuhn. Design principles for tamper-resistant smartcard processors. In *Proc. of the USENIX Workshop on Smartcard Tech.*, vol.12, pp. 9–20, 1999.

[114] L. Changlong, Z. Yiqiang, S. Yafeng and G. Xingbo. A System-On-Chip bus architecture for hardware trojan protection in security chips. In *Int. Conf. of Electron Devices and Solid-State Circuits*, pp.1–2, 2011.

[115] E. Castillo, U. Meyer-Baese, A. Garcia, L. Parilla and A. Lloris. IPP@HDL: Efficient intellectual property protection scheme for IP cores. *IEEE Trans. VLSI Systems*, vol. 16, no. 5, pp. 578–591, 2007.

[116] R. S. Chakraborty and S. Bhunia. Security against hardware trojan through a novel application of design obfuscation. In *Proc. ACM Int. Conf. on Computer-Aided Design*, pp. 113–116, 2009.

[117] J. B. Wendt and M. Potkonjak. Hardware obfuscation using PUF-based logic. In *Proc. IEEE/ACM Int. Conf. on Computer-Aided Design*, pp. 270–277, 2014.

[118] M. Ni and Z. Gao. Watermarking system for IC design IP protection. In *Int. Conf. on Commun., Circuits and Systems*, vol. 2, pp.1186–1190, 2004.

[119] D. A. Osvik, A. Shamir and E. Tromer. Cache attacks and countermeasures: The case of AES. In *Topics in Cryptology*, Lecture Notes in Computer Science, vol. 3860, Springer-Verlag, pp. 1–20, 2006.

[120] E. Brickell, G. Graunke, M. Neve and J.-P. Seifert. Software mitigations to hedge AES against cache-based software side channel vulnerabilities. In *IACR Cryptology ePrint Archive*, 2006.

[121] D. Page. Partitioned cache architecture as a side-channel defense mechanism. In *IACR Cryptology ePrint Archive*, 2005.

[122] Z. Wang and R. B. Lee. New cache designs for thwarting software cache-based side channel attacks. In *int. ACM Symp. on Computer Architecture*, Vol. 35, no. 2, pp. 494–505, 2007.

[123] Z. Wang and R. B. Lee. A novel cache architecture with enhanced performance and security. In *Int. IEEE/ACM Symp. on Microarchitecture*, pp. 83–93, 2008.

[124] D. Page. Defending against cache-based side-channel attacks. In *Information Security Technical Report*, vol. 8, no. 1, pp. 30–44, 2003.

[125] A. E. Caldwell, H.-J. Choi, A. B. Kahng, S. Mantik, M. Potkonjak, G. Qu and J. L. Wong. Effective iterative techniques for fingerprinting design IP. In *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 23, no. 2, pp. 208–215, 2004.

[126] Y. Alkabani and F. Koushanfar. Active hardware metering for intellectual property protection and security. In *USENIX Security Symp.*, pp. 291–306, 2007.

[127] S. Moein, F. Gebali, T. A. Gulliver and M. W. El-Kharashi. Hardware attack risk assessment. In *Proc. Int. Conf. on Computer Engineering and Sys.*, 2015.

[128] P. C. Kocher, J. M. Jaffe and B. C. Jun. Using unpredictable information to minimize leakage from smartcards and other cryptosystems. *US Patent No. 6,327,661*, 2001.

[129] J. M. Jaffe, P. C. Kocher and B. C. Jun. Balanced cryptographic computational method and apparatus for leak minimizational in smartcards and other cryptosystems. *U.S. Patent No. 6,510,518*, 2003.

[130] G. Taylor, S. Moore, R. Anderson, R. Mullins and P. Cunningham. Improving smart card security using self-timed circuits. In *IEEE Symp. on Asynchronous Circuits and Systems*, pp. 211–218, 2002.

[131] Z. C. Yu, S. B. Furber and L. A. Plana. An investigation into the security of self-timed circuits. In *IEEE Symp. on Asynchronous Circuits and Systems*, pp. 206–215, 2003.

[132] F. E. McFadden and R. D. Arnold. Supply chain risk mitigation for IT electronics. In *IEEE Int. Conf. onTech. for Homeland Security*, pp. 49–55, 2010.

[133] R. Maes, D. Schellekens, P. Tuyls and I. Verbauwhede. Analysis and design of active IC metering schemes. In *Proc. Int. Workshop on Hardware-Oriented Security and Trust*, pp. 74–81, 2009.

[134] T. S. Messerges, E. A. Dabbish and R. H. Sloan. Examining smart-card security under the threat of power analysis attacks. In *IEEE Trans. on Computers*, Vol. 51, pp. 541–552, 2002.