Enhancements in Embedded Systems Security using Machine Learning

Ritu Ranjan Shrivastwa

INSTITUT POLYTECHNIQUE DE PARIS

NNT : 2023IPPAT051

Thèse de doctorat

TELECOM Paris

IP PARIS

Enhancements in Embedded Systems Security using Machine Learning

Thèse de doctorat de l'Institut Polytechnique de Paris
Préparée à Télécom Paris

Ecole doctorale n° 626 Ecole doctorale de l'Institut Polytechnique de Paris (ED IP Paris)
Spécialité de doctorat: Réseaux, informations et communications

Thèse présentée et soutenue à Paris, le 05/12/2023, par

RITU RANJAN SHRIVASTWA

Jury composition:

M. Jean-Max DUTERTRE Professeur, Ecole des Mines de Saint-Étienne, France	Président/Examinateur
M. Yossi OREN Professeur, Ben-Gurion University, Israel	Rapporteur
M. Philippe MAURINE Maître de conférence HDR, LIRMM, Université de Montpellier, France	Rapporteur
M. Gilles SASSATELLI Professeur, LIRMM, Université de Montpellier, France	Examinateur
M. Giorgio DI NATALE Professeur, TIMA, Université Grenoble Alpes, France	Examinateur
M. Jean-Luc DANGER Professor, Télécom Paris, France	Directeur de thèse
M. Sylvain GUILLEY CTO, Secure-IC et Professeur Invité de Télécom Paris, France	Co-directeur de thèse

II

Contents

Acknowledgements

It has been an adventurous, rewarding and at the same time arduous journey working on this part-time thesis deemed under the significant collaboration of the esteemed organization Secure-IC and the prestigious university Télécom-Paris. I took great pleasure in the endeavours to write this thesis of my four years of industrial research journey and I can in many words thank my thesis director and supervisor Prof. Jean-Luc DANGER and my thesis co-director and co-supervisor Prof. Sylvain GUILLEY for their tremendous support, guidance and encouragements during this whole period. It is with their kind supervision, through many meetings, discussions and brainstorming sessions, that I am able to complete my thesis work.

I also take this opportunity to sincerely thank all the jury members for agreeing to be a part of the defense jury, to review and examine the thesis manuscript. I express my grateful thanks to the reviewers Prof. Yossi OREN and Assoc. Prof. Philippe MAURINE, and the examiners Prof. Gilles SASSATELLI, Prof. Jean-Max DUTERTRE, and Prof. Giorgio DI NATALE.

In my company, Secure-IC, I must thank my manager Dr. Youssef SOUISSI for his encouragements and support during the period of my PhD work. I would also like to thank my CEO M. Hassan TRIQUI for his support in this process. Additionally, I would like to extend my thanks to all my colleagues and my teammates in Secure-IC, who in one way or the other helped me during my thesis works and supported me for the same.

I must also thank the administrative body of the doctoral school at Télécom-Paris and EDIPP, Prof. Adriana TAPUS and Ms. Florence Besnard, and also to the director of LTCI Prof. Talel ABDESSALEM.

I would like to express my gratitude to my friends in Secure-IC and beyond who had been with me through thick and thin during this period.

Finally, I would like to express my heartfelt gratitude to my wife, my mother, my father and my brother, without the support of whom this journey would not have been made possible.

Abstract

The cybersecurity requirements and concerns have grown exponentially piggy-backing on the growing market of embedded and integrated systems rolling out in the market through many verticals including IoT, Industrial-IoT, smart cities, healthcare, automotive, armed forces, remote intelligence, etc. to name a few. The end devices have become more sophisticated over time to provide high-end user services. This has been the result of the advancements in the semiconductor industry by enabling high-end computations at the edge or end-nodes. However, the current market demands the capability of managing the devices remotely to enable many features such as firmware updates and remote life-cycle management. This high sophistication of the connected-device architecture in many forms including Vehicle-to-Anything (V2X) have also paved the way for threat actors to inject and exploit vulnerabilities that usually pass unseen from the verification phases or are introduced in the design because of distributed development culture in the semiconductor industry. To cope with the growing size of the attack surfaces and exposed vulnerabilities, it is pertinent to introduce cybersecurity verification mechanisms that identify and manage those weaknesses and vulnerabilities and save the equipment manufactures from obsoletion due to security issues as well as protect user data to maintain privacy and confidentiality. However, the size of data in most cases is also a big challenge and therefore the reliance on Machine Learning (ML) and Artificial Intelligence (AI) comes into play.

In this PhD manuscript, the challenges associated with the protection of different devices in terms of physical, internal threats in embedded/integrated designs, as well as network security in some cases, have been addressed. The focus of the work is to provide generic frameworks, through evaluation and experimentation, for different security challenges that are generally encountered in the embedded and integrated systems domain. Therefore, the work has been divided into three major areas viz. External Fault Injection detection, Hardware Trojan detection, and Intrusion Detection System (IDS) with a working example of automotive use-case in a simulated V2X architecture. The major contribution is in the system design and defense upgrade to enhance security of different types of devices from baremetal to the operating systems level while extending it to network level with server-side defense monitoring. To prove the effectiveness of the proposed architectures, design and validation/testing methodologies, results and comparisons with similar works have been presented for the proposed defense mechanisms.

Thesis summary

Chapter 1 – Introduction

The modern semiconductor industry allows the flexibility for a semiconductor company to develop all the parts of a device in-house by allowing the outsourcing of the tasks to which it does not hold the expertise in such as fabrication and packaging. The recent trend for chip manufacturers is to develop the design and produce the synthesizable RTL which then is forwarded to a partner industry for fabrication, packaging and testing services. This has created a massive outsourcing network in the entire industry where there are different companies focused on various stages in the VLSI design flow. This being said, there are two major outcomes of this distributed infrastructure viz. a) the production rate is significantly increased and the processing time is highly reduced as multiple stages can be carried out in parallel, and b) the vulnerability of tampering the design is increased manifold because most of the processing and different parts originate or are modified in third-party locations which can be considered insecure. As a result, we currently see a rise in the number of connected devices.

This modality of development and production has boosted the productivity of the connected devices but at the same time allowed for a rise in the hardware Trojan vulnerability among these devices and the risk of other threats associated with distributed development and production such as overproduction. To that end, it is essential to have robust countermeasures against the growing threats and also to have a strong defence mechanism that keeps up to the growing threat level in terms of detection accuracy. This means we are looking for smart solutions for both protection and evaluation that have low footprint to reduce overhead on the overall design as well as does not decrease production time with overly complicated security gates at each phase of the production. The amount of data handling is high and the detection is expected to have high accuracy with reliability over different types of data. The immediate answer that comes to mind is Artificial Intelligence (AI) and Machine Learning (ML). In this thesis work, the author propose different solutions to counter the various challenges that arise in the semiconductor development and manufacturing process by utilizing AI techniques. The solution for protection can range from reactive to proactive techniques viz. creating countermeasures at hardware as well as software level that actively resides inside the chip or the system to be protected, as well as evaluation methodologies that can be used to determine if any perturbation activity as fault injection or presence of a Hardware Trojan (HT) persists in a fabricated chip or chip batches. This topic is very timely since normalization in this respect is on-going (see e.g., the ISO/IEC DTR 5891.2[1]: Information security, cybersecurity and privacy protection – Hardware monitoring technology for hardware security assessment, which is in the draft technical report stage [1]).

[1]This upcoming standard discusses hardware monitoring aspects for security at the hardware layer in the post-silicon phase during runtime.

Contributions. This thesis focuses on the various challenges associated with the protection of embedded and integrated devices against physical, internal and network originating threats. To this end, three separate axes of work have been conducted:

The fist axis targets Fault Injection physical attacks at the chip level and the objective is to devise a detection methodology that can detect such attacks preemptively and report security incidents to the chip control to take remedial actions. For the first time it is demonstrated how aggregation of sensors can be used on-the-fly for abnormal conditions detection, with controlled false positive rate. This contribution is covered in the **chapter 3**.

The second work is concerning Hardware Trojans which is a serious practical concern and is an ever present danger in the current distributed development practice in the semiconductor industry. This work focuses on detecting the presence of extremely stealthy Hardware Trojans in the design, that are never triggered throughout the data acquisition process, with the help of Machine Learning modelling of the ElectroMagnetic (EM) emanation data collected from the active chip. It is shown that even tiny Hardware Trojans can be efficiently detected by this method, which leverages multidimensional (spatial & temporal) measurements. **Chapter 4** corresponds to the contributions in this axis of the thesis work.

The third axis of contribution is focused on threats in an Internet-of-Things (IoT) infrastructure. The V2X (Vehicle to anything) scenario is considered where an emulated vehicle is constructed with sensing capabilities and through an IoT communication protocol it is connected to a Server running some applications. The emulated vehicle consists of a sensing unit and detection units running on Machine Learning engines that are trained to detect sensor and network based intrusions or perturbations. Such system is shown to work very reliably even with modest computation power. **Chapter 5** presents the contributions from this axis of the thesis work.

Chapter 2 – Related works

This chapter includes a survey of different modalities in which embedded system security is assessed and averted in the state-of-the-art. Additionally, how machine learning techniques are used towards security at the hardware level is also discussed. All the topics involved in this thesis manuscript are studied for their security implications in embedded systems which include:

- Fault injection attacks
- Hardware Trojans
- Intrusion Detection Systems
- Edge to cloud systems
- AI and ML for chip and device security
- Communication protocols in IoT
- IoT related standards and regulations (e.g., [1])

With the growing use of AI in multiple applications including security analysis as well as, in some cases, security applications (with the major example being in the automotive industry for smart car navigation systems), it has become necessary to have standardized approaches to creating AI driven software or hardware applications with the aspect of security built-in the systems. There are already semiconductor manufacturers of AI based ASICs that are solely made for the purpose of Advanced Driver Assistance Systems (ADAS) or Automated Driving Systems (ADS) with AI powered image detection and processing blocks integrated within the System-on-Chip (SoC). The security certification aspects of such use cases solely rely on the existing security standards and schemes that are out of date compared to the parameters that come into play within the security boundary when such AI applications are included. The lack of standardized methodology is also the reason why the trust in AI is difficult to achieve. Additionally, it is also seemingly difficult to create a generalized framework for developing AI based security applications. To that end many certification bodies are working towards standardizing the use of AI in various applications and domains. A detailed aspect of all such efforts of AI standardization shall also be detailed in this thesis chapter.

Chapter 3 – External Fault Injection detection

The exposed devices in insecure locations are vulnerable to perturbation attacks that account for high severity in terms of threats arising from an adversary. These attacks comprising of Fault Injections (FI) in the chip or device through various means such as Electro-Magnetic (EMFI), Clock-Glitch (CGFI), voltage, temperature, Laser (LFI), Focused Ion-Beam (FIB), etc. are invasive and have the potential to disrupt the system as well as, with some precision. They allow the adversary to perturb specific areas within the chip that can allow unauthorized control switching such as granting access without authentication, authentication, or security bypass and in most cases lead to Denial-of-Service (DoS), all of which are catastrophic for the system when amplified as Distributed DoS on device classes. Indeed, these identified threats are relevant to the security critical applications such as the ones described in the general section. Therefore, through the introduction of multimodal attack detection IPs (such as digital sensors) within the design and a machine learning based inference unit within the chip, it is proposed to create a solution of interconnected system of perturbation sensitive sensors with an aggregation unit collating the inputs from all the digital sensors to feed an integrated AI module capable of predicting the presence of any FI related threats is proposed. The solution is tested experimentally with real data (captured from experimental benches) to prove the efficiency of such an approach.

Chapter 4 – Insider (Hardware Trojan) detection

Another area of perturbation of system behavior to maliciously achieve a targeted function, usually placed discretely by malicious third parties with an automated or manual trigger mechanism integrated in the design, which is originally unintended from the system functionality point of view, is the presence of internal system threats popularly known as Hardware Trojans (HT). The detection of HTs becomes more difficult (cf. ISO/IEC TR 5891) since the insertion can take place even in the last phases of the semiconductor manufacturing process. Thereby, in this work a ML based HT detection mechanism is proposed to detect different types of Trojans such as in the Register Transfer Level (RTL) or the Place and Route (P&R) level. The data

collected is through Electro-Magnetic (EM) cartography of a chip containing the Trojans (real data) and through precise ML engineering and classification it is shown that it is possible to identify the presence of even a tiny (down to 0.1%) HT with this verification process. Several methodologies are applied and tested on the large data sets and the effectiveness in detecting the HTs are demonstrated through experimented inferences and compared with existing state-of-the-art research.

Chapter 5 – Intrusion Detection System

Another aspect of external perturbation is at the device level by conflicting the system stability through adversarial attacks. A popular use-case is autonomous smart cars in the V2X (Vehicle to anything) infrastructure (see risk analysis in Common Criteria (CC)[2] Protection Profile PP0114 [3]). To this end, it is highly critical to secure such devices as human lives are involved in case of adversary induced perturbation such as failure of braking systems, unwanted acceleration or stopping, etc. The functional safety aspect of automotive development indeed tackles failures such as through ISO 26262 standard. However, the cybersecurity aspect is still relied upon classical security mechanisms and a strong cybersecurity culture in automotive development. Therefore, a more advanced form of an intelligent system is required such as the one proposed in this thesis through an AI capable smart IDS. The IDS proposed is a parallel architecture of AI engines that are trained on real data collected from a miniature smart car model with built-in sensor mechanisms mimicking a real smart-car. The AI based IDS is capable of detecting perturbations received at the sensor level as well as intrusions from the network level. To extend the security concept beyond V2X to the whole IoT segment, the smart car is connected to a server that runs a monitoring and a firmware upgrade application through a standard IoT protocol. Through many data collection sessions of both nominal as well as perturbed conditions the AI models are trained and are shown to achieve good results in terms of detection accuracy.

Résumé de la thèse

Chapitre 1 – Introduction

L'industrie moderne des semi-conducteurs offre à une entreprise de semi-conducteurs la flexibilité de développer toutes les parties d'un dispositif en interne en permettant l'externalisation des tâches pour lesquelles elle ne possède pas l'expertise, telles que la fabrication et l'emballage. La tendance récente pour les fabricants de puces est de développer la conception et de produire le RTL (abbréviation de "Register Transfer Level") synthétisable qui est ensuite transmis à une industrie partenaire pour les services de fabrication, de conditionnement et de test. Cela a créé un vaste réseau d'externalisation dans l'ensemble du secteur, où différentes entreprises se concentrent sur différentes étapes du flux de conception VLSI (abbréviation de "Very Large Scale Integration"). Cela étant dit, cette infrastructure distribuée a deux conséquences majeures, à savoir. a) le taux de production est considérablement augmenté et le temps de traitement est fortement réduit car plusieurs étapes peuvent être effectuées en parallèle, et b) la vulnérabilité de falsification de la conception est considérablement accrue car la plupart du traitement et les différentes pièces proviennent ou sont modifiées dans des établissements tiers qui peuvent être considérés comme non sécurisés. D'autre part nous constatons actuellement une augmentation du nombre d'appareils connectés qui utilisent des circuits VLSI.

Cette modalité de développement et de production a augmenté la productivité des appareils connectés, mais a en même temps permis une augmentation de menaces potentielles associées au développement et à la production distribués, telles que les chevaux de Troie matériels (HT, abbréviation de "Hardware Trojans") ou la surproduction illégale. Pour améliorer la sécurité face à ces menaces croissantes, il est essentiel de disposer de contre-mesures robustes et/ou d'un mécanisme de détection de menaces solides notamment en termes de précision de détection. Il faut donc rechercher des solutions efficaces pour la protection et l'évaluation mais aussi des solutions de faible complexité afin de réduire les coûts de conception sans impacter le temps de production. Comme la quantité de données traitées est élevée, la détection devrait avoir une grande précision et fiabilité quelque soit le type et la taille des données. Pour satisfaire cette exigence, une solution envisagée est l'emploi d'intelligence artificielle (IA), plus précisément l'apprentissage automatique (ML, abbréviation de "Machine Learning"). Dans ce travail de thèse, je propose différentes solutions pour relever les différents défis qui se posent dans le processus de développement et de fabrication des semi-conducteurs en utilisant des techniques d'IA. La solution de protection peut utiliser des techniques réactives (détection) ou proactives (prévention). Les techniques de prévention reposent sur le rajoût de contre-mesures au niveau matériel ou logiciel offrant une résilience à l'intérieur de la puce ou du système à protéger. Les techniques de détection nécessitent des méthodologies d'évaluation qui peuvent être utilisées pour déterminer si une activité de perturbation, telle qu'une injection de fautes (détecton à la volée) ou la présence d'un cheval de Troie matériel (détection à la

volée ou avant utilisation) a lieu dans une puce. Ce sujet est d'actualité puisque la normalisation à cet égard est en cours (voir par exemple, la norme ISO/IEC DTR 5891.2[2] : Sécurité de l'information, cybersécurité et protection de la vie privée – Technologie de surveillance du matériel pour l'évaluation de la sécurité du matériel, qui est au stade du projet de rapport technique [1]).

Contributions. Cette thèse se concentre sur les différents défis associés à la protection des appareils embarqués et intégrés contre les menaces physiques, internes et provenant du réseau. Pour cela, trois axes de travail distincts ont été menés :

Le premier axe cible les attaques physiques par injection de fautes (FI, abbréviation de "Fault Injection") au niveau de la puce et l'objectif est de concevoir une méthodologie de détection capable de détecter de telles attaques de manière préventive et de signaler les incidents de sécurité au contrôle de la puce pour prendre des mesures correctives. Pour la première fois, il est démontré comment l'agrégation de capteurs peut être utilisée à la volée pour la détection de conditions anormales, avec un taux de faux positifs contrôlé. Cette contribution est couverte dans le **chapitre 3**.

Le deuxième travail concerne les HT, qui constituent un problème pratique sérieux et un danger toujours présent dans les pratiques actuelles de développement distribué dans l'industrie des semi-conducteurs. Ce travail se concentre sur la détection de la présence de chevaux de Troie matériels extrêmement furtifs dans la conception, qui ne sont jamais déclenchés tout au long du processus d'acquisition de données, à l'aide d'une modélisation d'apprentissage automatique des données d'émanation électromagnétique (EM) collectées à partir de la puce active. Il est démontré que même les plus petits chevaux de Troie matériels peuvent être détectés efficacement par cette méthode, qui exploite des mesures multidimensionnelles (spatiales et temporelles). **Chapitre 4** correspond aux contributions dans cet axe du travail de thèse.

Le troisième axe de contribution est axé sur les menaces dans une infrastructure Internet des Objets (IoT, abbréviation de "Internet-of-Things"). Le scénario V2X (abbréviation de "Vehicle to everything") est envisagé dans lequel un véhicule émulé est construit avec des capacités de détection et via un protocole de communication IoT, il est connecté à un serveur exécutant certaines applications. Le véhicule émulé se compose d'une unité de détection et d'unités de détection fonctionnant sur des moteurs d'apprentissage automatique qui sont formés pour détecter les intrusions ou les perturbations basées sur les capteurs et le réseau. Il est démontré qu'un tel système fonctionne de manière très fiable, même avec une puissance de calcul modeste. **Chapitre 5** présente les contributions de cet axe du travail de thèse.

Chapitre 2 – Œuvres liées

Ce chapitre comprend une étude des différentes modalités dans lesquelles la sécurité des systèmes embarqués est évaluée et pratiquée dans l'état de l'art. De plus, la manière dont les techniques d'apprentissage automatique sont utilisées pour la sécurité au niveau matériel est également abordée. Tous les sujets impliqués dans ce manuscrit de thèse sont étudiés pour leurs implications en matière de sécurité dans les systèmes embarqués, notamment :

[2]Cette prochaine norme traite des aspects de surveillance du matériel pour la sécurité au niveau de la couche matérielle dans la phase post-silicium pendant l'exécution.

- Attaques par injection de fautes
- Chevaux de Troie matériels
- Systèmes de détection d'intrusion
- Systèmes Edge à Cloud
- IA et ML pour la sécurité des puces et des appareils
- Protocoles de communication dans l'IoT
- Normes et réglementations liées à l'IoT (par example [1])

Avec l'utilisation croissante de l'IA dans de multiples applications, y compris l'analyse de sécurité ainsi que, dans certains cas, les applications de sécurité (le principal exemple étant dans l'industrie automobile pour les systèmes de navigation intelligents), il est devenu nécessaire d'avoir des approches standardisées pour créer de l'IA. applications logicielles ou matérielles pilotées avec l'aspect de sécurité intégré aux systèmes. Il existe déjà des fabricants de semi-conducteurs d'ASIC basés sur l'IA qui sont uniquement conçus pour les systèmes avancés d'aide à la conduite (ADAS, abbréviation de "Advanced Driver Assistance Systems") ou les systèmes de conduite automatisée (ADS, abbréviation de "Automated Driving Systems") avec des blocs de détection et de traitement d'images alimentés par l'IA intégrés dans le système sur puce (SoC, abbréviation de "System-on-Chip"). Les aspects de certification de sécurité de tels cas d'utilisation reposent uniquement sur les normes et systèmes de sécurité existants qui sont obsolètes par rapport aux paramètres qui entrent en jeu dans les limites de sécurité lorsque de telles applications d'IA sont incluses. Le manque de méthodologie standardisée est également la raison pour laquelle la confiance dans l'IA est difficile à instaurer. De plus, il semble également difficile de créer un cadre généralisé pour développer des applications de sécurité basées sur l'IA. À cette fin, de nombreux organismes de certification s'efforcent de normaliser l'utilisation de l'IA dans diverses applications et domaines. Un aspect détaillé de tous ces efforts de normalisation de l'IA sera également détaillé dans ce chapitre de thèse.

Chapitre 3 – Détection d'Injection de Fautes Externe

Les appareils exposés dans des emplacements non sécurisés sont vulnérables aux attaques de perturbation qui représentent une menace très puissantes pour retrouver un secret. Ces attaques comprennent des FI dans la puce ou le dispositif par divers moyens tels que le rayonnement électromagnétique (EMFI, abbréviation de "ElectroMagnetic Fault Injection"), le parasitage de l'horloge (CGFI, abbréviation de "Clock-Glitch Fault Injection"), la tension, la température, le laser (LFI, abbréviation de "Laser Fault Injection"), le faisceau d'ions focalisé (FIB, abbréviation de "Focused Ion-Beam"), etc. sont invasifs et peuvent potentiellement perturber le système, avec une certaine précision. Ils permettent à l'adversaire de perturber des zones spécifiques de la puce qui peuvent permettre une commutation de contrôle non autorisée, comme l'octroi d'un accès sans authentification, authentification ou contournement de sécurité, et conduisent dans la plupart des cas à un déni de service (DoS, abbréviation de "Denial-of-Service"), qui sont tous catastrophiques pour le système lorsqu'il est amplifié en tant que DoS distribué sur les classes d'appareils. En effet, ces menaces concernent les applications critiques pour la sécurité telles que celles décrites dans la section générale. Par

conséquent, grâce à l'introduction de blocs de détection d'attaques multimodales (telles que des capteurs numériques) dans la conception, et l'ajoût d'une unité d'inférence basée sur l'apprentissage automatique au sein de la puce, il est proposé de créer une système interconnecté de capteurs sensibles aux perturbations avec une unité d'agrégation. Ce système permet la collecte des données de tous les capteurs numériques pour alimenter un module d'IA intégré capable de prédire la présence de toute menace liée à la FI. La solution est testée expérimentalement avec des données réelles (capturées sur des bancs expérimentaux) pour prouver l'efficacité d'une telle approche.

Chapitre 4 – Détection d'initiés (chevaux de Troie matériels)

Les chevaux de Troie matériels (HT) permettent de perturber le comportement du système pour atteindre de manière malveillante une fonction ciblée, généralement. Le HT est placé discrètement par des tiers malveillants avec un mécanisme de déclenchement automatisé ou manuel intégré dans la conception. La détection des HT devient plus difficile (cf. ISO/IEC TR 5891) puisque l'insertion peut avoir lieu même dans les dernières phases du processus de fabrication des semi-conducteurs. Ainsi, dans ce travail, un mécanisme de détection HT basé sur les techniques ML est proposé pour détecter différents types de chevaux de Troie, comme au niveau RTL ou P&R. Les données collectées proviennent de la cartographie EM d'une puce contenant les chevaux de Troie (données réelles) et grâce à une ingénierie et une classification ML précises, il est démontré qu'il est possible d'identifier la présence même d'un infime (jusqu'à 0,1%) HT avec ce processus de vérification. Plusieurs méthodologies sont appliquées et testées sur de grands ensembles de données et l'efficacité de la détection des HT est démontrée par des inférences expérimentées et comparée aux recherches de pointe existantes.

Chapitre 5 – Système de Détection d'Intrusion

Un autre aspect de la perturbation externe est de compromettre la stabilité d'un système interconnecté par le biais d'attaques contradictoires. Un cas d'utilisation populaire est celui des voitures intelligentes autonomes dans l'infrastructure V2X (voir l'analyse des risques dans CC[2] Profil de protection PP0114 [3]). À cette fin, il est obligatoire de sécuriser ces dispositifs, car des vies humaines sont impliquées en cas de perturbation induite par un adversaire, telle qu'une défaillance des systèmes de freinage, une accélération ou un arrêt non désiré, etc. L'aspect sécurité fonctionnelle du développement automobile s'attaque en effet aux défaillances telles que Norme ISO 26262. La voiture étant un élement d'un système interconnecté, il est devenu indispensable de transposer la culture de "cybersécurité" dans ce domaine. Par exemple, une forme de sécurisation du système autonome peut reposer sur un système de détection d'intrusion (IDS) intelligent à base d'IA comme proposé dans cette thèse. L'IDS proposé est une architecture parallèle de moteurs d'IA entraînés sur des données réelles collectées à partir d'un modèle de voiture intelligente miniature avec des mécanismes de capteurs intégrés imitant une véritable voiture intelligente. L'IDS basé sur l'IA est capable de détecter les perturbations reçues au niveau des capteurs ainsi que les intrusions au niveau du réseau. Pour étendre le concept de sécurité au-delà du V2X à l'ensemble du segment IoT, la voiture intelligente est connectée à un serveur qui exécute une application de surveillance et de mise à niveau du micrologiciel via un protocole IoT standard. Grâce à de nombreuses sessions de collecte

de données dans des conditions nominales et perturbées, les modèles d'IA sont entraînés et obtiennent de bons résultats en termes de précision de détection.

Acronyms

ADAS Advanced Driver Assistance Systems. 30

ADS Automated Driving Systems. 30

AES Advanced Encryption Standard. 11, 18, 29

AI Artificial Intelligence. 7, 29–32, 65

ANN Artificial Neural Networks. 22, 24

API Application Programming Interface. 22, 70

ARP Address Resolution Protocol. 67, 68, 72

ASIC Application Specific Integrated Circuit. 16, 24, 66

CAD Computer Aided Design. 16

CC Common Criteria. 11, 30, XIV, XVIII

CEM Common Evaluation Methodology. 11

CGFI Clock-Glitch Fault Injection. 6, 12, 33, 36, 42–46, 49, 67, 78, IV

CNN Convolutional Neural Networks. 22

CoAP Constrained Application Protocol. 21, 65, 70, 75, 79

CPU Central Processing Unit. 16, 20, 68

CSP Critical Security Parameter. 1, 16, 20

CWE Common Weakness Enumeration. 10

DFA Differential Fault Analysis. 67

DNN Deep Neural Networks. 22, 53

DoS Denial of Service. 2, 5, 12, 16, 67, 72, 73, 75, 78, XXVI

DS Digital Sensor. 34–36, 39–41, 43, 45–49, XXV

DTC Decision Tree Classifier. 53, 59, 61

DTLS Datagram Transport Layer Security. 21

DUT Design-Under-Test. 34, 36, 46, 50, 53, 54, 64

DVFS Dynamic Voltage and Frequency Scaling. 10

EDA Electronic Design Automation. 4, 14, 16, 46

EE Elliptical Envelope. 54, 62

EM Electro-Magnetic. 18, 29, 34, 35, 41, 43, 55–57, 59, 61, 64, 78, XIV, XVIII, XXV, XXVI

EMFI Electro-Magnetic Fault Injection. 6, 11, 12, 33–37, 40–46, 49, 52, 55, 67, 78, IV, XXV

FBBI Forward Body Biasing Injection. 6

FFT Fast Fourier Transform. 70

FIA Fault Injection Attacks. 10, 20, 27, 34–37, 39, 41–46, 48–50, 67, 78–80, XXV

FIB Focused Ion Beam. 6, 30

FPGA Field Programmable Gate Array. 16, 18, 29, 34, 46, 48, 54–56, 59

FSM Finite State Machine. 22

GNBC Gaussian Naive Bayes Classifier. 22, 37, 38, 40–42, XXV

GPU Graphics Processing Unit. 24

HDL Hardware Descriptive Language. 16, 47, 56

HIDS Host-Based Intrusion Detection System. 18, 65, 70, 74, 75, XXVI

HLS High-Level Synthesis. 33, 46–48

HSM Hardware Security Module. 10

HT Hardware Trojan. 4, 7, 12–16, 28, 29, 51–54, 58–64, 78–80, III, IV, XXV, XXVI

I2C Inter-Integrated Circuit. 69

IAM Identity Access Management. 21

IC Integrated Circuit. 4, 11

ICMP Internet Control Message Protocol. 70

IDS Intrusion Detection System. 6, 18, 29, 65–71, 74, 75, 79, III, XXVI

IF Isolation-Forest. 54, 62, 70

IoT Internet of Things. 2, 7, 18, 65, 71, 75, 78, 79, III, XXVI

IP Internet Protocol. 67

IR Infra-Red. 69

iSE Integrated Secure Element. 1, 20

LDR Light-Dependent Resistor. 69

LFI Laser Fault Injection. 6, 12, 67

LOF Local Outlier Factor. 54, 62

LR Logistic Regression. 22

LRC Logistic Regression Classifier. 37, 38, 59, 61

LSTM Long Short-Term Memory. 79

LwM2M Light-weight Machine-to-Machine. 21

MAC Media Access Control. 67, 68

MCU Micro-Controller Unit. 20, 66, 75

MDP Markov Decision Process. 25

MIMT Man-in-the-Middle. 67

ML Machine Learning. 1, 5–7, 18, 22–31, 33, 34, 36–41, 43, 45–47, 49, 52, 54, 58–66, 69–71, 75, 77–80, III, IV, XXV, XXVI

MLP Multi-Layered Perceptron. 37, 38, 40, 52, 53, 59, 61, XXV

MPFI Micro-Probing Fault Injection. 6

MPU Micro-Processor Unit. 20

MQTT Message Queuing Telemetry Transport. 21

NIDS Network-Based Intrusion Detection System. 18, 65, 69, 70, 72, 73, 75, XXVI

NIST National Institute of Standards and Technology. 30

NN Nearest Neighbor. 31, 38, 53, 59, 61

NVM Non-Volatile Memory. 1

OEM Original Equipment Manufacturer. 5, 16

OMP Orthogonal Matching Pursuit. 22

OS Operating System. 7, 21, 66, 68, 75, 79

OSI Open System Interconnection. 21

P&R Place and Route. 4, 16, 51, 56, XIII, XVIII

PCA Principle Component Analysis. 27, 70

PPA Performance, Power and Area. 10

RAM Random Access Memory. 47

RBF Radial Basis Function. 53, 54

RO Ring Oscillator. 29

RSA Rivest-Shamir-Adleman. 11

RTL Register Transfer Level. 14, 28, 47, 51, 80, XIII, XVIII

SCA Side-Channel Analysis. 6, 20

SCR Smart-car Robot. 66, 68, 69, 74, XXVI

SE Secure Element. 20

SFR Security Functional Requirement. 4

SoC System-on-Chip. 3, 10, 20, 33, 65

SQL Structured Query Language. 22

SVM Support Vector Machines. 18, 22, 29, 37, 38, 52–54, 59, 61, 62

TCP Transmission Control Protocol. 72, 75

TCP/IP Transmission Control Protocol/Internet Protocol. 68

TEE Trusted Execution Environment. 1, 10, 20

TFI Temperature Fault Injection. 6

TLS Transport Layer Security. 21

TOE Target of Evaluation. 12, 30

TPM Trusted Platform Module. 20

TPU Tensor Processing Unit. 25

USS Ultra-Sonic sensor Spoofing. 74, 75

V2X Vehicle-to-Anything. 65, 66, 68, 71, 79, XXVI

VGFI Voltage-Glitch Fault Injection. 6

WLAN Wireless Local Area Network. 69

XSS Cross-Site Scripting. 22

List of Figures

Chapter 1

Introduction

Contents

1.1 Context

Security and privacy have been a big issue from the time the internet was invented. Since then the concerns have only grown, as attested by new regulations (e.g., NIST SP 800-193 [4] and the forthcoming EU Cyber Resilience Act [5]) and the increasing need of sovereignty in the silicon market. As technology becomes more sophisticated, more methods to fault such systems appear through all the three modalities of actors i.e. white-hat, black-hat and gray-hat originating from state-of-the-art vulnerability analysis research, security testings, and adversarial means. To that end, many cryptographic algorithms and protocols have been designed and globally accepted through standards organizations to tackle cybersecurity threats and issues in various scenarios, such as:

- Critical Security Parameter (CSP) storage and transfer in/from embedded or Integrated Secure Element (iSE) or Trusted Execution Environment (TEE) to the other parts of the chip
- Critical and sensitive memory area isolation
- Secure booting of security critical systems
- Secure retrieval and storage of data in and from Non-Volatile Memory (NVM) and other external memories

- Intra-device communication
- Inter-device communication
- Network communication
- Private area network (corporate and public)
- Device to edge/server
- Heterogenous protocol communication
- Peer to peer communication
- Industrial Internet of Things (IoT) communication
- Connected smart healthcare devices to communicate medical data
- and so on.

Not surprisingly, many of these transactions are highly confidential due to their nature, such as that of governmental organizations, critical infrastructure (gas, power grid, smart city, satellite, maritime, etc.), state and private owned agencies for public service, healthcare, etc. and therefore, requires high grade security infrastructure for information interchange and operation.

While thinking of security the most important steps are to identify the assets involved and then the identification of attack surfaces and the attack scenarios in which the likelihood of any threat is theorized. For any connected device systems, the first point of infraction is the device itself which are in-field and highly accessible to any public user or adversary. While the most import aspects of security involve Confidentiality, Integrity and Authenticity (CIA) of the transacted data, there are other aspects involved that is highly critical for the device and chip manufacturers as well as OEMs, such as:

- Reverse engineering
- Compromising the device through malicious modifications in order to lead to Denial of Service (DoS)
- Sniffing of confidential non-user data through back-doors
- Injection of malware

This not only compromises a single device, but it has the potential to disrupt the entire production lot leading to decommissioning of the chips and or devices incurring heavy costs through chip re-spinning. This not only consumes time and currency, but also impacts heavily on the market trust of the company and therefore, has the potential to lead to the major damage impacts on the company such as Safety, Financial, Operational & Privacy (SFOP) [6]. Additionally, through reverse engineering attacks as well as overproduction due to outsourced production, counterfeit technologies disrupt global supply chain and market and this has become a serious concern to the market players.

1.2 Problem definition

As described in the context, the severity of cybersecurity threats and challenges are not limited to a particular use case and therefore there has been several attempts in security science to manage and mitigate them. As a result of the different challenges faced in the hardware, software, firmware, network and server levels, there are non-profit organizations such as *MITRE* [7, 8] that provide vulnerabilities and weakness database to properly categorize and maintain traceability of such threats for the purpose of improvement in security of the products and applications. However, by only maintaining the database of events and individual resolutions are not sufficient to encounter the threat origins and reduce the numbers and impacts caused by such threats. Therefore, in this thesis is targeted towards providing real-world impacting solutions to generalize as much as possible several security domains to anticipate and mitigate security threats through proactive and reactive approaches.

To provide a broader aspect of development of embedded or integrated systems, the entire life-cycle of semiconductor process should be studied. Figure 1.1 provides a high level scenario of the entire development, deployment and decommissioning life-cycle stages in the semiconductor manufacturing process to understand the threat impacts on each stages.

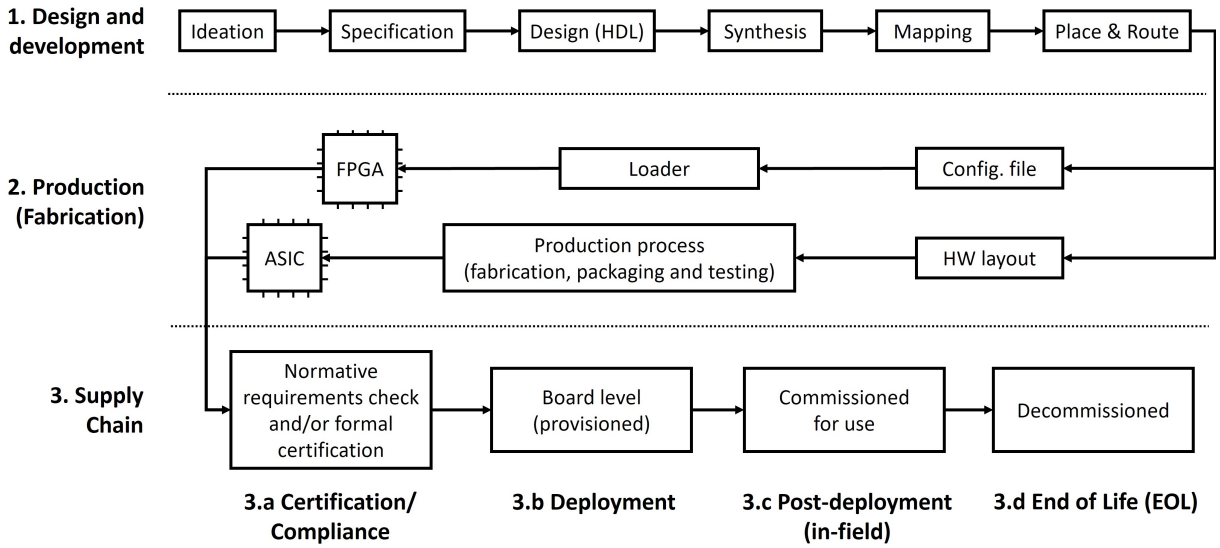

Figure 1.1: High level semiconductor development life-cycle

The Figure 1.1 presents three main phases of development:

1. Design and development
2. Production
3. Supply chain

Let us look at each of these phases individually to understand the various security challenges associated with the same. For the purpose of understanding the challenges within the development process, we shall dub the different phases with either ”secure” or ”insecure”, depending upon their locality in the development process flow and the trust of the people involved. For instance, the ”ideation” phase is within the System-on-Chip (SoC) developer’s premises and therefore, would be considered as secure.

1.2.1 Design and development

The preliminary activity of any semiconductor product is the ”idea” where the functional requirements of the product are envisioned. This process is performed by the principal people involved in the product development and therefore, this activity is considered to be secure.

The next step is to design the specification of the functional and Security Functional Requirement (SFR)s with the architecture of the chip or secure element. This step is also considered safe because the specifications directly map to the idea and all the people involved in the idea phase work closely to realize the specification and the architectural design. However, if security aspects are involved, there is a chance of human error to introduce vulnerabilities, even if security design principles are followed. Therefore, this step is slightly insecure with some chances of vulnerability or weakness introduction in the product.

The design and synthesis steps are very prone to cause security abnormalities in the product since in today's industry, Electronic Design Automation (EDA) tools are widely used that could be subject to bugs and vulnerabilities that may automatically introduce design flaws or vulnerabilities that could be exploited in later phases. Additionally, there is a slight chance of the presence of adversarial actors in this stage that might introduce stealthy malicious design modifications that evades the regular testing phases and could aid in the back-door related attacks in the later stages of the product life-cycle. Thereby, this step is slightly more insecure as compared to the previous step.

Finally, in the Place and Route (P&R) step, it is possible to easily introduce some additional logic, remove some wires or logic, or modify certain logical behaviour in the design to enable an adversary to exploit these changes to launch and attack or steal information.

1.2.2 Production

This phase is entirely considered insecure since all the steps involved are in an insecure location, usually outsourced to a third-party company or establishment, where it is exceedingly easy to introduce abnormalities in the design in the form of back-doors or Hardware Trojan (HT) circuits, as well as copy designs for counterfeiting, and even overproduction.

Therefore, it is a big challenge in today's semiconductor production process to ensure the integrity of the design and protect the interests of the Integrated Circuit (IC) vendors.

1.2.3 Supply-Chain

The go-to-market is facilitated through the supply chain. A crucial aspect that plays a significant role in the security of the product up to the production aspect with all the sites involved in the process is the certification and regulatory compliance verification. The compliance to normative requirements from standards and schemes may comprise in:

- Verification of design and architecture
- Verification and review of the source code to ensure security principles are respected and required security assurance is reached through sufficient security mechanisms
- Development process validation through gap analysis and audits
- Standard testing requirements for full coverage of all security features and protection of all assets within the sub-system

- Threat analysis and/or vulnerability assessment

However, there are several levels of security levels and assurances and the requirement of higher levels is subject to the market needs, Original Equipment Manufacturer (OEM) and OEM's customer's requirements, as well as governmental regulations. And whilst this provide high assurance to identify potential vulnerabilities in the product, there scope is limited and it does not ensure a hundred percent security against these threats.

Therefore, to address these challenges, different strategies and solution to mitigate, with minor impacts on design, process, and cost, are presented in this thesis and the applicability is shown with real world use-cases while comparing the results to existing state-of-the-art solutions, and also paving a path to further improvements through research and development in this direction to unify the solution utilizing Machine Learning (ML) following compliance to security standards and regulations for secure product development. The major areas covered in this thesis are briefly described in the following sections of this chapter.

1.3 External Fault Injection

Perturbation attacks are one of the most disruptive physical attacks on any system or chip. The principle of any external fault injection attack also known as perturbation attacks relies in using external sources such as precisely calculated electro-magnetic or laser probes to inject external pulses to interfere with the internal signal transmission within the chip to interrupt or modify the internal signal behaviour. The intention behind such attacks could be to either:

- Disrupt the functioning of the chip's subsystem
- Perform precise modifications in the signals to change the information it carries
- Alter stored information within memory elements such as registers

Indeed, such attacks with a precise purpose are difficult by nature as locality of reference on the chip as well as timing are most crucial factors to launch such attacks. Therefore, not only the attacker needs to know about the target's architectural planning on the chip fabric, but also in the precise time which operation is happening in order to not miss the window of opportunity. This would also call for utilizing sophisticated instruments designed for precise experimentation purposes. The intent of such attacks could be characterized broadly as to:

- create DoS to shut down the system
- bypass security checks
- shut down the security protocol within the chip to facilitate coordinated attacks
- allow unauthorized access to restricted areas on the chip
- alter original functionality of the logic to inject abnormalities that would create access points for adversarial attacks
- retrieve sensitive information by modeling or profiling based on behaviour
- to demonstrate system weakness against such attacks ([1])

[1] such attacks are usually the motivation of academic research to help make the solution more robust

There are various methods in which a perturbation or external fault injection attack could be realized and over the years many new techniques have been proposed for the same. A list of such attack modalities [9, §5.4] is provided below:

- Electro-Magnetic Fault Injection (EMFI)
- Clock-Glitch Fault Injection (CGFI)
- Laser Fault Injection (LFI)
- Voltage-Glitch Fault Injection (VGFI)
- Temperature Fault Injection (TFI)
- Focused Ion Beam (FIB)
- Micro-Probing Fault Injection (MPFI)
- Forward Body Biasing Injection (FBBI)

1.4 Hardware Trojans

As described in section 1.2, there are many instances in the semiconductor development lifecycle where, through human or tool error or malicious intent, it is possible to make slight modifications in the design that go unverified through the testing phases and persist within the design. These modifications could be either known to some adversarial agents or could be discovered post production. In either case, they can lead to exploitation of the system and thereby compromise and disrupt the supply chain. Hardware Trojans pose one of the major security risks which, if escalated at an international level, may trigger global situational crisis and affect trade between nations. Such an example was the news reported by Bloomberg in 2018 [10].

1.5 Intrusion Detection System

An Intrusion Detection System (IDS) refers to any system or its ability to detect incoming threats. Usually it is referred to those existing at the network level. In the classical approaches, intrusions were detected based on anomalous behaviour through classical filtering techniques which were usually comparative in nature. In this thesis, the focus of introducing IDS is to show how the well known concept of the latter can be utilized in more advanced and sophisticated systems such as in a V2X infrastructure and at the core having a ML operated engine to continuously monitor, analyze and report intrusions that can originate from many sources at the same time or at different times.

1.6 Countermeasures using ML

The use of ML is not new in security analysis and for over the past decade, meany researchers have attempted and proposed different security deductive analysis methodologies assisted by ML such as profiled Side-Channel Analysis (SCA) on hardware systems. While many works exist in utilizing ML to perform data segmentation and analysis, only few proposed works actually utilize it for the purpose of enhancing the security itself.

In this thesis, the preventive and protective countermeasures, against threats targeted at embedded systems, using ML are proposed and demonstrated to work significantly better compared to classical approaches with higher efficiency in terms of aggregated multiple different types of threats at the same time and also providing smart on-board analysis such as classification. With advanced remote life-cycle management capabilities, it is also shown to incorporate continuous integration through remotely monitoring and updating the parameters of detection.

1.7 Objectives

The objectives of the thesis include study of different challenges in embedded and integrated systems security and propose countermeasures with respect to proactive as well as reactive approaches utilizing Artificial Intelligence (AI) and ML for not only statistical analysis and data segregation but also towards the active detection and reporting with enhanced state-of-the-art capabilities such as secure firmware update over the air.

Objective 1. Modelling of embedded system security problems into machine learning models

Objective 2. Unify detection of fault injection attacks in embedded systems using modular machine learning engines for easy integration into designs

Objective 3. Generalize Hardware Trojan detection methodology to easily determine differences in an unknown design model either with or without a golden (reference) model for Trojans inserted at various levels of the design

Objective 4. Show how using machine learning based Intrusion Detection Systems perform efficiently in IoT networks

Objective 5. Prove efficiency of all proposed methods through real-world use-case examples

Objective 6. Demonstrate the usability of machine learning in solving security challenges

Objective 7. Provide analysis of certifications and regulations towards standardization of the usage of machine learning in security applications

Objective 8. Provide background on the problem statement concerning security in embedded systems

This thesis work is broadly divided into three major contributions targeting from the baremetal chip level, up to the system or Operating System (OS) level connected with a server.

C1 **The first contribution** is focused on mitigating external perturbation threats at the hardware level through on-board sensing and smart detection even if the perturbation arises from multiple sources. This contribution is presented in the chapter 3.

C2 **The second contribution** is about HT detection at post-silicon stage as a proactive approach, presented in chapter 4. The evaluation methodology is also proposed to integrate within the testing phases of the manufacturing process to not have big impacts on the supply chain time-to-market.

C3 Finally, **the third contribution** is focused on providing on-board detection of intrusions originating from multiple sources. It is covered in the chapter 5.

All the contributions in this thesis manuscript are demonstrated with real world emulated data and, therefore, exist as proofs of concepts. The contributions are also published as scientific research papers. Table 1.1 presents the objectives of this thesis covered by the contributions and their corresponding chapter, with the associated research publications. There were additional research activities performed during this thesis duration that are not covered in this manuscript but the associated publications of the same could be found in the section 6.5.

Contribution	**Objectives**	**Chapter**	**Associated Publications**
C1	1, 2, 5, 6	3	[11, 12]
C2	1, 3, 5, 6	4	[13, 14]
C3	1, 4, 5, 6	5	[15]
-	7, 8	2	-

Table 1.1: A mapping between the objectives and contributions in this thesis manuscript with associated chapters and research publications.

Chapter 2

Related Works

Contents

2.1 Introduction to Embedded Systems security

Characterization of faults in embedded systems has been studied for a long time and thereby deeming it pertinent to ensure security of the digital connected devices [16]. This is seemingly important since the growth in the number of connected devices and their popularity in use of almost every smart product in any market domain, has been massive. Even at the consumer level, people would want to purchase devices that are smart and could be connected with their smartphone or home/office intra-network, and are future proof with upgrade or update options. While these devices offer very high usability, they achieves so through a labyrinth of inter-communications and through those communication channels flow sensitive data. The global smart connected device manufacturing is also characteristic of the advancement in the integrated circuit technology that is providing the means to increase capabilities of these devices by enhancing the processing capabilities allowing data to be processed on the edge rather than channeled to the server. Even with the processing on the node or edge side, a seamless communication to the server is maintained, thanks to the advancements in the wireless communication infrastructure. This sophistication brings in major challenges for cybersecurity providers as it becomes more easy for threat actors to play around with different attacks due to the high availability of the system in adversarial location.

With the onset of serious concerns about cybersecurity, the main focus was towards the software. While the research in this direction progressed, it was realized that hardware attacks are more exploitable. Going forth many the attack methodologies at hardware level evolved and many vulnerability analysis techniques of the hardware design and implementation was

standardized for commercial products under various certification schemes. Till this date, every hardware manufacturer deals with secure implementation issues and therefore, compliance validation and testing has become one of the major steps in the manufacturing of SoCs. In 2021 the *MITRE* Common Weakness Enumeration (CWE) [8] released the first ever list of most important hardware weaknesses [17] to spread global awareness of the hardware vulnerabilities in different products.

The main topics or areas of interest in this thesis as defined in section 1.7 is presented in the following sections.

2.2 Fault Injection attacks

Fault Injection Attacks (FIA) can be generally described as an external forced stimulus in the hardware of a system, functioning within its operational environment, to cause abnormal changes in the behaviour of the system software. These attacks are motivated to achieve certain goals as described in the section 1.3. Such attacks are termed as faults for the reason that they tend to fault the internal signals or memory locations at precise timing and location, respectively, to achieve the attack objectives. These faults can be ephemeral or transient in nature, thereby existing for only a certain period of time such as few calculated clock cycles (depending upon various parameters such as attack strength or duration of attack), or they can persist for indefinite amount of time within the system such as a bit flip in a target register. What makes these attacks interesting is the fact that a modification at the hardware layer can trigger a change in the system behaviour at the software level. For instance, tampering the control flow register can compromise the integrity of the system-level program execution flow.

Due to their disruptive nature and capability to impact multiple layers, FIAs pose a high risk in security sub-systems, Hardware Security Module (HSM), or cryptographic modules. As a result, the major focus of security engineering in chip manufacturing business is to come up with countermeasures to tackle such fault injections. Having an overhead of Performance, Power and Area (PPA) to meet market expectations is another optimization topic which the industry is struggling in. The magnanimity of fault attacks is also evident from the fact that most compliance checks for security certifications, such as the widely acclaimed AVA_VAN (Vulnerability Analysis from the AVA class of Common Criteria Certification scheme [2]), imply stronger focus on system's security robustness against FIA.

FIA have been around for quite some time now and have been extensively used to cause glitches in the Integrated Circuits (IC) to eventually extract the secret parameters or cause malfunction to the system. One of the early fault analysis was done in [18]. Several vulnerabilities exposed due to fault attacks have been reported along with their countermeasures like in [19] and [20]. A new class of fault attack was introduced in 2017 by Tang et. al. in [21], known as CLKSCREW, where they exposed the vulnerabilities of the energy management mechanisms, basically the Dynamic Voltage and Frequency Scaling (DVFS), to break security without the need for physical access to the devices or any equipment to inject fault by overclocking at the software level in ARM processors thereby compromising the TEE. More recent attacks, targeting the DVFS, have been proposed such as Plundervolt [22] and VoltJockey [23]. Unlike CLKSCREW, Plundervolt abuses the voltage scaling feature of the Intel SGX enclave computations to inject faults to jeopardize the memory encryption and authentication of Intel SGX. VoltJockey, similar to Plundervolt, utilizes the voltage scaling to inject the fault. In it, a controlled fault generation facilitates differential fault analysis and breaks the ARM TrustZone.

The fault injection techniques leverage many different techniques and input source such as the ones listed in the section 1.3. Some of these are detailed in the following sub-sections.

2.2.1 Electro-Magnetic Fault Injection

EMFI is the technique to use short and long wave electromagnetic pulses, of specifically crafted waveform, spatially controlled by an experimental setup utilizing precise measurements, through an electromagnetic antenna or probe that are injected over the chip surface at the desired targeted location. This electromagnetic pulses emitted through the chip surface penetrates the layers within the chip and interferes with the signals traversing through the internal circuitry of the IC. The target could be a particular bus or memory area. The typical setup of an EMFI bench requires a computer with controller program for a connected pulse generator that is connected to a precise electromagnetic probe that generates the pulse over a chip mounted over a stable surface (usually an XYZ table) in a controlled environment such as a Faraday cage. Figure 2.1 represents the typical EMFI setup [24].

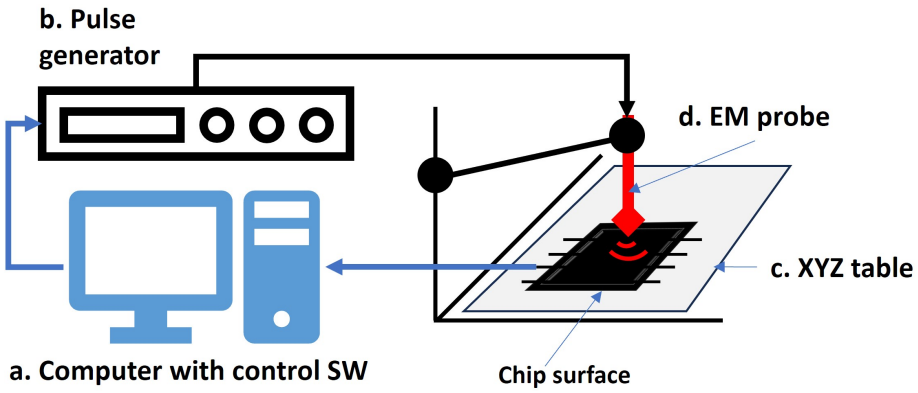

Figure 2.1: A typical EMFI setup

EMFI has several implications in retrieving secret data using fault analysis. There have also been works to tamper the control flow of a program by causing instructions skip [25] with high precision. Another detailed study of EMFI impact on Instruction Set Architecture (ISA) was done in [26]. A recent work on the effect of data transfer due to EMFI was done in [27] with a byte-level precision. [24] presents data flow faults and program flow faults using the EMFI attack on a widely used ARM cortex based micro-controller unit. Other fault attacks on heavily used ciphers include optical and electromagnetic faults on Rivest-Shamir-Adleman (RSA) [28], and Advanced Encryption Standard (AES) [29].

The precision required in faulting a cipher or encryption algorithm is paramount since it relies on a predefined fault model. The timing needs to be perfect in a very small window to impact the targeted section of the algorithm. If the wrong bits are impacted the entire attack iteration would fail and would require to reset the attack process. These attack complexities could be characterized that is a standard metric of evaluating an attack feasibility within a particular frame of security assurance by the Common Criteria (CC) certification scheme. The Common Evaluation Methodology (CEM) of the CC[2] provides five such metrics[1] in which any attack is assessed, namely:

[1] these metrics can be utilized for determining attack feasibility of any perturbation attacks

1. Elapsed time - time needed to setup and implement the attack
2. Expertise - the level of expertise of the attacker needed to perform the attack
3. Knowledge of the Target of Evaluation (TOE) - the level of information about the underlying system available to the attacker based on either public, private or confidential sources
4. Window of opportunity - the time window available to launch a successful attack that can also be sometimes referred to as access time to the TOE
5. Equipment - The sophistication of the equipment needed to setup the experiment to perform the attack

2.2.2 Clock-Glitch Fault Injection

CGFI is relatively easier to perform as compared to EMFI since the setup requires only for a glitched clock input supplied to the target device or chip. However, the timing constraints of the EMFI are still implied in the CGFI attacks i.e., in the identification of the precise moment at which the clock pulse should be glitched in order for the attack to be successful. The principle of clock-glitch fault attacks is to modulate the clock frequency such that it is either sped up or slowed down during an instruction execution in order to introduce the fault. The technique can be in temporarily increasing the clock frequency to either cause some flip-flops to sample their inputs before the new state is reached [30] or reduce the processor's time to write a jump address and prevent the branch execution [31]. A precise clock glitching generator is proposed in [32] and a study showing relationship between the generation of genuine and faulty ciphertext and the variation of clock frequency is presented. A typical setup for a CGFI attack includes a computer with a control software, a clock-glitch generator connected to the target device.

2.2.3 Laser Fault Injection

Laser induced faults work on the same principle as the EMFI except for the fact that instead of the electromagnetic pulse, an optical source in the form of laser is blasted on the chip surface that results in inducing an abnormal current in the target circuit that that results in toggling the value it carries [33]. Coordinated multi-order laser fault attacks are an ever present serious concern for the design of security sub-systems and therefore, the security testing laboratories focus more on the testing against such attacks. Additionally, LFI has gained more popularity due the fact of it being easier to setup, perform and exploit. However, there has been subsequent efforts in the industry and state-of-the art to mitigate such attacks as well such as with the use of time-to-digital converters as shown in [34, 35].

2.3 Hardware Trojans

2.3.1 Hardware Trojan Introduction

A Hardware Trojan (HT) is a malicious modification or insertion in the original design of a chip that is inserted during the design or fabrication phases of the chip manufacturing process. The objectives behind HTs is to, in the later stages of the chip life-cycle, cause DoS attacks, compromise system integrity, or retrieve sensitive and confidential information stored or processed in the chip through some back-door channels usually known only to the adversary or group of

adversaries. A HT is generally constructed with two parts called as the trigger and the payload. The payload is the modification in the design or addition of the extra logic which remains dormant until the trigger is manually or automatically activated based on some input or system state patterns such as some specific register configuration. The figure 2.2 shows a simplified example of a HT with a trigger and a payload part.

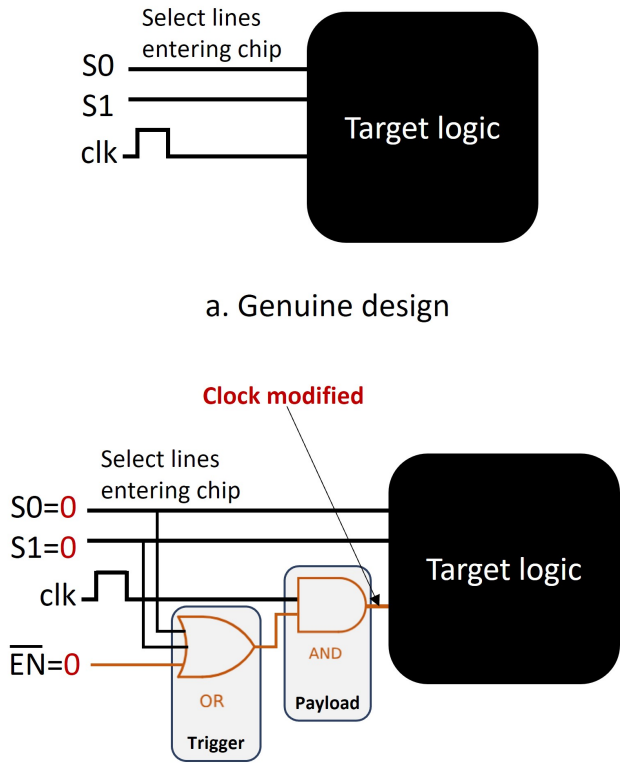

Figure 2.2: An example of an HT infected design with manual trigger

In the figure 2.2 the genuine design takes two select lines as input S0, S1 and a clock signal. The HT is designed to modify the behaviour of the clock signal entering the target logic circuit. This is achieved by introducing two gates and a new signal EN (Trigger Enable with active low) for enabling or disabling the HT. The select lines and EN signals are connected to the trigger OR gate, the output of which is connected to the AND gate which receives its other input as the clock input. The output of the AND gate is the input clock for the target circuit. The trigger is based on the state when both the select lines S0 and S1 are low. When either of the select signals is low, the Trojan circuit does not interfere with the original expected behaviour. However, when the select lines entering the OR gate are both in low state, and the EN is also low (i.e. enabled) the output of the OR gate is 0 and therefore, the clock is inverted, if the original clock was in high pulse state, When entering the original circuit after passing through the AND gate, since the outputs of the OR gate and the original clock are inputs to the AND gate; which means that the AND gate will pull low the high clock pulse since the other input is low (as AND logic gate output is low if any input is low).

2.3.2 Types of Hardware Trojans

A typical HT consists of a trigger mechanism and a payload. However, a HT can also be designed without the trigger. The trigger design therefore, is very important considering the fact that without it, the payload could not be activated or the design be exploited. The trigger part can indeed be a manual trigger or an automated one. It is rather difficult to manipulate the design to incorporate a HT payload without the trigger as it corresponds to an "Always On" state which has higher probability to get identified during the testing phase, since with the manual or automated triggers, it is usually designed to activate the payload with an extremely rare condition which is generally outside the intended functionality of the chip, and therefore, it evades the test coverage scope.

The National Science Foundation (NSF) of the United States of America (USA) maintains one of the largest informational database of HTs at Trust-HUB [36] and also provide the taxonomy for the same. The figure 2.3 provides the characteristics of any HT with respect to:

- the semiconductor development step in which they are inserted,
- the level of abstraction in the design or development process of the chip,
- the mechanism to activate the HT payload with or without the trigger,
- the effect that the HT would have on the design or the system,
- the location in the chip where the HT could be placed in, and
- the physical characteristics of the HT such as their size, type or distribution in the system

From the figure 2.3, the Insertion phase refers the method in which a HT can be inserted, for instance by modifying the design specification to degrade the design dependability such as temperature, or directly in the design during development, in the design netlist, by adding malicious gates or modifying the masks, or fabrication[2] stages. The testing step is also subject to HTs since the testing is also done at "trustworthy" locations where the adversary can modify the testing criteria to let the HT go undetected. The chips are also vulnerable to HT via unprotected interconnections between chips despite the fact that they themselves are built safe.

The abstraction levels corresponds to the degree of control the adversary has over the insertion of the HT in the chip. For instance, at the system level, the adversary is limited to the modules and their interfaces with other modules. If the threat actor has access to the development environment, then they may insert HTs through the EDA tools and scripts, that can go unnoticed in the later stages. At the Register Transfer Level (RTL) the adversary can tamper easily with the Boolean functional implementations and registers as well as signals. The HTs can be added in the gate level where Trojan logic gates can be inserted with obscurity or in the layout where the adversary can control the parameters of the original circuit's transistor parameters such as power consumption or delay characteristics. At the physical level it is observable which areas on the fabric are non-impacting to the design characteristics for inserting the HT.

As discussed earlier, the Trojan can be activated or kept always on depending on the choice of the HT design. The always on Trojan is activated as soon as the chip is powered on. The triggered HT can be again internally or externally triggered. In the internally triggered Trojan, it

[2]since this step is mostly done in third party locations

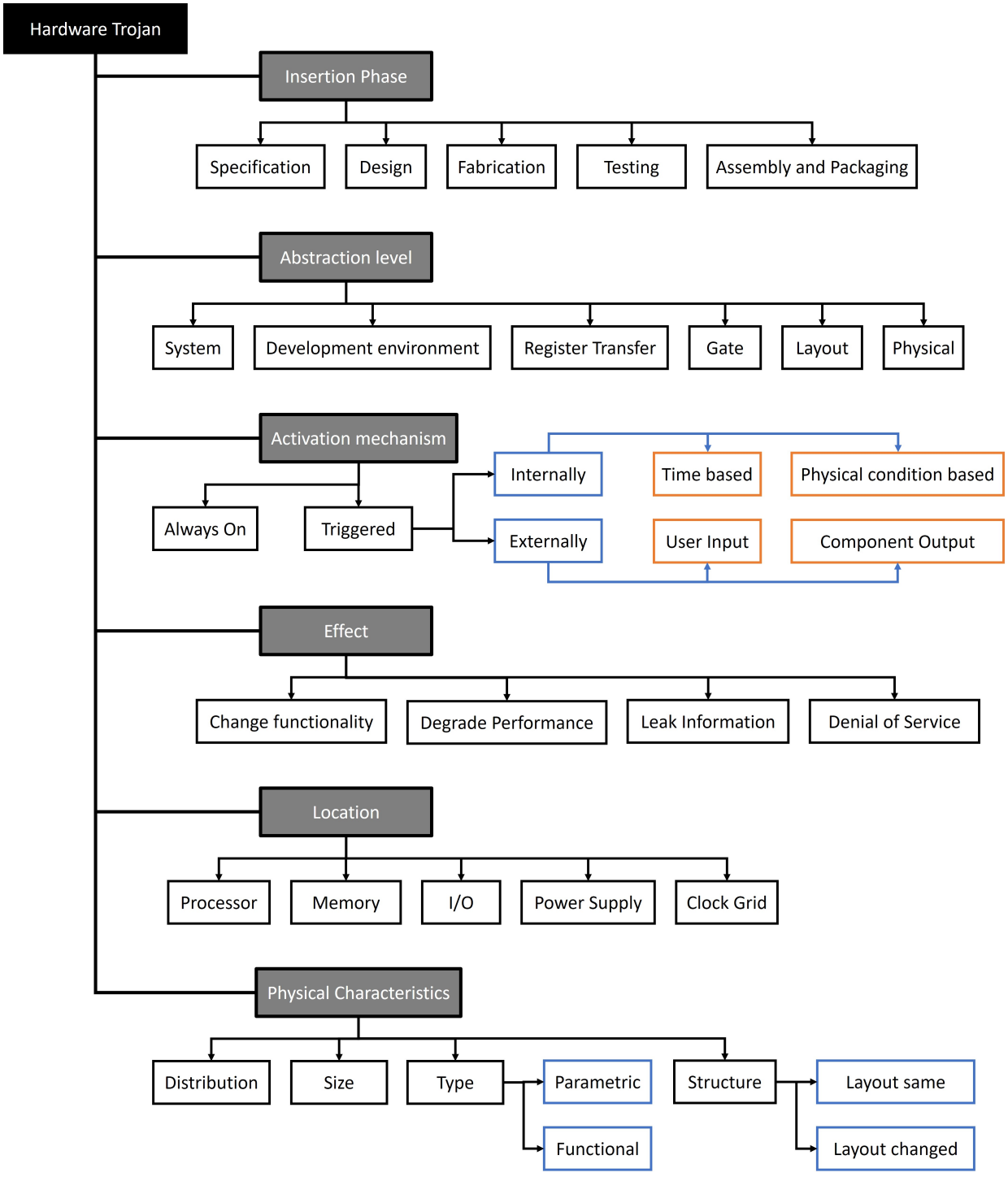

Figure 2.3: Taxonomy of HTs

can be based on time (for instance activated after certain clock cycles) or it can be based on a physical condition such as a certain temperature, etc. The externally triggered Trojans could be activated by either a specific user input that is chosen in the HT design which is outside the input scope of the system's original functionality, or when a specific condition in the system is met or reached.

The effects of the Trojans could be in either changing the functionality of the chip such as if the data path of the Central Processing Unit (CPU) is changed, or just to degrade the performance of the system altogether and even cause DoS. A HT can also be crafted to leak general or specific information from the chip when activated such as CSPs like keys, etc.

The HT can be located in a particular module or area in the chip or it can be distributed throughout the chip to achieve a common objective. Finally, the physical characteristics of Trojan corresponds to the number of additional gates or number of gates or transistors removed from the original design, the area overhead incurred due to the HT addition, as well as if the addition of Trojan changes the layout of the original design or retains the same. For example, one of the major characteristic of a Trojan is to not impact the original functionality of the system unless triggered to evade testing and evaluation phases. Therefore, there can be certain instances where the layout might be altered to incorporate the Trojan logic but the functionality in the modified design is retained.

2.3.3 How and where in the life-cycle can a HT be inserted

As discussed earlier, the risk of hardware Trojan inserting in the design is largely due to the distributed development and manufacturing process adopted by the semiconductor industry. A little, albeit still, factor is the adversary being located in own premises. In the figure **??** the different steps in the semiconductor development and production are explained. The figure 2.4 it is shown with examples that in those steps which of them are highly likely for the Trojans to get inserted and what could be the means to insert the Trojans. For instance, if we talk about the design and development phase, the risks start as early as the specification phase of the design where intentional mistakes could be introduced in the design given that the adversary is located within the vendor's premises. Similarly, in the Hardware Descriptive Language (HDL) design phase, the design could be tampered either due to the use of pre-infected EDA tools or by the adversary using Computer Aided Design (CAD) tools and scripts to tamper. Before the synthesis step some IP cores could be already malicious which is inserted in the design when the synthesis tools synthesize the RTL. The tools used for synthesis, mapping or P&R could be tampered themselves.

In the Production phase, there could be tampered files also used for configuration or layout or the design could be manipulated during loading in case of Field Programmable Gate Array (FPGA) or production process for Application Specific Integrated Circuit (ASIC) which could be during fabrication or packaging or even adversarial testing to evade presence of already inserted Trojan in earlier steps. The supply chain is also infected. Although the government accredited certification laboratories are trusted but it depends upon the selected security certification scope that the vendor or the OEM would choose to certify for and therefore would make their detection difficult. In this step of the supply chain, there is no modification done in the design and only verified and validated for compliance, so there is no risk of HT insertion in this step.

2.3.4 Hardware Trojan Detection

Since HTs pose serious threats in the IC manufacturing, they have become a very important and key research topic. Covered areas are: threat analysis, HT architecture, prevention and detection methods. Regarding HT detection, numerous methods and approaches have been proposed in the state-of-the-art. To mention a few, optical methods [37, 38], testing based

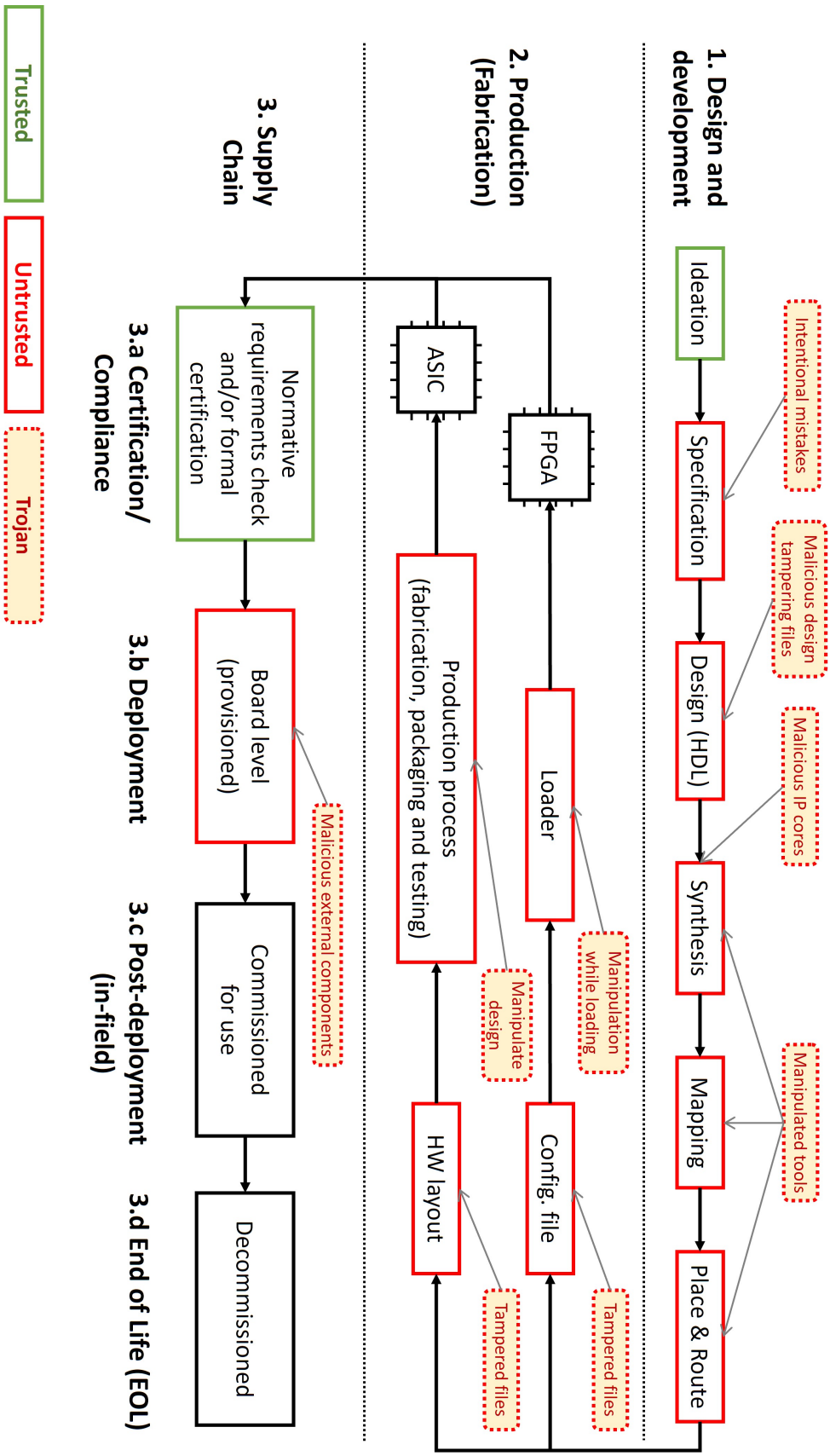

Figure 2.4: Depiction of possible insertions within the design at different levels of semiconductor development phases

detection methods [39, 40], run-time based detection [41] or side-channel based detection methods [42, 43, 44]. Among these approaches, side-channel based detection methods seem to be the most suitable one for various reasons. First of all, side-channel methods are non-invasive and unlike optical methods they do not require chip chemical preparation. Second, they can work without the need of additional logic for run-time detection. Third and most important, efficiency in detection is relatively high. The side-channel based detection methods can detect HTs even if they are not activated during the experimental process.

In the state-of-the-art, different works have been proposed to detect purported HTs using side-channel analysis. In [42], the authors propose an Electro-Magnetic (EM) cartography detection method. The experiment has been performed on an FPGA and the detection method is based on the visual comparison of T-test coefficient between the genuine and infected design. In [45], the authors have used a golden chip-free EM side-channel methodology to detect the HT. Their technique has been limited to utilize the difference in the response between the simulated trace and chip's actual traces from the experiments. In [46], the authors propose a method based on the integration of sensor matrix used to measure the supply voltage in the circuit and T-test metric. The test is performed on a 128-bits AES and validated on a HT with an overhead of 3.2% of the target FPGA. Using the T-test, they obtained a success rate of 80%. In [47], the authors also propose a detection method based on a Ring Oscillators (ROs) matrix (used to measure the power) combined with supervised ML methods such as K-Nearest Neighbors and Support Vector Machines (SVM). With this approach, they have a success rate greater than 88%.

2.4 Intrusion detection systems

Traditionally an IDS was introduced first for network monitoring against unauthorized usage, misuse of the network system, and the abuse of the connected device by internal as well as external entities [48]. With the growing times and the ways of handing computing devices and network infrastructure given the sheer volume of data generated per second, the use of IDS have evolved as well wherein they are often coupled with the firewall systems at the ingress of any computer infrastructure or intranet of an organization. The definition however, remained the same for any IDS even though the categorization grew with host based IDS or network based IDS or a hybrid of both. A Host-Based Intrusion Detection System (HIDS) is a system that is similar to a Network-Based Intrusion Detection System (NIDS) except it is decentralized to a particular device in the network unlike NIDS which monitors the traffic for the entire network it is responsible for. Additionally, the HIDS also takes input from multiple sources and not just from the network. The different types of IDS could be understood from the figure 2.5 which is presented in this thesis after studying the different means in which IDSs are being used in today's world.

2.4.1 Application of IDS in IoT

A typical use case of the HIDS is applicable in the IoT domain to enable detection at the node as well as the edge or server levels. The IDS presented in this thesis, in chapter 5, is a similar one. An IoT system contains typically three system agents, that boils down to the end node or the end device or the IoT objects, the edge and the server. In a typical Edge-to-cloud system, an additional agent, the user, is added. The edge device acts as the gateway between the IoT objects and the server which is accessible to the user.

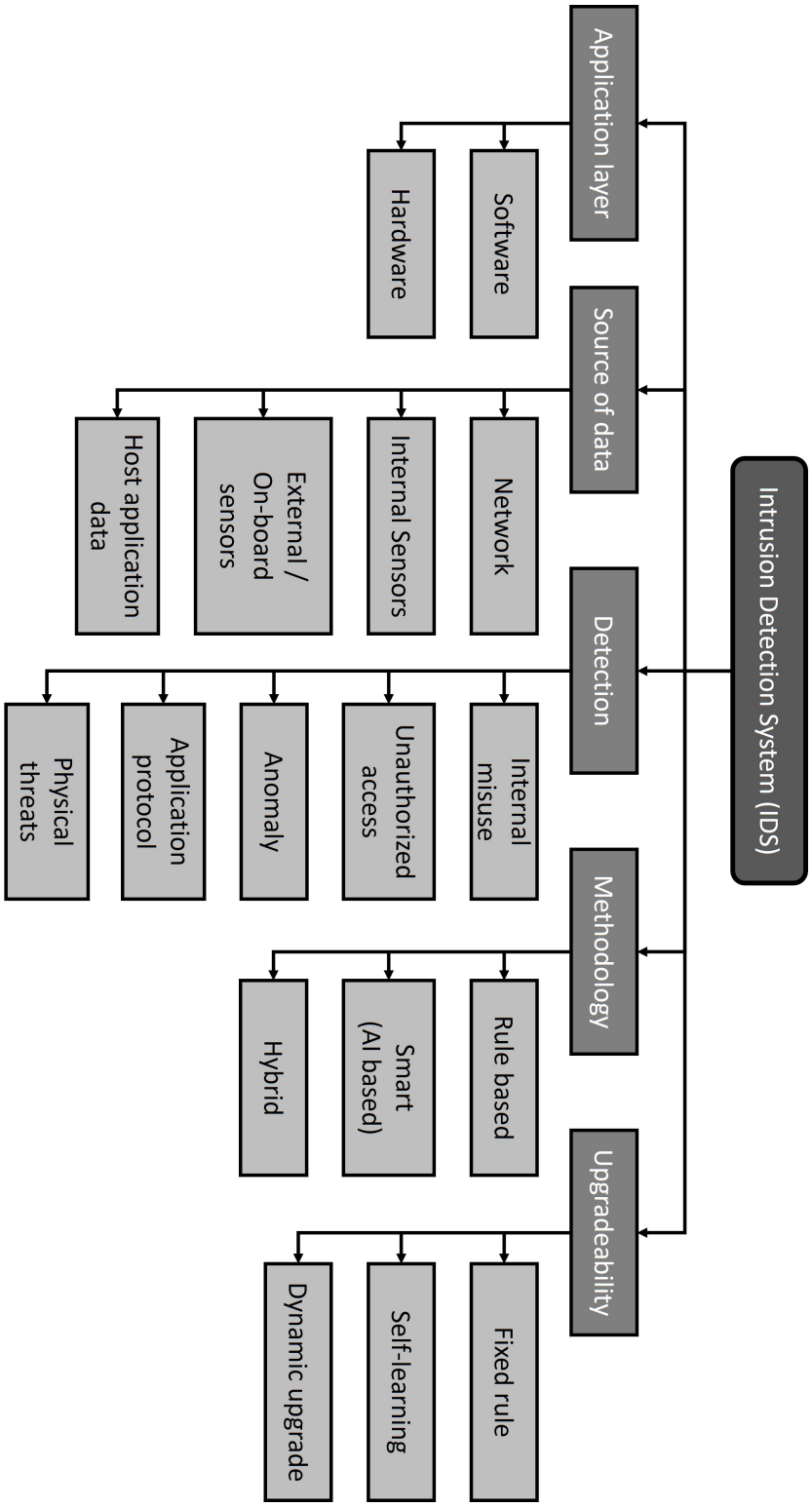

Figure 2.5: Different forms of Intrusion Detection Systems

Edge devices have the capability to ensure back-and-forth connectivity with other devices or with a central system known as cloud server. The edge device is basically composed of a processing unit that can be a Micro-Controller Unit (MCU) with low resources or a Micro-Processor Unit (MPU) with more power and computing resources. Typically, we can define an Edge-to-Cloud system as a technology composed of three main actors as follows:

1. the edge device with a connectivity module, alongside a host CPU for the software part with the hardware layer.

2. the sever side, with significantly higher computing capabilities as a central element that talks with a fleet of edge devices accompanied with application services to manage and monitor the connected devices.

3. the user interacting with the server to send requests to, and monitor, the fleet of edge devices. Users could have different privileges and roles with regards to the server.

An illustration of such system is depicted in figure 2.6.

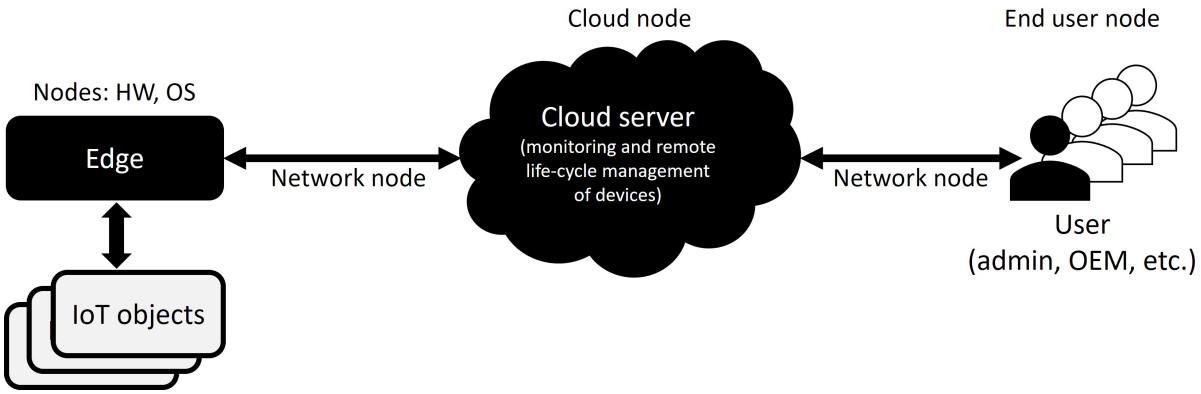

Figure 2.6: The main nodes and actors in a typical Edge-to-Cloud

Based on the figure 2.6, we can distinguish six nodes where end-to-end security should be considered in an Edge-to-Cloud context, as follows:

- **Node 1: at the hardware layer of the edge device.** The security at this node is generally managed by a technology dependent secure layer as a TEE, Trusted Platform Module (TPM), or a dedicated Secure Element (SE) [49]. It ensures a strong security level as per data isolation, secure storage of secrets, etc. A TEE consists in separating the host processor into two spaces viz. regular and secure. All security operations execute within the secure space by ensuring an isolation with the normal space. However, such security mechanism is less secure than a dedicated hardware as TPM or SE. In fact, within a TEE, the shared components, such as internal Host processor memories, might leak sensitive data such as CSPs. The TPM is theoretically more secure than a TEE as it comes with a separated hardware chip. However, the leakage might come from the link between the host processor and the TPM. In fact, the data in transit might be probed and stolen if not encrypted. Thereby, an iSE could be considered the most secure as it is integrated within the same SoC as the host processor. That said, physical attacks such as SCA [50] or FIA [51] are still serious challenges at the hardware layer.

- **Node 2: at the CPU Host layer of the edge device.** The security of data is considered by the host processor that implements a software bridge handling a secure channel with the server side. The Host processor should be able to use cryptographic software engines if security hardware components are not available. For this purpose, the processor shall manage the secure communication with the server side by supporting software clients for security protocols such as Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) alongside crypto libraries such as OpenSSL [52]. The processor might use cryptographic embedded hardware accelerators for performance purposes. Globally, the goal is to ensure that the edge device is identified, authenticated, and authorized relatively to the server side. For this purpose, the host might hold and manipulate device identities (IDs) considered to be sensitive and that needs to be protected. Moreover, the host layer is in charge of all the software, typically running at bare metal or OS layer. That software might be obfuscated or signed and encrypted for more security. In fact, threats like malware and binary reverse engineering are still security threats to the host layer.

- **Node 3: at the connectivity between the edge device and the server.** The connectivity layer is all about network stack ranging from the physical channel to application layer protocols. Edge devices are basically communicating over Internet Protocol (IP)-based channels such as Ethernet, WiFi, Cellular connectivity (4G, 5G, 6G), etc. Some Radio Frequency (RF) protocols use an encapsulation technique to allow IP-based communication. The security shall consider all the layers of the network. The Open System Interconnection (OSI) model, for instance, suggests securing the lowest layers with MACSec (for data link) or IPSec (for transport). Then application protocols are proposed as TLS and DTLS. Higher application frameworks for connectivity like Light-weight Machine-to-Machine (LwM2M) and Message Queuing Telemetry Transport (MQTT) come with a set of schemes to securely manage a device. LwM2M is based on Constrained Application Protocol (CoAP) and DTLS protocols to initiate a communication with an edge device.

- **Node 4: at the server core layer including its data storage components.** The server is the central element in the system. It manages the input and output data from edge devices. Security should at least be ensured for the data at rest like edge devices' logs and users' credentials, that are often stored in databases, data in transit like direct requests from users to edge devices, or even the server components and interactions between those components themselves. In fact, the server is the most impacted node as it is exposed to the internet. In other words, it is the target of a large number of cyberattacks. A list of cyberattacks is regularly updated by the OWASP web pentesting framework group [53]. Hence, security should be thoroughly checked from the server infrastructure level to application micro-services. The literature has recently proposed a new approach with several security requirements, called "zero-trust", that aims at maximizing the security at cloud server node.

- **Node 5: the connectivity layer between the server and the user machine.** Similar to node 3, the connectivity in this node is more about the communication between the user and the server. The security of this node is crucial as it deals with user credentials and device registration's initial inputs alongside secret data such as keys and certificates. A known approach for the same, known as Identity Access Management (IAM), comes with a set of tools, protocols, and frameworks to securely authenticate, and authorize users to access the server [54]. In addition to that, security could be reinforced by a double authentication technique as it is proposed by FIDO2 [55]. Moreover, the data security

could be maximal by combining such software-based solutions with hardware tokens.

- **Node 6: at the user system level.** Most commonly known attacks at this node are performed on software web browsers and interfaces. Technically, this represents the front-side of the server solution that can be a web interface, a web application, a command line interface, an exposed Application Programming Interface (API), etc. The security scope is about all the known attacks as Structured Query Language (SQL) injections against databases, Cross-Site Scripting (XSS) attacks, directory traversal attacks, etc.

2.5 Machine Learning

Machine learning is the notion of enabling a software based system designed for learning from a single or multiple sources of data to train itself and derive meaningful information from it and make probabilistic decisions based on the construction of the learning algorithm [56]. Computational technology relies heavily on configuration, optimization, programming and task driven approaches. This require careful planning to theorize all possible scenarios in which a system should operate and also in which it should not. Any such system could be broadly defined as a Finite State Machine (FSM), first introduced by Prof. Edward F. Moore in [57] paving the way for advanced Automata Theory.

2.5.1 Introduction to ML

The principles of ML is to be non-programmable and self-learning. This is only possible if there are sufficiently available example data which are evenly distributed and available for the ML algorithm to learn from. The first notion of self-learning capability of algorithms was introduced in 1957 [58] which focused on training by using solution to previous problems as input for the solution to next problem which laid down the foundation for training in ML. There are numerous types of ML algorithms, however the basic idea is to fit a function to the data points available. The function can be both linear as well as non-linear. The choice of ML based on linearity depends upon the ML engineer who makes the choice usually based on the type of data available. Usually, for simple data with less features and sufficient distribution and distinction, linear models are used. Non-linear models are more suited for complex or multi-dimensional data that would not linearly separate. To understand linear and non-linear data, let's look at the figure 2.7.

To separate non-linear data, the non-linear ML algorithms are used that takes the data in a higher dimension and separates them using a higher dimensional plane such as the one shown in figure 2.8. Another aspect of ML is based on the output activity categorized under classification, regression and clustering. Clustering could be considered as an extension of classification algorithms. Examples of linear ML algorithms include Logistic Regression (LR), K-means clustering, Gaussian Naive Bayes Classifier (GNBC), Orthogonal Matching Pursuit (OMP), etc. Similarly there are multiple non-linear ML algorithms such as Artificial Neural Networks (ANN), Deep Neural Networks (DNN), Convolutional Neural Networks (CNN), non-linear SVM, and so on.

As mentioned earlier about the types of ML models based on the output activity or usage, the following are the three main categories into which they could be classified:

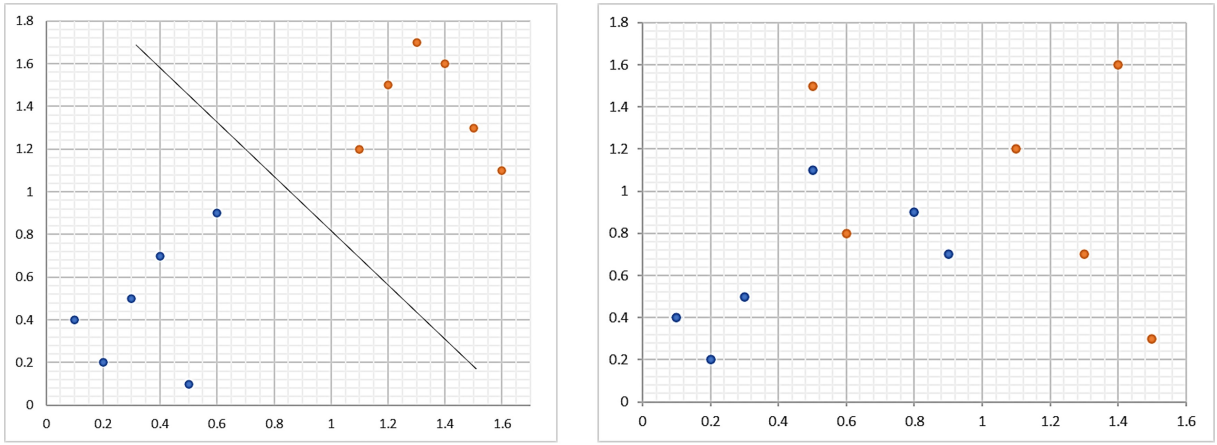

Figure 2.7: An example of linear and non linear data with separation. (a) linearly separable data (b) non-linearly separable data

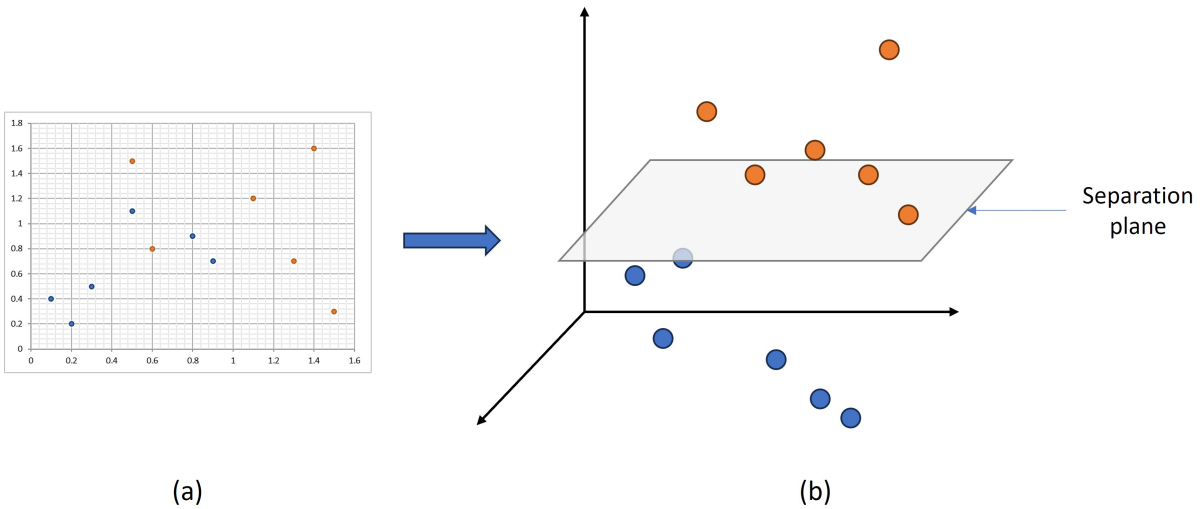

Figure 2.8: An example of a non-linear separation. (a) Samples could not be separated linearly in two dimensions (b) taking non-linearly separable data into higher dimension and separating with a hyperplane

- **Classification** Classification is the type of ML technique that helps differentiate and predict discreet classes based on the input data. This is a classic form of supervised learning method where the model is trained with labelled data[3] and deployed in the operational environment where it classifies unknown incoming data sample among the classes it was trained into. A simple example of classification method is handwritten digit recognition where the ML model predicts a handwritten image among either of the ten digits in the decimal number system as shown in figure 2.9.

[3]Labelled data corresponds to the type of data where all the samples are marked as to which class it belongs. This also gives the notion of how many classes exist in which the data sample should be classified into.

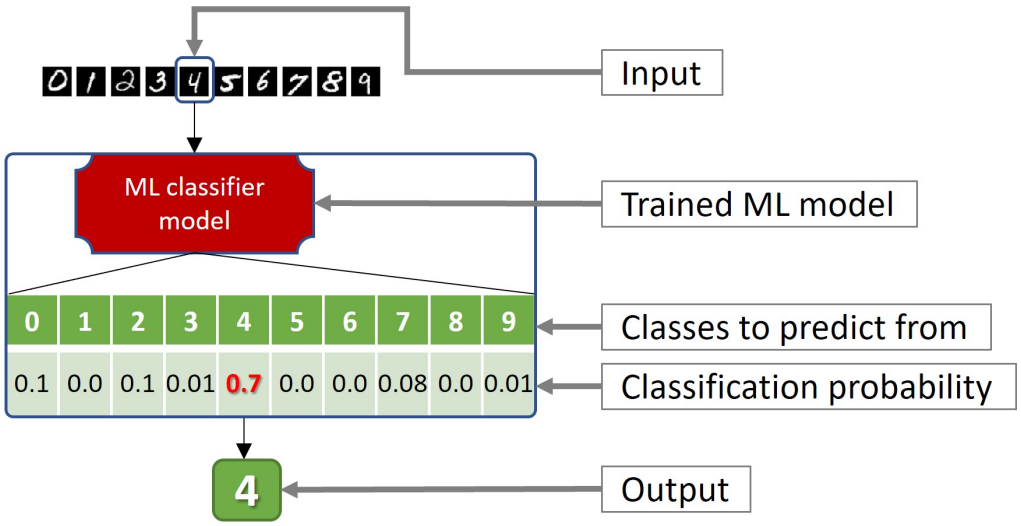

Figure 2.9: An example of ML classification for handwritten digit classification

- **Regression** Oftentimes the problem is not to classify but to predict the next outcome such as in the case of signal reconstruction or predicting rates of house sales in a neighborhood in the upcoming years based on previous collected information. These methods are used for generating continuous real values which has great significance in many fields like healthcare, for example, in predicting geographical region-wise mortality rates of cancer patients from a big database of previously collected data. Other use cases include weather prediction, simulation of any continuous moving object's trajectory given the physical conditions around it, etc.

- **Clustering** Clustering as in interesting example of making sense from a large data of random samples. The principle is to segregate or cluster data points or samples that are related to each other with some pre-defined set of characteristics such as weights, nearest neighbour, etc. Such methods are useful in many ways such as grouping of cell types in a given image, clustering of files based on type, categorizing network packets from a dump file based on protocol, etc. Clustering techniques tend to be unsupervised since they derive relationships between points through iterations or generations of computation.

It is worth mentioning that the popularity of ML rose in the past few decades even if the introduction of the concept dates back much earlier. This is due to the fact that training a ML model to achieve high accuracy is based on the fact that sufficient data is available. Additionally, the training phase involves multiple complex mathematical computations such as matrix multiplication in ANN based models, which requires high processing capabilities from the system. Simply put, the technology at the time of introduction of the concept was not capable of handling such high amount of processing tasks. With the growing advancements in semiconductor technology, it has been made possible to integrate high performance multimedia processing in specialized processor engines. An example is the Graphics Processing Unit (GPU) that can render 3-dimensional objects for the display. With the invention of the GPUs, the researchers found a way to use their parallel processing capabilities to perform the mathematical computations and perform ML training with ease. In today's time, there even exists ML ASICs with smaller technology nodes (as small as 5nm) built for the purpose of solving ML problems that

even allow training the models in the edge nodes. Some examples include Qualcomm's Snapdragon processor or Hexagon AI processor, and Google's Tensor Processing Unit (TPU) which is built for Google's TensorFlow ML development framework.

2.5.2 Types of ML algorithms

The ML algorithms can be categorized, based on how they learn from the training samples, in broadly four categories namely:

- **Supervised learning** In this type of training methodology, the model is fed with labelled data for example in the case of classification or regression. The model during its training phase in each iteration tries to make a prediction of the output class or value. It then, using its loss function, compares the correct answer (label or value) with its prediction and then makes adjustments to the learning parameters. This continues until it can successfully make correct prediction to an acceptable degree. This type of learning methodology generally produces the best results in terms of model accuracy.

- **Unsupervised learning** Unsupervised learning is with the absence of labelled data. This means the model does not have any any example to compare from, it just has the data in unlabelled form. In this type of learning, the model starts building some relationships between the samples or find similarities in the training dataset and tries to categorize them in some order. The best example of unsupervised learning is clustering.

- **Semi-supervised learning** The semi-supervised learning is, as the name suggests, a hybrid of supervised and un-supervised learning methods. This is a special case in which the available data are labelled but only a small amount of them. The model tries to understand the characteristics of the labelled data and apply them over the unlabelled ones. The accuracy of this method is better than the unsupervised learning approach.

- **Reinforcement learning** This is a different class of ML techniques where the learning is based on rewards for making correct actions. The problem is modelled similar to a Markov Decision Process (MDP) where there are some set of states in which the agent, the ML model, can be in, a set of actions that could lead to the different states, and a reward function that awards a reward to the agent after making any state transition with a given action. The goal is for the agent to learn as optimally as possible, a policy that cumulatively gives the maximum reward.

A typical ML model preparation, as shown in figure 2.10, consists of two phases viz. training and testing. In the training phase the model is monitored for progress in model prediction accuracy in every iteration of training until acceptable accuracy is reached or the accuracy does not increase any further. During the training phase, another dataset known as validation dataset, which is typically around 10% in size of the original data pool, is used to ensure that the model is not "overfitting" to the training data.

Overfitting is referred to the state in which a model during training fits very tightly to the training dataset and would not recognize well any other data. This is observed when the model predicts the training data with extremely high accuracy but at the same time performs very poorly on the testing dataset which is a part of the original data pool. This is overcome with the introduction of validation data, that keeps the accuracy of the model in check by remove model "bias"-ness over the training data.

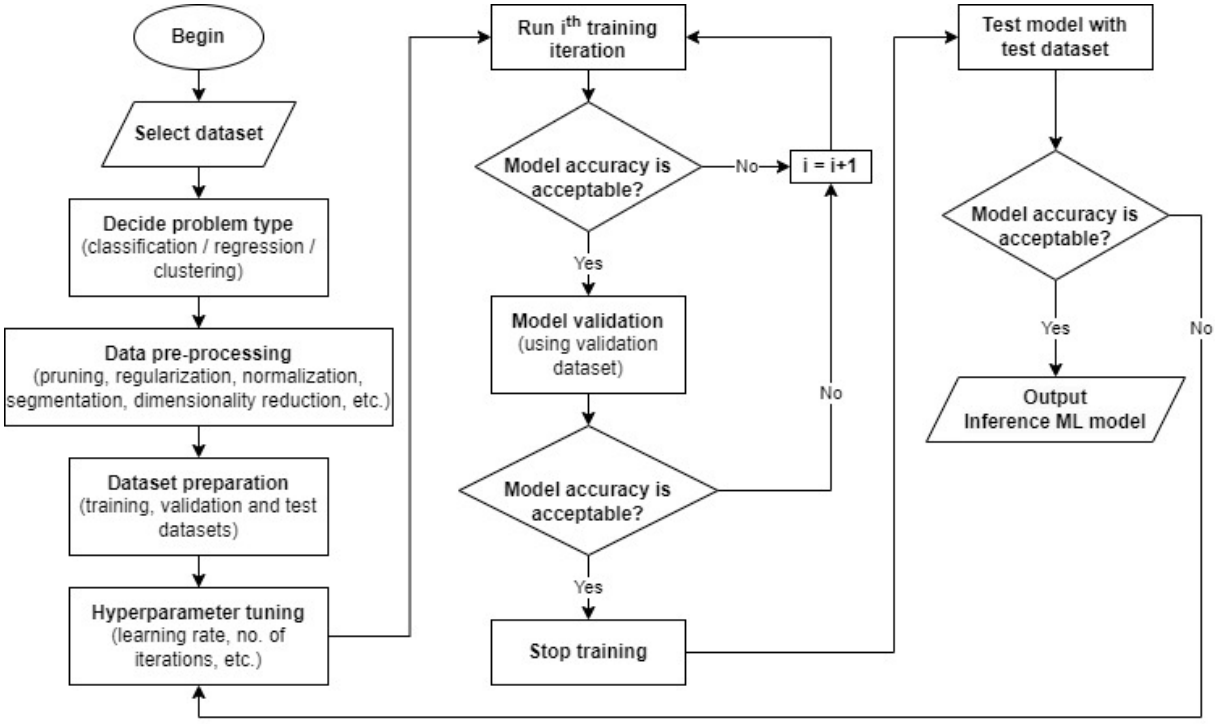

Figure 2.10: A typical Machine Learning model training flow

The accuracy of any model is determined by the model's capability in producing true negatives and true positives. The optimization of the model during the training phase is in reducing the false positive and false negative rates. The accuracy of a typical ML model is represented in the form of a "Confusion matrix" as shown in the table 2.1

	Actual Positive	**Actual Negative**
Predicted Positive	**True positive**	**False Positive**
Predicted Negative	**False negative**	**True negative**

Table 2.1: Confusion matrix; Rows are predicted values by the ML Inference model and columns are actual values.

2.5.3 Non-security applications of ML

ML can be considered as the wizards stick of the modern day computational intelligence. With its way into almost every domain including physical systems and weather forecasting [59], DNA Sequencing [60], Drug development [61], so on and so forth. Apart from image analysis, object tracking in video, multi-object image classification, etc., ML can be now used to draw hyper-realistic drawings using Generative Adversarial Networks (GAN) [62]. Therefore, the application of ML is just limited to imagination.

2.5.4 Embedded systems security using ML

The applications of ML in hardware security has seen a lot of growth since ML for pattern recognition became popular [63]. In hardware security there are two aspects of usage of machine learning i.e., to either use the assistance of the pattern matching and problem solving capabilities of ML to break the security using techniques such as profiling attacks [64, 65, 66, 67], while the other case is in creating countermeasure solutions with the help of ML such as that of detecting Hardware Trojans, with or without golden or reference models [68, 69, 70], or Fault Injections [12, 71], prevention of Side Channel Attacks [72], etc. The ability of Machine Learning methods to distinguish and identify patterns which are otherwise extremely difficult to mathematically characterize and use for analysis of systems for security purpose. The main challenge is, however, in modelling of the security specific problems into machine learning problems. As we understood from figure 2.10 about the ML engineering process, which is more focused on preparing the data and the model to be able to successfully identify the patterns otherwise the accuracy would be around 50%[4]. This is the reason why most of the time spent in employing a Machine Learning method is in making sure the following objectives are met during the ML engineering design:

- The data features are prominent otherwise techniques such as Principle Component Analysis (PCA) should be used to highlight the features for the model to learn.

- The choice of the ML algorithm corresponds to the type of data since a model with high accuracy on one dataset does not guarantee high accuracy on other dataset of a different type. To make this choice is rather difficult and is based on expert analysis and observation, as well as comparison among multiple different models.

- Quality of the data is good i.e., even distribution of classes to avoid class-imbalance that interferes with the learning process of the model, it is normalized and correctly fed into the model for it to recognize it, and so on.

- Finally, the hyper-parameters such as learning rate, number of layers in the Neural-Network, model architecture, number of neighbors to consider in a nearest-neighbor based model, etc.

2.5.5 Detecting Fault attacks with Machine Learning

The use of ML in the security domain is relatively new. Major works have been done in the Side Channel Analysis [73]. In [74], the authors provide a ML-assisted technique to explore and characterize the fault attack space and use the knowledge of a known fault attack on a cipher in understanding new attack instances. While most fault analyses are based on the characterized faults from known attacks [75], this work is based on a completely different approach of live identification of attacks using real-time sensors. To the best of the knowledge of the authors, our work is the premiere in providing a hardware based framework to dynamically detect FIA from multiple sources.

[4] A value of around 50% in the accuracy of a ML model signifies random guessing in the normal distribution of samples which in other words mean the model is not learning anything useful.

2.5.6 Detecting Hardware Trojans with Machine Learning

The detection of HTs using machine learning requires supervision for best results. However the challenge in the detection is to have a golden reference model to characterize the internal parameters for data collection and repeat the same procedure for the data collection from an unknown or Trojan infected chip model to make the classification. The reason for this is the preparation of the test chip or golden model is done in the same process as the rest of the chip and there is no guarantee that the so called golden model will not contain the Trojan itself. In this case the classification will yield negative results for the presence of hardware Trojan in the unknown lot. There are several workarounds to overcome this challenge in real world for instance, the use of a more trusted or in-house foundry to produce the test chip for the golden model reference with same technology node as the production lot would be. However, there would be a question as to why not produce all the lots from this trusted source. The answer is two-fold for this question; firstly, using two production units can prove beneficial if it is ensured that both the units are mutually exclusive to each other and are not related, and secondly, the reason for outsourcing the production to third-party foundries is to save cost in production and while getting the golden model in a more trusted location can be costlier than producing the whole lot in a standard fabrication unit. To tackle the challenge of trust with the golden model not containing any Trojan produced in a more trusted location, another test in the pre-validation stage could be performed which is to have simulated readings from the RTL of the design with the same characteristics that would be used for ML classification testing, and make the comparison with some pre-processing in the simulated readings to match the physical characteristics with the chip. Incidentally, this is also a technique for ML classification testing of unknown lot of chips without the availability of a golden model, that brings us to the second method of ML testing i.e. without a golden reference model.

There are few methods in the literature that propose methods to detect the presence of HTs without using a golden model. This is achieved via similar methods proposed above which consists in creating a reference model through simulation or other techniques such as statistical side-channel fingerprinting [69].

Other methods of detection includes:

- optical methods ([37], [38])
- testing based detection methods ([39], [40])
- run-time based detection ([41])
- side-channel based detection methods ([42], [43], [44]).

Among these approaches, side-channel based detection methods seem to be the most suitable approach for the following reasons:

- Side-channel based methods are non-invasive and unlike optical methods they do not require chip chemical preparation.
- They can work without the need of additional logic for run-time detection. Third and most important, efficiency in detection is relatively high.
- side-channel based detection methods showcase the ability to detect HTs even if they are not activated during the experimental process.

In the state-of-the-art, different works have been proposed to detect purported HTs using side-channel analysis such as in [42] the authors propose an EM cartography based detection method. The experiment has been performed on an FPGA and the detection method is based on the visual comparison of T-test coefficient between the genuine and infected design. In [45], the authors have used a golden chip-free EM side-channel methodology to detect the HT. Their technique has been limited to utilizing the difference in the response between the simulated trace and chip's actual traces from the experiments. In [46], the authors propose a method based on the integration of sensor matrix used to measure the supply voltage in the circuit and T-test metric. The test is performed on a 128-bits AES and validated on a HT with an overhead of 3.2% of the target FPGA. Using the T-test, they obtained a success rate of 80%. In [47], the authors also propose a detection method based on a Ring Oscillator (RO) matrix (used to measure the power) combined with supervised ML methods such as K-Nearest Neighbors and SVM. With this approach, they have a success rate greater than 88%.

2.5.7 Statistical methods

For the purpose of this thesis, a statistical metric, known as the T-Test (or Student's test), is explored as it provides some significant pre-processing capabilities for ML modelling of certain problems such as HT detection which is detailed in the later chapters.

T-test is a metric used in the field of statistics to detect if the mean of a population has a value specified in a null hypothesis or if the means of two different populations are equal.

For the HT detection application, the T-test is already used in the state of the art to determine if the reference dataset and the dataset under test have the same means (no HT presence) or not (HT presence) using the following formula:

$$t = \frac{\mu_0 - \mu_1}{\sqrt{\frac{\sigma_0^2}{N_0} + \frac{\sigma_1^2}{N_1}}} \tag{2.1}$$

where μ_0 is the genuine sample mean, μ_1 is the HT sample mean. σ_0 is the genuine sample variance, σ_1 is the HT sample variance. N_0 is the cardinality of genuine set and N_1 is the cardinality of HT set. The T-test is also used for the side-channel analysis to break the cryptography IPs [76].

2.5.8 Machine Learning based IDS

AI and particularly ML provides powerful prediction algorithms that constitute state-of-the-art techniques in several research areas: image processing, natural language processing, medical diagnosis, etc. Naturally, those methods are also drawing increasing interest in the cyber-security landscape as explained earlier. Indeed, the advanced modelling capabilities of ML algorithms allow to leverage on large quantities of available data and knowledge to improve security systems in various fields of application. ML approaches are perfectly suited for attack or failure detection applications as they allow creating a model of the normal predictable behavior of a system [77]. After this profiling phase, it becomes possible to detect significant deviations from the model. A typical example of application are intrusion detection systems which analyse a network traffic to detect, block and report malicious packets.

A "traditional" IDS uses a database of known malicious signatures that compares with the incoming packets to detect attacks. This approach presents a significant drawback i.e., it detects attacks based on known threats and is unable to handle new attacks. On the other hand,

it is possible to use ML algorithms to create a model of the normal behaviour of a network and to detect abnormal activities based on the observed deviations from the base profile. This approach has the advantage of detecting unknown or even zero-day attacks.

The same idea can be applied to sensor data analysis. Standard deployment of a fleet of sensors requires calibration, and threshold-based analysis is necessary to process sensor values, which often leads to false positives. AI-based sensor aggregation and analysis enable detection of fault injection attacks, anomalies and failures, and advanced diagnosis [11, 12], while reducing the number of false alerts. To build such a system, a test chip is characterized in controlled environment, in order to generate sample data and train a detection model to be deployed on the final chip. Then, in operation, the model classifies new data, provide useful information such as the presence of an attack or anomaly, if any failures occur, or even the type of attack, and report to the upper layers in the system stack. Based on desired security policy or user feedback, the detection sensibility can also be adapted after deployment.

2.6 Certification and Standardization of AI

This section in the thesis manuscript presents an overview of the certification and standardization efforts taken by various governmental, national and international organizations towards generalizing the development of AI and ML based implementations. This bears significance due to the fact that ML is a probabilistic approach to predict various outcomes. While accuracy is a factor, the presence of false positives and negatives are serious concerns in security critical applications, especially when humans are involved. The biggest example is the autonomous smart car industry which is on a fast-track mode for development of Automated Driving Systems (ADS) and Advanced Driver Assistance Systems (ADAS) based systems.

2.6.1 Introduction

Standardization is an important step towards market trust as well as ensuring verifiability. The process of standardization as mentioned in the figure 1.1 is an important part of the supply chain for any embedded or integrated cybersecurity product. During the certification or standardization phase, a product is entrusted to an accredited third-party certification laboratory to perform all forms of applicable security testings on the TOE mainly fault injection evaluation. The assurance of security, however, is determined based on the security levels provided in the followed standards. For example, National Institute of Standards and Technology (NIST) FIPS 140-3 provides 4 security levels while the CC certification scheme allows 7 levels of assurance and 5 levels of vulnerability assurance. The highest security levels targets the most sophisticated attacks such as that using a FIB.

With the growing popularity in the use of AI in cybersecurity and general application, there is a competitive rush amongst the chip manufacturers to produce AI enabled chips as well as dedicated AI chips and many big market players have already introduced such products which are being used mostly in the smartphone industry. The use of AI is being normalized at even the governmental level, such as in Europe [78], and its importance being highlighted while also driving the narrative towards its safe and trustworthy usage. However, the regulations and schemes are far behind the development pace of the products which creates a big challenge and a security risk in using AI in general applications within the connected device infrastructure.

Nonetheless, there are several efforts being made in the direction of standardizing the verification and framework of AI by different national and international schemes for safe and secure implementation and usage of AI in the semiconductor industry and embedded or integrated system. In this section, an overview of the available standards and schemes, including the ones which are under development, are presented for responsible usage of AI in cybersecurity domain for various applications.

2.6.2 Standards and Guidance on AI

Among the many efforts being made to publish standards for AI development and implementation some of the major ones are listed in this section. The biggest challenge in developing in standards is due to long process of generalizing the concept followed by a lengthy process of drafting, commenting, correcting, voting, and finally publishing, that takes on an average up to two to three years or even more.

The current notable published standards related to AI development within cybersecurity context are as follows:

1. ISO/IEC 23053:2022 Framework for Artificial Intelligence (AI) Systems Using Machine Learning (ML)
   This standard focuses on general framework and necessary segments in the development of ML models.

2. ISO/IEC 25059:2023 Software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - Quality model for AI systems
   This standard mainly focuses on measuring and evaluating the quality of an AI system.

3. IEEE P2247.3 - Recommended Practices for Evaluation of Adaptive Instructional Systems
   This guidance from IEEE provides the outlines for ethical system design for using AI in Adaptive Instructional Systems.

4. IEEE P2840 - Standard for Responsible AI Licensing

5. P2976 - Standard for XAI – eXplainable Artificial Intelligence - for Achieving Clarity and Interoperability of AI Systems Design

6. IEEE P7018 - Standard for Security and Trustworthiness Requirements in Generative Pretrained Artificial Intelligence (AI) Models, and many more

Additionally, as expressed earlier, there are several ongoing efforts towards publishing AI centric standards. The ISO/IEC JTC1/SC (Joint Technical Committee/Sub-Committee) 42 is focused on development of AI standards and have produced around 20 published standards already that includes standards for assessment of performance of classification algorithms, frameworks for AI, verification of robustness of Nearest Neighbor (NN)s, etc.

EU's High Level Expert Group on AI (AI HLEG), which is an expert group appointed by the European Commission for advice on AI strategy, had already published many recommendations such as ethics guidelines on trustworthy AI.

A group of experts from different organizations from within the industry and academia published a guidance book concerning trustworthy AI development [79] that provide hardware

mechanisms and recommendations for the development of security features for AI hardware accelerators. There are, additionally, many other works in the literature around trust in AI such as in [80, 81].

2.6.3 Discussion

The need for standardized approaches for AI development is an essential factor. Even though several standards have been proposed, it would take some time to mature as well as for the industry to adopt those standards who have already been developing AI hardware based solutions for quite some time. The biggest challenge in verification of the AI systems for the claimed performance metrics in security critical applications.

2.7 Conclusion

In this chapter, the background and the related works covering all the aspects of this thesis work is presented, including the major attacks and threat categories viz. fault injections, Hardware Trojans, and hardware & network intrusions, covered by the presented detection mechanisms in the contributions of this manuscript. Additionally, a background on machine learning is given along with their capability in detecting those threats. In the next chapter, the first contribution in detecting fault attacks with machine learning methods is presented.

Chapter 3

External Fault Injection Detection

Contents

3.1 Introduction

In this chapter, two important types of fault injections on embedded or integrated systems are studied and different methodologies based on ML to detect them are presented with experimental results. These fault injection types include the EMFI and CGFI. The proposed framework for detection is intended for on-chip deployment and the main purpose of the presented methodology is to enable easy integration of such smart detection engine in an existing design following plug-and-play approach. To that end, a High-Level Synthesis (HLS) based hardware IP is also generated that can be easily interfaced with existing controller modules in a SoC. The detection mechanism is universal and not depended upon any architecture or design and is subject to availability of time-to-digital sensors or simply Digital Sensors that are spatially distributed within the chip at sensitive locations, such as secure internal storage or bus interconnect. The solutions are fully digital and the configuration required is only during the offline training of the ML model(s) to account for technology or operational environment dependent changes, to establish the nominal boundary or normal working conditions for the chip.

Two methodologies are presented viz. for fault injections from a single source, and for multiple fault injections from different sources. The detection mechanism in the first method for single source fault injection focuses on identifying the presence of an attempt to induce faults while in the second method for detecting fault injection from multiple sources, a two-step approach is presented that in the first step identifies the presence of an attempt to fault

injection and in the second step also provides forensics to the type of perturbation source i.e., classification between type of fault injection. To compare the ML based smart solution's efficacy, a classical sensor threshold based method is also used in which the threshold is optimized, based on the same training dataset used for the ML model, and compared with the results of the ML models. The results are presented in later sections of this chapter.

Machine Learning based Fault Injection Detection
The ML technique used for the detection of fault injections is based on classification algorithms. The training of the ML model is done offline on a standard desktop computer and the inference is performed both offline to simulate the model behaviour and online to test the model in a more realistic setting. The details are provided in the next sections.

3.2 Data acquisition and description

For the purpose of data acquisition, an FPGA based setup is prepared with integrated digital sensors with some sample logic[1] to protect against.

3.2.1 Digital Sensors

Digital Sensor (DS) is a light weight delay chain unit that can be placed anywhere on the chip fabric. The DSs have delay chains longer than the critical path and thereby detecting delay faults before they affect the user logic [82] . The DS is designed to detect various FIAs, such as clock glitch, power glitch, underfeeding, heating, laser as well as electromagnetic fault attacks. A DS converts all observed stresses into a timing stress for measurement. It is extremely sensitive to variations in temperature and voltage as well as to internal activities of the Design-Under-Test (DUT) which makes it a generic sensor that can detect multiple perturbation types. For a United Microelectronics Corporation's (UMC) design kit with 28nm HPC (High Performance Computing), with a frequency of 100MHz, each Digital Sensor is 2.93 kGE (kilo Gate Equivalent), which means 730µm² (micro-meters squared) for this technology node.

3.2.2 Experimental setup

The setup comprised of a controller application running in a computer connected to a frequency pulse generator, a state-of-the-art oscilloscope, an XYZ-table for adjusting the position of the board automatically based on the controller application for spatial localization of the EM probe on the chip, an EM probe for injecting the EM pulse in the chip, and finally the FPGA board withe the DUT as shown in figure 3.1.

3.2.3 Dataset description

EMFI dataset
The EMFI dataset is recorded from sixteen Digital Sensors (DS) on a chip executing AES encryption (shown in figure 3.1). Same sized data is recorded for both Nominal (no EMFI) and with fault injection that are classified as classes 0 and 1 respectively (a binary classification

[1] A generic AES implementation is used to demonstrate as the design under test since the effect of Fault Injection Attack detection is independent to the design being protected

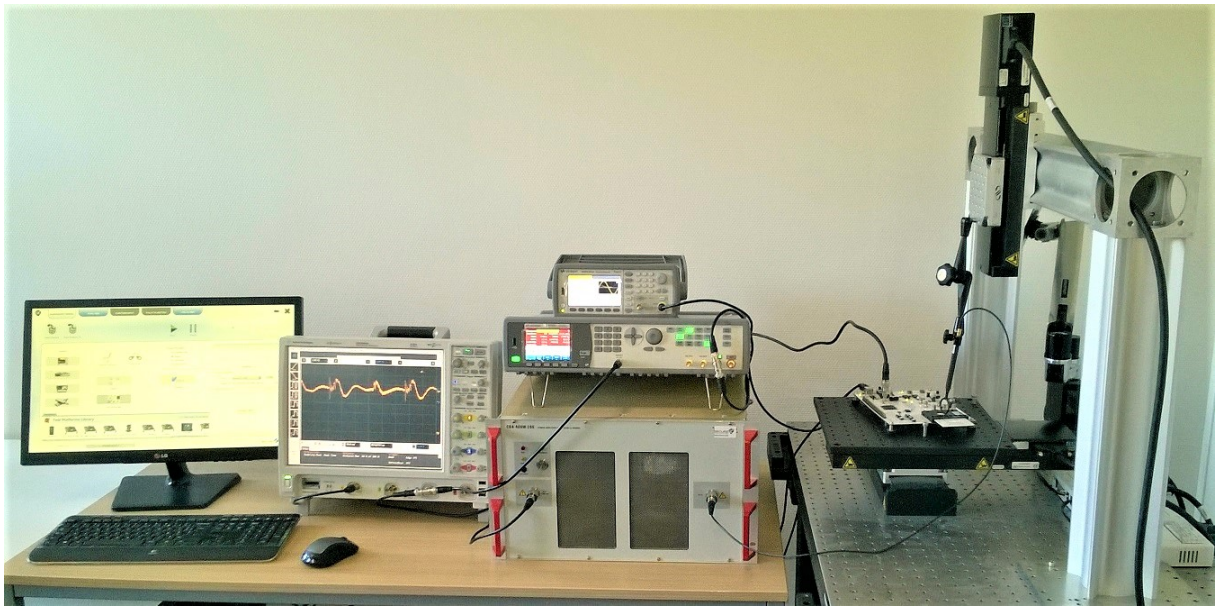

Figure 3.1: Experimental Setup for EMFI data acquisition

problem). The same experiment is repeated with the EM probes placed at four[2] different arbitrary locations on the chip. The parameters for the FIA are presented in the table 3.1. The amplitude indicates the attenuation of the signal. The sine burst consists in the shape of the signal, that looks like a sequence of 10 arches. The wave frequency is 320 MHz, meaning that the injection pattern lasts $10/320 = 31.25$ ns. The polarity is normal, which means that the pulse is positive (i.e., of positive height, but non-negative).

Amplitude (dB)	Sine burst	Wave frequency (MHz)	Polarity
0	10	320	Normal

Table 3.1: Electro-Magnetic Fault Injection parameters for all the four sessions

The choice of selecting the locations is based on the location of the Digital Sensors in the design i.e. from closest to farthest from the fleet of Digital Sensors, in order to have maximum coverage guarantee. The four locations in terms of the X and Y coordinates on the chip surface of the Sakura-G FPGA evaluation board are presented in the table 3.2.

Chip Location	X position	Y position
Location 1	5.890000e+01	5.620000e+01
Location 2	5.940000e+01	5.620000e+01
Location 3	5.990000e+01	5.620000e+01
Location 4	6.540000e+01	5.370000e+01

Table 3.2: Chip locations for EMFI on the chip surface of Sakura-G FPGA evaluation board

Thus, there are four parts of the dataset with each having data for nominal and injected scenarios. Each part contains 1000 Test runs with each run comprising 13 cycles of sensor

[2]usually in a FIA a single location is chosen, but for the sake of analysis of effect on DSs of FIA at various locations, four different locations are chosen and studied separately as well as together

statuses[3] from each DS. This can be understood from the directory tree below:

```
EMFI_Dataset/
├── Part1/ (Chip Location 0)
│   ├── Nominal (Class=0 : 1000 Tests)
│   └── Injected (Class=1 : 1000 Tests)
├── Part2/ (Chip Location 1)
│   ├── Nominal (Class=0 : 1000 Tests)
│   └── Injected (Class=1 : 1000 Tests)
├── Part3/ (Chip Location 2)
│   ├── Nominal (Class=0 : 1000 Tests)
│   └── Injected (Class=1 : 1000 Tests)
└── Part4/ (Chip Location 3)
    ├── Nominal (Class=0 : 1000 Tests)
    └── Injected (Class=1 : 1000 Tests)
```

CGFI dataset

The design is kept coherent for both EMFI and CGFI experiments since later the study for detection of FIA from multiple sources are conducted. The acquisition for clock-glitch data is different from the EMFI since the attack surface is limited to the clock input instead of the whole chip surface in case of EMFI attack. However, for non-biased data collection as well as to eliminate experimental abnormalities and errors, two sessions of data acquisition (parameters as shown in table 3.3) are executed with varying number of tests per session and merged together to create a dataset of 6000 measurements with varying delay between 379 nanoseconds (ns) to 511 ns with a pulse width of 5 ns, 8 ns and 1.5 ns to inject clock-glitches in the DUT.

	Physical Parameters		
FIA Session	**Pulse delay (PD)**	**Pulse width (PW)**	**Chip Voltage (V_C)**
A	357.00-405.00 ns	5.00 ns	1.47 V
B	92.00-385.90 ns	1.5-15.50 ns (step-up by 1ns)	1.147 V

Table 3.3: Clock-Glitch Fault Injection parameters

The acquired data from the EMFI, CGFI and nominal sessions is processed to have an aggregated sensor data from all the digital sensors integrated in the DUT. This is termed as "sensor aggregation" hereafter. To understand the structure of the input data recorded from the digital sensors and sensor aggregation to represent the dataset for the ML model(s) can be understood from the figure 3.2.

3.3 Modelling EMFI and CGFI detection as ML problems

As shown in the figure 2.10 the first step in the process is to model the fault injection detection problem as a ML problem. In the table 3.4, few characteristics of the problem statement where

[3]the size of window of DS values per sampling is optimized to 13 in order to retain minimum feature in the data to be trainable with. Increasing the window size will provide better detection but at the trade-off cost of increased latency

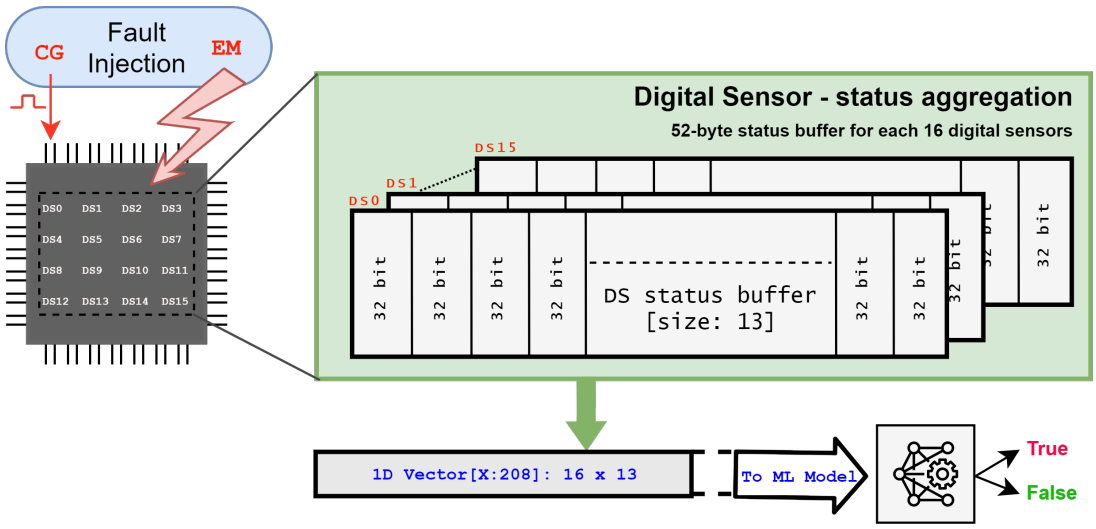

Figure 3.2: Illustration of data acquisition from multiple sensors (sensor aggregation) for EM and CG fault injections and dimension pre-processing for ML algorithms

the FIA detection problem is modelled as ML problem, are presented.

ML Problem type	Classification
Data type	Continuous, represented as discrete one-dimensional vectors
Training type	Supervised
Available classes	Nominal operation scenario and Fault Injection scenario
Data labelled?	Yes
Data feature(s) to train model	Variation in sensor readings
Optimization goals	No false positive and maximize detection accuracy (True Positive and True Negative)

Table 3.4: Modelling Fault Injection detection problem as ML problem

3.4 ML detection methodology for EMFI

3.4.1 ML classification models used for detection

The ML algorithms used for the EMFI detection are two linear classifiers namely Logistic Regression Classifier (LRC) and GNBC, and two non-linear classifiers namely SVM and Multi-Layered Perceptron (MLP) which are detailed below.

- LRC - Although the name Logistic Regression suggests otherwise, the LRC is a classification algorithm. A logistic regression is exemplary of a single perceptron with a *sigmoid* activation function which can be replaced with a *softmax()* function to obtain multi-class classification. The equations 3.1 and 3.2 provide equations for binary and multi-class

LRC respectively.

$$\begin{aligned} y &= sigmoid(wx + b) \\ sigmoid(t) &= \frac{1}{1 + e^{-t}} \end{aligned} \tag{3.1}$$

where, y is output of LRC, w and b are weight and bias respectively, and x is the input.

$$\begin{aligned} y &= softmax(wx + b) \\ softmax(y_i) &= \frac{e^{y_i}}{\Sigma_j e^{y_j}} \end{aligned} \tag{3.2}$$

For the purpose of evaluation as shown in the comparison figure 3.4 the solver used in the multinomial LRC classifies model is Limited-memory Broyden–Fletcher–Goldfarb–Shanno algorithm (BFGS) or LBFGS with a maximum iteration for convergence as 200.

- GNBC - Gaussian Naive Bayes classifier works on the probablistic model of Bayes theorem and assumes the input to have Gaussian or normal distribution. Since the data source is continuous in nature (refer 3.4), Gaussian model is used instead of classic model of Naive Bayes classifier. For binary classification (classification between two classes), the input is differentiated with the mean of the first class and divided by the standard deviation of that class to obtain the distance between mean of that class and the input variable. The same is repeated with the other class mean and standard deviation to get both the distances. The least distance is chosen as the candidate class for the input variable for the output.

- SVM - SVMs are supervised learning methods used for classification and regression and fall under the category of linear classifiers. SVMs work on the principle of machine learning with inbuilt over-fitting correction to increase accuracy of prediction. They use hypothesis space of any linear functions in a high dimensional feature space, trained with a learning algorithm from optimization theory that implements a learning bias derived from statistical learning theory. For the purpose of evaluation, the Radial Basis Function (RBF) kernel with One-vs-Rest decision function, is chosen for the classification.

- MLP - A multi-layered perceptron as the name suggests is the arrangement of perceptrons in multiple layers such that each layer has multiple perceptrons and multiple layers are connected to each other (example figure 3.3). A perceptron is an independent unit that contains a non-linear activation function such as a Rectilinear Unit (ReLU), Signmoid, Hyperbolic Tangent (Tanh), etc. The input of the perceptron passes through the activation function after being multiplied by the weight associated with the input and the bias that is added to the multiplication. Both the weights and bias in the network are subject to optimization as part of the ML training. The equation 3.3 is representative of the ReLU activation function and the equation 3.4 is the ReLU function with weights and bias i.e., a perceptron.

$$relu(x) = max(0, x) \tag{3.3}$$

$$P(x) = relu(x * w + b) \tag{3.4}$$

The parameters of the MLP NN model used for evaluation in this chapter are as follows:

1. Network structure:
   Input Layer
   - Size: 208
   - Activation: ReLU
   - Dropout: 0.1

   Hidden Layer 1
   - Size: 1000
   - Activation: ReLU
   - Dropout: 0.1

   Hidden Layer 2
   - Size: 2000
   - Activation: ReLU

   Hidden Layer 3
   - Size: 500
   - Activation: ReLU

   Hidden Layer 4
   - Size: 100
   - Activation: Sigmoid

   Output Layer
   - Size: 1
2. Batch size: 200
3. Loss function: Binary Cross-Entropy
4. Optimization function: Adam (similar to Stochastic Gradient Descent for Deep Learning models)
5. Learning Rate: 0.001
6. Iterations: 50

3.4.2 Classical threshold optimization method for comparison with the ML model

The extreme sensitivity of DSs allows very accurate detection of FIA, but obligates the IP designer to set a precise threshold (derived through simulations or empirically evaluated) which is far to be an easy task impacting directly the balance between false negative and false positive event detection. Additionally, sensors calibration are usually highly dependant of the target architecture and by essence hard to be transposed owing to technological dispersion.

For the purpose of comparison, an optimized version of the classic threshold based detection is computed with the same input dataset and tested for detection accuracy to finally compare with the ML models' performance. The disadvantage of the classical method is in choosing the threshold value, which is often chosen empirically over multiple test cases, therefore leading to non-generic coarse-tuned setting which may fail due to lack of sufficient test cases. To overcome this, the threshold for each individual DS is optimized and the collective result to predict the class is used i.e. if ≥ 8 (half of total number of sensors involved) DSs predict positive, the result is taken as FIA case, else nominal.

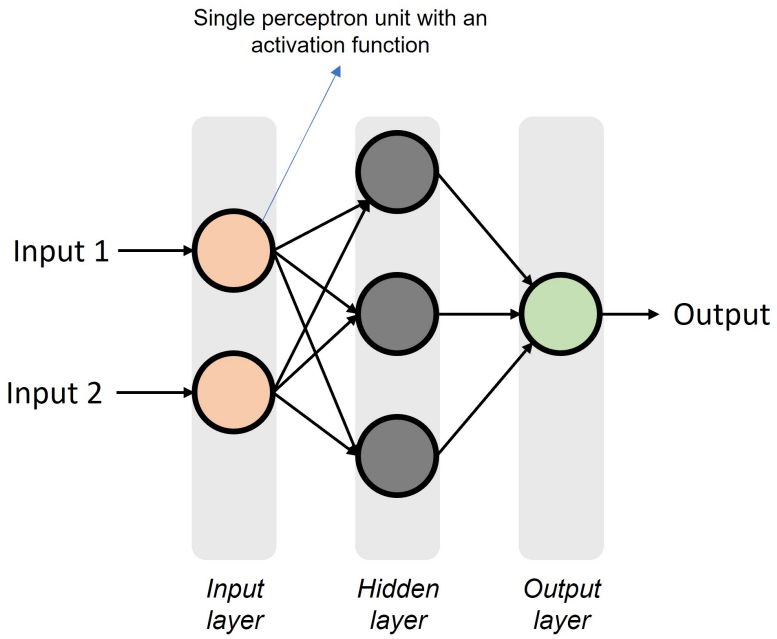

Figure 3.3: An example MLP

To optimize the thresholds for *each* DS, the buffer data (13×4 bytes[4]) is converted to an average form as shown below in equation 3.5, where X is the input vector and X' is the vector obtained after the averaging process:

$$\begin{gathered} \forall i \in \{0, 1, ..., 12\} \text{ and,} \\ X = \{V_i\} \text{ and } X' = \{V_i'\}, \text{ where,} \\ V_i' = V_i \text{ if } i = 0, \text{ else } V_i' = (V_i + V_{i-1})/2. \end{gathered} \tag{3.5}$$

Thereafter, the bounds (Lower/Upper for class Zero/One) of both the classes (0: Non-injection/Nominal, 1:Injection) are calculated by running a linear search over 80% (similar to training set ratio in ML methods) of the dataset. These bounds are used as threshold for the test set to detect FI states. The accuracy is measured by testing the optimized DS thresholds against the remaining 20% test data to record an aggregated accuracy over all the sensors.

3.4.3 Results

The results for EMFI detection are presented in this section. For the four different ML algorithms tested, the figure 3.4 presents a comparative study in performance. It is noticeable that the linear algorithm GNBC outperforms the others due to the fact that this algorithm is best suited for such kind of data as well as the widely accepted notion through empirical evidence that for any deep learning based approach to function with good accuracy, a dataset of minimum 10000 samples are necessary.

Since the GNBC works better than the rest of the tested ML algorithms, let us compare its results with the threshold based detection results as shown in figure 3.5 and in table 3.5 for true/false positive/negative rates.

[4]where 13 is the window size for one measurement and each of the 13 values are of size 4 bytes

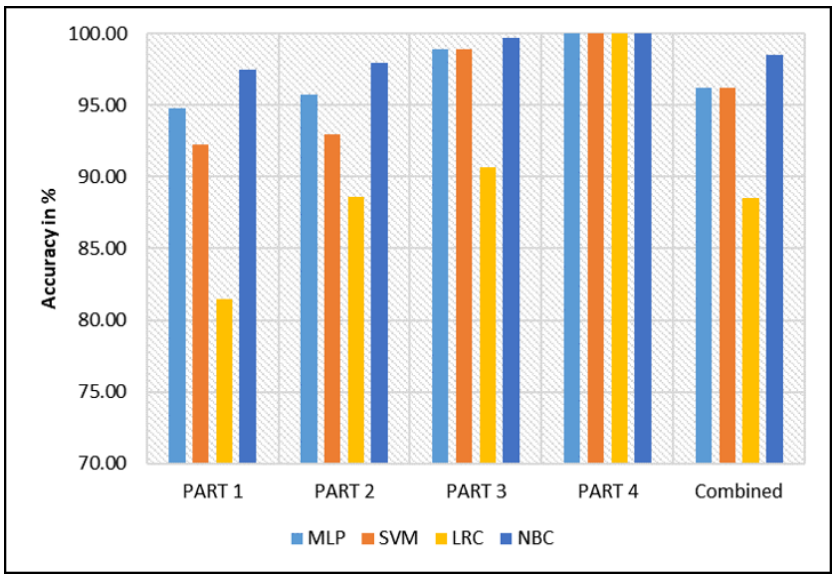

Figure 3.4: Performance comparison as accuracy in predicting EM Fault Injection from DS states over four different ML methods. Each method is tested separately over the four different parts of EMFI dataset, as well as over the combined dataset (all data merged together as one and randomized). Naive Bayes Classifier (NBC) outperforms other methods.

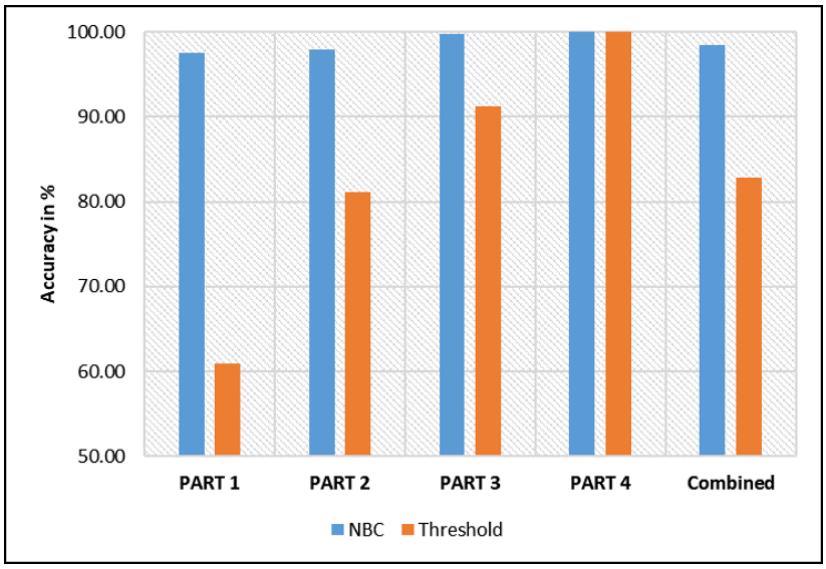

Figure 3.5: Performance comparison in accuracy of predicting EM FIA, from aggregated DS states, between GNBC and Threshold based method on the EMFI dataset. While there is minimal difference in accuracy for the GNBC over different parts of the dataset, the accuracy of threshold method is significantly affected and is always significantly less than that of the linear classification algorithm.

	(VALUES IN %)				
	False Positive	**False Negative**	**Acc: w/o injection**	**Acc: with injection**	**Overall**
	Naive Bayes Classifier				
PART 1	0.00	2.50	100.00	95.00	**97.50**
PART 2	0.00	2.02	100.00	95.96	**97.98**
PART 3	0.00	0.28	100.00	99.45	**99.72**
PART 4	0.00	0.00	100.00	100.00	**100.00**
Combined	0.00	1.49	100.00	**97.00**	**98.51**
	Thresholding Method				
PART 1	0.08	39.00	99.85	22.00	60.92
PART 2	0.00	18.83	100.00	62.35	81.17
PART 3	0.01	8.76	99.99	82.43	91.23
PART 4	0.00	0.00	100.00	100.00	100.00
Combined	0.00	17.18	100.00	**65.32**	82.82

Table 3.5: Detailed comparison of Threshold and GNBC results on EMFI detection.

3.5 ML detection for faults from combined sources of EMFI and CGFI

The classification problem presented in the section 3.3 is extended to perform forensic analysis of the attack in identifying the type of perturbation among the input multiple sources of fault injection along with the detection of presence/attempt to perform a fault injection. This problem could be modelled as a ternary classification but since the intended application of this module is in the hardware, a complex model would only consume more resources to compute classification for three classes viz. no-attack, EMFI and CGFI. Additionally, not all applications would require the classification between the perturbation type.

3.5.1 Two-stage detection framework

As stated above, the solution to predict the attack presence is crucial and the analysis of the type of attack is optional, although desirable. Additionally, the detection of any attack presence is a high priority task. Therefore, an efficient two stage detection framework is presented with the first stage being a fault attack presence detector which is a binary classifier giving only two types of outputs i.e., either presence of an attack or no attack presence. The second stage is activated based upon the output of the first stage if it detects an attack presence and then classifies the type of attack. This classifier can be multi-class classifier and would depend upon the number of different FIA datasets available to train from. In this proposal it is still a binary classifier since only two major fault injection sources are used as seen in the flowchart in figure 3.7. The modalities of operation of the two-stage detection framework can be understood from the simple illustration in figure 3.6.

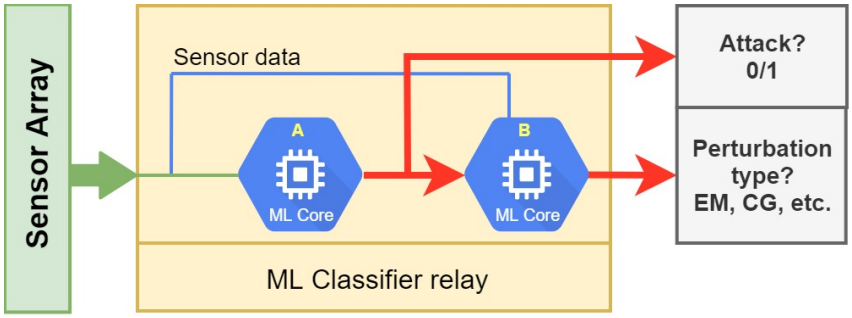

Figure 3.6: Two stage multi-source FIA detector

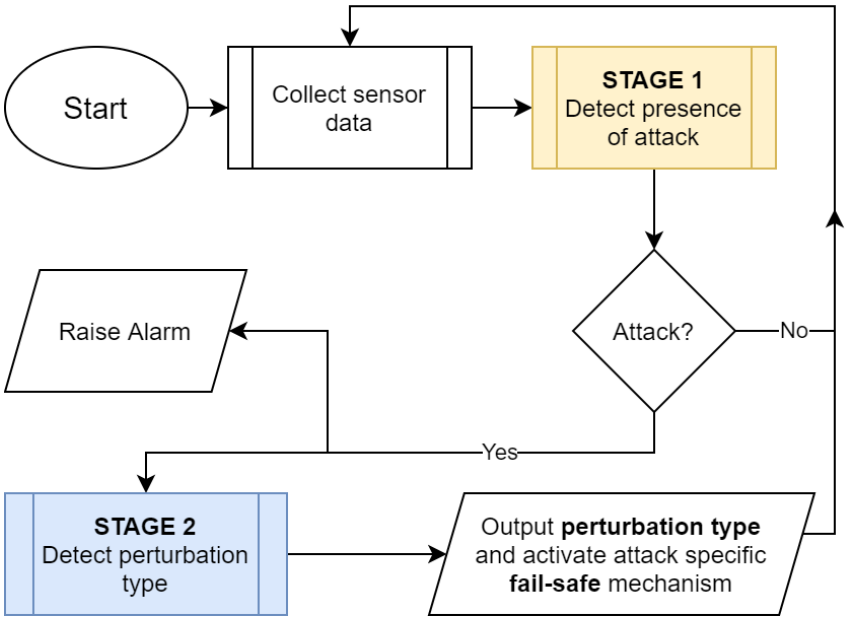

Figure 3.7: The proposed two-stage detection framework's modalities of operation and control flow.

3.5.2 Modes of evaluation and detection of EMFI and CGFI

To describe the detection capabilities (with scores) of the ML models with the EM- and CGFI datasets, multiple modes of evaluation are performed. Primarily, the classifier should be able to differentiate between nominal and injection classes (labelled as: 0 and 1 respectively). Additionally, the model should be able to detect the source of attack based on selected features from the training datasets. Since the sensors currently used have status saturation directly proportional to the strength of the attack, the problem of classifying the type of attack becomes harder. This is due to the fact that at highest saturation levels of the sensors, it becomes difficult to differentiate between the types of attack. This is the reason why sensor-aggregation strategy makes more sense where multiple sensors are placed at different locations and, even though some sensors might be saturated, some sensors might not be, allowing to characterize these behaviour of a group of sensor as features during the ML model training phase. The status values of the 16 DSs in nominal, CGFI and EMFI case is presented in Figure 3.8.

The classification task is divided into four parts, as mentioned below, and each of them is inferred separately over the same ML models.

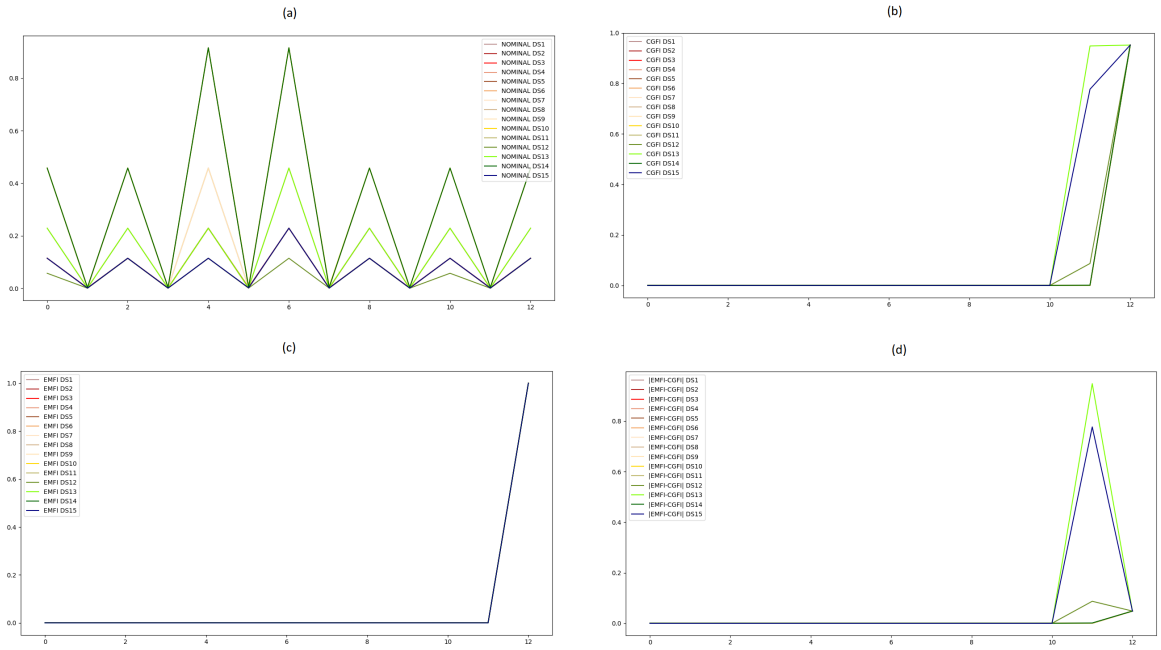

Figure 3.8: Comparison between states of 16 DSs for nominal as well CGFI and EMFI cases. The x-axis is the status buffer for each DS. (a) represents the nominal state of the DSs when no injection is performed, (b) represents the state of the DSs when CGFI is performed, (c) represents the state of the DSs when EMFI is performed, and (d) represents the difference in values of the DSs from CGFI and EMFI cases. It can be seen that some DSs behave similarly in both EMFI and CGFI cases (for example DS12). In case of one DS based system, it would not have been possible to differentiate between the type of attack. Therefore, sensor aggregartion provides more features which can be utilized by a classification algorithm to differentiate between the type of attack.

i. Detection of EMFI by performing binary classification between nominal and fault injected classes of EMFI dataset

ii. Detection of CGFI by performing binary classification between nominal and injected classes of CGFI dataset

iii. Combining the FIA datasets of both EMFI and CGFI and classification between the combined FIA datasets and nominal dataset from both EMFI and CGFI sessions

iv. Classification between EMFI and CGFI datasets to detect the perturbation type (by combining both FIA datasets and classifying between them)

3.5.3 Results

3.5.3.1 Classification between EMFI and nominal condition

The results are already provided in the section 3.4.3.

3.5.3.2 Classification between CGFI and Nominal condition

Similar comparison of performance between the ML and Threshold based method, shown in Table 3.6, is made for the CGFI datasets. The ML method produces 100% accurate predictions over the test dataset.

	Detection Accuracy (%)	
Dataset	**ML (GNBC)**	**Threshold**
Session 1 data	100	98.64
Session 2 data	100	99.27
Session 1 & 2 combined	100	98.24

Table 3.6: Detection accuracy comparison of CGFI between ML and Threshold based methods over CGFI datasets

3.5.4 Classification between combined EMFI and CGFI against Nominal condition

In this case the attack datasets from both EMFI and CGFI sessions are combined to form the attack class data. Similarly, the nominal class is also created for training/inference of the binary classifiers. The evaluation is also performed with the threshold method and the results are compared as shown in Table 3.7. The combination of diverse datasets increases the linear classification complexity manifold which can be observed from the performances of the ML and Threshold based models over the hybrid datasets.

	Classification Accuracy (%) of nominal dataset from		
Method	**EMFI**	**CGFI**	**EMFI+CGFI**
ML (GNBC)	98.51	100	91.98
Threshold	82.82	98.24	89.36

Table 3.7: Detection accuracy of multi-source FIA from nominal case with combined EMFI and CGFI attack case. (The EMFI and CGFI columns contain results of combined dataset of all sessions)

3.5.5 Classification of perturbation type: Forensic analysis to classify between EMFI and CGFI

Finally, for perturbation detection, the results are presented in Table 3.8. It is important to note that this classification is performed with just the attack case (fault injection datasets only) and the DSs, as mentioned earlier, have linear saturation directly proportional to attack strength. Thus, in the attack dataset with the two classes being CGFI and EMFI, the difference is minimal where the linear classifiers tend to fail in precisely optimizing the separation plane which is why the accuracy is low.

Finally, table 3.9 shows the accuracy percentages of the various classifications performed in one table. Since all the classifications are binary, two datasets are used for all cases. The attack diversity column of the table refers to the different conditions in which the data were recorded

	Values in %		
Method	**Accuracy**	**False Pos.**	**False Neg.**
ML (GNBC)	77.25	22.75	0
Threshold	39.61	0	60.39

Table 3.8: Accuracy comparison of perturbation detection between EMFI and CGFI of ML and Threshold methods

viz. four chip locations for EMFI and two different settings for CGFI. If both the datasets are used for either classification from nominal condition or between both the attack conditions, the diversity is indicated as 6. For classification between the types of perturbation the DS saturation leads to class overlapping leading to misclassification.

Classification between		**Attack**	**Detection Accuracy (%)**		
Data A	**Data B**	**Diversity**	**True detection**	**False Pos.**	**False Neg.**
EMFI	Nominal	4	98.51	0	1.49
CGFI	Nominal	2	100.00	0	0
⋆EMFI+CGFI	Nominal	6	91.98	0	8.02
⋆⋆EMFI	CGFI	6	77.25	22.74†	0

Table 3.9: Detection accuracy of the best performing ML model in various classification tasks. †This value denotes the percentage of tests where the ML model predicts CGFI for EMFI cases. ⋆Stage 1 detection result, ⋆⋆Stage 2 detection result

3.6 HLS based Hardware IP

As mentioned earlier, the purpose of having such modular approach to FIA and FIA source detection is to enable for its usage in hardware. The main objective of the proposed methodology and framework is for on-chip ML inference for FIA detection having trained offline with real data and thereby optimizing the ML core parameters for the operational environment of the DUT. To that end, the performance of the FIA detector is validated on hardware by utilizing HLS methodology for evaluation and testing with the DS activity data recording from the acquisition experiment performed on the Sakura-G FPGA board.

3.6.1 Experimental Setup

The evaluation setup is composed of a computer running Xilinx Vivado HLS 2019.2 EDA tool [83] connected to a Xilinx Vivado HLS compatible FPGA evaluation board, Digilent Arty S7-50. The design is run at 10nanoseconds clock-cycle. The on-board clock supports 100MHz clock frequency. Other features of the tiny yet capable FPGA includes 52,160 logic cells, 8,150 slices, 65,200 Flip-Flops, 2,700 Kilo-bits (Kb) of Block RAM, and 120 Digital Signal Processing (DSP) slices.

3.6.2 Methodology

The framework is composed of four main stages (please refer to figure 3.9) and can be understood from the list below:

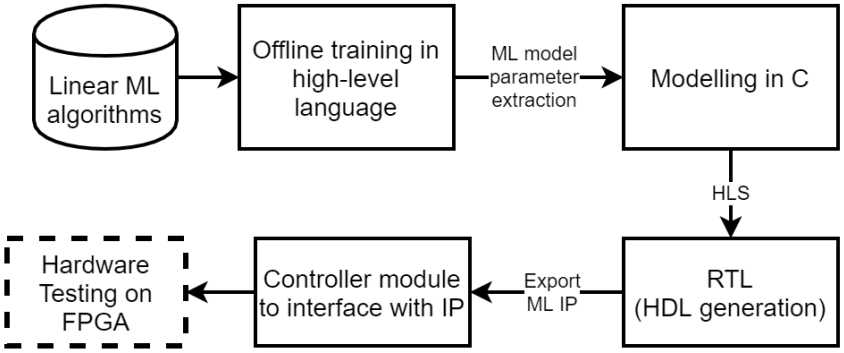

Figure 3.9: HLS Framework for testing the design on hardware

i Firstly, the training is carried out to train the ML model parameters using standard software ML frameworks in any high level language (in this case Python) after which the learnt parameters are extracted.

ii A C/C++ implementation of the design is created from scratch with no external library support and the ML inference model is initialized with the learnt parameters extracted at stage **i**.

iii The C source, along with a testbench written in C to validate the design performance at simulation level, is used to perform HLS using Vivado HLS design platform to generate RTL without any additional optimization, such as the usage of HLS pragmas, other than the ones already integrated in the design flow, thereby having no control upon the generated HDL code structure.

iv The generated RTL is packed into an IP and imported in the Vivado HLX suite where a controller program is written in Verilog to interface with the IP. Real DS test dataset for the ML IP is stored in a Random Access Memory (RAM) block which is used by the controller program to segment and test the IP for classification accuracy (see figure 3.10).

3.6.3 Experimental Results

The purpose of evaluating the inference results on hardware was to validate the detection accuracy which was successfully validated to be matching with the inference or tests performed offline and results presented in the sections 3.4.3 and 3.5.3.

The test dataset (test vectors) used to validate the classification accuracy of the ML IP created using HLS is same as used for offline inference in the ML testing phase. Upon evaluation, the classification capability remains unaltered at the hardware level. The resource utilization report for the HLS implementation is shown in Table 3.10 for the design with the controller module and test data as shown in figure 3.10.

Each DS consumes $\approxeq 400$ LUT (Look-Up Table) slices. In this design the minimum number of DSs required to achieve sensor aggregation and improved accuracy for the classification

Resource	Utilization	Available	Utilization (%)
LUT	8898	32600	27.29
LUTRAM	266	9600	2.77
FF	8478	65200	13.00
BRAM	1.50	75	2.00
DSP	43	120	35.83
IO	6	210	2.86
BUFG	1	32	3.13

Table 3.10: Post-implementation resource utilization of the Arty S7-50 FPGA for the whole design including the controller module and test data

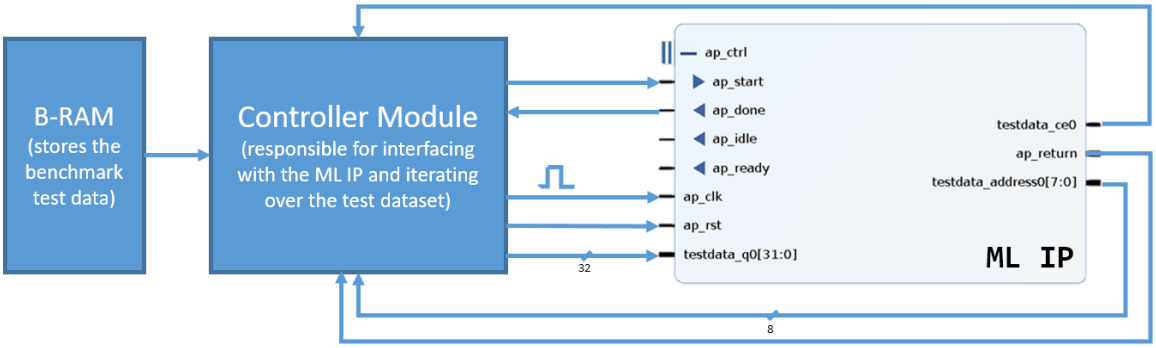

Figure 3.10: High level block diagram of the test setup with a controller module interfacing the ML HLS IP with the benchmark test data stored in a B-RAM

are chosen. Sixteen DSs which is approximately 6400 LUT slices on the FPGA fabric, are used. The FIA detection engine consumes 8898 LUT slices which is 39% greater than the total consumption of all the deployed DSs. However, it is to note that the FIA detector design is not optimized during the HLS and an optimized design can be similar in area of the total number of DSs deployed. In terms of throughput, the total number of cycles required to perform sensor aggregation and one classification, with a DS buffer length of 13 statuses, is 38275 clock cycles. The objective of the FPGA test is to justify the use-case and establish a proof of concept while keeping room for optimization.

3.7 Discussion

The advantage of having a two-stage detector is in quick and controlled response (with minimum false positive, which in our case is 0%). The first stage derives a presence of an attack and informs the mother system to take first-aid countermeasures, like stall execution or incident reporting, to quickly prevent any leak of critical security parameters. The second stage then reports the guessed perturbation type which may be used by the control system to activate any further fail-safe mechanism, while the system is already in alert state. Indeed, attack detection (Yes/No) is a matter of survival (hence must be accurate and fast), whereas “CG or EM” is more of interest for forensics purposes or for software/operating-system level decision making. This is analogous to two-tier firewall in networking. The control flow with detection probability is shown in the Figure 3.11. Additionally, in the two stages, different ML algorithms can be used. While in the first stage speed of detection along with accuracy is important, the second stage

can be more sophisticated in terms of complexity to enhance the detection accuracy at the cost of computation time. In the proposed method the ML model of the same algorithm with different trained parameters in both the stages are used but another axis of investigation could be to test the framework with a more sophisticated (Deep Neural Networks based) model employed in the second stage, depending upon the available resources.

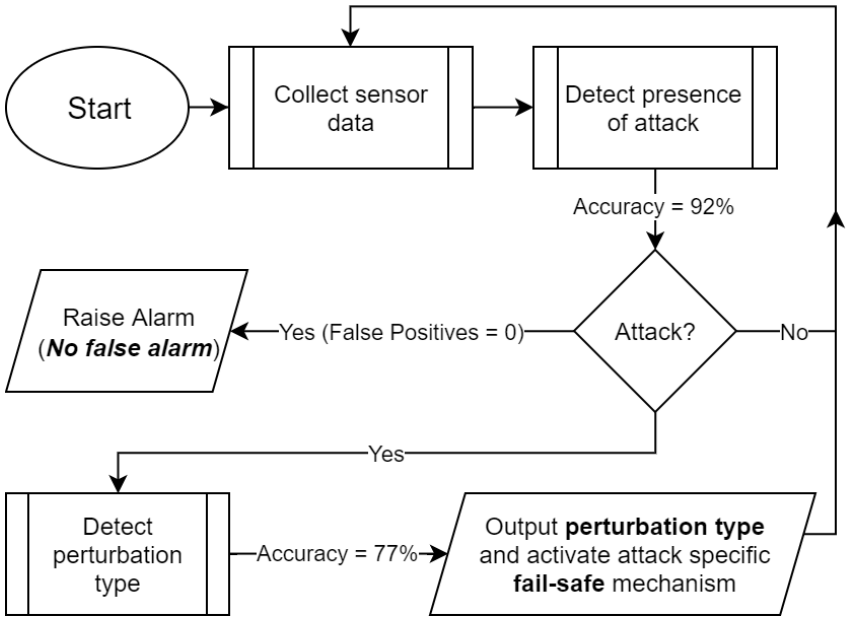

Figure 3.11: A high level control and data flow diagram of the multi-sourced attack detection with two-stage detector performance

3.8 Conclusion

In this chapter a two-stage fault injection detection framework for EMFI and CGFI is proposed with individual detection accuracy as well as for joint or multiple-soure fault injection detection accuracy. The proposed framework is not limited to only two sources and while the first stage is agnostic of the perturbation type since the problem is simplified to identify either the presence of a fault attack or not. This keeps room for validation and testing for more attack sources.

Furthermore, analysis of fault detection for individual attacks of EMFI and CGFI is performed and the accuracy of the ML based model with a classic threshold based method, where the threshold is trained instead of manually choosing a value, is compared. The ML model is evaluated on a FPGA with benchmark testing, using the same test dataset from the offline evaluation, to record the classification accuracy of the ML model on hardware. The ML model performs very well in detecting individual attacks with accuracy of 98.51% and 100.00% for EMFI and CGFI respectively, and 92% in detecting both CGFI and EMFI combined. Furthermore, to detect the perturbation type, classification is performed between CGFI and EMFI FIA dataset that prodcues 77.25% detection accuracy.Finally, all the methods are combined to form a two-stage FIA detector where the stage one predicts the existence of an attack when the DSs inputs can either be nominal, EMFI or CGFI. The second stage is enabled if the first stage detects an attack and, thus, it predicts the perturbation type.

The goal is to provide a solution that is integrate-able and scalable in terms of number of digital sensors and could be deployed as a countermeasure solution in any design against FIA, thereby improving the embedded systems security for the DUT.

In the next chapter, the second contribution of this thesis work is presented i.e., the detection of Hardware Trojans in a design using machine learning based techniques without even triggering the Trojan circuit in the DUT. The detection methodologies are based on offline evaluation using electromagnetic emanations data chip the chip.

Chapter 4

Insider (Hardware Trojan) Detection

Contents

4.1 Introduction

This chapter focuses on the detection of HTs in a design with a block-box approach i.e., it is not known to evaluator where in the design the HT is inserted, if the HT is activated or not during the evaluation phase, and even the impact of the HT on the actual design. To that end, there are three different types of HTs studied in this method which are also designed and inserted within the original logic at two different levels viz. the RTL and the P&R level mimicking the different phases in the semiconductor manufacturing process where a HT could be inserted as presented in the figure 2.4.

Hardware Trojan detection problem can be addressed in two ways i.e. by prevention or by detection. Through empirical evidence it is realized that despite all efforts, it is not possible to have an assurance that a HT would not be inserted in some phase of the design due to the lack of control in the operations in the different phases of the development and production processes. Therefore, the only reasonable method to stay safe from a HT attack on the design is by detecting it early in the supply chain before the product hits the market. With that in mind there could be a pre-silicon detection as well as post-silicon detection strategy for detection the HT. However, the threat is not resolved completely at the pre-silicon stage since there is still possibility of HT insertion at the last steps of fabrication that would let some Trojan circuit be in the design at the post-silicon stage.

Therefore, to counter this issue, a final security gating method is presented as solutions to detect HT in this chapter that does not expect the evaluator to have the knowledge of the HT insertion specifics. For the sake of complete transparency, during the data acquisition in the

proposed framework, the HT is not even activated. The design of the HT is also kept at minimal where for all the Trojan designs, they do not exceed even 1% of the original design in area overhead.

4.2 Machine Learning based Hardware Trojan detection

Similar to the fault injection detection, hardware Trojan detection can be modelled as a ML classification problem to be precise, binary classification with the two classes being the golden or reference model and the Trojan infected model. Furthermore, another methodology is presented that is based on outlier detection. Outlier detection is a ML method which falls under semi-supervised learning. In this case only one class data is available and using which the model is trained. Inference of such models depend on whether the unknown data is classified as an inlier, similar to the features of the training data, or an outlier, not matching with the data which was used to train the model.

In this chapter we will see why there is a need for outlier detection when supervised binary classification is an already available approach. This question can be answered with two points:

1. Firstly, supervised ML classification produce good detection accuracy in most cases but in few cases the accuracy is poor

2. The outlier detection method can be used to compare the design difference between two chip batches which can be an excellent metric to compare if any of the chip lot has a Trojan infection. Even if both the chip lots might contain Trojan, still the outlier detector will distinguish between them since both lots can originate from different production units and thus, cannot contain the same Trojan.

The proposed methods in the state-of-the-art have either no detection rate for the evaluation performed or a detection rate less than 70% even using the statistical approach. With the first method presented in this chapter using the supervised ML algorithm, a detection performance of $\tilde{9}0\%$ is achieved, while the second method, which is a combination of pre-processing using T-test and outlier detection algorithms, to obtain a very high detection rate of nearly 100%.

As mentioned earlier, the presented methodologies utilize both supervised and unsupervised learning approaches. The ML algorithms used to that end are detailed in the following subsections.

4.2.1 Supervised ML algorithms used for HT detection

The supervised ML algorithms are widely used for the classification and detection analysis. The definition of supervised ML is presented in the section 2.5.2. Here are some examples of supervised ML algorithms that also include SVM and MLP that were used in 3 for the purpose of EMFI attack detection.

- **Support Vector Machine** analyzes data used for classification and regression analysis. A SVM constructs a hyperplane or a set of hyperplanes in a higher dimensional space which can be used for classification, regression, or other tasks like outliers detection (also see 3.4.1). During the training phase, the SVM tries to find the hyperplane that has

the largest distance to the nearest training-data point of any class (so-called functional margin) [84].

SVMs are particularly effective in higher dimensional spaces and the computational complexity is also low. In some cases, they still outperform DNNs. The problem of HT detection involves overlapping data from both classes that are not linearly separable in two dimensions and, thus, SVM comes into play. The choice of kernel is very important in the process. A kernel or kernel function in a SVM transforms the data into a required form and returns the inner product between two points in a suitable feature space. These functions can be of type linear, nonlinear, polynomial, Radial Basis Function (RBF) and sigmoid, depending upon the type of the data distribution. The kernel function in the presented method in this chapter is chosen to be RBF with degree 3, gamma value or the kernel coefficient of RBF as 'scale' (which is equivalent to the value of $1/(number\_of\_featuers \times variance\_of\_X)$, where X is the input from the genuine Trojan free design), the regularization parameter as 1, and no limit on maximum iterations[1].

- **MLP**s as described in section 3.4.1, are efficient in learning patterns and important features from even poorly arranged data. It can be conveniently used to classify loosely formatted data as it automatically learns the principal features. The problem of HT detection is similar since the methodology lies in performing cartography of the chip with the DUT during operation without activating the HT and, therefore, the region of interest in the signals is unknown. The MLP is expected to identify and learn the intricate differences that is very difficult to estimate by statistical or heuristic methods. The results presented in this chapter comprise the Nearest Neighbor (NN) details of the MLP as provided below:
  - Fully connected layers: 3
  - Activation function: Rectilinear Units (attached to layers 1 and 2)
  - Regularization: Dropout (10% on layer 1 and 20% on layer 2)
  - Optimizer: ADAM
  - Error criterion (Loss function): Binary Cross-Entropy Loss (BCE)
  - Number of iterations (epochs): 400
  - Learning rate: 0.001

- **Decision Tree Classifier (DTC)** This algorithm creates tree models where the target variables can take a discrete set of values which are called classification trees. In these structures, "leaves" represent class labels and branches represent conjunctions of features that lead to those class labels. For the evaluation presented in this chapter based on DTC, the minimum number of samples required to split a node is kept as 2 with the minimum number of values in a leaf node as 1. The splitting function is "best" (it chooses the best split that provides the maximum information gain).

- **K-Nearest Neighbors** This is a non-parametric method used for classification and regression. In K-NN classification, the output is a class membership. An object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its *k* nearest neighbors (*k* is a positive integer, typically small). If $k = 1$, then the object is simply assigned to the class of that single nearest neighbor.For the second presented method in this chapter, the number of neighbors parameter is kept at 5 with uniform weight distribution.

[1] training stops when accuracy does not increase any further

For the purpose of the second method presented in this chapter that utilizes un-supervised learning approach (namely Outlier detectors[2]), the following list of ML algorithms have been evaluated with.

- **One Class SVM** This type of SVM is trained on data that has only one class, which is the "normal" class. It infers the properties of normal cases and from these properties, it is able to predict which test cases are unlike the normal case [85]. The kernel function used in the presented methodology in this chapter is same as the supervised SVM i.e. RBF with gamma value as 0.1, and the upper bound on the fraction of training errors as well as lower bound of the fraction of the support vectors at 0.001.
- **Isolation-Forest (IF)** An IF builds a set of trees for a given data set. These trees are also known as iTrees form the basis of detection of anomalies. It isolates observations by randomly selecting a feature and then randomly selects a split value between the maximum and minimum values of the selected feature [86]. In the presented methodology using unsupervised method for the IF based evaluation, the random state is decided from a deterministic random number generator with seed value 42, the maximum number of samples to draw from the input to train the base estimators is 100, and the contamination percentage is 0 since we used the genuine or normal dataset which does not contain any outliers.
- **Elliptical Envelope (EE)** This algorithm models the data as a high dimensional Gaussian distribution with possible co-variances between feature dimensions. It attempts to find a boundary ellipse that contains most of the data. Any data outside of the ellipse is classified as anomalous. For the elliptical envelope evaluation method in this chapter, the random state is constant at 0, and the contamination in the training data is also 0 since normal dataset with no outliers is used.
- **Local Outlier Factor (LOF)** LOF is an unsupervised anomaly detection method which computes the local density deviation of a given data point with respect to its neighbors. It considers the test samples as outliers that have a substantially lower density than their neighbors [87]. For the LOF algorithm based evaluation in the second method presented in this chapter, the number of neighbors are kept to be 20 with novelty detection mode to train with the genuine or normal input.

The ML parameters for all the algorithms listed above are selected for the training or testing is empirically chosen for best performance.

4.3 Hardware Trojan Design

Three different HTs with the corresponding sizes of 0.56%, 0.27% and 0.09% (as compared to the whole DUT) are implemented on a DE1 SoC Cyclone V FPGA and an Arty-7 FPGA for the experimentation. The characteristics of all the three HTs are presented in table 4.1.

For the evaluation, two RISC-V implementations are selected:

- PicoRV32 [88] (programmed in the DE1-SoC board with a Cyclone V FPGA)

[2]Outlier/Novelty detection is a sub-class of ML which is generally used to detect abnormal/unusual observations or data.

- Freedom E310 [89] (programmed in the Arty-7 FPGA board).

In the experiments, three HTs (HT1, HT2 and HT3) with different sizes are inserted in the above two target designs.

4.3.1 P&R Level

HT1 and HT2 are inserted in the PicoRV32 design in the DE1-SoC Cyclone V FPGA board.

- **Payloads** Once activated, HT1 and HT2 are able to arbitrarily modify the program counter of the processor.
- **Triggers** The trigger is based on a specific DIV instruction in HT1 and on registers in HT2.

These two HTs are inserted at Place & Route level: the reference design (without HT) and infected design (with HT) have the same layout except where the HTs are inserted. The overhead of the HT1 and HT2 are respectively 0.53% and 0.27% of the reference design (PicoRV32).

4.3.2 RTL level

The HT3 is inserted in the Freedom E310 Risc-V processor in the Arty-7 FPGA board.

- **Payload** Once activated, the HT3 modifies arbitrarily the privileged level of the processor hence performing a privilege escalation attack.
- **Trigger** The trigger of the HT3 is also based on the specific DIV instruction.

The overhead of the HT3 is 0.09% of the reference design (Freedom processor). The HT3 is inserted at the RTL level and therefore, using the automatic tool for the FPGA floorplan generation, the layout of HT3 and genuine design is completely different.

	Target design	Insertion phase	Trigger	Payload	Overhead
HT1	PicoRV32	P&R	Specific Instruction	Modify PC	0.53%
HT2	PicoRV32	P&R	Registers	Modify PC	0.27%
HT3	Freedom	RTL	Specific Instruction	Modify privilege level	0.09%

Table 4.1: HT designs for the experimentation on RISC-V processors

4.4 Hardware Trojan Detection Using Electromagnetic Emanation

4.4.1 Experimental Setup

The method of collecting the data is based on EM cartography. The elements of the setup include a computer with controller software, an EM sensor probe, a displacement table (XYZ table), and an oscilloscope. The setup is similar the EMFI platform as shown in figure 2.1. The setup on the XYZ table of the FPGA board is shown in figure 4.1.

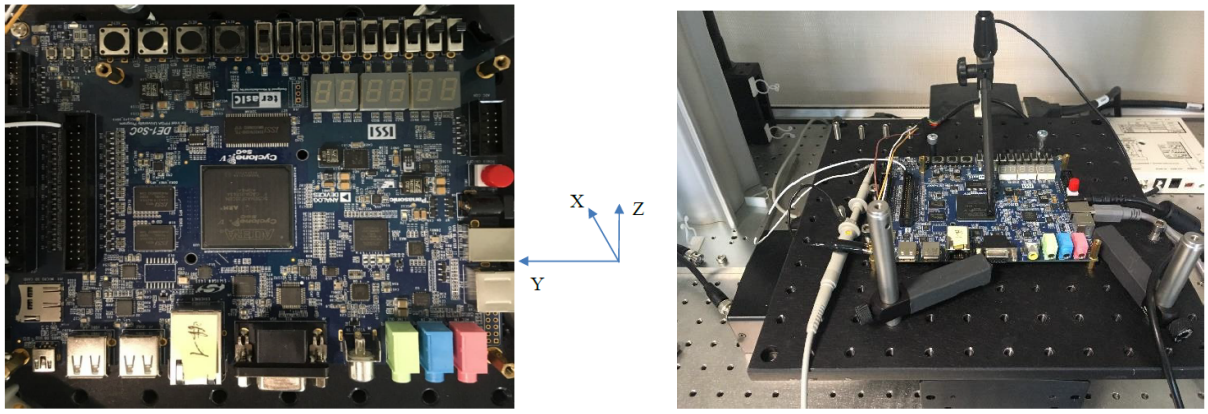

Figure 4.1: (Left) Top view of Altera board oriented in the plate plan XY. (Right) EM Cartography trace acquisition setup for HT detection

4.4.2 Acquisition

Figure 4.2 presents the overview of the RISC-V development process and the cartography setup used for the data acquisition. The RISC-V design is defined using an HDL (Verilog) implementation, and then the design is synthesized followed by P&R to obtain the floorplan and finally generate the bitstream for programming the corresponding FPGAs (Cyclone V for PicoRV32 and Arty-7 for Freedom).

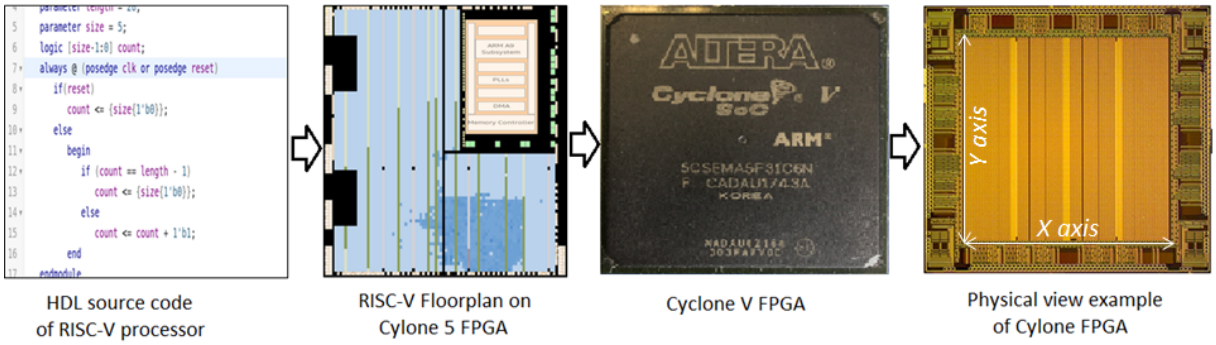

Figure 4.2: Cartography process overview

In the experiments, EM cartography (2D, automated by an XY moving stage) is used in order to measure the EM emanations on each area of the FPGA chip (as shown in the figure 4.2).

One cartography consists of multiple measurements at several points on the target circuit. In order to cover the whole FPGA, N_x steps of 2mm (for DE1 SoC board) and 1mm (for Arty-7 board) along X-axis and N_y steps (2mm for DE1 SoC and 1mm for Arty board) along Y-axis are performed. One cartography consists in $P = N_x \times N_y$ measurement points.

For each measurement point, N EM traces are acquired where N corresponds to the number of cartographies performed. Finally, each EM trace contains T temporal samples.

Cartogrpahy on DE1-SoC board
For the DE1 SoC board, $N = 50$ cartographies are performed of size $N_x \times N_y$ where $N_x = 13$ and $N_y = 13$ (i.e., $P = 169$) for each design.

Figure 4.5 gives an overview of this dataset (on the left) where $N = 50$ represents the number of cartographies, 13×13 represents the N_x steps of 2 mm along X-axis and N_y steps along Y-axis of the cartography and 5000 is the amount of samples of each EM trace.

Cartogrpahy on Arty-7 FPGA board
On the Arty-7 board, $N = 50$ cartographies of size $N_x \times N_y$ where $N_x = 10$ and $N_y = 10$ (i.e., $P = 100$) are performed for each design.

For both boards, the acquired traces consist in $T = 5000$ temporal samples.

4.4.3 Some state-of-the-art detection methodologies for the purpose of comparison

4.4.4 Raw EM traces comparison

The most straightforward approach is comparing the EM cartography traces of genuine and infected designs and trying to visually identify or detect the difference caused by the HT insertion.

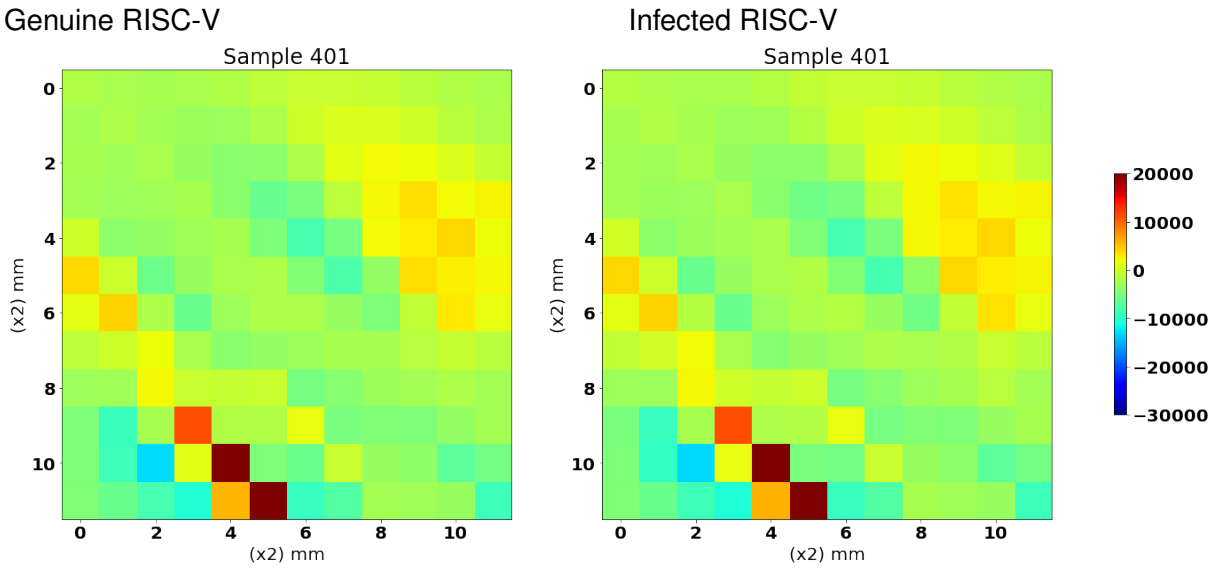

Figure 4.3: (left) EM cartography of Genuine design on PicoRV32 and (right) proposed HT1 Infected design for sample 401 in the cartography dataset

The figure 4.3 presents an example of the EM cartography result for the temporal sample 401 of the genuine design (PicoRV32 on the left) and of the HT1 infected design (presented on the right). It is clearly evident that it is not possible to visually distinguish the difference created by the HT insertion. The same comparisons for other temporal samples and for other HTs (HT2 and HT3) give the same results. In conclusion, we cannot detect the HT inserted in the RISC-V processor just based on visual comparison of the raw traces. Maybe in some cases where

the effect of the HT is very significant which is not the case usually since the main motivation behind the design of a HT to make it stealthy.

4.4.5 HT detection based on T-test metric

As discussed in the section 2.3.4, one of the approaches in the state of the art is using the T-test metric for the detection.

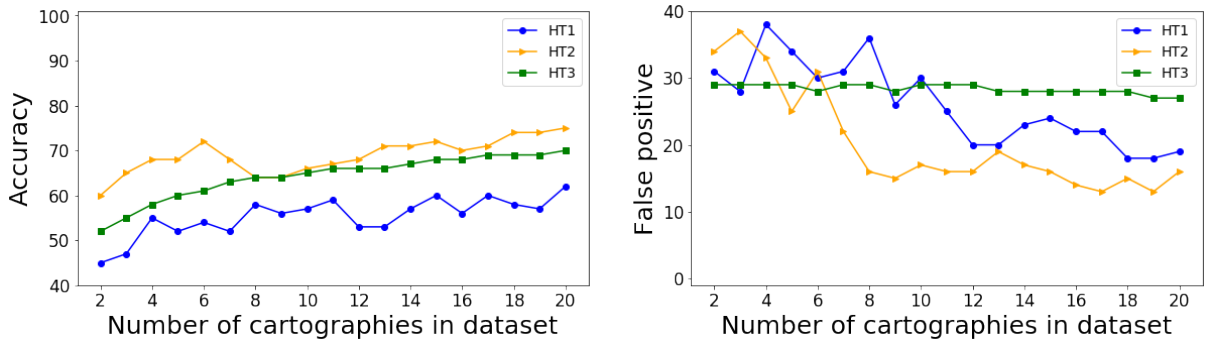

Figure 4.4: Detection of HTs using T-test metric

In order to evaluate the performance of T-test detection method, an evaluation is performed to record its detection rate for different numbers of cartographies that are selected for the T-test computation. The results of the T-test metric based HT detection are presented in figure 4.4 that shows the detection and the false positive rates of this method for all the 3 HTs in a function of the number of cartographies used for the T-test computation.

For detection purpose, a parameter c is used as the threshold coefficient. In each case, if the corresponding T-test value for one selected measurement point and one selected temporal sample is greater than $c = 2.0$ times of the reference T-test value, it can be considered as a HT. We can notice (in Figure 4.4) that the detection results for all the 3 HTs are between 55% and 75% with a very high false positive rate (between 15% and 30%). So we can infer that the T-test performance is poor and, moreover, the performance of the T-test metric also depends upon the selected temporal sample for the test, which is a difficult manual selection task. In some specific samples, we can observe a big difference between the T-test coefficient of the genuine design and the infected design. But in many other samples, we cannot see the difference between them. The measurement point of the cartography can, therefore, impact the detection results.

4.4.6 Novel ML based detection methodologies

4.4.6.1 Dataset description and Cartography Data preparation for ML processing

The Figure 4.5 presents the data formatting performed for the ML based methods for detecting the HT1 and HT2 on DE1 SoC board. In this figure, N represents the number of cartographies, 13×13 represents the N_x steps of 2mm along X-axis and N_y steps along Y-axis of the cartography and 5000 is the amount of samples of each EM trace. In the presented method, the cartography of each sample is used as input. Therefore, for one cartography, there are 5000

input vectors of length 169 (5000x169) for HT1 & HT2 (see paragraph 4.4.2). For the HT3 on Arty board, there are 5000 input vectors of length 100 (5000x100) (see paragraph 4.4.2).

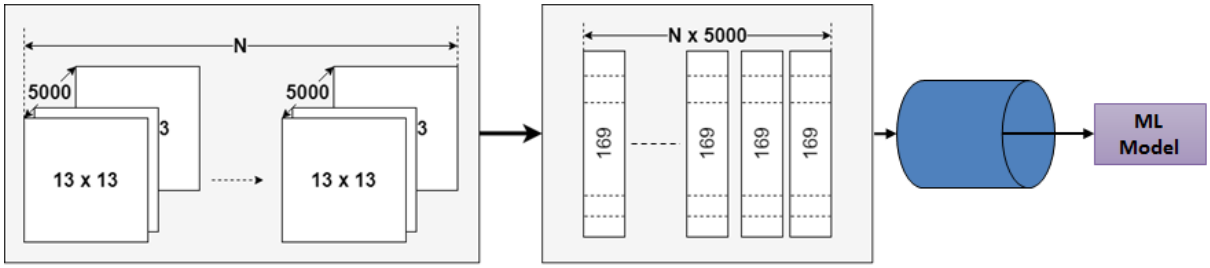

Figure 4.5: Data preparation for the machine learning methods

4.4.7 Detection methodology based on supervised ML models

For the supervised ML based detection method, the learning is performed directly using the raw EM traces with dimensionality reduction as shown in figure 4.5 for the detection of HTs for all the datasets of HT1, HT2 and HT3 recorded from the cartography sessions of both the FPGA platforms.

The methodology, including the cartography traces acquisition, comprises the following succinct steps:

1. Acquire the EM cartography traces of the reference design (EM_{ref}) as well as the HT infected design (EM_{HT})
2. Perform dimensionality reduction and data formatting to prepare the training dataset for the ML models
3. Using the formatted EM cartography traces (EM_{ref} and EM_{HT}) train the supervised machine learning algorithms
4. Acquire the EM cartography traces of the test design EM_{test}
5. Perform inference on the trained models using EM_{test} dataset
6. Record ML model performance accuracy on the test cartography dataset

4.4.7.1 Detection Results

The detection results using all five supervised ML algorithms viz. SVM, MLP, DTC, LRC and K-NN classifier are computed based on the ”detection probability” and the ”false positive rate” of the methods when the number of cartographies used in the training phase are varied. This means that the selection is of (x) cartographies amongst 50 cartographies of genuine dataset and HTs (HT1 & HT2 and HT3) or infected datasets while the number of cartographies used for the detection phase is $50-x$[3]. For this test, the number of x is varied from 1 to 22 cartographies (amongst the 50 cartographies of each datasets).

The results are presented in figure 4.6.

[3]Training data should not be used for inferenc of a ML model for performance benchmarking

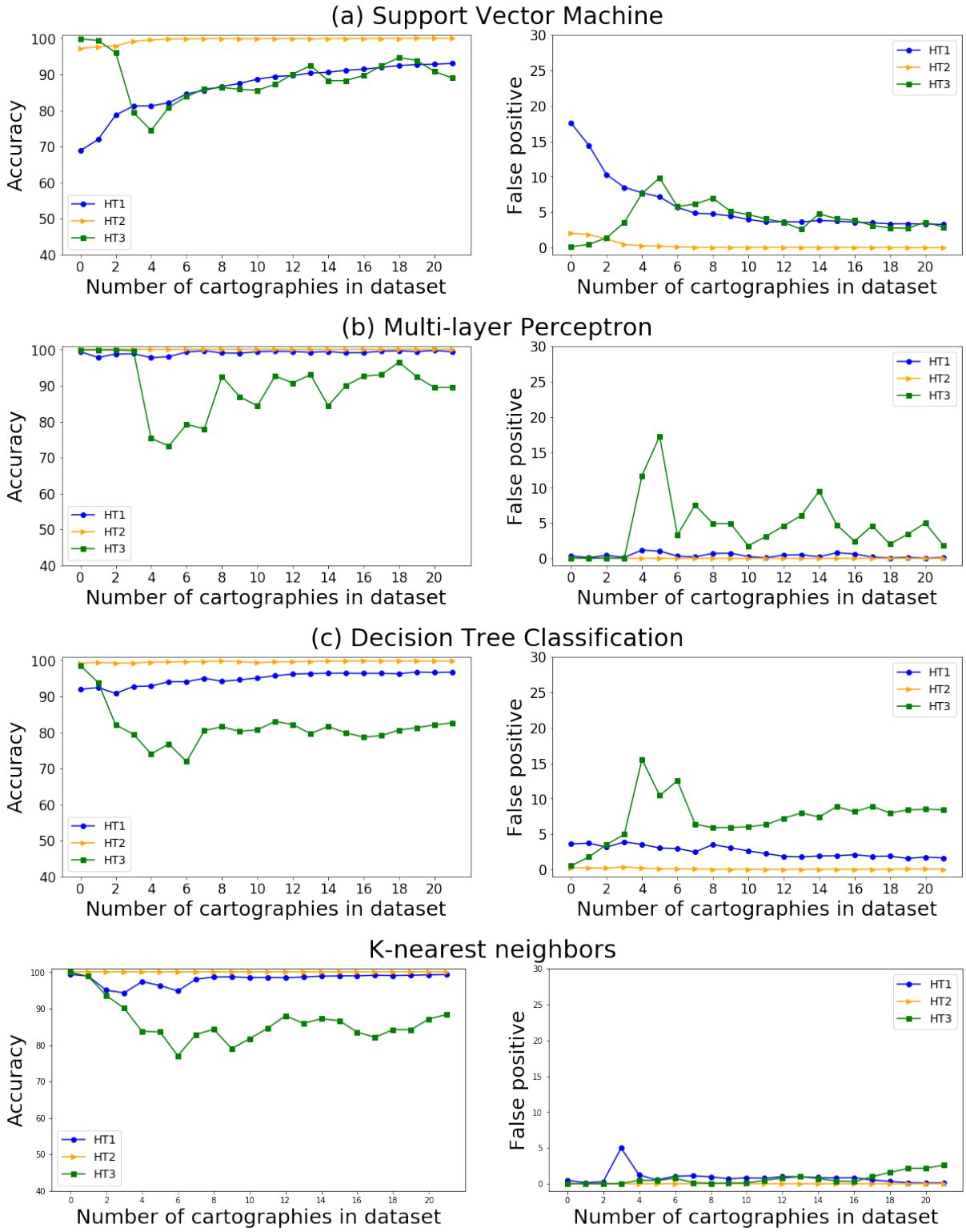

Figure 4.6: Detection of HTs using supervised ML algorithms

From the test results in figure 4.6, the detection rate of SVM and MLP are greater than 90% or even above with varying number of cartographies in the training phase from 1 to 20, and the false positive rate is less than 4%. For the DTC, the detection rates for the HT1 and HT2 are greater than 95% with a false positive less than 1%. However, the detection rate for the HT3 is less than 80% with a false positive of 4%. With the LRC, the detection rate of the HT1 is good (greater than 90%) with a false positive rate of 3%. However, the detection results for HT2 and HT3 are poor (between 50% to 60%) with a false positive rate of 20%. For the last algorithm (K-NN Classifier), the detection rates of all 3 HTs are greater than 80% with a false positive rate less than 3%. Therefore, the SVM and MLP based ML detection methods can detect the HTs with a good accuracy as compared to the T-test based method in section 4.4.5.

4.4.8 Detection methodology based on unsupervised ML models

The accuracy of detection observed with the supervised ML models are good, however, there are still rooms for improvement. To improve the accuracy as well as removing the need for a second design for comparison, as described in the section 4.2, a new hybrid T-Test and Outlier-detector ML based method is presented in this section. To revisit the second motivation (removing the need for a second dataset) we can further identify them and apply to two scenarios, such as:

1. When we want to detect if two different chip batches are the same or not. It can detect any difference between two chip batches.

2. When we want to detect if a test data (from a test chip) belongs to the same distribution as the reference observations (from the reference chips). If the test data is detected as an outlier, we can conclude that there is some modification (HT) on the test chip.

For the sake of establishing the utility of combining T-Test metrics with the outlier ML models, direct training of dataset on the outlier ML models is also performed. The methodology for this is same as the supervised ML method based detection by simply plugging in the outlier detection ML models instead of the supervised learning ML models and using the raw EM cartography traces as shown in figure 4.5. However, using the raw EM values as input, a low performance with a high false positive and false negative rate (between 60% to 40%, depending on the selected outlier detection algorithms) is obtained.

Therefore, to overcome this issue of high false positive and negative rates, the T-Test metric is introduced instead of directly using raw EM cartography data. The methodology for this hybrid approach could be listed into the following steps:

1. Acquire EM cartography traces of the reference design

2. Compute the T-test value of the reference design (T_{ref})

3. Train the Outliers detection algorithms using the T-test values (T_{ref})

4. Acquire the EM cartography traces of the test design

5. Compute the T-test value of the test design (T_{test})

6. Test the trained Outlier detection algorithms with (T_{test}) for inference

This methodology follows the same principle explained in the section 2.5.7. For the T-test of reference design, two datasets measured at 2 different moments are used for the evaluation (say Gt1 and Gt2). Then for each new test dataset (T), the T-test between T and Gt1 is computed.

4.4.8.1 Detection Results

The performance of these algorithms are evaluated by varying the number of cartographies used for the T-test computation from 3 to 22 cartographies (amongst the 50 cartographies for each dataset). Then the four outlier detection ML methods (One-Class SVM, EE, IF, and LOF) are applied for the learning part. The detection results of the 3 HTs using these methods are presented in figure 4.7. The obtained results show that the detection rates of all selected algorithms are greater than 95%. Particularly, the detection rate of One-Class SVM, EE and LOF detection rates are close to 100%. However, the One-Class SVM has a poor false positive rate (between 10% to 30% for HT2 and HT3) compared to other algorithms (nearly 1%). Therefore, the results show that it is possible to effectively detect all the three designed HTs using the EE and LOF detection algorithms. It is also noticeable that the new detection methods are much more efficient than T-test and supervised ML based detection methods individually. Table 4.2 demonstrates the performance comparison of the new hybrid method with those in the state of the art.

4.5 Discussion

Many methods and techniques are studied and proposed for the detection of HTs, but there is no universal method that can detect all HTs successfully. Atleast, there is no viable means to guarantee that. Because of the complexity of the HT, the combination of different techniques may be required in order to increase the coverage of the detection over various different types of HTs, since one of the observations in this chapter is also about the differences in detection accuracy of the same algorithm for different HT designs.

The supervised ML algorithm based methods are efficient to detect the HT but the drawback in these methods is the need for the genuine design dataset, or the golden model, and also the dataset for all the HT infected designs. For the second method, the outlier detection algorithms, such as IF and LOF detector, produce promising results comparing to those in the state of the art (table 4.2). Therefore, it provides an exciting exploratory field for the study of this method for HT detection.

	Method	**Target**	**HT Size** (%)	**Detection rate**
State-of-the-art	Raw trace comparison [44]	RISC-V	0.53, 0.27, 0.09	nc
	T-test [90]	RISC-V	0.53, 0.27, 0.09	70%
	One-Class SVM [91]	RISC-V	0.53, 0.27, 0.09	60%
Presented methods in this chapter	Supervised ML methods	RISC-V	0.53, 0.27, 0.09	$\approx 90\%$
	Hybrid T-Test & Outlier detection methods	RISC-V	0.53, 0.27, 0.09	$\approx 100\%$

Table 4.2: Comparison of detection methods

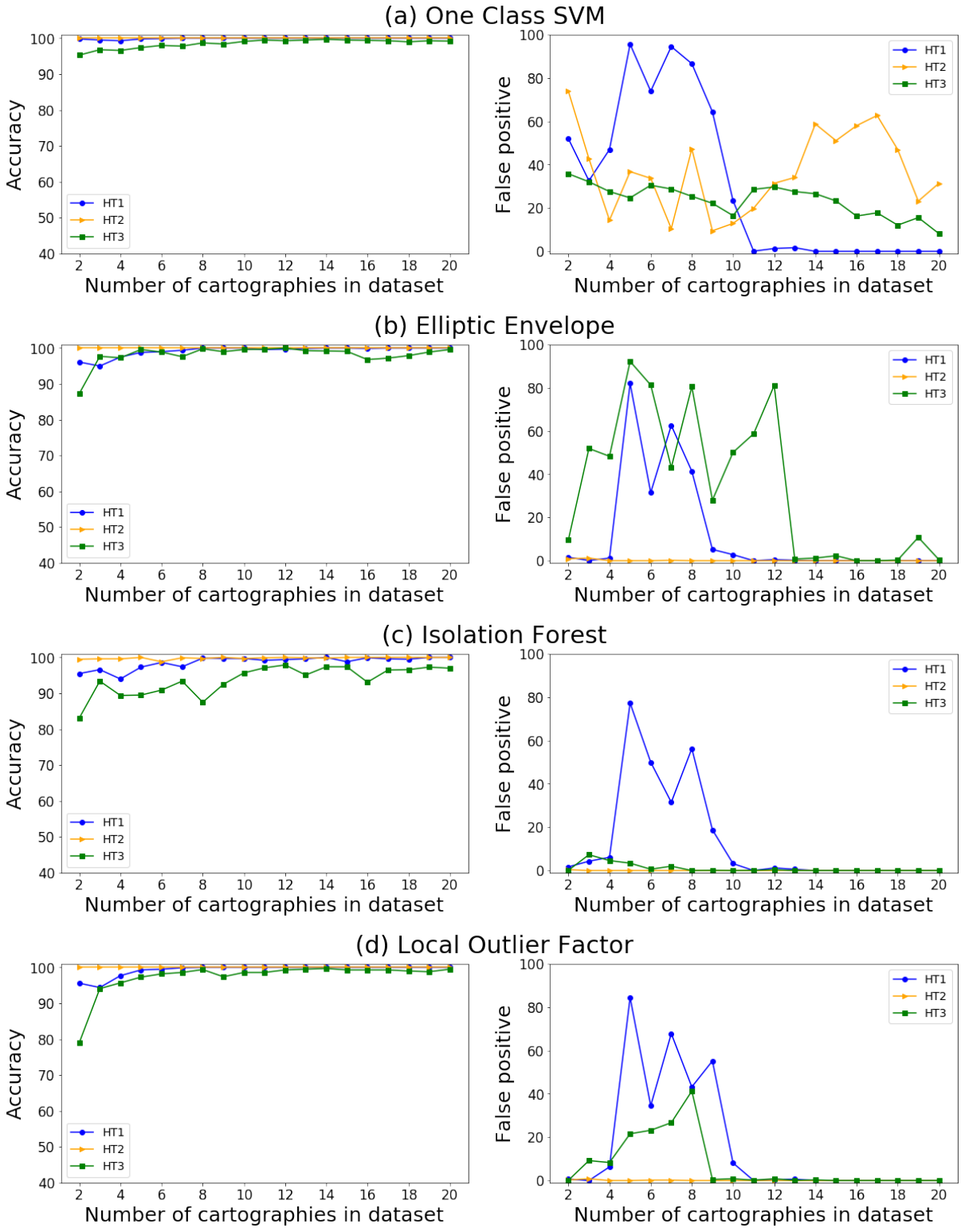

Figure 4.7: Detection of HTs using Outlier detection ML models

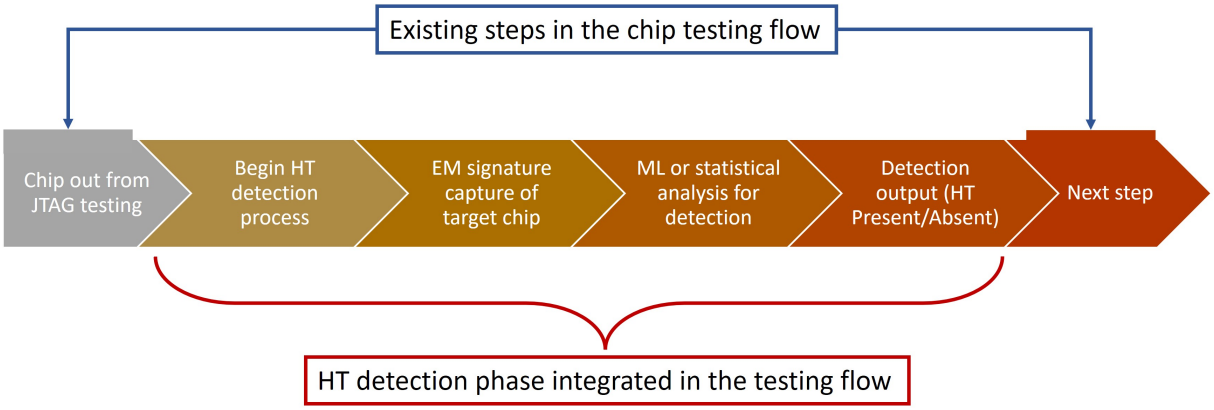

Figure 4.8: Integration of HT testing for chip lots in the standard production testing process flow

4.6 Conclusion

In this chapter, various different approaches of ML based HT detection is presented wherein leveraging supervised as well as unsupervised ML algorithms and even the use of T-Test metric for improving the accuracy of the outlier based detection methods. It is observed that the side channel based evaluation through EM cartography provides promising data since it is possible to identify all the three different Trojan designs presented in this chapter with the novel hybrid T-Test and Outlier detection based methodology without even trigger the Trojan circuit. It is impressive given the fact that the HTs presented in this chapter are very stealthy with a very minimal overhead in the design area (as low as 0.1%).

The motivation of providing such approach is for ease of integration in the development and production process (refer figure 1.1) such as shown in figure 4.8 with a minimum impact on the time to market while incorporating the assurance of high accuracy HT testing and thereby improving the embedded systems securing for the DUT using ML based approaches.

In the next chapter, an AI based intrusion detection system is presented as the final contribution of this thesis work, that focuses towards improving the security of embedded systems from the operating system level, by aggregating the collected data from a multitude of sensors on-board as well as inside the chip, along with the data from the network activity, and then detecting the presence of any intrusion by using machine learning methods for detection.

Chapter 5

Intrusion Detection System

Contents

5.1 Introduction

In this chapter, an interesting and very practical study and experimentation is performed wherein, using AI techniques, a smart parallel architecture of an Intrusion Detection System (IDS) is presented targeting multiple sources of attacks. As mentioned in the section 2.4, the classification of an IDS could be broadly in two different categories i.e., HIDS and NIDS. The HIDS can have inputs from internal sensors from within the SoC or from on-board as well as external sensors. The presented IDS in this chapter is built in mind with a global IDS for edge devices that can be deployed in any connected device infrastructure. This means that all the types of IDS are incorporated within the presented IDS except the host application data source is not utilized for the detection purpose. Additionally, the design is enabled to interface with a server via CoAP IoT protocol to send telemetry data of activity from the device to the cloud, send temporal intrusion detection status, as well as receive updates from the cloud.

The presented IDS can, thus, be featured as shown in the table 5.1 following the taxonomy of IDSs shown in figure 2.5. The detection is based on ML based methods and is split between network and sensors IDS since the number of features in the combined data from the sensors as well as network packets are too many to be trained for a good and acceptable accuracy from one ML model. Therefore, the network attacks are identified by the network IDS, also known as NIDS, and the attacks targeted at the physical layer (at the sensors) is detected by the sensor IDS. also known as HIDS.

The evaluation of the presented IDS is done mimicking a Vehicle-to-Anything (V2X) infrastructure and the ML detection core is trained using the normal operating condition of the

Category	Features
Application Layer	Software (OS)
Source of data	Network packets Internal chip sensors External sensor array connected to the core
Detection type	Anomalies Intrusion through sensor faults Physical threats Network intrusion attacks
Methodology	Unsupervised ML based intrusion detection
Upgradeability	ML parameter upgrade over the air

Table 5.1: Features of the presented IDS

different sensors and network communication without using any attack datasets. This is to ensure a nominal working boundary for the IDS that would detect any activity beyond the normal functioning condition as an outlier.

5.2 Automotive IDS use-case

A typical cybersecurity case includes:

- the system or assets to protect
- the threat actors who can compromise the system's integrity
- the available threat surfaces and different ways the system could be targeted by adversaries
- the impacts to the devices and user that would incur in case of a cybersecurity incident

In order to develop the presented IDS, an automotive cybersecurity use-case is chosen which is close to a real world scenario of recent times due to the increasing number of smart cars in the market with advanced functionalities and usability due to the connected V2X infrastructure (presented IDS environment high level diagram is shown in figure 5.1). To realize the scenario, it is not left to simulation based evaluation, but a real setting is created that includes a dedicated control environment (emulated inside a big room in a office building) and a Smart-car Robot (SCR) is placed in it to traverse like a real car on the street. The robot is equipped with almost all the sensors available in a real smart car and even the equipment to manipulate the car in the controlled environment to replicate various attack scenarios are arranged.

5.2.1 Security Threats and Attack Surface covered in the presented solution

The attack surface of an edge device varies from case to case, depending on its connectivity features (WiFi, Bluetooth, etc), its hardware and software architectures (OS layer, bare metal layer, MCU or ASIC based edge systems, etc.). The threat categories identified by the presented IDS can be listed as:

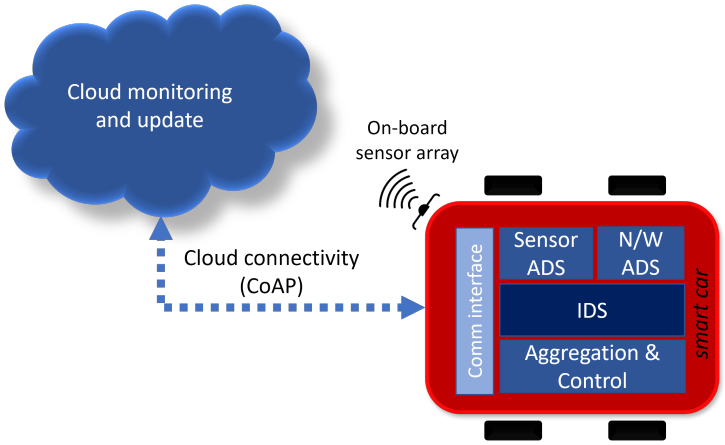

Figure 5.1: High level diagram of the presented V2X IoT emulated environment

- **FIA**
  As discussed in chapter 3, this class of attack consists in actively stressing a system in order to compromise its security. In short, when perturbing a security system, an attacker can induce faults during a computation or generate bit-flips in memory cells. Those effects can then be exploited for sensitive variable recovery, for example with Differential Fault Analysis (DFA) [92] or to skip specific instructions in order to bypass a security mechanism. There are several physical channels that can be used to create the perturbation: power glitching, CGFI, temperature glitching, EMFI, LFI, etc., as well as software or hybrid methods [93].

  Fault tolerant systems can be designed at the cost of performance by using redundancy as countermeasure for fault injection attacks, in various ways. Active defense against fault injection consists of analysing sensors values to detect attacks at `runtime`: this approach, enhanced with machine learning, is one of the focus of the presented IDS.

- **Connectivity related cyberattacks**
  These attacks target communication interfaces of the devices. Multiple attacks can be realized with different objectives such as:

  DoS attacks: aiming at flooding a service with unrealistic traffic in order to prevent the device to operate correctly, for example by occupying all the available bandwidth, consuming all the device resources, making the system crash or preventing legitimate traffic to reach its destination.

  Address Resolution Protocol (ARP) spoofing: This is a different type of attack where the adversary aims at impersonating a valid host device, causing the target device to send any traffic meant for the true host to the attacker instead. The attacker can then ”listen” or ”sniff” to the packets, discard them, or tamper them before sending back to the true intended host device in the network (such attack is also known as Man-in-the-Middle (MIMT) attack). ARP is a protocol used in Ethernet and WiFi to resolve a Media Access Control (MAC) address, given an Internet Protocol (IP) address. Devices can broadcast ARP requests to a network when they need the MAC address associated to a certain IP (in the attack case, the IP of the host). Anyone connected to the network can reply with an ARP response. Since ARP does not support any authentication mechanism, an attacker

can send fake ARP responses containing its own MAC address, causing the target to send packets to the attacker instead of the host. However, during such an attack, the attacker generates unusual activity on the network which can be detected by an intrusion detection system.

Port scanning: As the terminology suggests, this type of attack consists in scanning all the ports in a network in order to discover which ports are open and whether they give access to vulnerable applications. While this is a precursor to an actual attack, this malicious behaviour can still be detected by an intrusion detection system.

The IDS framework presented in this chapter is designed to handle such connectivity related cyber-attacks with a focus on the Transmission Control Protocol/Internet Protocol (TCP/IP) network interface.

5.2.2 Preparation of the emulated controlled environment

As described earlier, an real environment for the SCR is designed in a laboratory room space as shown in the figure 5.2.

Figure 5.2: (left) Arena for the SCR to traverse for data collection and attack testing on the presented IDS (right) The SCR with on board Raspberry Pi and various sensors and network communication units

The SCR is an advanced setup with high resource capability and sensing. It is capable of locomotion in any directional and is programmable. It also has on-board network connectivity to connect with a remote server or other devices such as in a V2X infrastructure. The different features of the SCR is listed as follows:

- **Control and computation:** Raspberry Pi 3B board
- **OS:** Raspberry Pi OS
- **Locomotion:** Freenove 4WD Smart car Kit [94]
- **Sensing:** Internal CPU sensors of Raspberry Pi B and external sensors attached to the Raspberry Pi board through a sensor array

- **Locomotion control:** Freenove's controller software
- **Sensors:** Ultrasonic range finder (HC SR04), Camera, Light-Dependent Resistor (LDR), Infra-Red (IR) sensors for lines-following, digital Gyroscope, Accelerometer, Magnetometer, Barometer, Temperature and Humidity Sensors.
- **Network communication:** Wireless Local Area Network (WLAN), Bluetooth, Inter-Integrated Circuit (I2C) (for serial communication with most of the sensors)

5.3 Presented methodology and design idea

5.3.1 Sources of data

The main challenge on the edge is to aggregate all the sensor information from various channels and detect abnormalities or falsified perturbation and detect a difference (glitch) using ML, pertaining to the whole system. The presented IDS tries to classify a normal scenario (unperturbed case) and an anomalous scenario while the SCR is in motion. In order to ensure good training, the data is significant. Therefore, separate sessions were run to collect different types of data. As mentioned earlier, two anomaly detectors are embedded within the edge system (SCR) viz. sensor anomaly detector and network anomaly detector. The sensors data recorded is from a variety of sensors as described in the section 5.2.2.

5.3.2 ML based IDS Structure

Both the network and host IDSs follow the training and inference flow as presented in the figure 5.3.

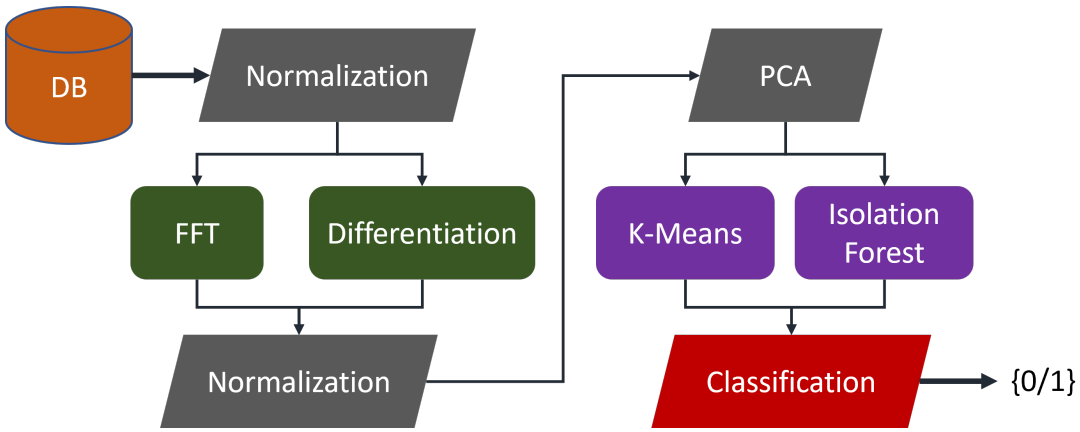

Figure 5.3: Presented ML based IDS training flow diagram

5.3.3 Network IDS (NIDS)

The NIDS is trained based on the network packet analysis. The network packet features include twenty-seven different features computed from the incoming traffic, some of which include Avg_syn_flag, Avg_ack_flag, Avg_DNS_pkt, Avg_TCP_pkt, Avg_UDP_pkt, Avg_ICMP_pkt, Avg_ARP_packet, Avg_pkts_length, Min_pkts_length, Max_pkts_length, etc.

5.3.4 Sensor or Host IDS (HIDS)

The HIDS connects with all the on-board sensors, interfacing using serial communication with the sensor array to collect all data. A total of thirty-one features are used for the monitoring and intrusion detection which include temperature, humidity, throttle status, cpu-voltage, cpu-temperature, gyroscope readings, obstacle distance, acceleration, and so on.

5.3.5 ML parameters

Two type of unsupervised ML algorithms have been introduced in the presented IDS after studying various different types of outlier and novelty detection algorithms. The best performing outlier detectors empirically chosen among the rest on the studied data are K-Means clustering and Isolation Forest (IF). The reason for using unsupervised learning approach despite having data from different sources since this approach generates more robust detection due the fact that they are ready for zero-day attacks since it is an extremely challenging task to anticipate all the different types of attacks that may be targeted on the system at present as well as in the future. The results presented in this chapter for the IDS are using the K-Means clustering method of unsupervised learning, where the ”K” represents the number of clusters which is a hyperparameter for the training that in this case is provided as an input to generate the IDSs.

An example set of hyperparameters for the generation of the NIDS or the HIDS using the nominal data collected is provided below:

- Standardize features from data source: Yes
- Transform data using: Fast Fourier Transform (FFT) with window size 5
- Components to chose from PCA: 5
- Choice of model: K-Means
- Number of clusters: 3

The presented IDS architecture is as shown in figure 5.4.

5.3.6 Network communication with the cloud

The API for Edge-to-Cloud communication is built around CoAP protocol. This communication protocol does not require any form of initialization or handshake since the messages are not encrypted and no identity is verified. The Cloud makes sure that the requested IP is alive by sending an Internet Control Message Protocol (ICMP) (ping) request to check the readiness status of the device, followed by other requests. Every message sent from the Cloud requires at least one acknowledgement from the Edge in the absence of which the Cloud will continuously keep sending the message until and acknowledgement is received.

For the presented framework, the Cloud will either send data (POST) to the Edge or ask for data (GET) from it, depending upon the choice from the user. A typical communication between a CoAP client and server is shown in the figure 5.5.

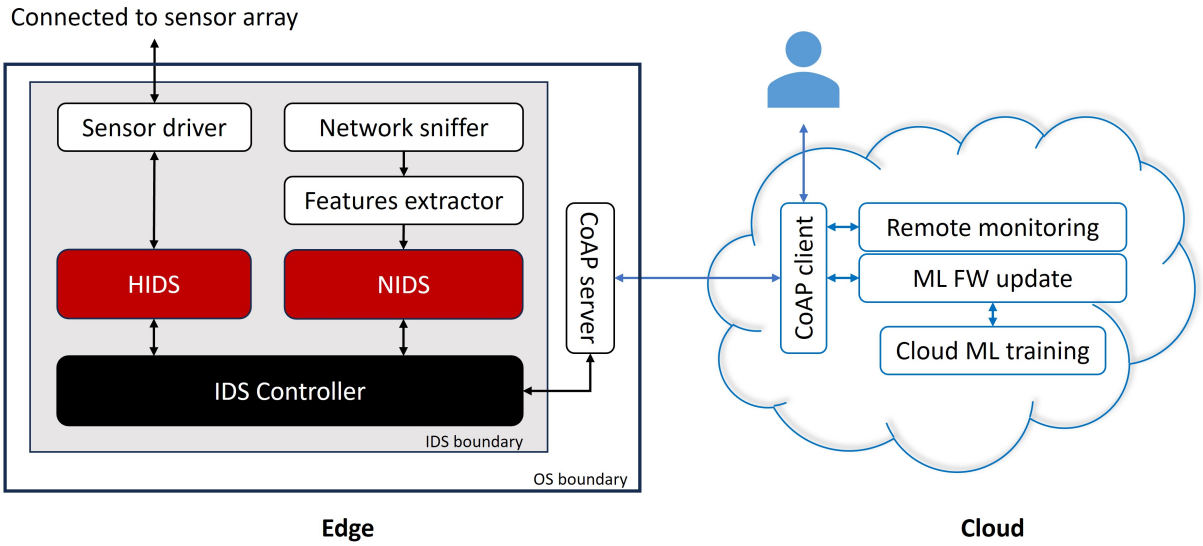

Figure 5.4: Architecture diagram of the presented edge IDS for V2X IoT

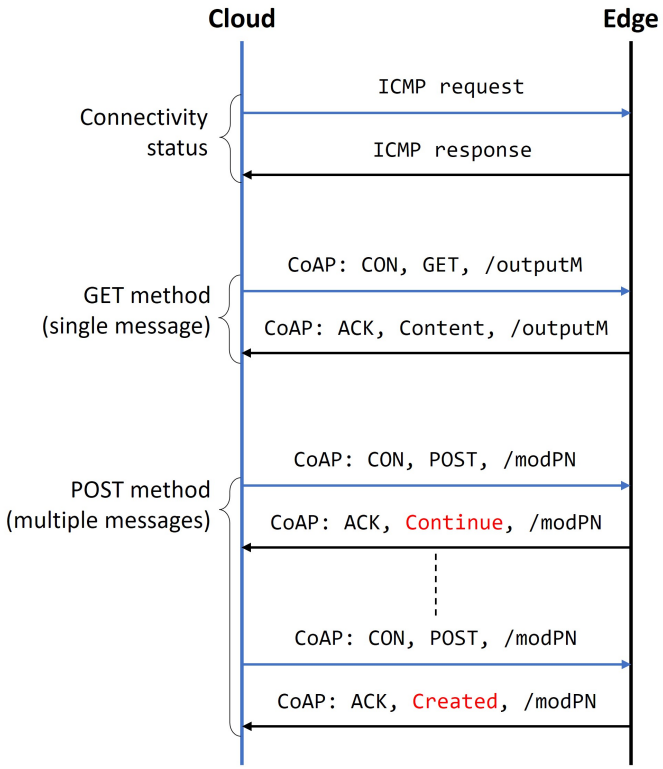

Figure 5.5: Edge-to-Cloud communication using CoAP

5.3.7 Results

As explained earlier, the ML algorithms used are unsupervised ML algorithms that do not use the attack datasets for identification of abnormal scenarios and rather the training is focused on the nominal or no-attack working condition of the system. The IDS then, during inference,

tries to detect any abnormal activity or intrusion as an outlier. The results of such detection is presented below.

5.3.7.1 NIDS detection results

The figure 5.6 and table 5.2 presents the detection results and confusion matrix for the NIDS behaviour over nominal (no-attack) data, respectively. This is expected to not detect any anomaly and the outlier detection algorithm performs as expected.

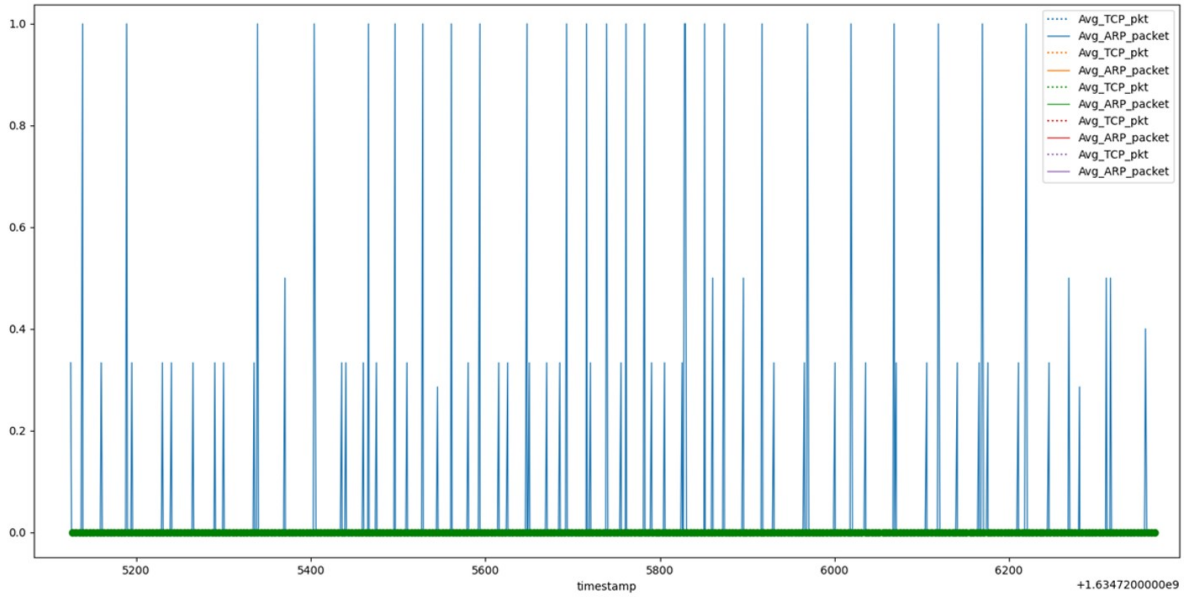

Figure 5.6: The detection behaviour of presented NIDS over nominal data (no attack scenario)

	Actual Positive	Actual Negative
Predicted Positive	0	0
Predicted Negative	0	100

Table 5.2: Confusion matrix for NIDS detection performance of nominal data

The figure 5.7 and table 5.3 presents the detection results and confusion matrix for the NIDS detection of Transmission Control Protocol (TCP) DoS attack, respectively.

	Actual Positive (%)	Actual Negative (%)
Predicted Positive	1.78 (TP)	0.92 (FP)
Predicted Negative	4.4 (FN)	92.91 (TN)

Table 5.3: Confusion matrix for NIDS detection performance of TCP DoS attack

The figure 5.8 and table 5.4 presents the detection results and confusion matrix for the NIDS detection of port scanning, respectively, which is not essentially an attack but it is the first step to identify the ports and their status to launch an attack.

There were no noticeable detection for the ARP spoofing attack.

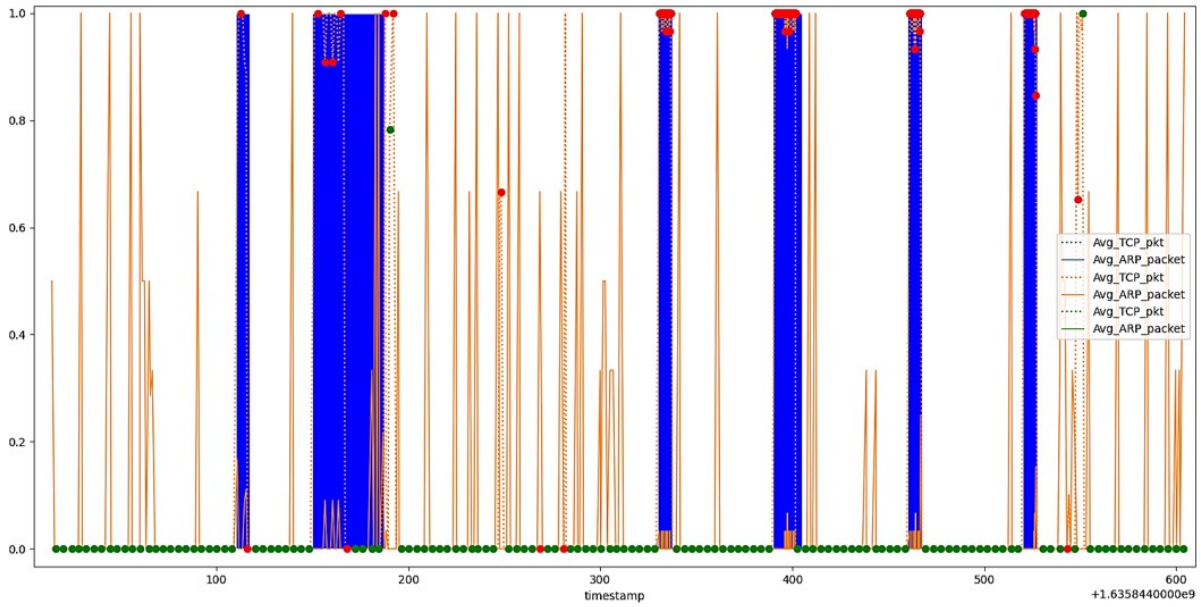

Figure 5.7: The detection behaviour of presented NIDS against TCP DoS attack

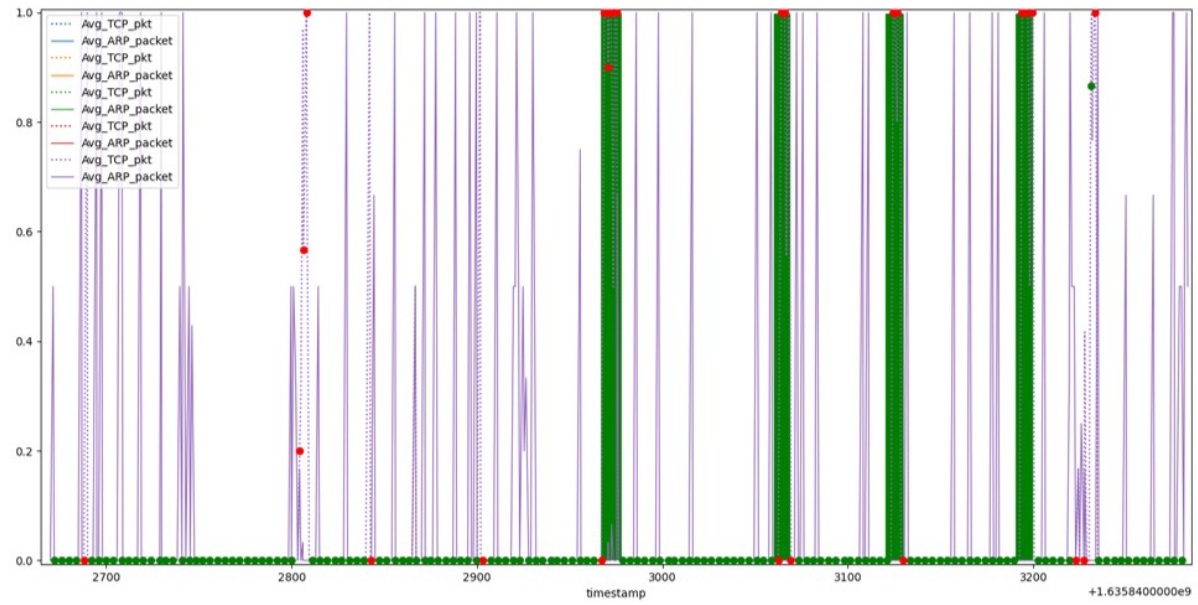

Figure 5.8: The detection behaviour of presented NIDS against Port Scanning using NMAP tool

	Actual Positive (%)	Actual Negative (%)
Predicted Positive	1.82 (TP)	1.95 (FP)
Predicted Negative	0.17 (FN)	96.06 (TN)

Table 5.4: Confusion matrix for NIDS detection performance of Port Scanning

5.3.7.2 HIDS detection results

Some of they physical attacks implemented were detected by the HIDS outlier detector based on K-Means clustering algorithm. The adversarial attacks to manipulate the sensors can make the smart car to go into unwanted states during operation. Some of those attacks that was tested on the implemented IDS in this chapter include:

- Jerking the SCR (emulating a forced collision)
- Lifting the SCR (emulating a jump off a cliff scenario)
- Increasing the humidity
- Ultra-Sonic sensor Spoofing (USS) attack (example real scenario includes manipulating autonomous smart car to make a maneuver by tampering the proximity sensors)

The figure 5.9 presents the detection capability of the HIDS. In the figure the labelled numbers represent:

1. A collision attack that was not detected since it was too minor to be picked up by the sensors
2. Another collision attack that was successfully detected
3. An event of lifting off the SCR and was successfully detected
4. Sudden large variation in luminosity (achieved by turning off the light in the controlled operational environment area).
5. A collision attack that was detected a bit late probably because of the size of the FFT window (latency due to computational complexity)

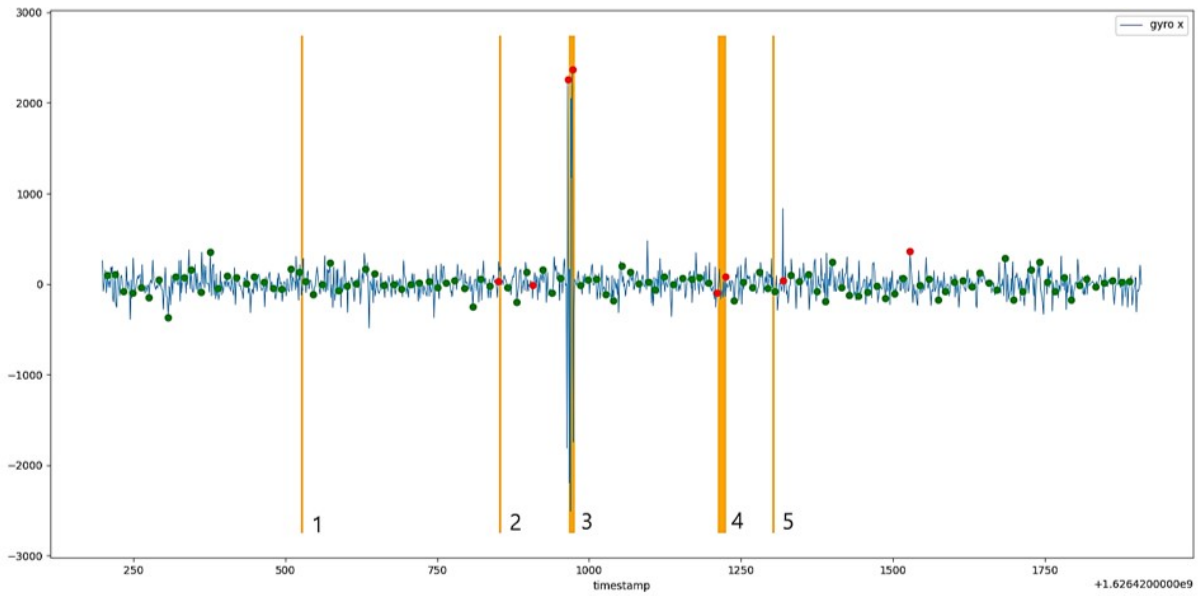

Figure 5.9: Motion based physical attacks detection by presented HIDS

Similarly, the humidity change was also detected. However, the ultra-sonic sensor spoofing attack was not detected successfully by the HIDS.

The table 5.5 presents a summary of the attacks that were implemented to test the efficacy of the presented IDS.

Attacks	Lifting off	Collision	Humidity	USS	TCP DoS	Port scan	ARP poison
Detected	Yes	Yes	Yes	No	Yes	Yes	Yes (poor)
Sensor / **Network**	pressure thermal humidity light accelerometer gyroscope magnetometer ultrasonic	pressure temperature humidity light accelerometers gyroscope magnetometers ultrasonic	humidity	Ultrasonic	Network	Network	Network

Table 5.5: A summary of the adversarial attacks implemented on the presented IDS with their detect-ability

5.4 Discussion

It is to note that the presented methodology is based on outlier ML algorithm which is an unsupervised learning method and the number of features to learn from in each case of HIDS and NIDS is over 25. However, not all the features are involved in different types of attacks. In order to improve the detection of multi-source intrusions or faults, it is necessary to perform feature extraction in order to elevate the quality of the ML detection, especially in the case of unsupervised learning where the data is not labelled. This is done in this case. However, as we also notice, that in some instances it can cause a computation overhead leading to added latency in detection. Therefore, it is upon the discretion of the designer to ensure the trade-off between detection and latency is well managed to keep the performance within acceptable ranges.

Furthermore, the reason for presenting the framework in this chapter is to provide a full end-to-end IDS for an IoT based system that can be leveraged further to implement or improve in other similar applications especially in an Edge-to-Cloud context.

5.5 Conclusion

In this chapter, a complete framework for an intrusion detection system with remote monitoring and parameter upgrade from the cloud is presented that includes all the three basic entities of an edge-to-cloud system viz. an edge system, a standard protocol based connectivity to the cloud, and the cloud application. The solution was tested in a standard MCU running an OS that hosts the IDS application which contains various modules such as a sensor data aggregator, a network sniffer, a NIDS, a HIDS, and a communication unit for the connectivity to the server using CoAP protocol. The solution can be deployed in any embedded system in parts (just the edge) or completely (with cloud monitoring and update) and thereby improving the security of the embedded system against intrusion attacks.

This chapter concludes the contributions part of this thesis manuscript along with the contributions presented in the chapters 3 and 4. The final concluding remarks are presented in the next chapter with some perspectives for future works along with a list of the research publications and patents that were published, accepted or submitted during this thesis period.

Chapter 6

Conclusion & Perspectives

Contents

6.1 Conclusion

In this thesis manuscript actual cases where ML do assist security of embedded systems have been presented in many different variants. The motivation behind this thesis is to present the different cybersecurity challenges that are faced in the chip industry in production of embedded system products starting from the design and development up to the in-field deployment of the products where they are exposed to the public infrastructure with the highest degree of possibility of opportunistic adversarial threats and attacks. It is shown how at many levels of production, testing and commissioning of embedded and integrated systems it is possible to tamper with the design and process to create security loopholes that could be exploited at later stages. Additionally, how the deployed secured solutions are still vulnerable to adversarial attacks in a real setting through various physical means and side channels. The connected device infrastructure generates large volumes of data at a constant rate due to the various services offered at end node level and the data contains sensitive information that is stored, computed or even transmitted to other devices.

In order to efficiently handle the various different characterization of attacks and the modalities in which the attacks could occur, ML is shown to be the solution to detect and in many cases allow the liable actors in the supply chain to take proper actions to mitigate the threats through threat conscious testings. The goal is to minimize the overhead in design and process for easy and smooth integration into the existing systems and make them smarter in managing adversarial threats by themselves. The solutions presented not only improve the security of the design but also provide future prospects of improvement and redeployment of the security solutions to make the offers more robust.

Cybersecurity in hardware solutions such as cryptographic accelerators, integrated secure elements, IoT objects, edge devices, and security chips in today's era are one of the biggest challenges that have troubled chipmakers and embedded system manufacturers. Some of the biggest challenges that are ever-present in hardware systems are FIAs, presence of HTs and other forms of intrusions in the system such as from the network layer or the physical layers. To all these known threats, ML based approaches are presented in this thesis that are shown to be able to proactively and reactively detect those threats and allow security improvements in the design and process. This thesis work not only provides measures to detect such threats but also presents use-cases and process solutions that would allow easy integration to the embedded system designs as well as their development and evaluation processes.

In chapter 2 a comprehensive background on all such threats mentioned above from the literature is provided along with description of ML techniques and how they could be applied in resolving those security threats. The chapters 3, 4, and 5 present solutions with experimental validations to detect and handle those threats and overall show how the solutions help alleviate the embedded systems security in general.

The chapter 3 is dedicated to a class of threat known as Fault Injection Attacks (FIA) which are physical perturbations made to a chip using equipment such as EM probe for EMFI and a clock-signal modulation generator to cause CGFI. The effects of these attacks on the chip are driven by the motivation behind the attacks. Some attacks could be simply disruptive i.e., to cause DoS scenarios and lead the chip to malfunction by modifying signals and values stored in memory elements. Other attacks could be more precise in trying to perturb minor functions that would be leveraged to exploit other attacks performed in conjunction to them to exploit the assets within the chips. To detect such attacks, an on-chip detection mechanism is presented that runs a data aggregator and a ML classifier algorithm at its core. The aggregator collects data from many digital sensors spread across the entire chip and feeds them to the ML module which pre-processes the data and then analyzes it to identify if there is any presence of FIA. This could be an early detection and the alarm raised by this smart security monitoring unit could help the central system to take appropriate measures such as reaching a non-exploitable state within the system based on some error policies. The major concern in such security systems is the presence of false positives which in the presented methodology is shown to be non-existent with a high detection rate.

In the chapter 4 the terror of Hardware Trojans (HT) is dealt. As described earlier, a HT is a threat that is almost unavoidable based on the current competitive trend of the semiconductor industry where chipmakers and manufacturers outsource several aspects in the development and manufacturing processes to third-party entities which are often the source on insertion of the Trojan circuits in the design. On the other hand, intentional mistakes in the design is also a threat that still exists in the so called "secure" locations where the design is initiated. To handle such threats, two different methods of evaluating the design at the post-silicon stage of the chip is presented that leverages ML based techniques. The methods rely on data recorded of the chip activity, without activating the Trojan circuit, through EM cartography, in a black-box setting. The through supervised ML techniques classification is performed on the raw traces with some data modelling to classify the difference between the genuine and infected design. The second method is presented in case there is no availability of a second design i.e., only one chip lot is available. In this case an unsupervised ML approach is taken where using T-Test statistical metric from the available design EM cartography data, the unsupervised outlier detector algorithm is trained. Then for any new chip lots, their cartography data is used to calculate the T-Test metrics and then inference it on the trained outlier detection ML algorithm

to see if it matches the previous design or not. This could be especially helpful to ensure if two different chip lots are similar or not which would mean either or both of them contain HTs. It is shown that the detection accuracy of the second method is very good with a minimum false positive or false negative rate.

Finally, in the chapter 5 another threat to embedded systems is addressed. The solutions presented in chapters 3 and 4 are at the baremetal level where the FIA detection is performed on-chip as a reactive approach to threats and the HT detection is performed as an off-line evaluation as a proactive approach being part of the security testing phase. In this chapter 5 the presented solution is for intrusion and anomaly detection targeted to IoT systems where the Edge or the end-nodes (such as IoT objects) have computational capability. Edge devices are prone to adversarial attacks due to availability of many attack surfaces both at the physical as well as network levels. To create a cybersecurity case, the scenario of an automotive cybersecurity (V2X) is chosen where a complete experimental setup including a robot smart car and the operational environment is created to mimic the real world threats. The data is collected from the setup for the ML training. An outlier detector is used in this case to allow for zero-day threat protection despite the availability of labelled data to provide a future proof solution. The outlier ML algorithms, precisely a clustering method, is used to learn the nominal operational conditions from over fifty features originating from both the on-board sensor array and network traffic. After the learning phase, multiple attacks are executed on the system that include both sensor based attacks as well as network based attacks to let the system identify them as outliers. A cloud server is also emulated that would connect to the Edge device using the CoAP IoT communication protocol to monitor the IDS activity as well as perform over the cloud training of new models based on collected data and, based on the choice of the user/administrator, could push back the updated parameters to the edge to improve the detection accuracy of the ML models. Thus, a complete end-to-end framework is presented that is easily integrateable in any OS based Edge device.

6.2 Perspectives

It is to be noted that, in all the presented solutions in this thesis manuscript, the evaluation is done in a very restrictive manner to uncover the real potential of the proposed solutions. These can be be further improved in many ways focusing on the virtues of ML, such as:

1. Increasing the number of digital sensors in the chip to allow more features to be trained from for the FIA detection problem
2. Relaxing the computational resource constraints and allowing to use more advanced non-linear ML algorithms in the detection of FIA attacks such as Long Short-Term Memory (LSTM) networks for the continuous sensor data for a better detection of fault injections
3. Increasing the size and diversity of the datasets to improve learning of the ML algorithms
4. Utilizing semi-supervised learning approaches instead of unsupervised learning methods to increase the detection accuracy of IDS systems

Additionally, the presented frameworks and methodologies can be further extended into the same as well as branched directions. Some of those are described in the following sections.

6.3 Timing based Hardware Trojan detection IP

A parallel work during this thesis duration had been conducted in the area of detecting hardware Trojans using circuit delay characterization which shows efficiency in detection some hardware Trojans and is, therefore, a promising area of exploration. Based on the this methodology, it could be further extended to create an on-chip Hardware Trojan detection IP.

6.3.1 Methodology and design idea

The methodology lies in using on-board detection module based on a Known-Answer-Test (KAT) like critical circuit path verification using a stored set of characterized circuit delay parameters derived from RTL simulation, with some provisions for process variations, and then an on-board controller module conditionally or periodically triggering circuit delay measurements of critical paths from pre-selected points A to B and comparing the measured delay with the stored delay for the path. This would allow for an identification of an activated Trojan circuit along the particular path if the path delay is greater than the stored value for the same. The classification of the two parameters is a precise problem as presented in this thesis manuscript for detecting faults using binary ML classification models. The proposed methodology is shown in the figure 6.1.

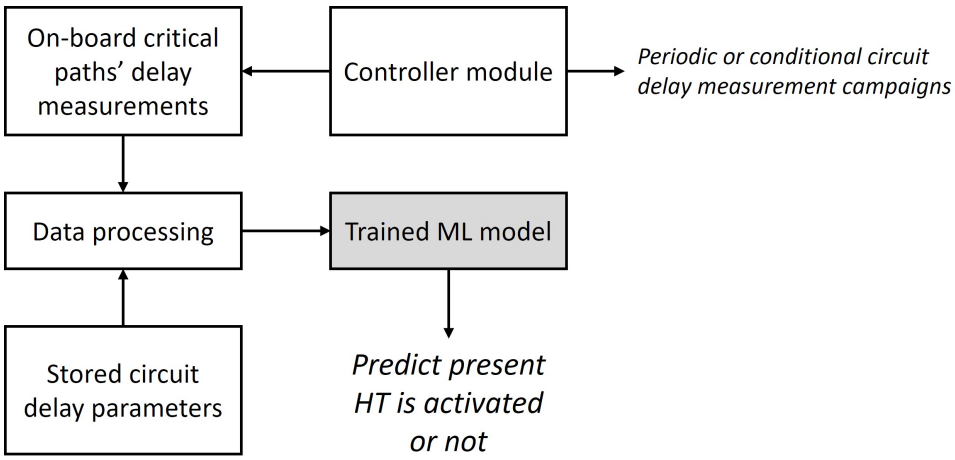

Figure 6.1: High level diagram of a circuit-delay characterization based ML HT detector IP

6.4 Novelty/Outlier detection for FIA

Another interesting area of study is the methodology of detection of fault injection attacks using one-class classification methods based on ML utilizing outlier detection or similar algorithms. The notion would be to eliminate the requirements of training the model with data from fault injection data and instead only train the nominal behavior of the chip to the ML model. Another approach would be to follow the methodology presented in the section 4.4.8 using T-Test metrics. However, the validity of such approach is unverified, and due to the unavailability of FIA data, it would be extremely challenging to achieve zero false positives as naturally unsupervised ML are weaker than supervised ML learning approaches because the example features of all pre-determined classes are known and learnt from beforehand.

6.5 List of Publications

6.5.1 Conference

An Embedded AI-Based Smart Intrusion Detection System for Edge-to-Cloud Systems *Shrivastwa, R. R., Bouakka, Z., Perianin, T., Dislaire, F., Gaudron, T., Souissi, Y., Karray, K., Guilley, S. In International Conference on Cryptography, Codes and Cyber Security (I4CS) 2022. DOI: 10.1007/978-3-031-23201-5_2*

Multi-source Fault Injection Detection Using Machine Learning and Sensor Fusion *Shrivastwa, R. R., Guilley, S., Danger, J. L. In International Conference on Security and Privacy (ICSP) 2021, DOI: 10.1007/978-3-030-90553-8_7*

High Precision EMFI Detector using Machine Learning and Sensor Fusion *Facon, A., Guilley, S., Ngo, X., Nguyen, R., Perianin, T., Shrivastwa, R. R. In: Cesar-Conference 2019. https://www.cesar-conference.org/editions-precedentes/slides-recordings-acts/*

Machine Learning Based Hardware Trojan Detection Using Electromagnetic Emanation *Takahashi, J., Okabe, K., Itoh, H., Ngo, X. T Guilley, S., Shrivastwa, R. R., Ahmed, M. Lejoly, P. In International Conference on Information and Communications Security (ICICS) 2020, DOI: 10.1007/978-3-030-61078-4_1* ***Best Paper award***

From substitution box to threshold Baksi, A., Guilley, S., Shrivastwa, R. R., Takarabt, S. In Indocrypt 2023 https://crsind.in/indocrypt2023/ ***Accepted***

6.5.2 Journal

Side-Channel Evaluation Methodology on Software *Guilley, S., Karray, K., Perianin, T., Shrivastwa, R. R., Souissi, Y., Takarabt, S. Cryptography, 2020, vol. 4, no 4, p. 27.*

Machine Learning Based Hardware Trojan Detection Using Electromagnetic Emanation *Takahashi, J., Okabe, K., Itoh, H., Ngo, X. T Guilley, S., Shrivastwa, R. R., Ahmed, M. Lejoly, P. In IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences 2021, DOI: 10.1587/TRANSFUN.2021CIP0011*

6.5.3 Patent

Hardware Trojan Detection Method
Shrivastwa, R. R., Guilley, S. – **submitted**

Bibliography

[1] Technical Committee: ISO/IEC JTC 1/SC 27 Information security, cybersecurity and privacy protection, "ISO/IEC DTR 5891.2 Information security, cybersecurity and privacy protection – Hardware monitoring technology for hardware security assessment."

[2] Common Criteria, "Common Criteria." https://www.commoncriteriaportal.org/index.cfm.

[3] Car-2-Car Communication Consortium, "Protection profile v2x hardware security module," 2021. https://www.commoncriteriaportal.org/files/ppfiles/pp0114b_pdf.pdf.

[4] "NIST SP 800-193 Platform Firmware Resiliency Guidelines," standard, National Institute of Standards and Technology, Gaithersburg, MD 20899, USA.

[5] "European Union Cyber Resilience Act," Regulation, National Institute of Standards and Technology, Gaithersburg, MD 20899, USA.

[6] "ISO/SAE 21434:2021 Road Vehicles – Cybersecurity Engineering," Standard, International Organization for Standardization, Geneva, CH, August 2021.

[7] The Mitre Corporation, "Common Vulnerabilities and Exposures (CVE)." https://cve.mitre.org/.

[8] The Mitre Corporation, "Common Weakness Enumeration (CWE)." https://cwe.mitre.org/data/index.html.

[9] Joint Interpretation Library, "Application of Attack Potential to Smartcards, Version 3.1," June 2020. https://www.sogis.eu/documents/cc/domains/sc/JIL-Application-of-Attack-Potential-to-Smartcards-v3-1.pdf.

[10] Bloomberg, "The Big Hack." https://www.bloomberg.com/news/features/2018-10-04/the-big-hack-how-china-used-a-tiny-chip-to-infiltrate-america-s-top-companies#xj4y7vzkg.

[11] A. Facon, S. Guilley, X.-T. Ngo, R. Nguyen, T. Perianin, and R.-R. Shrivastwa, "High Precision EMFI Detector using Machine Learning and Sensor Fusion," https://www.cesar-conference.org/wp-content/uploads/2019/11/20191120_J2_220_R-R-SHRIVASTWA_High_Precision_EMFI_Detector_using_Machine_Learning_and_Sensor_Fusion.pdf.

[12] R.-R. Shrivastwa, S. Guilley, and J.-L. Danger, "Multi-source fault injection detection using machine learning and sensor fusion," in *International Conference on Security and Privacy*, pp. 93–107, Springer, 2021. DOI: 10.1007/978-3-030-90553-8_7.

[13] J. Takahashi, K. Okabe, H. Itoh, X.-T. Ngo, S. Guilley, R.-R. Shrivastwa, M. Ahmed, and P. Lejoly, “Machine learning based hardware trojan detection using electromagnetic emanation,” in *International Conference on Information and Communications Security*, pp. 3–19, Springer, 2020.

[14] J. Takahashi, K. Okabe, H. Itoh, X.-T. Ngo, S. Guilley, R.-R. Shrivastwa, M. Ahmed, and P. Lejoly, “Machine learning based hardware trojan detection using electromagnetic emanation,” *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 105, no. 3, pp. 311–325, 2022.

[15] R.-R. Shrivastwa, Z. Bouakka, T. Perianin, F. Dislaire, T. Gaudron, Y. Souissi, K. Karray, and S. Guilley, “”an embedded ai-based smart intrusion detection system for edge-to-cloud systems”,” in *”International Conference on Cryptography, Codes and Cyber Security”*, pp. 20–39, Springer, 2022.

[16] P. W. Singer and A. Friedman, *”Cybersecurity: What everyone needs to know”*. Oxford University Press, USA, 2014.

[17] The Mitre Corporation, “2021 CWE Most Important Hardware Weaknesses,” October 2021. `https://cwe.mitre.org/scoring/lists/2021_CWE_MIHW.html`.

[18] E. Biham and A. Shamir, “Differential fault analysis of secret key cryptosystems,” in *Annual international cryptology conference*, pp. 513–525, Springer, 1997.

[19] G. Barthe, F. Dupressoir, P.-A. Fouque, B. Grégoire, and J.-C. Zapalowicz, “Synthesis of fault attacks on cryptographic implementations,” in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1016–1027, ACM, 2014.

[20] D. Boneh, R. A. DeMillo, and R. J. Lipton, “On the importance of checking cryptographic protocols for faults,” in *International conference on the theory and applications of cryptographic techniques*, pp. 37–51, Springer, 1997.

[21] A. Tang, S. Sethumadhavan, and S. Stolfo, “{CLKSCREW}: exposing the perils of security-oblivious energy management,” in *26th {USENIX} Security Symposium ({USENIX} Security 17)*, pp. 1057–1074, 2017.

[22] K. Murdock, D. Oswald, F. D. Garcia, J. Van Bulck, D. Gruss, and F. Piessens, “Plundervolt: Software-based Fault Injection Attacks against Intel SGX,” in *Proceedings of the 41st IEEE Symposium on Security and Privacy (S&P’20)*, 2020.

[23] P. Qiu, D. Wang, Y. Lyu, and G. Qu, “VoltJockey: Breaching TrustZone by Software-Controlled Voltage Manipulation over Multi-core Frequencies,” in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pp. 195–209, 2019.

[24] N. Moro, A. Dehbaoui, K. Heydemann, B. Robisson, and E. Encrenaz, “Electromagnetic fault injection: towards a fault model on a 32-bit microcontroller,” in *2013 Workshop on Fault Diagnosis and Tolerance in Cryptography*, pp. 77–88, IEEE, 2013.

[25] L. Riviere, Z. Najm, P. Rauzy, J.-L. Danger, J. Bringer, and L. Sauvage, “High precision fault injections on the instruction cache of armv7-m architectures,” in *2015 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pp. 62–67, IEEE, 2015.

[26] J. Proy, K. Heydemann, F. Majéric, A. Cohen, and A. Berzati, "Studying EM Pulse Effects on Superscalar Microarchitectures at ISA Level," *arXiv preprint arXiv:1903.02623*, 2019.

[27] A. Menu, S. Bhasin, J.-M. Dutertre, J.-B. Rigaud, and J.-L. Danger, "Precise spatio-temporal electromagnetic fault injections on data transfers," in *2019 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, pp. 1–8, IEEE, 2019.

[28] J.-M. Schmidt and M. Hutter, *Optical and EM Fault-Attacks on CRT-based RSA: Concrete Results*. Verlag der Technischen Universität Graz, 2007. Austrochip 2007 ; Conference date: 11-10-2007 Through 11-10-2007.

[29] A. Dehbaoui, J.-M. Dutertre, B. Robisson, and A. Tria, "Electromagnetic transient faults injection on a hardware and a software implementations of aes," in *2012 Workshop on Fault Diagnosis and Tolerance in Cryptography*, pp. 7–15, IEEE, 2012.

[30] H. Bar-El, H. Choukri, D. Naccache, M. Tunstall, and C. Whelan, "The sorcerer's apprentice guide to fault attacks," *Proceedings of the IEEE*, vol. 94, no. 2, pp. 370–382, 2006.

[31] S. W. Moore, R. J. Anderson, and M. G. Kuhn, "Improving smartcard security using self-timed circuit technology," in *Fourth ACiD-WG Workshop, Grenoble*, 2000.

[32] H. Liu, Z. Liu, Y. Qiao, Z. Lu, *et al.*, "Clock glitch fault injection attacks on an fpga aes implementation," *Journal of Electrotechnology, Electrical Engineering and Management*, vol. 1, no. 1, pp. 23–27, 2017.

[33] A. H. Johnston, "Charge generation and collection in pn junctions excited with pulsed infrared lasers," *IEEE Transactions on Nuclear Science*, vol. 40, no. 6, pp. 1694–1702, 1993.

[34] M. Ebrahimabadi, S. S. Mehjabin, R. Viera, S. Guilley, J.-L. Danger, J.-M. Dutertre, and N. Karimi, "Delfines: Detecting laser fault injection attacks via digital sensors," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 1–1, 2023.

[35] M. Ebrahimabadi, S. S. Mehjabin, R. Viera, S. Guilley, J.-L. Danger, J.-M. Dutertre, and N. Karimi, "Delfines: Detecting laser fault injection attacks via digital sensors," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2023.

[36] National Science Foundation, "Trust-HUB." `https://www.trust-hub.org/#/home`.

[37] R. Torrance and D. James, "The State-of-the-Art in IC Reverse Engineering," in *CHES*, vol. 5747 of *LNCS*, pp. 363–381, Springer, September 6-9 2009. Lausanne, Switzerland.

[38] F. Courbon, P. Loubet-Moundi, J. J. A. Fournier, and A. Tria, "A high efficiency hardware trojan detection technique based on fast SEM imaging," in *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition, DATE 2015, Grenoble, France, March 9-13, 2015* (W. Nebel and D. Atienza, eds.), pp. 788–793, ACM, 2015.

[39] M. Banga and M. Hsiao, "ODETTE : A Non-Scan Design-for-Test Methodology for Trojan Detection in ICs," in *International Workshop on Hardware-Oriented Security and Trust (HOST), IEEE*, pp. 18–23, 2011.

[40] S. Jha and S. K. Jha, "Randomization Based Probabilistic Approach to Detect Trojan Circuits," in *Proceedings of the 2008 11th IEEE High Assurance Systems Engineering Symposium*, HASE '08, (Washington, DC, USA), pp. 117–124, IEEE Computer Society, 2008.

[41] X. T. Ngo, J.-L. Danger, S. Guilley, Z. Najm, and O. Émery, "Hardware property checker for run-time hardware trojan detection," in *2015 European Conference on Circuit Theory and Design (ECCTD)*, pp. 1–4, Aug 2015.

[42] R. Rad, J. Plusquellic, and M. Tehranipoor, "Sensitivity analysis to hardware Trojans using power supply transient signals," in *Proceedings of the 2008 IEEE International Workshop on Hardware-Oriented Security and Trust*, HST '08, (Washington, DC, USA), pp. 3–7, IEEE Computer Society, 2008.

[43] M. Banga and M. Hsiao, "A Novel Sustained Vector Technique for the Detection of Hardware Trojans," in *International Conference on VLSI Design, IEEE*, pp. 327–332, 2009.

[44] O. Söll, T. Korak, M. Muehlberghuber, and M. Hutter, "EM-based detection of hardware trojans on FPGAs," in *2014 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, pp. 84–87, May 2014.

[45] J. He, Y. Zhao, X. Guo, and Y. Jin, "Hardware trojan detection through chip-free electromagnetic side-channel statistical analysis," vol. 25, pp. 2939–2948, Oct 2017.

[46] M. Lecomte, J. Fournier, and P. Maurine, "An on-chip technique to detect hardware trojans and assist counterfeit identification," vol. 25, pp. 3317–3330, Dec 2017.

[47] K. Worley and M. T. Rahman, "Supervised machine learning techniques for trojan detection with ring oscillator network," in *2019 SoutheastCon*, pp. 1–7, April 2019.

[48] B. Mukherjee, L. T. Heberlein, and K. N. Levitt, "Network intrusion detection," *IEEE network*, vol. 8, no. 3, pp. 26–41, 1994.

[49] M. Sabt, M. Achemlal, and A. Bouabdallah, "Trusted execution environment: What it is, and what it is not," in *2015 IEEE Trustcom/BigDataSE/ISPA*, vol. 1, pp. 57–64, 2015.

[50] A. Krasovsky and E. Maro, "Actual and historical state of side channel attacks theory," pp. 1–7, 09 2019.

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

[52] The OpenSSL Project, "OpenSSL: The open source toolkit for SSL/TLS." `www.openssl.org`, April 2003.

[53] OWASP, "OWASP Top 10." `https://owasp.org/www-project-top-ten/`.

[54] D. Hardt, "The OAuth 2.0 Authorization Framework." RFC 6749, Oct. 2012.

[55] FIDO Alliance, "FIDO2." `https://fidoalliance.org/fido2/`.

[56] T. M. Mitchell, *Machine learning*. McGraw Hill, 1997.

[57] E. F. Moore *et al.*, "Gedanken-experiments on sequential machines," *Automata studies*, vol. 34, pp. 129–153, 1956.

[58] R. J. Solomonoff, "An inductive inference machine," in *IRE Convention Record, Section on Information Theory*, vol. 2, pp. 56–62, Institute of Radio Engineers New York, 1957.

[59] H. Hamann and S. Lu, "Modeling complex physical systems with big data and machine-learning," *Bulletin of the American Physical Society*, vol. 65, 2020.

[60] Y. Yang, K. E. Niehaus, T. M. Walker, Z. Iqbal, A. S. Walker, D. J. Wilson, T. E. Peto, D. W. Crook, E. G. Smith, T. Zhu, *et al.*, "Machine learning for classifying tuberculosis drug-resistance from dna sequencing data," *Bioinformatics*, vol. 34, no. 10, pp. 1666–1671, 2017.

[61] Q. Liu, H. Zhu, C. Liu, D. Jean, S.-M. Huang, M. K. ElZarrad, G. Blumenthal, and Y. Wang, "Application of machine learning in drug development and regulation: Current status and future potential," *Clinical Pharmacology & Therapeutics*.

[62] T. Park, M.-Y. Liu, T.-C. Wang, and J.-Y. Zhu, "Gaugan: semantic image synthesis with spatially adaptive normalization," in *ACM SIGGRAPH 2019 Real-Time Live!*, pp. 1–1, 2019.

[63] T. Ç. Köylü, C. R. Wedig Reinbrecht, A. Gebregiorgis, S. Hamdioui, and M. Taouil, "A survey on machine learning in hardware security," *ACM Journal on Emerging Technologies in Computing Systems*, vol. 19, no. 2, pp. 1–37, 2023.

[64] H. Maghrebi, T. Portigliatti, and E. Prouff, "Breaking cryptographic implementations using deep learning techniques," in *Security, Privacy, and Applied Cryptography Engineering: 6th International Conference, SPACE 2016, Hyderabad, India, December 14-18, 2016, Proceedings 6*, pp. 3–26, Springer, 2016.

[65] E. Cagli, C. Dumas, and E. Prouff, "Convolutional neural networks with data augmentation against jitter-based countermeasures: Profiling attacks without pre-processing," in *Cryptographic Hardware and Embedded Systems–CHES 2017: 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*, pp. 45–68, Springer, 2017.

[66] S. Picek, I. P. Samiotis, A. Heuser, J. Kim, S. Bhasin, and A. Legay, "On the performance of deep learning for side-channel analysis.," *IACR Cryptol. ePrint Arch.*, vol. 2018, p. 4, 2018.

[67] S. Bhasin, A. Chattopadhyay, A. Heuser, D. Jap, S. Picek, and R. Ranjan, "Mind the portability: A warriors guide through realistic profiled side-channel analysis," in *NDSS 2020-Network and Distributed System Security Symposium*, pp. 1–14, 2020.

[68] M. Muehlberghuber, F. K. Gürkaynak, T. Korak, P. Dunst, and M. Hutter, "Red team vs. blue team hardware trojan analysis: detection of a hardware trojan on an actual asic," in *Proceedings of the 2nd International Workshop on Hardware and Architectural Support for Security and Privacy*, pp. 1–8, 2013.

[69] Y. Liu, K. Huang, and Y. Makris, "Hardware trojan detection through golden chip-free statistical side-channel fingerprinting," in *Proceedings of the 51st Annual Design Automation Conference*, pp. 1–6, 2014.

[70] R. Yasaei, S.-Y. Yu, and M. A. Al Faruque, "Gnn4tj: Graph neural networks for hardware trojan detection at register transfer level," in *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1504–1509, IEEE, 2021.

[71] E. Khalastchi, M. Kalech, and L. Rokach, "A hybrid approach for improving unsupervised fault detection for robotic systems," *Expert Systems with Applications*, vol. 81, pp. 372–383, 2017.

[72] H. Wang, S. Salehi, H. Sayadi, A. Sasan, T. Mohsenin, P. S. Manoj, S. Rafatirad, and H. Homayoun, "Evaluation of machine learning-based detection against side-channel attacks on autonomous vehicle," in *2021 IEEE 3rd International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, pp. 1–4, IEEE, 2021.

[73] G. Hospodar, B. Gierlichs, E. De Mulder, I. Verbauwhede, and J. Vandewalle, "Machine learning in side-channel analysis: a first study," *Journal of Cryptographic Engineering*, vol. 1, no. 4, p. 293, 2011.

[74] S. Saha, D. Jap, S. Patranabis, D. Mukhopadhyay, S. Bhasin, and P. Dasgupta, "Automatic characterization of exploitable faults: a machine learning approach," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 4, pp. 954–968, 2018.

[75] J. Breier, X. Hou, and S. Bhasin, *Automated Methods in Cryptographic Fault Analysis*. Springer, 2019.

[76] A. A. Ding, C. Chen, and T. Eisenbarth, "Simpler, faster, and more robust t-test based leakage detection," in *Constructive Side-Channel Analysis and Secure Design - 7th International Workshop, COSADE 2016, Graz, Austria, April 14-15, 2016, Revised Selected Papers* (F. Standaert and E. Oswald, eds.), vol. 9689 of *Lecture Notes in Computer Science*, pp. 163–183, Springer, 2016.

[77] M. R. Shahid, G. Blanc, Z. Zhang, and H. Debar, "Anomalous communications detection in iot networks using sparse autoencoders," in *2019 IEEE 18th International Symposium on Network Computing and Applications (NCA)*, pp. 1–5, IEEE, 2019.

[78] European Commission, "A European approach to artificial intelligence." `https://digital-strategy.ec.europa.eu/en/policies/european-approach-artificial-intelligence`.

[79] M. Brundage, S. Avin, J. Wang, H. Belfield, G. Krueger, G. Hadfield, H. Khlaaf, J. Yang, H. Toner, R. Fong, *et al.*, "Toward trustworthy AI development: mechanisms for supporting verifiable claims," *arXiv preprint arXiv:2004.07213*, 2020.

[80] S. Thiebes, S. Lins, and A. Sunyaev, "Trustworthy artificial intelligence," *Electronic Markets*, vol. 31, pp. 447–464, 2021.

[81] B. Li, P. Qi, B. Liu, S. Di, J. Liu, J. Pei, J. Yi, and B. Zhou, "Trustworthy AI: From principles to practices," *ACM Computing Surveys*, vol. 55, no. 9, pp. 1–46, 2023.

[82] N. Selmane, S. Bhasin, S. Guilley, and J.-L. Danger, "Security evaluation of application-specific integrated circuits and field programmable gate arrays against setup time violation attacks," *IET Information Security*, vol. 5, pp. 181–190, December 2011. DOI: 10.1049/iet-ifs.2010.0238.

[83] Xilinx, "Xilinx vivado hls 2019.2 support documentation." `https://www.xilinx.com/support/documentation-navigation/design-hubs/2019-2/dh0012-vivado-high-level-synthesis-hub.html`.

[84] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf, "Support vector machines," *IEEE Intelligent Systems and their Applications*, vol. 13, pp. 18–28, July 1998.

[85] A. Bounsiar and M. G. Madden, "One-class support vector machines revisited," pp. 1–4, May 2014.

[86] F. T. Liu, K. M. Ting, and Z. Zhou, "Isolation forest," pp. 413–422, Dec 2008.

[87] A. L. M. Chiu and Ada Wai-chee Fu, "Enhancements on local outlier detection," pp. 298–307, July 2003.

[88] Clifford Wolf, "PicoRV32 - A Size-Optimized RISC-V CPU." `https://github.com/cliffordwolf/picorv32` Accessed on March 8, 2021.

[89] Sifive, "Freedom." `https://github.com/sifive/freedom` Accessed on March 8, 2021.

[90] J. Balasch, B. Gierlichs, and I. Verbauwhede, "Electromagnetic circuit fingerprints for Hardware Trojan detection," pp. 246–251, Aug 2015.

[91] Y. Liu, Y. Jin, A. Nosratinia, and Y. Makris, "Silicon demonstration of hardware trojan design and detection in wireless cryptographic ics," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 4, pp. 1506–1519, 2017.

[92] P. Dusart, G. Letourneux, and O. Vivolo, "Differential fault analysis on aes," in *International Conference on Applied Cryptography and Network Security*, pp. 293–306, Springer, 2003.

[93] P. Qiu, D. Wang, Y. Lyu, and G. Qu, "Voltjockey: Breaking sgx by software-controlled voltage-induced hardware faults," in *2019 Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*, pp. 1–6, 2019.

[94] Freenove, "Freenove 4WD smart car kit for Raspberry Pi." `https://github.com/Freenove/Freenove_4WD_Smart_Car_Kit_for_Raspberry_Pi`.

ECOLE
DOCTORALE

Titre: Améliorations de la sécurité des systèmes embarqués avec Apprentissage automatique

Mots clés: Internet des objets (IdO); Intelligence Artificielle (IA); Apprentissage Automatique (AA); Détection des menaces;Cyberprotection;Processus de prise de décision;Sécurité intégrée;Attaques cyber-physiques;Synthèse de haut niveau.

Résumé: La liste des appareils connectés (ou IoT) s'allonge avec le temps, de même que leur vulnérabilité face aux attaques ciblées provenant du réseau ou de l'accès physique, communément appelées attaques Cyber Physique (CPS). Alors que les capteurs visant à détecter les attaques, et les techniques d'obscurcissement existent pour contrecarrer et améliorer la sécurité, il est possible de contourner ces contre-mesures avec des équipements et des méthodologies d'attaque sophistiqués, comme le montre la littérature récente. De plus, la conception des systèmes intégrés est soumise aux contraintes de complexité et évolutivité, ce qui rend difficile l'adjonction d'un mécanisme de détection complexe contre les attaques CPS. Une solution pour améliorer la sécurité est d'utiliser l'Intelligence Artificielle (IA) (au niveau logiviel et matériel) pour surveiller le comportement des données en interne à partir de divers capteurs. L'approche IA permettrait d'analyser le comportement général du système à l'aide des capteurs , afin de détecter toute activité aberrante, et de proposer une réaction appropriée en cas d'attaque. L'intelligence artificielle dans le domaine de la sécurité matérielle n'est pas encore très utilisée en raison du comportement probabiliste. Ce travail vise à établir une preuve de concept visant à montrer l'efficacité de l'IA en matière de sécurité. Une partie de l'étude consiste à comparer et choisir différentes techniques d'apprentissage automatique (Machine Learning ML) et leurs cas d'utilisation dans la sécurité matérielle. Plusieurs études de cas seront considérées pour analyser finement l'ibtérêt de l' IA sur les systèmes intégrés. Les applications seront notamment l'utilisation des PUF (Physically Unclonable Function), la fusion de capteurs, les attaques par canal caché (SCA), la détection de chevaux de Troie, l'intégrité du flux de contrôle, etc.

Title: Enhancements in Embedded Systems Security using Machine Learning

Keywords: Internet of Things (IoT); Artificial Intelligence (AI); Machine Learning (ML); Threat Detection; Cyber-Protection; Decision Making Process; Embedded Security; Cyber-Physical Attacks; High-Level Synthesis (HLS).

Abstract: The list of connected devices (or IoT) is growing longer with time and so is the intense vulnerability to security of the devices against targeted attacks originating from network or physical penetration, popularly known as Cyber Physical Security (CPS) attacks. While security sensors and obfuscation techniques exist to counteract and enhance security, it is possible to fool these classical security countermeasures with sophisticated attack equipment and methodologies as shown in recent literature. Additionally, end node embedded systems design is bound by area and is required to be scalable, thus, making it difficult to adjoin complex sensing mechanism against cyberphysical attacks. The solution may lie in Artificial Intelligence (AI) security core (soft or hard) to monitor data behaviour internally from various components. Additionally the AI core can monitor the overall device behaviour, including attached sensors, to detect any outlier activity and provide a smart sensing approach to attacks. AI in hardware security domain is still not widely acceptable due to the probabilistic behaviour of the advanced deep learning techniques, there have been works showing practical implementations for the same. This work is targeted to establish a proof of concept and build trust of AI in security by detailed analysis of different Machine Learning (ML) techniques and their use cases in hardware security followed by a series of case studies to provide practical framework and guidelines to use AI in various embedded security fronts. Applications can be in PUF predictability assessment, sensor fusion, Side Channel Attacks (SCA), Hardware Trojan detection, Control flow integrity, Adversarial AI, etc.