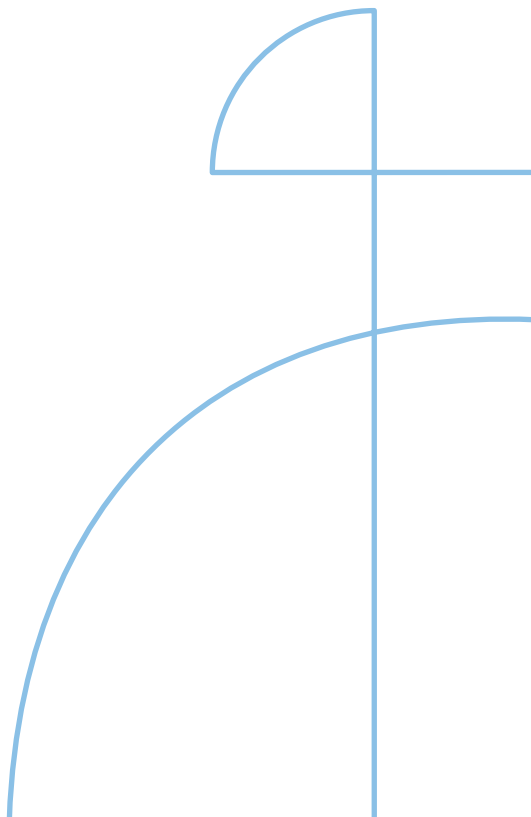
Doctoral Thesis in Information and Communication Technology

Protecting Remote FPGAs and Embedded Devices from Non-Invasive Physical Attacks

CAN AKNESIL

KTH ROYAL INSTITUTE OF TECHNOLOGY



Protecting Remote FPGAs and Embedded Devices from Non-Invasive Physical Attacks

CAN AKNESIL

Academic Dissertation which, with due permission of the KTH Royal Institute of Technology, is submitted for public defence for the Degree of Doctor of Philosophy on Wednesday the 11th of June 2025, at 9:00 a.m., in Ka-Sal C, Kistagången 16, 164 40 Kista, Sweden.

Doctoral Thesis in Information and Communication Technology
KTH Royal Institute of Technology
Stockholm, Sweden 2025

© Can Aknesil

TRITA-EECS-AVL-2025:59
ISBN 978-91-8106-294-6

Printed by: Universitetsservice US-AB, Sweden 2025

Abstract

The remote computing market has been growing rapidly for more than ten years, and this growth is expected to continue in the future. A considerable portion of this market is held by data centers, which today offer a diverse range of acceleration technologies for cloud computing, from massive multiprocessing to field-programmable gate arrays (FPGAs). Another significant portion is held by embedded devices in specialized mechanical and electronic systems from motor vehicles to home security systems. Valuable intellectual properties and sensitive information are being deployed in the cloud and embedded devices, which require strong protection. Besides the benefits, the industry's shift toward remote computing comes with increased vulnerability to physical attacks. FPGAs and embedded devices are among the electronic devices that are most vulnerable to physical attacks, because they may be deployed in a location physically accessible by potential adversaries. This thesis aims to protect FPGAs and embedded devices from physical attacks by exploring the boundaries of possible attack vectors and introducing new countermeasures.

This thesis contains six research papers. The first paper presents an FPGA implementation of a novel arbiter physically unclonable function (PUF) with 4×4 switch blocks. The PUF provides a more resource-efficient solution to secure key generation and storage on FPGAs. The second paper presents near-field electromagnetic deep learning-based side-channel analysis performed on Raspberry Pi 3, a widely-used single-board computer. The paper investigates the generalizability of side-channel analysis by focusing on the extraction of data in memory operations. The third and fourth papers present covert transmitting antennas and covert near-field EM sensors, respectively, both implemented entirely within the FPGA configurable fabric. The results highlight wireless covert channels as a plausible attack vector for cloud FPGAs and point to the need for further research on the topic. The fifth paper aims to improve IP security in FPGA clouds by introducing circuit disguise, a new method that enables FPGA design checks to be performed in the cloud without requiring the disclosure of the clients' unprotected designs. Last but not least, the sixth paper presents a hybrid method for fingerprinting neural networks by combining power side-channel measurements with information domain metrics.

Keywords FPGA security, embedded systems security, cloud security, physically unclonable functions, side-channel attacks, covert-channel attacks, neural network fingerprinting

Sammanfattning

Marknaden för fjärrberäkning har vuxit snabbt i över tio år, och denna tillväxt förväntas fortsätta i framtiden. En betydande del av marknaden innehas av datacenter, som idag erbjuder ett brett utbud av accelerationstekniker för molnberäkning, från massiv multiprocessering till påplats-programmerbar grindmatriser (FPGAs). En annan viktig del utgörs av inbyggda enheter i specialiserade mekaniska och elektroniska system, från motorfordon till hemlarmsystem. Värdefulla immateriella tillgångar (IP) och känslig information distribueras i molnet och i inbyggda enheter, vilket kräver starkt skydd. Förutom fördelarna medför industrins övergång till fjärrberäkning också en ökad sårbarhet för fysiska attacker. FPGAs och inbyggda enheter är bland de elektroniska enheter som är mest sårbara för fysiska attacker, eftersom de kan placeras på platser som är fysiskt tillgängliga för potentiella angripare. Denna avhandling syftar till att skydda FPGAs och inbyggda enheter mot fysiska attacker genom att undersöka gränserna för möjliga attackvektorer och introducera nya motåtgärder.

Avhandlingen innehåller sex forskningsartiklar. Den första artikeln presenterar en FPGA-implementering av en ny typ av arbiter-baserad fysiskt icke-klonbar funktion (PUF) med 4×4 switchblock. Denna PUF erbjuder en mer resurseffektiv lösning för säker nyckelgenerering och lagring på FPGA. Den andra artikeln presenterar närfälts elektromagnetisk djupinlärningsbaserad sidokanalsanalys utförd på Raspberry Pi 3, en allmänt använd enkortsdator. Artikeln undersöker generaliserbarheten av sidokanalsanalyser med fokus på extrahering av data i minnesoperationer. Den tredje och fjärde artikeln presenterar dolda sändarantennor respektive dolda närfälts EM-sensorer, båda helt implementerade inom FPGAs konfigurerbara struktur. Resultaten belyser trådlösa dolda kanaler som en trovärdig attackvektor för moln-FPGAs och pekar på behovet av vidare forskning inom området. Den femte artikeln syftar till att förbättra IP-säkerheten i moln-FPGAs genom att introducera ”circuit disguise”, en ny metod som möjliggör verifiering av FPGA-designer i molnet utan att klienternas oskyddade designer behöver avslöjas. Slutligen presenterar den sjätte artikeln en hybridmetod för fingeravtrycksbestämning av neurala nätverk genom att kombinera effektsidokanalsmätningar med informationsdomänmetrik.

Nyckelord FPGA-säkerhet, säkerhet i inbyggda system, molnsäkerhet, fysiskt icke-klonbara funktioner (PUF), sidokanalattacker, dolda kanalattacker, fingeravtrycksbestämning av neurala nätverk

Özet

Uzaktan hesaplama pazarı, on yılı aşkın süredir hızla büyümekte ve bu büyümenin önümüzdeki yıllarda da devam etmesi beklenmektedir. Günümüzde, pazarın önemli bir kısmını, çok çekirdekli işlemcilerden alanda programlanabilir kapı dizilerine (FPGA'lar) kadar çeşitli hızlandırma teknolojileri sunan veri merkezleri oluşturmaktadır. Diğer bir önemli kısmını ise, motorlu taşıtlardan ev güvenlik sistemlerine kadar özel amaçlı mekanik ve elektronik sistemlerde kullanılan gömülü cihazlar oluşturmaktadır. Bulut ortamında ve gömülü sistemlerde kullanılan değerli fikrî mülkiyetler ve hassas veriler, güçlü güvenlik önlemleri gerektirmektedir. Uzaktan hesaplamaya geçiş, sağladığı avantajların yanı sıra, fiziksel saldırılara karşı artan bir savunmasızlığı da beraberinde getirmiştir. FPGA'lar ve gömülü cihazlar, potansiyel saldırganların fiziksel erişimine açık ortamlara yerleştirilebildikleri için, fiziksel saldırılara karşı en savunmasız elektronik donanımlar arasında yer almaktadır. Bu tez, FPGA'lar ve gömülü cihazları fiziksel saldırılara karşı korumayı amaçlamakta; bu doğrultuda, olası saldırı vektörlerinin sınırlarını inceleyip yeni karşı önlemler geliştirmeyi hedeflemektedir.

Tez altı adet araştırma makalesini içermektedir. İlk makale, 4×4 anahtar bloklarıyla tasarlanmış, yeni bir tür arbiter tabanlı fiziksel klonlanamayan fonksiyonun (PUF) FPGA üzerindeki uygulamasını sunmaktadır. Bu PUF, FPGA üzerinde güvenli anahtar üretimi ve depolanması için daha verimli bir çözüm sunmaktadır. İkinci makale, yaygın olarak kullanılan tek kartlı bilgisayarlardan Raspberry Pi 3 üzerinde gerçekleştirilen, yakın alan elektromanyetik ölçümlere dayalı derin öğrenme tabanlı yan kanal incelemelerini ele almaktadır. Bu çalışma, özellikle bellek işlemlerinden veri çıkarımı konusuna odaklanarak, bu tür incelemelerin genellenebilirliğini araştırmaktadır. Üçüncü ve dördüncü makalelerde, tamamen FPGA'nın ayarlanabilir donanımı içerisinde gerçekleştirilen gizli iletim antenleri ve gizli yakın alan elektromanyetik sensörler tanıtılmaktadır. Elde edilen sonuçlar, kablosuz gizli kanalların bulut tabanlı FPGA'lar için gerçek bir saldırı vektörü oluşturabileceğini göstermekte ve bu alanda daha fazla araştırmaya ihtiyaç olduğunu ortaya koymaktadır. Beşinci makale, bulut FPGA'larında fikrî mülkiyet güvenliğini artırmak amacıyla, "devre kamuflajı" adını taşıyan yeni bir yöntem önermektedir. Bu yöntem, müşterilerin korumasız tasarımlarını ifşa etmeden FPGA tasarımlarının bulutta doğrulanabilmesini mümkün kılmaktadır. Son olarak, altıncı makale, sinir ağlarının parmak izi tespitine yönelik olarak güç yan kanal ölçümleriyle bilgi alanı metriklerini birleştiren hibrit bir yöntem sunmaktadır.

Anahtar Kelimeler FPGA güvenliği, gömülü sistemler güvenliği, bulut güvenliği, fiziksel klonlanamayan fonksiyonlar (PUF), yan kanal saldırıları, gizli kanal saldırıları, sinir ağı parmak izi tespiti

Contents

Abstract **i**

Acknowledgement **ix**

List of Papers **xi**

List of Figures **xiii**

Acronyms **xv**

1 Introduction **1**
1.1 Motivation 1
1.2 Problem Addressed 2
1.3 Research Methodology 3
1.4 Organization of the Thesis 4
1.5 Contributions 4

2 Background **7**
2.1 Hardware Security 7
2.2 Physical Attacks 8
2.3 FPGAs 9
2.4 Basics of Cryptography 12
2.5 Deep Learning 15

3 Secure Key Generation and Storage on FPGAs **17**
3.1 Cryptographic Keys on FPGAs 17
3.2 Existing Key Generation and Storage Methods 18
3.3 Physically Unclonable Functions (PUFs) 19
3.4 Usage of PUFs 20
3.5 PUF Security 22
3.6 4×4 Arbiter PUF 23
3.7 Summary 24

4 EM Side-Channel Attacks **25**
4.1 Growth of Side-Channel Attacks . . . 25
4.2 Profiled Deep Learning-Based Side-Channel Attacks . . . 26
4.3 Divide-and-Conquer Attack Strategy . . . 27
4.4 Leakage Models . . . 28
4.5 Noise and Averaging . . . 28
4.6 Trace Synchronization . . . 29
4.7 Leakage Assessment . . . 31
4.8 Countermeasures . . . 32
4.9 Memory Leakage on General-Purpose Computers . . . 32
4.10 Summary . . . 33

5 Attacks on Remote FPGAs **35**
5.1 FPGAs in Embedded Systems . . . 35
5.2 FPGA Acceleration in the Cloud . . . 36
5.3 Hardware Trojans . . . 36
5.4 Time-to-Digital Converters . . . 38
5.5 Transmitting Antenna in Configurable Fabric . . . 38
5.6 Receiving Antenna in Configurable Fabric . . . 39
5.7 FPGA Design Checks and IP Protection . . . 41
5.8 Circuit Disguise . . . 42
5.9 Summary . . . 42

6 Fingerprinting of Neural Networks **45**
6.1 IP in Machine Learning . . . 45
6.2 Adversarial Machine Learning . . . 46
6.3 Defenses against Model Extraction . . . 47
6.4 Hybrid Fingerprinting . . . 49
6.5 Summary . . . 50

7 Conclusions **51**
7.1 Recap . . . 51
7.2 Future Work . . . 52

References **55**

Included Papers **65**
Paper I . . . 67
Paper II . . . 75
Paper III . . . 83
Paper IV . . . 91
Paper V . . . 99
Paper VI . . . 109

Acknowledgement

First and foremost, I am grateful for my mother Ayşe Erdem Aknesil who was the greatest inspiration for me to pursue a path in academia. In addition to my mother, I am grateful to my father Güngör Aknesil, my brother Hakan Aknesil, and my grandparents Akın Erdem (the late), Sema Erdem, Ceyhun Aknesil, and Ayten Aknesil for their endless support throughout my studies.

I thank my primary supervisor Elena Dubrova for giving me the opportunity to be her doctoral student, providing me with an open research environment where I could pursue my own research ideas and shape my own academic path, and guiding me seamlessly from the beginning to the end of my degree. She greatly helped me to develop myself technically, academically, and pedagogically.

I also thank my secondary supervisor Zhonghai Lu, along with the other faculty members including Johnny Öberg, Ingo Sander, Ahmed Hemani, Carl-Mikael Zetterling, Ana Rusu, Panagiotis Papadimitratos, Pontus Johnson, and Mårten Björkman, for contributing to my education.

I thank Didem Unat, Barış Bozkurt, Thomas Magnusson, and Christian Schulte (the late), who believed in me and had an invaluable influence on me and my career.

I am deeply grateful to Thilanka Thilakasiri for her unwavering love, understanding, and encouragement throughout this journey.

I appreciate my colleagues and friends Huanyu Wang, Yang Yu, Michail Moraitis, Ruize Wang, Yanning Ji, Kalle Ngo, Martin Brisfors, Linus Backlund, Sönke Jendral, Fahimeh Bahrami, Rodolfo Jordão, Rui Chen, Shiqian Chen, Trishna Rajkumar, Jordi Altayó González, Ritika Ratnu, Dimitrios Stathis, Nooshin Nosrati, Yu Yang, Sofia Olsson, Maah Paara, Saba Yousefzadeh, Paul Delestrac, Jiawei Xu, Björn Lindqvist, Ivan Sraichuk, Shriraj Hegde, Vahid Geraeinejad, and Gracieth Cavalcanti Batista who were with me during my studies.

I thank our collaborators from Ericsson AB Niklas Lindskog, Jakob Sternby, and Håkan Englund for their contribution to our research.

I acknowledge KTH for enabling my doctoral studies. In addition to KTH, I thank the Sweden's Innovation Agency Vinnova, the Swedish Research Council, and Ericsson AB for their support in my research.

List of Papers

Paper I

An FPGA Implementation of 4×4 Arbiter PUF

Can Aknesil, Elena Dubrova

2021 IEEE 51st International Symposium on Multiple-Valued Logic (ISMVL), Nur-sultan, Kazakhstan, 2021, pp. 160-165.

Paper II

Towards Generic Power/EM Side-Channel Attacks: Memory Leakage on General-Purpose Computers

Can Aknesil, Elena Dubrova

2022 IFIP/IEEE 30th International Conference on Very Large Scale Integration (VLSI-SoC), Patras, Greece, 2022, pp. 1-6.

Paper III

Is Your FPGA Transmitting Secrets: Covert Antennas from Interconnect

Can Aknesil, Elena Dubrova, Niklas Lindskog, Håkan Englund

2023 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), St. Petersburg, FL, USA, 2023, pp. 79-84.

Paper IV

A Near-Field EM Sensor Implemented in FPGA Configurable Fabric

Can Aknesil, Elena Dubrova, Niklas Lindskog, Håkan Englund

2023 IEEE 22nd International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), Exeter, United Kingdom, 2023, pp. 1908-1913.

Paper V

Circuit Disguise: Detecting Malicious Circuits in Cloud FPGAs without IP Disclosure

Can Aknesil, Elena Dubrova

2024 27th Euromicro Conference on Digital System Design (DSD), Paris, France, 2024, pp. 361-368.

Circuit Disguise: Detecting Malicious Circuits in Cloud FPGAs without IP Disclosure (Extended version)

Can Aknesil, Elena Dubrova

Submitted to the Journal of Computers & Electrical Engineering.

Paper VI

Hybrid Fingerprinting for Effective Detection of Cloned Neural Networks

Can Aknesil, Elena Dubrova, Niklas Lindskog, Jakob Sternby and Håkan Englund

Accepted to 2025 IEEE 55st International Symposium on Multiple-Valued Logic (ISMVL).

Hybrid Fingerprinting for Effective Detection of Cloned Neural Networks (Extended version)

Can Aknesil, Elena Dubrova, Niklas Lindskog, Jakob Sternby and Håkan Englund

Cryptology ePrint Archive, Paper 2025/673, 2025.

List of Figures

2.1 Categorization of physical attacks in the field of hardware security. Attacks that are in the scope of this thesis are colored green. 8
2.2 A LUT with four inputs $in[0]$ to $in[3]$ and one output out, implemented using 2×1 multiplexers. The LUT is configured via setting the truth table entries tt. 10
2.3 Compilation of one-bit counter from RTL code to FPGA bitstream. . 11
2.4 Symmetric encryption and decryption of message M using shared key K. 13
2.5 Authentication of message M using symmetric encryption. 13
2.6 Public-key encryption of a message M using public-private key pair $(K_{\text{pub}}, K_{\text{priv}})$. 14
2.7 Diffie-Hellman key exchange algorithm. 14
2.8 A neural network with one input layer, one output layer, and two fully connected hidden layers. 15

3.1 Bitstream extraction from Flash memory using a passive probe. 17
3.2 Silicon chip at design stage (a) [1] and after manufacturing (b) [2]. . . 19
3.3 Ring oscillator consisting of an odd number of inverters. 20
3.4 Key exchange and authentication using a PUF. 22
3.5 An example RFID tag [3]. 23
3.6 High-level design of 4×4 arbiter PUF from Paper I (a), example switch block configurations (b). 24

4.1 The EM side-channel attack from Paper II. Target device with the EM probe on the left. A side-channel measurement on the right. . . . 25
4.2 A power trace from the execution of AES before averaging (top), averaged trace (middle), and Student's t-test result showing the leakage of all 16 bytes of the first S-box output (bottom). 29
4.3 Raspberry Pi 3 Model B front (left) and back (right). The front contains the processor chip. The back contains the memory chip. . . . 33

5.1 A replica of the Trojan horse [4]. Hardware Trojans, a.k.a. malicious modifications to an electronic circuit, take the name from the Trojan horse. 35
5.2 The cause and effect diagram from Paper V, which describes types of malicious circuits, the physical phenomena they result in, and the attacks they enable. 37
5.3 A tapped delay line-based TDC from Paper V. 38
5.4 Device view of an interconnect antenna on FPGA (left) and the near-field EM probe used to capture the transmitted signal (right). 39
5.5 The high-level schematic of the EM sensor presented in Paper IV. . . 40
5.6 Experimental setup in Paper IV. A function generator connected to a hand-made near-field loop antenna, whose signal is received from the FPGA that implements an EM sensor. 40
5.7 Configuration of a cloud FPGA using circuit disguise. D: compiled design netlist, D': disguised design, B: bitstream, S: digital signature, sk and pk: private and public keys of the trusted enclave, respectively. 41
5.8 An example design of a combinational cycle (left) and its disguised version (right). 42

6.1 Experimental setup for fingerprinting of neural networks using power side-channel information, from Paper VI. A microcontroller board that runs neural network inference (bottom left), a capture board for capturing power traces (top right), and a computer to drive the experiments and to analyze the collected information (top left). 45

Acronyms

ADC Analog-to-Digital Converter.

AES Advanced Encryption Standard.

API Application Programming Interface.

ASIC Application-Specific Integrated Circuit.

ATM Automated Teller Machine.

CPU Central Processing Unit.

DoS Denial of Service.

DSP Digital Signal Processing.

eFuse Electronic Fuse.

EM Electromagnetic.

FaaS FPGA as a Service.

FPGA Field-Programmable Gate Array.

GPU Graphics Processing Unit.

I/O Input/Output.

IC Integrated Circuit.

IoT Internet of Things.

IP Intellectual Property.

ISA Instruction Set Architecture.

LLM Large Language Model.

LUT Lookup Table.

NIST National Institute of Standards and Technology.

ORAN Open Radio Access Network.

OS Operating System.

PCB Printed Circuit Board.

PUF Physically Unclonable Function.

RAM Random-Access Memory.

RF Radio Frequency.

RFID Radio-Frequency Identification.

RSA Rivest-Shamir-Adleman.

RTL Register-Transfer Level.

SHA Secure Hash Algorithm.

SNR Signal-to-Noise Ratio.

SPI Serial Peripheral Interface.

SRAM Static Random-Access Memory.

SSH Secure Shell.

TDC Time-to-Digital Converter.

USB Universal Serial Bus.

1

Introduction

This chapter summarizes the motivation behind the thesis, the problem addressed, the research methodology, the organization of the thesis, and the contributions of the author.

1.1 Motivation

With the ongoing trends in computing, more and more computations are performed on third-party clouds, as well as on embedded systems that are physically accessible to the general public. Clouds, today, offer technologies ranging from massive multiprocessing to custom field programmable gate array (FPGA) acceleration, accordingly, large computations are being pushed from on-premise resources to the cloud [5]. From the other perspective, large data centers distribute their workload on edge devices that are located closer to the end users [6]. Valuable neural networks are being deployed on clouds, mobile phones, and embedded systems [7, 8]. Many internet of things (IoT) devices are deployed for various applications from home security [9] to healthcare [10].

The security of a computer, server, or more generally, an electronic device becomes highly challenging when the device is physically accessible by parties who do not own the computation executed on the device or the information stored and processed by it. A computation can be implemented as software, firmware, or hardware, and is typically considered an intellectual property (IP) due to the engineering effort and resources spent to develop it, thus requiring protection. Computer programs, neural networks, and FPGA bitstreams are examples of IP. Likewise, the information processed in a computation can be part of the IP or otherwise can be sensitive. To illustrate, in a computing cloud, the client's program runs on the premises of the cloud provider, consequently, the program and its data are vulnerable to attacks that may take place in the cloud environment. Similarly, a base station can be physically accessible to the general public; accordingly, its firmware and the information processed by the firmware, such as the communication data, are prone to physical attacks.

Securing an electronic device entails (1) the confidentiality of the information stored and processed on the device, (2) the integrity of the device's operation

and, accordingly, the processed information, and (3) the availability of the service provided by the device. Confidential information can be IP, cryptographic keys, or other sensitive information, such as medical and financial information.

Modern cryptography offers cryptographic algorithms, such as Advanced Encryption Standard (AES) and Rivest-Shamir-Adleman (RSA), that are still impractical to break via cryptanalysis powered by classical computers today, see [11] and [12], respectively. However, the protection offered by cryptographic algorithms relies on two important assumptions: (1) The device that runs the algorithm should not leak secret information through a side channel. (2) Key generation should be truly random. With the evolution of physical attacks on electronic devices, summarized in section 2.1, these assumptions are no longer negligible, accordingly, implementations of most widely used cryptographic algorithms that were considered secure, including AES and RSA, are no longer secure, as demonstrated in [13] and [14], respectively.

Computer security in general is a well-researched topic in electrical engineering and computer science. On the other hand, hardware security, a branch of computer security, is only recently becoming popular due to the increasing surface of vulnerabilities [15]. We, computer security researchers, have to fill this gap to remain one step forwarder than our adversaries.

1.2 Problem Addressed

This thesis specializes in the category of non-invasive physical attacks (see section 2.1), which include side-channel attacks, fault attacks, and covert-channel attacks. Non-invasive physical attacks do not require partial destruction of the target device and are usually not expensive. Although cold boot attacks are categorized under non-invasive physical attacks, it is out of the scope of this thesis.

FPGAs and embedded devices are among the electronic devices that are most vulnerable to physical attacks. Notice that FPGAs and embedded devices are not disjoint concepts. An embedded system may contain one or more FPGAs. FPGAs can also be present in the cloud, which is not considered an embedded system. The term *remote*, used throughout this thesis, including the title, describes remotely operated electronic devices that are deployed at a location that is physically accessible by potential adversaries. A remote FPGA or a remote embedded device may be deployed in a public area, thus exposed to physical attacks. A remote FPGA, additionally, may be deployed in the cloud, be remotely operated by the clients, and be exposed to physical attacks that may occur in the cloud.

An FPGA is a configurable integrated circuit (IC) that enables the implementation of custom digital circuits after its manufacturing. One usage of FPGAs is accelerating cloud computing. FPGAs are increasingly deployed in computing clouds as an alternative to graphics processing units (GPUs).

Embedded devices are a broad category of devices that typically are a part of a specialized mechanical or electronic system, and implement some of the system's functionality. Most IoT devices can also be included in this category. *Embedded systems* is a more widely used related term that typically refers to the system as a whole. An embedded system may be very large and may integrate hundreds of devices, such as in a car. Such large-scale systems come with an additional set of security challenges that are not addressed in this thesis. The focus of this thesis is instead on the protection of single devices, thus the term *embedded device* was chosen to be used throughout this thesis, including the title. Some examples of embedded systems are cars, planes, traffic lights, pacemakers, wireless modems, and smart card readers.

Furthermore, a growing concern that requires immediate attention is the protection of neural networks that are deployed on embedded devices and in the cloud [8, 7]. Previous research showed that such neural networks can be stolen via various model extraction attacks, as explained in section 6.2.

The ultimate goal of security research is to provide countermeasures to all possible attacks. A crucial part of developing countermeasures is understanding the capabilities and limitations of an adversary. A countermeasure is unlikely to be effective if there is insufficient research on the possible attacks. Therefore, an essential part of security research is the exploration of possible attack vectors and the testing of new countermeasures via exhaustive attacks.

There is a big knowledge gap in understanding physical attacks and how they can be mitigated. This thesis introduces new countermeasures to these attacks and, in the meanwhile, pushes the boundaries of the possible attacks.

In a nutshell, this thesis tries to answer the following research questions:

Research Question 1 What are the capabilities and limitations of an adversary who launches a non-invasive physical attack on a remote FPGA or an embedded device?

Research Question 2 How can remote FPGAs and embedded devices be protected from non-invasive physical attacks?

1.3 Research Methodology

The state of security research continuously evolves with fast iterations of new attacks and countermeasures. The fact that this is a compilation thesis reflects the dynamic nature of the field. A monograph thesis, which would be published only at the end of several years of study, may have been too slow for the pace of security research. The work presented in this thesis takes its place in the race between attacks and countermeasures, and, accordingly, will be passed behind by future work.

The topics of the papers included in this thesis are selected according to a combination of factors including the level of urgency for the topic's progress, the research gap, the projected impact, the topic's alignment with the research questions and the totality of the thesis, the author's expertise, and also the course of creativity and inspiration. Besides the fact that the included papers reside under the umbrella of the research questions of this thesis, each paper had a separate, although related, research agenda. After all, each paper is self-contained, and published as a stand-alone study. Hopefully, this kappa helps the reader understand how the included papers relate to each other, and to see the big picture and the current state of the hardware security research.

1.4 Organization of the Thesis

This is a compilation thesis. It consists of six research papers that are included in the second half. The first half of the thesis, the so-called *kappa*, provides a summary of the entire area of the thesis, connects the included papers, and addresses material that is not covered in the included papers. The kappa offers a gentle introduction for an expert audience who seeks an overview of the thesis' material, as well as for a general audience who seeks to get familiar with the thesis' topic.

The kappa is organized as follows. Chapter 1 explains the motivation behind this thesis, the problem it addresses, the research methodology, the organization of the thesis, and the contributions of the author. Chapter 2 provides background on hardware security, physical attacks, FPGAs, cryptography, and deep learning. Chapter 3 discusses secure key generation and storage on FPGAs and presents physically unclonable functions (PUFs). Chapter 4 presents electromagnetic (EM) side-channel attacks. Chapter 5 describes hardware Trojans and IP protection on remote FPGAs. Chapter 6 summarizes attacks and defenses in adversarial machine learning and describes neural network fingerprinting. Chapter 7 concludes the thesis and presents future work.

1.5 Contributions

Paper I

Can Aknesil, Elena Dubrova, "An FPGA Implementation of 4×4 Arbiter PUF," 2021 IEEE 51st International Symposium on Multiple-Valued Logic (ISMVL), Nur-sultan, Kazakhstan, 2021, pp. 160-165.

This paper presents an implementation of a novel PUF design as a solution to the problem of secure key generation and storage on FPGAs.

The author developed the implementation, conducted the majority of the experiments, wrote the majority of the paper, and presented it at ISMVL 2021.

Paper II

Can Aknesil, Elena Dubrova, "Towards Generic Power/EM Side-Channel Attacks: Memory Leakage on General-Purpose Computers," 2022 IFIP/IEEE 30th International Conference on Very Large Scale Integration (VLSI-SoC), Patras, Greece, 2022, pp. 1-6.

This paper explores EM side-channel leakage of the memory operations on Raspberry Pi 3, a single-board computer widely used in embedded systems, and as an IoT device.

The author contributed to the conception of the research idea, conducted the majority of the experiments, wrote the majority of the paper, and presented it at VLSI-SoC 2022.

Paper III

Can Aknesil, Elena Dubrova, Niklas Lindskog, Håkan Englund, "Is Your FPGA Transmitting Secrets: Covert Antennas from Interconnect," 2023 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), St. Petersburg, FL, USA, 2023, pp. 79-84.

This paper presents several covert transmitting antenna implementations built solely within the FPGA configurable fabric, with the purpose of investigating the capabilities of an adversary who controls part of the FPGA design.

The author contributed to the conception of the research idea, developed the implementation, conducted the majority of the experiments, wrote the majority of the paper, and presented it at RAW 2023, a workshop within IPDPS.

Paper IV

Can Aknesil, Elena Dubrova, Niklas Lindskog, Håkan Englund, "A Near-Field EM Sensor Implemented in FPGA Configurable Fabric," 2023 IEEE 22nd International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), Exeter, United Kingdom, 2023, pp. 1908-1913.

This paper presents several covert EM sensors built solely within the FPGA configurable fabric, with the purpose of investigating the capabilities of an adversary who controls part of the FPGA design.

The author contributed to the conception of the research idea, developed the implementation, conducted the majority of the experiments, wrote the majority of the paper, and recorded a video presentation that was played at TrustCom 2023.

Paper V

Can Aknesil, Elena Dubrova, "Circuit Disguise: Detecting Malicious Circuits in Cloud FPGAs without IP Disclosure," 2024 27th Euromicro Conference on Digital System Design (DSD), Paris, France, 2024, pp. 361-368.

Can Aknesil, Elena Dubrova, "Circuit Disguise: Detecting Malicious Circuits in Cloud FPGAs without IP Disclosure (Extended version)," Submitted to the Journal of Computers & Electrical Engineering.

Paper V refers to two papers, the second an extended version of the first. Only the first is included in this thesis.

These papers present a novel method for detecting malicious circuits in cloud FPGA designs without disclosing the clients' IP, a.k.a. the FPGA design, to the cloud.

The author came up with the research idea, developed the implementation, conducted the majority of the experiments, wrote the majority of the papers, and presented the first one at DSD 2024.

Paper VI

Can Aknesil, Elena Dubrova, Niklas Lindskog, Jakob Sternby and Håkan Englund, "Hybrid Fingerprinting for Effective Detection of Cloned Neural Networks," Accepted to 2025 IEEE 55st International Symposium on Multiple-Valued Logic (ISMVL).

Can Aknesil, Elena Dubrova, Niklas Lindskog, Jakob Sternby and Håkan Englund, "Hybrid Fingerprinting for Effective Detection of Cloned Neural Networks (Extended version)," Cryptology ePrint Archive, Paper 2025/673, 2025.

Paper VI refers to two papers, the second an extended version of the first. Only the extended version is included in this thesis.

These papers present a neural network fingerprinting method that combines information-domain metrics, namely the model output, with physical-domain metrics, namely the power traces, for more effective detection of model cloning.

The author conducted the majority of the experiments, contributed to writing the papers, and plans to present the first one at ISMVL 2025.

Source code The source code associated with each included paper is publicly available at the locations indicated in the respective papers.

2

Background

This chapter provides a background on hardware security and physical attacks, FPGAs, cryptography, and deep learning.

2.1 Hardware Security

The main goals of computer security are the protection of confidential information stored and processed in a computer, network, or more generally, an electronic system, ensuring the system's integrity, and maintaining its availability. An electronic system consists of hardware and software. Computer security, accordingly, can be divided into software security and hardware security.

The line that separates the hardware from the software is not a fixed line. It depends on the application. In this thesis, *hardware security* is interpreted such that it deals with aspects at a lower abstraction level than an instruction set architecture (ISA), while *software security* deals with ISA and upper-level aspects. Most of the time, software is agnostic of hardware-level details such as cache hierarchy, instruction pipelining, layout of specific components on a printed circuit board (PCB), or hardware-level implementation of a processor on the IC. Software-level attacks do not require or exploit knowledge that is below the ISA level, while hardware-level attacks do. For example, cryptanalysis, exploitation of misconfiguration of file system access permissions, password brute forcing, and analysis of network packets are at the software level, while Meltdown [16] and Spectre [17] that exploit speculative execution, power-based side-channel attacks that exploit power consumption patterns [18], clock glitching attacks that cause a processor to skip an instruction [19], *cold-boot attacks* that freeze a static random-access memory (SRAM) memory chip to access its content after power off [20], tapping PCB traces (see Fig. 3.1), and radio jamming [21] are at the hardware level. According to the above classification, timing attacks are somewhat on the borderline between the software and hardware levels. Timing attacks based on the execution of different software branches [22] can be considered at the software level, while the timing of a sequential set of instructions to exploit, e.g., a multiplication module that has a shortcut for zero inputs, would be at the hardware level.

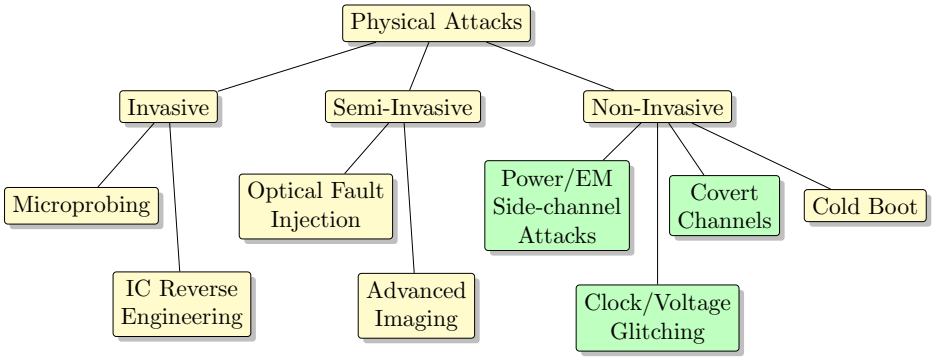


Figure 2.1: Categorization of physical attacks in the field of hardware security. Attacks that are in the scope of this thesis are colored green.

2.2 Physical Attacks

Physical attacks are defined, in this thesis, as a subset of attacks that monitor or interact with the target device through a physical medium. According to this definition, e.g., probing the pins of a target chip or monitoring EM radiation emanating from a target device can be a part of a physical attack. However, microarchitectural attacks such as Meltdown [16] and Spectre [17] are not, as they interact with the target device in the information domain.

The categorization of physical attacks used in this thesis is based on [23]. Physical attacks are divided into three categories, depending on the degree of physical access required. *Invasive attacks* require destruction of the package and the passivation layer, a.k.a. *decapsulation* and *depassivation*, respectively, of the target IC. Passive or active probing of individual wires in an IC, called *microprobing* [24], and reverse engineering of an IC [25] are examples of invasive attacks. *Semi-invasive attacks*, introduced in [23], require decapsulation but not depassivation, as electrical contact with the circuit is not necessary. Optical fault attacks [26] and observation of transistor states via microscopy [23] are semi-invasive attacks. *Non-invasive attacks*, the focus of this thesis, require neither decapsulation nor depassivation. Non-invasive attacks include power/EM-based side-channel attacks, fault attacks, covert channel attacks that are at the hardware level, and cold boot. Examples of non-invasive fault attacks are under/over voltage attacks, also known as voltage glitching [27], and clock glitching. Covert antennas presented in Papers III and IV are examples of hardware-level covert channel attacks. The categorization of physical attacks is illustrated in Fig. 2.1.

Although attacks in physical-layer security, e.g., radio jamming, involve interacting with the targets wirelessly through the physical medium and do not need partial destruction of the target devices, they are not categorized under non-invasive physical attacks in this thesis since it is a major topic in itself within

wireless communications.

2.3 FPGAs

A *field-programmable gate array* (FPGA) is a configurable IC on which custom functionality can be implemented after its manufacturing.

One of the applications of FPGAs is application-specific integrated circuit (ASIC) prototyping [28]. The time required to manufacture ASICs is typically high, and their design and manufacturing are expensive for small volumes [29]. On the other hand, FPGAs have a much lower initial cost as they can be reconfigured in seconds without remanufacturing and have smaller design complexity. Therefore, it is a common practice to implement and test ASIC designs on FPGAs during ASIC development.

FPGAs are also used in applications that can benefit from custom hardware for improved speed and power efficiency compared to central processing units (CPUs) and GPUs. In case the computational load of an application does not match the generic architecture provided by a CPU or a GPU, custom hardware tailored to the exact needs of the application can provide a better performance [30]. Both an FPGA and an ASIC can be used for this purpose. ASICs are faster and consume less power than FPGAs. However, an FPGA may be a better choice if the application has strict time and budget constraints, or is not large-scale enough to have a reasonable per-unit manufacturing cost.

FPGAs are also offered in computing clouds for acceleration as an alternative to CPU multiprocessing and GPUs. A client can configure a cloud FPGA or a portion of it according to their unique needs. An example use of cloud FPGAs is the acceleration of training and inference of neural networks on the cloud [31] and embedded systems [32]. More information on FPGA acceleration in the cloud is provided in section 5.2.

In addition to the above applications, FPGAs are also used in digital design courses at educational institutions, including KTH.

FPGA Architecture The *configurable fabric* of an FPGA contains various types of logic and storage elements that can be connected through configurable switch matrices to implement combinational and sequential logic. *Lookup tables* (LUTs) are fundamental logic elements in an FPGA. A LUT, illustrated in Fig. 2.2, has an associated truth table which can be configured to implement combinational logic functions with up to a certain number of inputs. Multiple LUTs can be connected to implement logic functions that require more inputs than a single LUT can handle.

Flip-flops and latches are fundamental storage elements in an FPGA. Together with LUTs, they can implement sequential logic.

Although nearly all digital circuits can be mapped onto LUTs, flip-flops, and latches, FPGAs may also contain more specialized logic blocks for improved speed

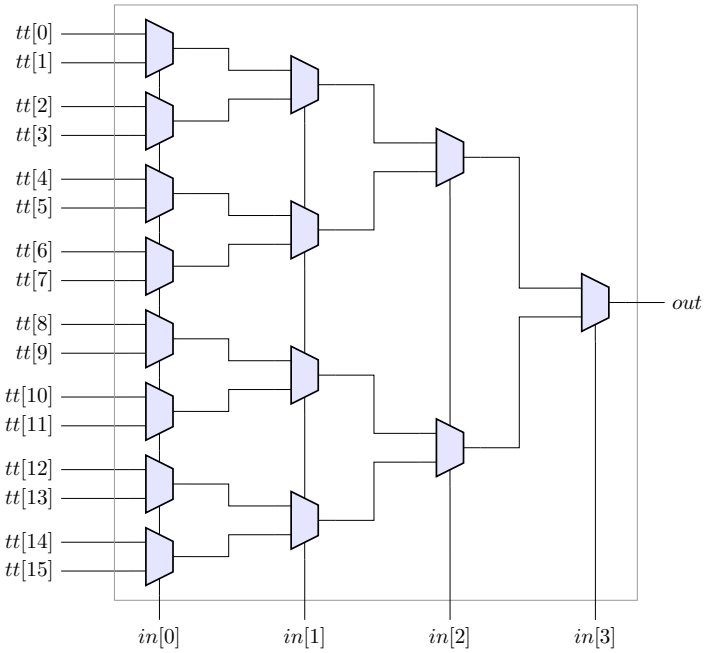


Figure 2.2: A LUT with four inputs $in[0]$ to $in[3]$ and one output *out*, implemented using 2×1 multiplexers. The LUT is configured via setting the truth table entries tt.

and power efficiency. These specialized blocks include random-access memory (RAM), digital signal processing (DSP) blocks, high-speed carry chains, shift registers, and wide multiplexers [33].

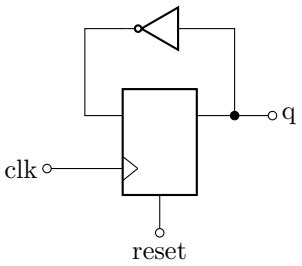
Compilation of FPGA designs The functionality that is to be configured in an FPGA, the so-called *FPGA design*, is usually described at register-transver level (RTL) via a hardware description language, such as VHDL [34] or SystemVerilog [35]. The RTL code is compiled into a *bitstream*. During compilation, the logic described at RTL is mapped onto the FPGA configurable fabric. A bitstream is analogous to a binary program, it depends on the underlying FPGA's architecture. After compilation, the bitstream is configured into an FPGA.

Compilation of an FPGA design consists of *RTL synthesis*, *placement*, *routing*, and *bitstream generation*. During RTL synthesis, the design is expressed in terms of the resource types of the target FPGA. Logic gates and sequential logic are converted into LUTs, flip-flops, and other, more specialized cells. During placement, cells in the synthesized design are mapped onto the resources available on the target FPGA. In routing, the switch matrix configurations are generated to

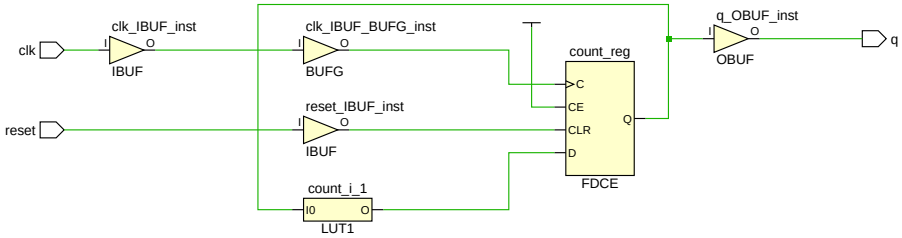
```
entity counter is
  Port (reset : in STD_LOGIC;
        clk : in STD_LOGIC;
        q : out STD_LOGIC);
end counter;

architecture Behavioral of counter is
  signal count: std_logic;
begin
  process(reset, clk)
    begin
      if (reset = '1') then
        count <= '0';
      elsif (rising_edge(clk)) then
        count <= not count;
      end if;
  end process;
  q <= count;
end Behavioral;
```

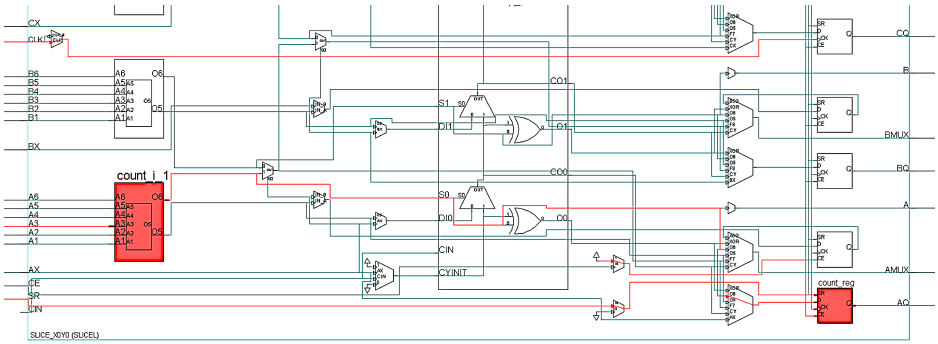
(a) RTL code for a one-bit counter in VHDL.



(b) Illustration of the RTL code of a one-bit counter.



(c) After synthesis.



(d) After placement and routing. Only the LUT and the flip-flop are shown.

Figure 2.3: Compilation of one-bit counter from RTL code to FPGA bitstream.

connect the placed cells. Finally, in bitstream generation, the placed and routed design is converted into a binary format recognized by the target FPGA. A one-bit counter design at RTL (a and b), after synthesis (c), and after placement and routing (d) are shown in Fig. 2.3. The compilation is done via the AMD/Xilinx Vivado design suite [36].

2.4 Basics of Cryptography

Modern cryptography is the study of techniques to achieve secure communication [37]. The three primary goals of cryptography are information confidentiality, integrity, and availability. *Confidentiality* refers to preventing the content of a transmitted message from being read by parties other than the sender and the receiver. *Encryption algorithms*, such as AES [38] and RSA [39], are used to ensure confidentiality. *Integrity* refers to ensuring that the transmitted message arrives at its destination without intentional modifications by an adversary or without unintentional modifications that may occur during the transmission. Integrity also entails the *authentication* of the sender. *Message authentication* enables the receiver to authenticate the sender and ensures that the message has not been modified. Finally, *availability* refers to the successful transmission of the messages without any disruption in the communication. The term is mostly used in the context of a server, and more generally, a service being available. Note that not all three, confidentiality, integrity, and availability, are required in every security application. For example, news feeds require integrity and availability, but not necessarily confidentiality.

Modern cryptography offers various primitives used in cryptographic systems to enable secure communication. Among these primitives, hash functions, symmetric encryption, and public-key cryptography are explained below.

Hash Function A *hash function* is a function that calculates a fixed-sized digest from binary information of arbitrary size. For a cryptographic hash function, it is required that the input to the function is infeasible to be recovered from a given output. Hash functions are used as basic blocks in cryptographic algorithms such as digital signatures and key encapsulation.

MD5 [40] and the Secure Hash Algorithm (SHA)-2 [41] are examples of widely used cryptographic hash functions.

Symmetric Encryption In *symmetric encryption*, illustrated in Fig. 2.4, a message (plaintext) is encrypted and decrypted using the same key K. The key must be distributed to the sender and the receiver beforehand and kept secret. An adversary who gets hold of the ciphertext $E_K(M)$ cannot calculate the plaintext M without the secret key.

AES [38] and ChaCha20 [42] are examples of widely used symmetric encryption algorithms.

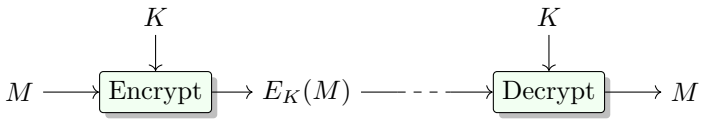


Figure 2.4: Symmetric encryption and decryption of message M using shared key K.

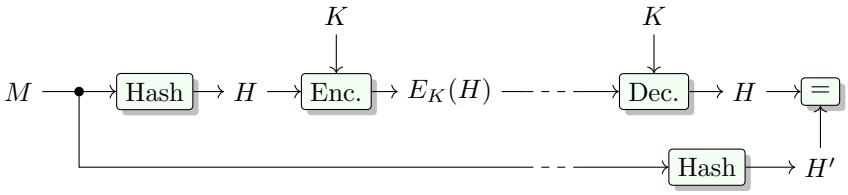


Figure 2.5: Authentication of message M using symmetric encryption.

Symmetric encryption can also be used as a building block for performing message authentication, as illustrated in Fig. 2.5. The sender generates a digest H of a message M using a cryptographically secure hash function, encrypts the digest using a shared key K, and sends the encrypted digest $E_K(H)$ to the receiver along with the message M. To authenticate the message, the receiver calculates the digest H' of the received message, decrypts the encrypted digest sent by the sender, and compares the two. An adversary who intercepts the message can see its content but cannot calculate a new digest for a modified message without the shared key K.

Secure exchange of shared keys is challenging. A trivial way for the sender and the receiver to have a common key is physically coming together and establishing a common secret without any other party being involved. However, this is not very practical. One side remotely sending the secret key to the other through a public network, such as the internet, is not secure, as an adversary may be eavesdropping on their communication. Here is where public-key cryptography comes in handy.

Public-Key Cryptography *Public-key cryptography* is a collection of primitives that make use of *public-private key pairs*. A party can own one or more key pairs. Private keys are meant to be kept secret, while public keys are publicly distributed. Public-key cryptography can be used for encryption, message authentication, and key distribution (establishment).

In *public-key encryption*, illustrated in Fig. 2.6, a public key K_{pub} is used for encryption while the associated private key K_{priv} is used for decryption. First, the receiver shares his/her public key with the sender. This can be done remotely as an adversary having access to the public key is not a security risk. After receiving the public key, the sender encrypts a message M using the receiver's public key

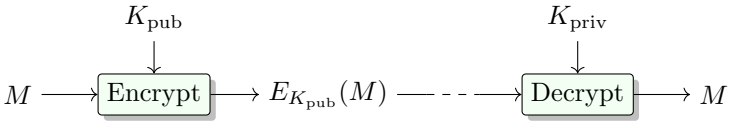


Figure 2.6: Public-key encryption of a message M using public-private key pair $(K_{\text{pub}}, K_{\text{priv}})$.

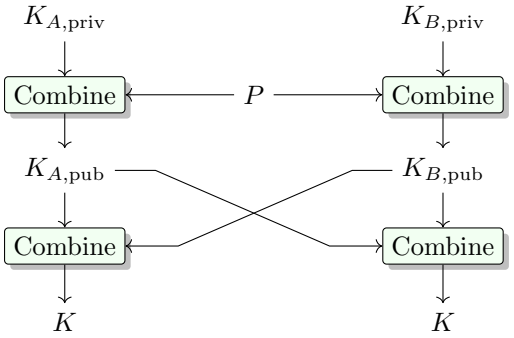


Figure 2.7: Diffie-Hellman key exchange algorithm.

and sends the ciphertext $E_{K_{\text{pub}}}(M)$ to the receiver, who then uses his/her private key to decrypt the message M. An adversary who intercepts the message cannot decrypt it without the receiver's private key.

RSA [39] and ElGamal [43] are examples of widely used public-key encryption algorithms.

Public-key encryption algorithms are typically slower and more computationally intensive than symmetric encryption algorithms. Therefore, it is a common practice to establish a shared key using public-key cryptography and use symmetric encryption for the rest of the communication, such as in the Secure Shell (SSH) protocol [44]. One way to share a symmetric key is by transferring it using public-key encryption. Another way is using a key exchange algorithm, such as the Diffie–Hellman [45], illustrated in Fig. 2.7. The algorithm starts with the sender and the receiver agreeing on a set of public parameters P. Each side then generates a private key, $K_{A,\text{priv}}$ and $K_{B,\text{priv}}$, respectively, combines the generated private key with P and sends the generated value, $K_{A,\text{pub}}$ and $K_{B,\text{pub}}$, respectively, to the other side. At the final stage, each side combines the received value with its private key to obtain the shared secret K.

Message authentication using public-key cryptography is similar to message authentication using a shared key, except that the sender encrypts the digest using his/her private key (contrary to public-key encryption where the public key is used for encryption) and the receiver uses the sender's public key for the

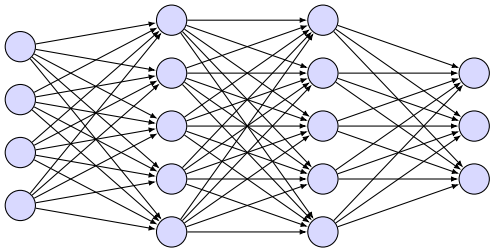


Figure 2.8: A neural network with one input layer, one output layer, and two fully connected hidden layers.

decryption of the digest. In public-key cryptography-based message authentication, the encrypted digest is called *digital signature*. Similar to a handwritten signature, a digital signature is associated with a single person or an entity, and it is difficult to forge.

One challenge with public-key cryptography is the authentication of public keys, in other words, ensuring that a public key belongs to the person or entity of interest, and has not been replaced by the adversary. A common solution for this challenge is to have a certification authority that binds public keys to respective entities and provides means to verify the authenticity of the public keys.

In modern cryptography, secret keys must be generated, stored, and processed securely because an extraction of the secret key by an adversary would compromise the whole cryptographic system.

2.5 Deep Learning

A *neural network* is a computational model inspired by the animal brain, consisting of layers of *neurons*, illustrated in Fig. 2.8. A neuron typically executes a weighted sum of its inputs, followed by an activation function:

$$f\left(b + \sum_{i=1}^{N} x_i w_i\right) \tag{2.1}$$

where x_i is the i^{th} input to the neuron, w_i is the weight of input i, b is the bias of the neuron, and f is the activation function.

Neural networks are widely used for optimization, data classification, pattern recognition, and anomaly detection in various areas including finance, engineering, security, agriculture, meteorology, medicine, education, insurance, marketing, management, and manufacturing [46].

A neural network needs to be trained to do a desired task. *Training* refers to the process of adjusting a neural network's parameters, w and b in eq. 2.1,

using a training dataset until the network learns the patterns in the training samples. The network can then be deployed to perform its task. For example, a neural network can be trained to recognize the type of a flower from its pictures, in which case the network needs to be trained using a collection of pictures of flowers of known types. *Inference* refers to the usage of a neural network after it has been trained.

Deep learning is a subcategory of machine learning that allows learning directly from the raw data without the need for human-engineered features [47]. Accordingly, training and inference of deep neural networks require more computational power and are more expensive.

3

Secure Key Generation and Storage on FPGAs

This chapter explains the need for secure key generation and storage on FPGAs, elaborates on existing methods, with a greater focus on PUFs, and presents the arbiter PUF with 4×4 switch blocks.

3.1 Cryptographic Keys on FPGAs

There are multiple scenarios where a cryptographic key needs to be stored in an FPGA. One is to enable bitstream encryption. SRAM-based FPGAs are widely used in the industry due to their reconfigurability and speed [48]. These FPGAs store the configuration in the internal SRAM memory. A disadvantage of SRAM-based FPGAs is that they need to be reconfigured at every power-up. To solve this problem, some FPGA vendors offer ways to download the configuration from an external source, such as from non-volatile storage using serial peripheral interface (SPI), or a computer using universal serial bus (USB) [33]. Regardless of the method, loading an FPGA from an external source requires the bitstream to travel through package pins and PCB wires. Therefore, an adversary who has

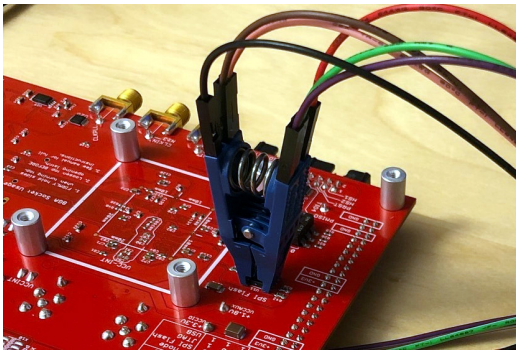


Figure 3.1: Bitstream extraction from Flash memory using a passive probe.

physical access to the FPGA can easily extract the bitstream by eavesdropping on the media the bitstream passes through, such as by probing the pins of the Flash configuration memory, see Fig. 3.1. This way, the adversary can create an unauthorized copy of the bitstream or an IP core used in the FPGA design. To solve the problem of the bitstream being exposed, the FPGA vendors offer *bitstream encryption* [33], a method in which the confidential part of the bitstream is encrypted during the compilation of the design, and decrypted on the FPGA after its transfer. For the FPGA to decrypt the bitstream a cryptographic key must be provisioned in the FPGA and kept through subsequent power cycles.

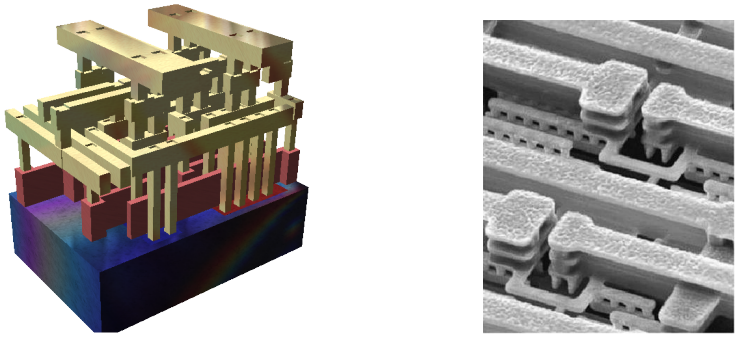
Another use of a stored cryptographic key is secure communication with the FPGA during its runtime. This is required if sensitive information is to be sent to or received from the FPGA during its operation. One example application is training or inference of a neural network, in which the inputs and outputs to the neural network, thus to the FPGA, are confidential. Secure communication to an FPGA can be implemented via a cryptographic key that is stored in the FPGA and that remains after power-off. Alternatively, public-key cryptography can be used for communication, which requires a public-private key pair to be generated on-chip using a true random number generator (TRNG). A TRNG may also be needed for key exchange, such as the Diffie-Hellman key exchange algorithm, see section 2.4.

A key stored on the FPGA can also be used for message authentication. This can be the authentication of the bitstream, which ensures the authenticity and integrity of the bitstream. Similarly, any information communicated with the FPGA can be authenticated. A related application is the authentication of the FPGA itself or the owner of the FPGA, similar to the way credit cards, and accordingly their owners, are authenticated.

3.2 Existing Key Generation and Storage Methods

Conventional pseudorandom number generators are not suitable for generating cryptographic keys due to their predictability [49]. Instead, cryptographic keys need to be generated by a TRNG. TRNGs exploit the noise inherent in the environment and the underlying electronic circuit. Various TRNGs have been proposed for FPGAs [50, 51, 52].

The state-of-the-art FPGAs provide two means for on-chip storage of bitstream encryption keys: a dedicated, battery-backed RAM and electronic fuses (eFuse) [33]. The battery-backed RAM is sustained by an auxiliary voltage source, a battery, and retains the key when power is down. eFuse is a non-volatile storage technology that is programmed by burning the fuse links with a high current. While the battery-backed RAM can be programmed multiple times during the lifetime of an FPGA, eFuse is one-time programmable. The disadvantage of both methods is that the key is present when power is down and can be extracted via physical attacks [53, 23].



(a) (b)

Figure 3.2: Silicon chip at design stage (a) [1] and after manufacturing (b) [2].

3.3 Physically Unclonable Functions (PUFs)

PUFs are a technology for secure key generation and storage on ASICs and FPGAs. A PUF is a digital circuit designed to exhibit a unique behavior on every chip on which it is implemented by exploiting random manufacturing differences. No two instances of the same chip model are identical at the analog level; accordingly, the behavior of a well-designed PUF implemented on them differs. Small segments of a silicon chip at the design stage and after manufacturing are shown in Fig. 3.2 and 3.2b, respectively. Random manufacturing variations are visible in the latter.

A PUF provides a challenge-response interface. The challenge is the input to the PUF and the response is the output. The same PUF design, implemented on two different instances of a chip, must generate a distinct response to a given challenge.

Various kinds of PUFs were proposed previously. The majority of the existing PUF designs can be categorized into two, according to the analog properties they use. The first category of PUFs exploits random delay differences in the electronic paths in a silicon device. Multiple paths in an IC can be designed such that they, in theory, have the same delay. However, these paths, after manufacturing, end up having small random delay variations. Some examples of delay-based PUFs are arbiter PUFs, such as the one presented in Paper I, and ring oscillator PUFs [54]. An arbiter PUF compares delay differences of multiple paths that, according to their abstract layout, have the same delay, but in reality, have a small and unique variation on every produced chip. The arbiter PUF implemented in Paper I is illustrated in Fig. 3.6. A ring oscillator PUF contains one or more combinational loops, each consisting of an odd number of inverters (see Fig 3.3). Such a loop provides a digital signal that oscillates between high and low logic levels in a frequency dictated by the total delay of

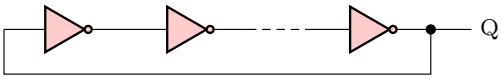


Figure 3.3: Ring oscillator consisting of an odd number of inverters.

the loop, which is different on every instance of the device. The second category of PUFs exploits metastability inherent in volatile memory technologies. Examples in this category include SRAM PUF [55] and Butterfly PUF [56].

Configurability is a big advantage of PUFs. Once a key is provisioned on an FPGA, it will have an expiration date. A key can only be used a limited amount times before it becomes feasible for an adversary to guess it through cryptanalysis [57], or side-channel attacks [58]. Therefore, a new key must be periodically provisioned. A reconfigurable PUF can be thought of as a large list of key-value pairs that would otherwise take a much larger area. A reconfigurable PUF removes the necessity of such big storage and periodic key provisioning.

PUFs are characterized via three metrics: *uniformity*, *reliability*, and *uniqueness*. Uniformity is the measure of how equally the output of a PUF is distributed across the set of possible responses, for a uniform set of challenges. Reliability is a measure of whether the PUF generates the same response to repeated application of a given challenge on a given device. A perfectly reliable PUF should generate the same response to a given challenge on a given device. And finally, uniqueness measures the capability of a PUF to generate unique responses to a given challenge across multiple devices.

3.4 Usage of PUFs

PUFs can be used as TRNGs to generate keys for encryption, authentication, and key exchange. PUFs that are used as TRNGs are required to have high uniformity and uniqueness, but not reliability. A ring oscillator-based PUF that does not take any input challenge can be used as an entropy source of a TRNG [59]. A non-reliable PUF that implements a challenge-response interface can also be used as a TRNG, with the condition of having non-perfect reliability. The entropy of the output responses of such a PUF can be improved by combining multiple responses, as suggested in Paper I. Another method of improving entropy is to use a PUF in combination with a pseudorandom number generator [60]. The design proposed in [60] requires a PUF with high reliability.

As the output of a TRNG cannot be replicated, a new key should be generated at every power cycle. PUF-based TRNGs are useful for establishing a secure communication channel with an FPGA after it powers up. Whereas, if an application requires a key that persists after power is down, such as in bitstream encryption, PUFs need to be used another way, as explained below.

Key storage is another use case of PUFs. A PUF that is used for key storage is required to have high uniformity, reliability, and uniqueness. One key storage

method using a PUF is to configure a PUF on a batch of FPGAs before their deployment and to create a database that contains a list of challenge-response pairs for every FPGA. The database must be kept secure throughout the lifetime of the FPGAs. After an FPGA has been deployed, the same PUF is configured on the FPGA and is instructed to use the response to a particular challenge chosen from the database as the secret key for secure communication. The response of the PUF, hence the key, is known through the database by the party who is to communicate with the FPGA. In the case of bitstream encryption, the logic that makes use of the PUF and takes care of the decryption of the bitstream must be within the FPGA chip but outside the configurable fabric.

A similar method can be used in applications where the FPGA itself needs to be authenticated. During the authentication process, the FPGA is asked to give a response to one or more challenges chosen from the database. If the received responses match the ones in the database, the FPGA is authenticated. No challenge should be used more than once for authentication, as an eavesdropping adversary who gets hold of the challenge-response pairs may attempt to bypass authentication by impersonating an authentic FPGA.

The processes of key exchange and authentication using a PUF are illustrated in figure 3.4.

The number of challenge-response pairs recorded in the database is limited by the available resources, accordingly, a PUF must have a limited *lifetime*, thus must *expire* after a certain number of challenge-response pairs are consumed. Another consideration for the lifetime of a PUF is the amount of challenge-response pairs that will enable an adversary to break the PUF's security. Setting a shorter lifetime for a PUF, thus reducing the number of challenge-response pairs that will be revealed, makes a PUF more secure.

Another application of PUFs is the identification of the device it is implemented on. There are other methods of FPGA identification, e.g., The AMD/Xilinx UltraScale FPGAs contain an embedded, 64-bit device identifier which is used to provide a Device DNA value [33]. The identifier is nonvolatile, permanently programmed into the FPGA, and is unchangeable, making it tamper-resistant. Note that Device DNA is not reconfigurable. It does not make sense to use a PUF to identify an FPGA if an identification method is already provided by the vendor. However, PUFs can be useful to identify smaller devices where non-volatile memory is unavailable or undesirable. For example, radio-frequency identification (RFID) tags are small and they typically need to be produced in large amounts, therefore putting a non-volatile memory in RFID tags would be very costly. A PUF, on the other hand, can be a much cheaper choice [61]. An example RFID tag is shown in Fig. 3.5. A method similar to authentication can be used for identification as well.

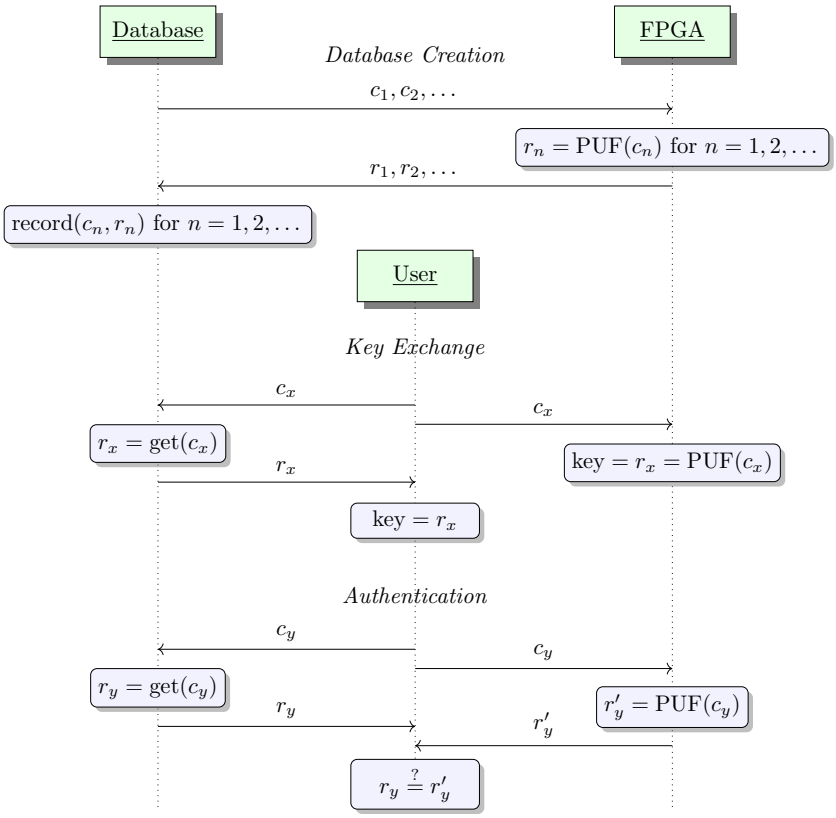


Figure 3.4: Key exchange and authentication using a PUF.

3.5 PUF Security

One big advantage of PUFs is that their content is not visible when power is off, reducing the risk of the challenge-response pairs being extracted via physical attacks.

Modeling attacks is one weakness of PUFs [62, 63]. An adversary can model a PUF's functionality and later use the model to replicate the PUF's behavior. Increasing a PUF's complexity generally makes modeling attacks more difficult, as more challenge-response pairs would need to be exposed and the adversary would need more computational resources. An alternative is to make a PUF bigger by increasing the value of internal parameters, such as by increasing the number of switch blocks in a 4×4 arbiter PUF (see Fig. 3.6a). Setting a shorter lifetime for a PUF is another countermeasure against modeling attacks since it bounds the number of challenge-response pairs that will be revealed during the PUF's use.

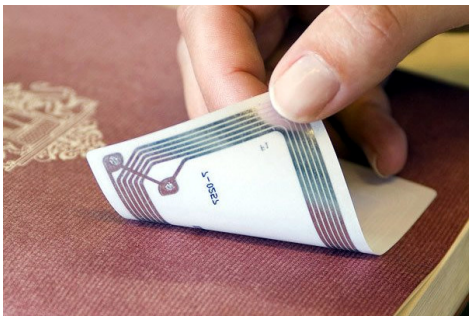


Figure 3.5: An example RFID tag [3].

A PUF must also be resistant to side-channel attacks. In [64], an exhaustive, deep learning-based power side-channel analysis is performed on a 2×2 arbiter PUF implemented on an AMD/Xilinx Artix-7 FPGA. The analysis was proved to be ineffective at recovering the PUF's output with high accuracy. The PUF's output could only be extracted after increasing the power leakage by replicating the output flip-flops through bitstream modification.

3.6 4×4 Arbiter PUF

Paper I presents an implementation of a novel PUF design, 4×4 arbiter PUF, as a solution to the problem of secure key generation and storage on FPGAs, and proposes methods that enable effective usage of the PUF in various real-world applications.

The high-level design of the 4×4 arbiter PUF is illustrated in Fig. 3.6a. It consists of four paths that pass through several switch blocks and arrive at the arbiter. Each switch block, configured by a challenge in the range $[0, 23]$, is capable of rerouting its four inputs in 24 distinct permutations. Example switch block configurations are illustrated in Fig. 3.6b. At the design phase, all possible configurations of the switch blocks result in the four paths having the same delay, however, due to random manufacturing differences of the underlying FPGA, the paths end up having a slightly different delay depending on the challenge provided by the user.

The implemented PUF is demonstrated to have sufficient uniformity, reliability, and uniqueness to be used for key generation, key storage, authentication, and identification. The area utilization is at most one-fifth of the area utilization of the previously proposed arbiter PUF variants.

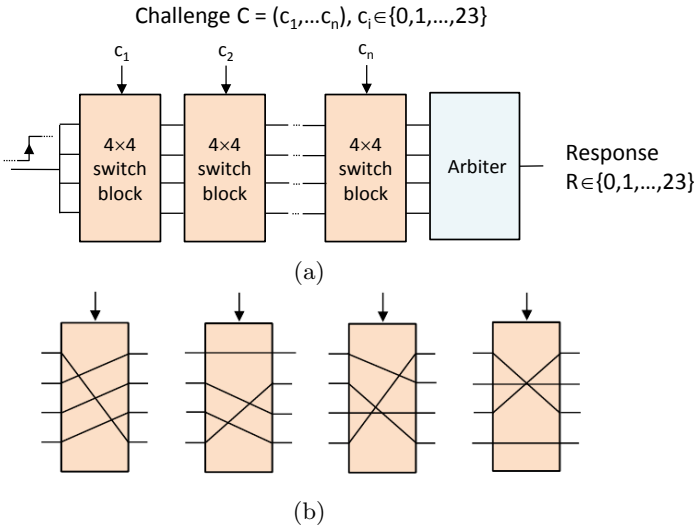


Figure 3.6: High-level design of 4×4 arbiter PUF from Paper I (a), example switch block configurations (b).

3.7 Summary

Secure key generation and storage on FPGAs is a challenging problem due to the inflexibility of the existing solutions and their vulnerability to physical attacks. PUFs exploit random manufacturing differences of the underlying FPGA to solve the problem. Paper I presents an implementation of a novel PUF design, which is demonstrated to be effective for applications that require key generation and storage. The presented implementation has a much smaller area utilization compared to the existing PUF variants.

4

EM Side-Channel Attacks

This chapter presents EM side-channel attacks and their countermeasures, along with the techniques involved, including deep-learning-based profiling, divide-and-conquer strategy, leakage models, averaging, trace synchronization, and leakage assessment. In addition, the generalization of side-channel attacks through the extraction of data in memory operations is presented. Although the chapter focuses on the EM side channel, the presented methods can also be applied to power side-channel attacks.

4.1 Growth of Side-Channel Attacks

Every electronic device, when it operates, has side effects observable from its environment. The side effects include EM radiation. According to Maxwell's equations [65], changing voltage levels in a circuit generates a changing electric field, changing current generates a changing magnetic field, and changing electric and magnetic fields induce each other, creating an EM radiation that originates from the circuit and propagates to its environment. Side effects other than EM radiation are noise that propagates back to the power supply from a circuit [18], sound [66], and computational side effects due to complex memory hierarchies and speculative execution [16, 17].

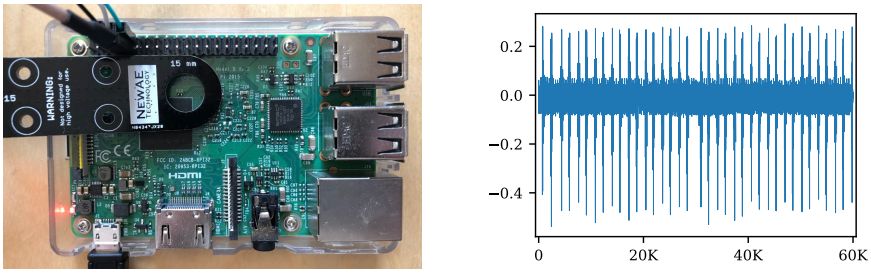


Figure 4.1: The EM side-channel attack from Paper II. Target device with the EM probe on the left. A side-channel measurement on the right.

The side effects depend on the device's operation and the information processed in it. By analyzing these side effects, it is possible to understand what the device is doing and to extract the information processed. This is a vulnerability for devices that process confidential information, such as cryptographic keys, medical records, and banking information. Attacks that exploit the side effects of an electronic device to extract confidential information are called *side-channel attacks*, and the side effects collected for this purpose are called *side-channel information*.

There have been game-changing discoveries in the field of side-channel attacks in less than two decades. One is machine learning-based side-channel attacks, which first appeared in 2011 [67], that use machine learning methods to extract the secret from side-channel information. Machine learning-based side-channel attacks have repeatedly been proven to be very powerful [68]. A second discovery is, *screaming channels* [69], first published in 2018. Electronic devices with a radio transmitter, e.g., Bluetooth or Wi-Fi, are found to be unintentionally broadcasting noise present on the device, coupled with the transmitter circuit. The noise created in the digital parts of the device that process secret information can leak into the transmitter circuit and be broadcast. The transmitted signal can be picked up by an adversary who places a far-field antenna at a distance of up to 15 meters from the target device, and the secret information can be extracted from the captured signal, as demonstrated in [70].

It has been shown that small embedded devices and FPGAs are most vulnerable to power/EM side-channel attacks due to the simplicity of their designs, specificity of the tasks they are typically used for, and, thus, consistency of the collected side-channel information, explained in more detail in Paper II.

4.2 Profiled Deep Learning-Based Side-Channel Attacks

Profiled deep learning-based side-channel attacks have two stages: *profiling*, or *training*, and *attack*. In the profiling stage, the adversary obtains a replicate of the target device and collects side-channel information by repeatedly executing a selected program or algorithm on the profiling device. For example, if the adversary aims to extract a cryptographic key from an encryption algorithm, he/she repeatedly executes the algorithm for a set of known keys. If the aim is to extract information from memory operations, as in Paper II, the adversary programs the target device to perform repeated memory operations. In the context of power and EM side-channel attacks, the collected information is typically called *traces*. One trace is the information collected from a single execution, it is typically a one-dimensional, finite, discrete signal. The collected traces are then used for the training of a neural network. The neural network's input is a single trace, and the output is the secret information to which the trace depends. In the attack stage, the neural network is used to predict the secret information from traces collected from a target device.

Deep learning-based side-channel attacks are, in their essence, very similar to image classification. The input trace is classified as one of the possible target values. Although, traces are typically one-dimensional, rather than two as with black and white images, or three, as with colored images.

The equipment needed to perform profiled deep learning-based side-channel attacks is listed below.

1. A profiling device, preferably similar to the target device.

2. Means to collect side-channel information. EM attacks require a near-field EM probe, as used in Paper II (see Fig. 4.1), or a far-field antenna, as in [70]. Power attacks require measuring the current or voltage of the devices' power line via, e.g., a voltage probe and a shunt resistor circuit, similar to the one present in CW305 target board [71]. For both power and EM attacks, a capturing device incorporating an analog-to-digital converter (ADC) is required to convert and store the side-channel information, such as an oscilloscope or a dedicated capture board similar to CW1173 [72]. Capturing computational side-channel information, e.g., cache timing, may not require additional equipment as the information can be collected via another process running on the same device, as in [17].

3. Computing resources to perform training and inference of the neural network.

4. In an experimental setup, a target device is also needed. The profiling device can be used as the target device. However, note that two instances of the same device may result in slightly different side-channel information, thus it is a common practice to have a separate target device for a more robust attack if the budget permits it [73].

4.3 Divide-and-Conquer Attack Strategy

The size of the secret information targeted in a single attack is limited by computing resources. For example, a 128-bit cryptographic key can take 2^{128} different values. It is computationally intensive to predict among 2^{128} classes. One way to tackle this problem is to partition the key and repeat the attack for each partition. For example, each byte of a 128-bit key (16 bytes in total) can be recovered separately, and the extracted bytes can be concatenated, as performed in [74], assuming that the target encryption algorithm allows such partitioning. A separate neural network can be trained for each byte of the key. As each neural network targets a single byte, the number of classes reduces to 2^8, which is manageable.

4.4 Leakage Models

The information that leaks through the side channel may not be directly the secret information itself, but other information that is related to the secret information. Such as a value calculated at an intermediate step of a cryptographic algorithm. The leaking value may also leak in an altered form. For instance, the power consumption or the EM radiation may correlate to the number of bits, so-called *Hamming weight*, of a value, rather than the value itself. In these cases, a *leakage model* can be defined. A leakage model maps the secret information to its expected leakage. The selection of the leakage model depends on the target device's nature, such as whether it is an FPGA, ASIC, or microcontroller. If the target information is processed in an algorithm, the leakage model also depends on the algorithm's implementation. If a leakage model is used in a deep learning side-channel attack, the neural network is trained to recognize the output of the leakage model. After the extraction of the value that depends on the secret value, the secret value needs to be reconstructed by reversing the leakage model.

Neural networks are sometimes capable of predicting the leaking value with a simpler leakage model, or even without the need for a leakage model. For example, neural networks in [75] predict the intermediate values of the AES algorithm directly. Likewise, neural networks in Paper II each predict a single byte of a memory operation directly without any leakage model. This is a strength of deep learning-based side-channel attacks.

4.5 Noise and Averaging

Traces that are collected from a device often contain noise picked up from the environment or other parts of the device, such as from an operation ongoing in parallel with the target operation. Due to the presence of noise, the traces end up having a finite signal-to-noise ratio (SNR). In the context of side-channel attacks, *SNR* is the power ratio between the component of a trace that depends on the secret information to the rest of the trace. Low SNR may, besides the noise, also be due to limitations related to trace collection, such as the distance of the antenna to the target device. Another reason for low SNR may be the presence of a noise-based countermeasure that aims to reduce the side-channel leakage.

Averaging is a common method to increase the SNR of traces. Averaging stands for collecting multiple traces using a given target value, and then averaging the resulting traces, thus reducing them to a single trace. An averaged trace has the same dimensions as a trace before averaging, and it is expected to have been cleaned from the noise.

In the presence of Gaussian noise, averaging by N improves the SNR by a factor of $\sqrt{N}$ [76].

Both the profiling and attack stages of deep learning-based side-channel attacks may benefit from averaging. However, averaging is costly because it in-

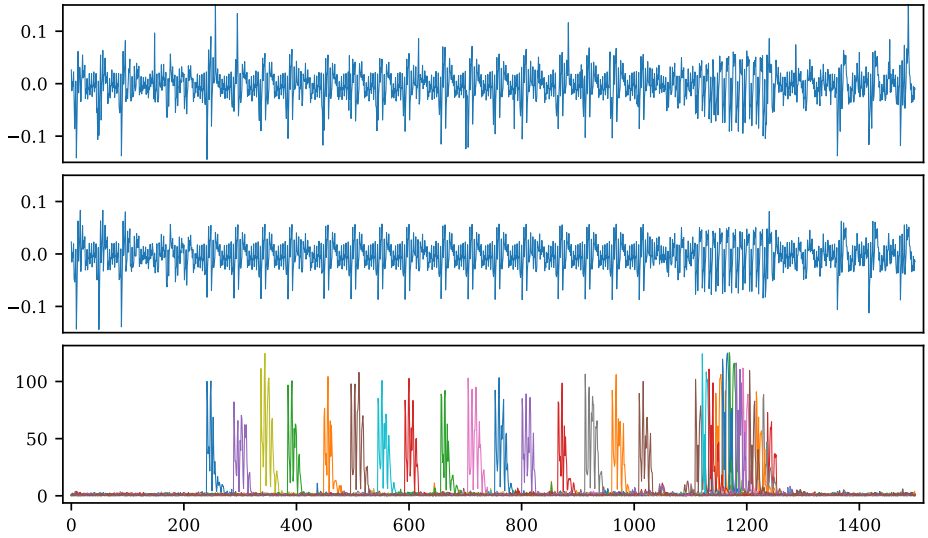


Figure 4.2: A power trace from the execution of AES before averaging (top), averaged trace (middle), and Student's t-test result showing the leakage of all 16 bytes of the first S-box output (bottom).

creases the number of traces that need to be collected, hence the time required to collect them. It may not be even possible to perform averaging in the attack stage, as the target device may not repeat the same execution more than once.

Averaging is also useful to perform analysis of a trace with the naked eye, as it will help reduce the unrelated content in the trace and make the interesting part of the trace visible. The effect of averaging can be seen in Fig. 4.2.

4.6 Trace Synchronization

The collected traces are useful only if they are collected at the right moment so that they belong to the desired part of the target device's execution. It is also necessary to align the traces so that parts of the target device's execution align across all traces. *Synchronization* can refer to both making the capturing device select the right moment to start capturing, and adjusting the individual traces, e.g., by shifting them right or left, after they have been captured. Synchronization makes averaging possible and makes a neural network's job much easier.

During the profiling stage of deep learning-based side-channel attacks, the profiling device may be programmed to provide an external trigger for the capturing device. This can make synchronization very easy, at least on simple devices that do not require a more sophisticated form of synchronization. An adversary

may not have such luxury in the attack stage, in which, the traces should be synchronized after they have been collected.

Cross-correlation-based synchronization is one method to synchronize traces after they have been captured. *Cross-correlation* of two traces u and v is defined as follows:

$$(u \star v)[m] = f(u[n], v[n - m + N_v]) \ , \quad m = 1, \ldots, N_u + N_v - 1 \tag{4.1}$$

where N_u and N_v are the number of points in u and v, respectively. f is conventionally the dot product [77], however, in Paper II, the Pearson correlation coefficient is used due to its non-susceptibility to amplitude scaling. The *Pearson correlation coefficient* is defined as follows [68]:

$$r_{X,Y} = \frac{\sum (x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum (x_i - \overline{x})^2 \sum (y_i - \overline{y})^2}} \tag{4.2}$$

where $r_{X,Y}$ is the Pearson correlation coefficient of random variables X and Y. x_i is i^{th} sample of random variable X. $\overline{x}$ is mean of random variable X. r takes a value in the range $[-1, 1]$, 1 and -1 indicating perfect direct and inverse linear correlation, respectively, and 0 indicating the lack of correlation.

The sum of absolute differences has also been used as an alternative to the dot product and the Pearson correlation coefficient, in which case, 0 indicates the maximum direct correlation [13].

Simply put, cross-correlation calculates the similarity of two traces u and v as a function of the amount of shifting applied to the second trace v. The location of the highest value in cross-correlation indicates how much the second trace v should be shifted to maximize similarity and synchronize the two traces.

For a set of traces, cross-correlation-based synchronization can be performed by selecting one trace as a reference, cross-correlating other traces with the reference trace, and shifting them according to the location of the highest value in their respective cross-correlations.

The synchronization accuracy is proportional to the quality of the reference trace. In Paper II, the reference trace, or the *signature* as it is called in the paper, is created by first synchronizing several traces using cross-correlation and averaging the traces afterward.

Another method of post-capture synchronization is *dynamic time warping*, which is more powerful than the cross-correlation-based method, as it tackles synchronization errors that cannot be solved by mear shifting [78].

The task of synchronization becomes more difficult as the complexity of the target device increases and the inner timing of the trace becomes less predictable. On a complex system such as a personal computer running a general-purpose operating system (OS) like Linux, the execution of the same binary twice may result in very different power traces even after the starting points are aligned,

due to features including the complex memory hierarchy, presence of concurrent and parallel processes, dynamic branch prediction, and out-of-order execution, as explained in Paper II.

4.7 Leakage Assessment

One way to determine if a target device leaks secret information is to perform side-channel attacks and observe the outcome. However, this would require a lot of effort and may not give much insight into the cause of the leakage. That's why it is a good idea to perform a leakage assessment before attempting an attack.

A side-channel attack typically starts with manual inspection of a small number of traces, e.g., by plotting them on a computer or by analyzing the signal on an oscilloscope, to understand the execution flow of the target device and to locate potential leakage points. Afterward, the adversary can move on to the leakage assessment.

Student's t-test is a leakage assessment method that assigns a measure of statistical significance to the average difference of two sets of information [79].

Student's t-test for unequal sample sizes and equal variances is defined as follows:

$$\begin{aligned} t &= \frac{\overline{x} - \overline{y}}{s_p \sqrt{\frac{1}{n_X} + \frac{1}{n_Y}}} \\ s_p &= \sqrt{\frac{(n_X - 1)s_X^2 + (n_Y - 1)s_Y^2}{n_X + n_Y - 2}} \end{aligned} \tag{4.3}$$

where t is the result of the test, called *t-statistic* or *t-value*. X and Y are two sample sets. $\overline{x}$ is mean of X. n_X is the number of samples in X. s_p is *pooled variance*. s_X is the variance of X.

A t-value is calculated for each sample point of traces.

Student's t-test is used as follows. Suppose that the adversary wants to know if the target device leaks a variable x during the execution of an algorithm. The first step is to collect traces where x takes different values. The collected traces are then separated into two groups according to a chosen condition on x. For instance, grouping can be done according to a single bit b of x: traces where $b = 0$ are separated from the traces where $b = 1$. The result of the Student's t-test should tell whether the selected bit of x leaks. Another example is separating traces where the Hamming weight of x is smaller than $B/2$ from traces where the Hamming weight is equal or larger than $B/2$, where B is the number of bits used to store the variable x. The test, in this case, should tell if the Hamming weight of x leaks. After the grouping, the t-value is calculated for every trace sample. If there is a leakage, a high t-value is expected for some samples. According to [79], usually a threshold of $|t| > 4.5$ is defined. When the calculated t-values are

plotted alongside a trace, samples with higher t-values should indicate the points in time when a leakage occurs. Example leakage assessment using Student's t-test is shown in Fig. 4.2.

4.8 Countermeasures

In a context where the target device runs a cryptographic algorithm, countermeasures can be proposed both at the algorithm level and the implementation level.

Developing algorithm-level countermeasures, that is, improving existing cryptographic algorithms and developing new ones, is very hard because cryptographic algorithms are typically highly standardized after years of heavy research, and are widely adopted around the world [11]. Widely used encryption algorithms, such as AES and RSA, are fundamentally vulnerable to side-channel attacks, as demonstrated in [13] and [14], respectively, because side-channel attacks were not taken into account in their development, as side-channel attacks were less known back then. National Institute of Standards and Technology (NIST) announced AES in 2001 [38]. RSA was introduced in 1976 [39]. They are still the most common encryption algorithms today due to the difficulty of the development and adoption of new cryptographic algorithms.

There can be multiple implementations of a cryptographic algorithm or, more generally, an application, whether it be in hardware or software. Implementations of applications that operate on confidential information must be secure against side-channel attacks.

A side-channel countermeasure typically aims for one or more of the following:

- Prevent attack partitioning. For example, the Dilithium digital signature scheme processes on 13-bit data blocks, compared to 8-bit blocks used in AES, thus, increasing the attack space by a factor of 32 [70].
- Force a leakage model that makes the reconstruction of the secret information more difficult, e.g., via *masking* [80].
- Reduce SNR of the side-channel information by reducing the leakage [81] or introducing noise [82].
- Make synchronization of the traces more difficult, e.g., via clock randomization [83].
- Limit the number of traces exposed to the adversary, e.g., via *key rolling* [33].

4.9 Memory Leakage on General-Purpose Computers

Paper II performs leakage assessment and a deep learning-based EM side-channel attack to extract the data in memory operations on Raspberry Pi 3, a single-

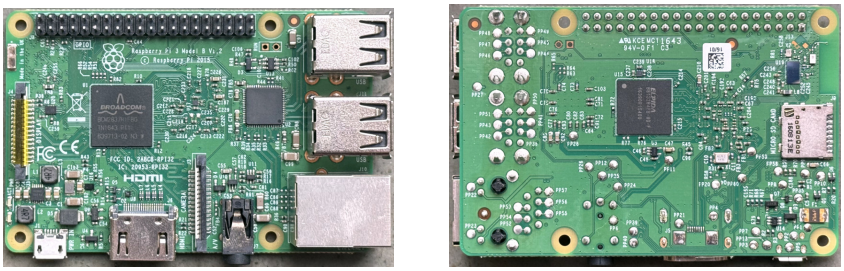


Figure 4.3: Raspberry Pi 3 Model B front (left) and back (right). The front contains the processor chip. The back contains the memory chip.

board computer widely used in embedded systems, and as an IoT device, shown in Fig. 4.3.

The major drawbacks of the existing power and EM side-channel attacks are as follows:

1. Limited applicability to only simple devices, such as small microcontrollers, or specialized hardware.
2. Lack of flexibility. A power or EM side-channel attack is usually applicable to a specific implementation of a specific algorithm that runs on a specific device. The effort necessary to port an existing attack to another algorithm, implementation, or device is high.
3. The amount of control required to have on the target device and the attack environment is high.

Paper II aims to work towards understanding the capabilities of side-channel attacks on general-purpose computers. To this end, the focus is on individual memory operations, rather than a specific cryptographic algorithm. This approach is considered, by the authors of this paper, to be one of the building stones that can enable side-channel analysis on more complex devices.

The attack presented in Paper II is capable of extracting, from a single trace, bytes of a 4-byte word written to or read from the main memory, with accuracy from 49% to 86%.

As explained earlier in section 1.2, understanding adversaries' capabilities is essential to be able to develop countermeasures. The contributions of Paper II are expected to be useful in securing complex devices, such as personal computers and clouds, from power and EM side-channel attacks.

4.10 Summary

Power and EM side-channel attacks are capable of breaking the most widely used cryptographic algorithms, such as AES and RSA. It is paramount to un-

derstand the capabilities and limitations of side-channel attacks and to develop countermeasures. Currently, a typical side-channel attack is only effective on simple devices, in tightly-orchestrated attack environments. Paper II aims to investigate the generalization of existing side-channel attacks on general-purpose computers, a necessary step towards securing them.

5

Attacks on Remote FPGAs

This chapter discusses the use of FPGAs in embedded systems and cloud environments, elaborates on hardware Trojans, attacks, and IP protection in remote FPGAs, and presents covert antennas implemented in FPGA configurable fabric along with the circuit disguise method.

FPGAs are used in embedded systems and the cloud [66, 84]. In both cases, the FPGA is physically accessible by people other than the design owners or those whose information is processed.

5.1 FPGAs in Embedded Systems

An embedded system is usually part of an electronic product that is physically accessible to the public after its deployment. Examples of these products are wire-

Figure 5.1: A replica of the Trojan horse [4]. Hardware Trojans, a.k.a. malicious modifications to an electronic circuit, take the name from the Trojan horse.

less routers, base stations, mobile phones, smart card readers, security cameras, door phones, automated teller machines (ATM), cars, medical equipment, and factory automation infrastructure. Being publicly accessible makes an embedded system, hence an FPGA that may be present in that system, vulnerable to physical attacks, including power/EM side-channel attacks [85], fault attacks [86], covert channels, and backdoors.

5.2 FPGA Acceleration in the Cloud

Today, many computing cloud providers offer FPGAs to be used by their clients, a.k.a. FPGA as a service (FaaS), as an alternative to CPU multiprocessing and GPUs [87, 88]. FPGAs can also be used internally within a cloud or a data center without being exposed to the clients, such as the ones used in Bing infrastructure to accelerate search ranking [89].

In FaaS, clients develop their FPGA designs locally and use the tools provided by the cloud to compile and remotely configure their designs into a cloud FPGA. Afterward, the cloud FPGA is operated remotely. A cloud FPGA can be shared by multiple clients in a queued manner, so-called *temporal multi-tenancy*, or a parallelized manner, so-called *spatial multi-tenancy*. In the latter case, each client configures only part of the FPGA through partial reconfiguration.

Using a cloud FPGA has advantages in performance, power, and flexibility [90]. In addition, FPGAs were shown to be environmentally friendlier than ASICs for low-volume applications [91]. On the other hand, sharing FPGAs in the cloud comes with additional challenges related to efficiency, scalability, and security. FPGAs are originally built to be owned, used, and maintained by a single party at a time. The rapid popularization of FPGAs in the cloud caught the FPGA vendors off guard, given the long development period and lifecycle of a typical FPGA model. For example, the typical lifecycle of the AMD/Xilinx FPGAs is more than 15 years [92]. As a result, the task of integrating FPGAs into the cloud infrastructure is a very challenging but necessary task borne by the cloud providers.

Cloud FPGAs have vulnerabilities similar to FPGAs used in embedded systems due to the cloud environment being physically accessible to others than the FaaS clients. This requires that clients trust the cloud provider with their confidential information. In addition, the cloud FPGAs have vulnerabilities due to the shared usage of the cloud FPGAs. Accordingly, the cloud provider must ensure isolation between its clients' computations.

5.3 Hardware Trojans

A *hardware Trojan* is a malicious modification to an electronic circuit. The purpose of a Trojan is to bypass a circuit's security by providing the adversary with a secret backdoor. The name hardware Trojan comes from the infamous

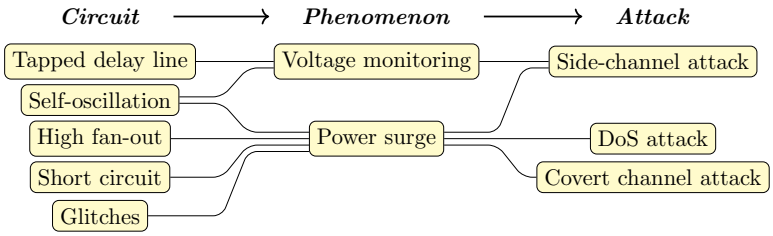


Figure 5.2: The cause and effect diagram from Paper V, which describes types of malicious circuits, the physical phenomena they result in, and the attacks they enable.

Trojan horse, shown in Fig. 5.1, used to breach the city of Troy in Greek mythology. ASICs and FPGAs are particularly vulnerable to hardware Trojans. ASICs typically have highly distributed design and manufacturing processes, making it easier for an adversary to insert a Trojan at a stage within the process that spans from the initial design to the delivery of the final product. Complex FPGA designs, similar to ASICs, may have a distributed design process where multiple parties contribute to a single design. As stated in [93], the FPGA and ASIC industries heavily rely on third-party IP modules. A cloud FPGA may exhibit malicious behavior not only due to a Trojan inserted into an unsuspecting client's design but also due to an adversarial client intentionally configuring the part of the FPGA with a malicious circuit to attempt attacks on the cloud or the other clients of the cloud.

Malicious circuits that are inserted in an FPGA design can enable side-channel attacks, denial-of-service (DoS) attacks, and covert-channel attacks. An adversary can implement a voltage sensor on a cloud FPGA to launch a power side-channel attack on other clients that share the same FPGA [94], die [95], PCB [96], or even another board connected to the same power supply [97]. Circuits that consume a very high amount of power can be used for DoS [98] and fault attacks [99]. Covert-channel attacks establish a secret communication channel between the target device and the adversary, which is used to extract confidential information. Various means of creating covert channels on FPGA have been proposed, including powerline crosstalk [97], interconnect crosstalk [100], residual heat [101], and bus contention [102]. In addition, Papers III and IV present transmitting and receiving antennas, respectively, implemented entirely on the FPGA configurable fabric without using any peripherals. An adversary can insert these antennas in an FPGA design to establish a secret communication channel with the FPGA. Malicious circuits that can be implemented on FPGAs are presented in more detail in Paper V. The types of malicious circuits and the attacks they enable are summarized in Fig. 5.2.

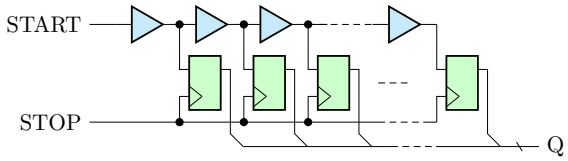


Figure 5.3: A tapped delay line-based TDC from Paper V.

5.4 Time-to-Digital Converters

A *time-to-digital converter* (TDC) is an electronic circuit that measures the delay of an electronic signal by comparing it to a reference signal. They are originally used in applications that require precise delay measurements [103]. Recently, TDCs have also been used in side-channel and covert-channel attacks. It has been shown that a TDC can be used to sense fluctuations in an electronic device's operating voltage, which gives an adversary a means to monitor ongoing operations on a device through powerline crosstalk [94]. In addition, it has been shown in Paper IV that a TDC can sense external EM radiation, enabling an adversary to establish a covert channel with the device having the TDC. Due to their use in various attacks, TDCs are considered as malicious circuits, and it is important to be able to detect them when used out of context, to ensure security.

A popular type of TDC that can be implemented on both ASICs and FPGAs is *tapped delay line-based TDC*, illustrated in Fig. 5.3.

5.5 Transmitting Antenna in Configurable Fabric

Paper III presents several covert transmitting antenna implementations built solely within the FPGA configurable fabric, to investigate the capabilities of an adversary who controls part of an FPGA design. In the paper's experiments, the implemented antennas are used in a wireless communication scenario. The ability to transmit information from an FPGA without any radio-frequency (RF) peripherals, such as an external antenna, ADC, or a resistor-capacitor circuit, is significant because it makes it possible for an adversary to covertly transmit information from an FPGA with a self-contained digital circuit.

Various FPGA resources, namely the interconnect (shown in Fig. 5.4 left), flip-flops, and unused input/output (I/O) pins are attempted to be used as an antenna. In the experiments, the wireless signal that is transmitted from the FPGA is received with a near-field EM probe (shown in Fig. 5.4 right) connected to an oscilloscope.

All implemented antennas enabled an error-free transmission. It has been demonstrated that an interconnect antenna can achieve error-free transmission only with 0.25% of the FPGA's (AMD/Xilinx Artix-7 100t) capacity. Furthermore, it is also shown that such an antenna can coexist with another design that

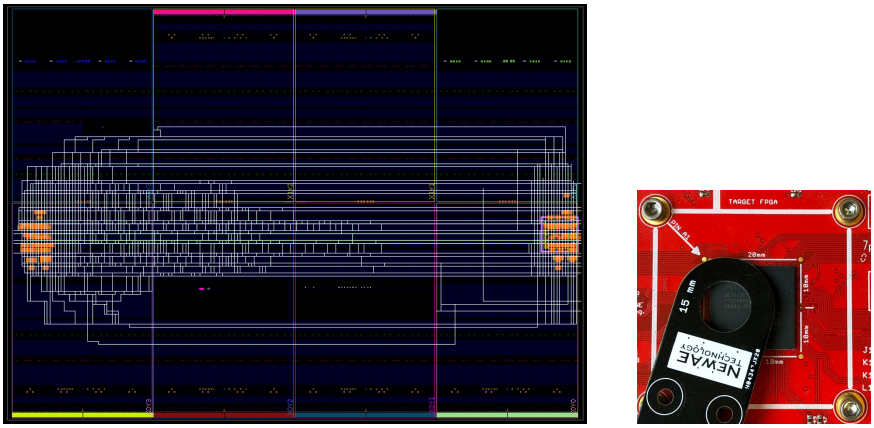


Figure 5.4: Device view of an interconnect antenna on FPGA (left) and the near-field EM probe used to capture the transmitted signal (right).

utilizes 80% of the FPGA's capacity and still deliver a good performance. In a real-world attack, the adversary would need to inject the antenna alongside another design, which is proven possible by the latter result.

The implemented antennas pose threats in FPGA clouds and open radio access networks (ORAN). A covert antenna injected into a cloud FPGA can cause the transmission of confidential information to the physical cloud environment. The ORAN architecture, by design, is open to a multivendor environment, making it vulnerable to malicious components, which can be an IP module, an entire FPGA design, or even an entire device. The results of Paper III indicate the need for a countermeasure against covert transmitters on FPGAs. Accordingly, a routine screening for covert antennas in FPGA designs is suggested.

5.6 Receiving Antenna in Configurable Fabric

Paper IV presents a series of covert EM sensors built solely within the FPGA configurable fabric, to investigate the capabilities of an adversary who controls part of an FPGA design. The sensor design is the first of its kind, presented for the first time in Paper IV. It is also the first time a TDC is used to sense external EM radiation. Being able to receive external EM radiation from the FPGA configurable fabric without any RF peripherals is significant, as in the case of transmitting from the configurable fabric, because it enables an adversary to secretly listen to EM radiation around a remote FPGA via a purely digital circuit.

The design of the sensor is illustrated in Fig. 5.5. The sensor works on the principle that external EM radiation influences FPGA path delays. A path with

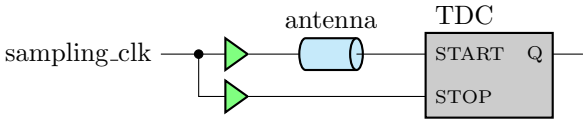


Figure 5.5: The high-level schematic of the EM sensor presented in Paper IV.

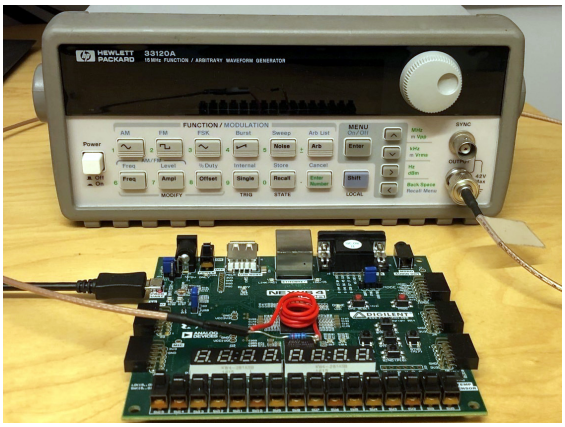


Figure 5.6: Experimental setup in Paper IV. A function generator connected to a hand-made near-field loop antenna, whose signal is received from the FPGA that implements an EM sensor.

a large interconnect delay is used as the receiving antenna. The delay of a digital signal that propagates through the antenna is measured using a TDC.

Paper IV explores the effectiveness and characteristics of the presented EM sensor. The experimental setup includes a function generator that generates a sine wave with varying amplitude and frequency, which is connected to a hand-made near-field loop antenna. The signal transmitted from this antenna is received from an FPGA that contains an EM sensor. The received signal is further processed after being transferred to a computer. The experimental setup of Paper IV is shown in Fig. 5.6.

The results of this paper bring wireless covert channels into attention as a plausible attack vector for cloud FPGAs. An adversarial client may establish a covert channel between his/her FPGA and a target FPGA at a proximity by configuring his/her FPGA with an EM sensor, and by injecting a covert transmitter, such as one that is presented in Paper III, in the target FPGA. Another attack vector is using the EM sensor to launch an EM side-channel attack on another FPGA.

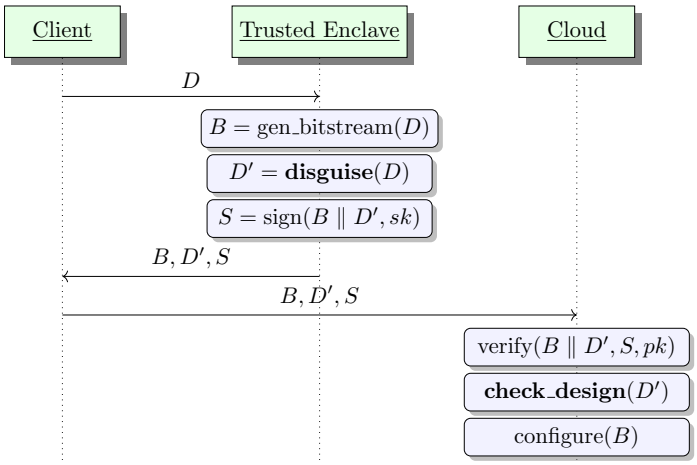


Figure 5.7: Configuration of a cloud FPGA using circuit disguise. D: compiled design netlist, D': disguised design, B: bitstream, S: digital signature, sk and pk: private and public keys of the trusted enclave, respectively.

5.7 FPGA Design Checks and IP Protection

Cloud FPGAs are vulnerable to malicious circuits that can be configured by an adversarial client. Consequently, cloud providers must protect themselves and their clients from malicious circuits. The current practice among major cloud providers is to perform design checks on clients' unprotected designs before bitstream generation [104]. After the design checks, if no malicious circuit is detected, the design is converted to a bitstream with the assistance of the cloud and loaded into an FPGA. The cloud having unrestricted access to the client's designs facilitates the design checks, however, this method comes with an increased risk of IP theft. An FPGA design typically requires considerable resources and engineering effort to develop, and thus, they are valuable IPs. The clients typically are very protective of their FPGA designs, and sharing an unprotected design with the cloud provider increases the risk of IP theft. In short, there exists a trade-off between protection against malicious circuits versus protection of IP.

Alternative methods that aim to protect IP during design checks are to perform the design checks on the bitstream, either by reverse engineering [105] or directly [106]. The former method requires that the bitstream format is known so that the recovery of the design netlist is possible. Unfortunately, the bitstream format is typically obfuscated by the FPGA vendors, requiring bitstream reverse engineering to recover the design netlist [107]. Bitstream reverse engineering is fundamentally unreliable due to its nature of being an attack towards bitstream obfuscation. Although the format is generally reverse-engineered after the release of a new FPGA model, the vendors improve their method of obfuscation

Figure 5.8: An example design of a combinational cycle (left) and its disguised version (right).

iteratively with the newer models. Therefore, relying on bitstream reverse engineering is not a sustainable approach. Furthermore, both former and latter methods require that the bitstream is not encrypted, which is another drawback.

Performing design checks without IP disclosure is an open problem that needs to be solved if cloud FPGAs are to be made secure. Paper V focuses on this problem.

5.8 Circuit Disguise

Paper V presents a novel method, called *circuit disguise*, for detecting malicious circuits in cloud FPGA designs without disclosing the client's intellectual property, a.k.a. the FPGA design, to the cloud. Circuit disguise is based on disclosing a transformed, or *disguised*, version of the design netlist to the cloud, rather than the netlist itself. Necessary design checks can be performed on the disguised version, however, the disguised circuit cannot be reversed to obtain the original netlist. Configuration of a cloud FPGA using circuit disguise is illustrated in Fig. 5.7.

An example of a malicious circuit, a combinational cycle, and its disguised version are shown in 5.8. The disguised circuit does not include details related to the functionality of the circuit, such as truth tables, however, malicious aspects of a circuit, combinational cycles in this example, remain. Paper V also considers various other types of malicious circuits, presented in the paper.

Furthermore, Paper V provides an implementation of circuit disguise that has a polynomial time complexity, and performs experimental evaluation with various benchmark designs.

5.9 Summary

FPGAs that are deployed in embedded systems and clouds are vulnerable to hardware Trojans. Papers III and IV investigate a novel type of hardware Trojan: a covert transmitter and a covert receiver, respectively, both implemented entirely in the FPGA configurable fabric. It is concluded that these Trojans can be used to establish a wireless covert channel, thus they need researchers' and the

industry's attention. Paper V proposes a solution to the problem of performing FPGA design checks in the cloud without the disclosure of the client's IP.

6

Fingerprinting of Neural Networks

This chapter elaborates on IP in machine learning and adversarial machine learning, with a greater focus on model extraction attacks and defenses. It also presents hybrid neural network fingerprinting that combines information domain and physical domain metrics.

6.1 IP in Machine Learning

In machine learning, the most valuable IPs are the parameters of a trained neural network and its training dataset. Collecting and labeling a training dataset, as well as training a neural network, typically require significant engineering effort and computing resources. According to a survey of the machine learning industry,

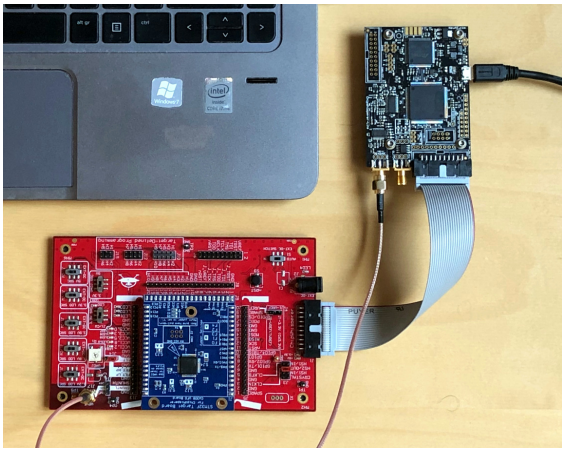


Figure 6.1: Experimental setup for fingerprinting of neural networks using power side-channel information, from Paper VI. A microcontroller board that runs neural network inference (bottom left), a capture board for capturing power traces (top right), and a computer to drive the experiments and to analyze the collected information (top left).

presented in [108], the integrity of the developed neural networks, protection of IP, and privacy of the client's information are the most significant issues that would affect a company's business.

Storing a neural network on the cloud and providing remote access to its users through a prediction application programming interface (API) is a popular deployment method for neural networks. Some of the most popular large language models (LLMs) are deployed on the cloud [7]. Another deployment method is to locally store a copy of a neural network on the device where it is used, such as a mobile phone [8]. In both cases, it is typically desired by the owner of the neural network that the neural network and its training dataset remain secure against theft.

6.2 Adversarial Machine Learning

This section presents a background on the security risks in machine learning, with a focus on model extraction.

The following types of attacks can break the security of a neural network, its training data, or information that belongs to a neural network's user. *Model evasion* aims to harm the integrity of a deployed neural network by misleading it via specifically crafted input data, called *adversarial samples* [109]. Adversarial samples are input data that are classified differently by humans and neural networks. An attacker can use adversarial samples to evade, e.g., face recognition software or spam filters. *Model inversion* refers to extracting information about a neural network's training dataset, thus breaking the training dataset's confidentiality [110]. A training dataset can contain sensitive information about the individuals to whom the data belongs, such as medical information. Furthermore, a compromised prediction API can break the privacy of a neural network's users. When a neural network is deployed in the cloud, the users are given the ability to make queries to the model via a prediction API. The cloud infrastructure that provides the remote API may collect or leak the users' information, thus violating their privacy [111]. *Training data poisoning* is another type of attack that aims to reduce the accuracy of the training of a neural network via injection of malicious samples into the training dataset [112]. For example, the adversary may inject wrongly labeled data into the training set to cause a neural network to mispredict. Finally, *model extraction* refers to the theft of the neural network itself. Model extraction attacks enable an adversary to replicate the functionality of a target neural network without fully going through the costly stages of creating a training dataset and training. Model extraction may lead to unauthorized usage and distribution of a neural network.

Various types of model extraction attacks have been previously demonstrated. They are summarized as follows. A neural network can be stolen via direct access to the file that contains its parameters. For example, through unauthorized access to the cloud that stores the neural network [113], or through the file sys-

tem of a local device that keeps a copy of the network [8]. A neural network that is stolen through direct access is expected to have the same parameters as the original network. The parameters of a neural network can also be extracted via *numerical analysis*. In one such method, the adversary creates a system of equations from queries made to a victim neural network and later approximates the hidden parameters from the system of equations [7]. A neural network that is extracted through numerical analysis typically has parameters equal to or close to the original network. The parameters of a neural network can alternatively be extracted via side-channel analysis. The architecture and parameters of a neural network can leak through side channels during inference in the form of power consumption patterns or EM radiation and, therefore, can be extracted via side-channel analysis [114]. Similar to the case with numerical analysis, a neural network that is cloned via side-channel analysis is expected to have equal or similar parameters compared to the original network. A more flexible model extraction attack is *retraining.* An adversary can train a *surrogate* network from scratch by using the results of queries made to the target neural network as the training dataset of the surrogate network [7]. Retraining can lead to a clone that performs as well as the target model, with a fraction of the cost. The parameters of a surrogate network would typically highly differ from the parameters of the original network, due to the random initialization of parameters before the training of the surrogate network.

According to the industry survey presented in [108], model extraction comes in second place among the attack categories thought to affect the industry the most, the first place being held by data poisoning.

6.3 Defenses against Model Extraction

This section explains defenses against model extraction, with a focus on fingerprinting.

A defense approach is to prevent possible attacks from happening. A measure in this category is encryption, where the neural network is stored in encrypted form and the inference is performed in a trusted execution environment [115]. The network is decrypted before the inference. Alternatively, a homomorphic encryption scheme can be deployed to perform inference directly on the encrypted network without the need for decryption [116]. Thanks to encryption, the meaningful content of a neural network stays hidden from adversaries. Yet another measure is access control, e.g., the protection of a model located in the cloud via a firewall or similar networked systems security mechanisms.

The preventive measures mentioned above will not work on attacks that only require a prediction API, such as retraining attacks. One defense against such attacks is to reduce the dependency of the query outputs on the neural network's parameters, e.g., by adding noise to the network's output [7]. As a result, the adversary would need to make more queries for a successful attack.

Another defense approach is to detect and stop an ongoing attack. A defense in this category is to distinguish queries made by an adversary from queries made by typical users. An adversary who aims to clone a neural network typically makes many, systematically crafted queries. For instance, an adversary may try to find the decision boundaries of the target neural network [117], thus ending up making a large number of queries that are classified with lower confidence. A skilled adversary, on the other hand, can try to evade detection by making normal-looking queries that are still useful for model extraction [118].

Yet another approach is the identification of a stolen model after an attack has been completed, through ownership resolution. Identification of cloned neural networks also protects a neural network's integrity by preventing customers from using a counterfeited neural network that does not perform as well as the original. To this end, two popular types of defenses are *watermarking* and *fingerprinting*.

In watermarking, additional information, called *watermark*, is embedded into the neural network. A watermark is expected to pass on to the clone during an extraction attack. Thus, the existence of the watermark in a suspected neural network indicates that the suspected network is a clone. An example watermarking method is to inject wrongly labeled samples into the training dataset of a neural network [119]. The resulting neural network is trained to make mistakes on a selected set of inputs, therefore, clones that are based on a watermarked model are expected to make the same mistakes that the original model does. A watermark can also be added to the prediction API of a neural network, without modifying the network [120]. Many of the existing watermarking schemes are proven to be ineffective [121].

In fingerprinting, on the other hand, a *fingerprint* is extracted from a given neural network. A clone is expected to have a similar fingerprint to the neural network that it is based on.

Fingerprinting in machine learning has differences from conventional fingerprinting. In a conventional fingerprinting method, e.g., hashing a file, a small modification on the file is expected to generate a different fingerprint. Thus, only two files that are exact copies of each other should generate the same fingerprint. On the other hand, the content of a cloned neural network is not necessarily equal to the neural network that it is based on. Consequently, a fingerprinting method for neural networks should relate a cloned network to the original one, even if the two networks are not the same. How a cloned neural network relates to the original network depends on the nature of the extraction method used. Therefore, a fingerprinting method for neural networks should consider the existing extraction attacks. It is also essential that a fingerprinting method be robust against the existing fingerprint removal attacks, such as the one presented in [122].

In fingerprinting, it is important to correctly identify not only a cloned neural network, a.k.a. *true positives*, but also a suspected neural network that is trained independently of the network claimed to be the original, a.k.a. *true negatives*.

The majority of the existing fingerprinting methods are based on comparing

two sets of outputs, namely the class predictions and confidence values, received from two neural networks queried with the same set of inputs. The result of the comparison is expected to tell how similar the two neural networks are. A high similarity may indicate that one neural network is a clone of the other. Previous work on these methods is presented in Paper VI.

The current state of fingerprinting is considered to be superior to watermarking, as stated in [123]. It should be also noted that fingerprinting research is still premature. The first work that presents a comprehensive review of fingerprinting removal attacks, [122], was published in 2024. The work indicates that much work needs to be done to increase the robustness of the state-of-the-art fingerprinting methods.

There is a fundamental similarity between the existing fingerprinting methods and model extraction attacks. For example, a fingerprinting method may involve making queries to the suspected neural network to extract the decision boundaries [117]. The same approach can be used to clone a model [7]. So, there is a risk that an adversary pretends to be the victim and claims that a neural network was cloned from one of their neural networks, accordingly, launching an extraction attack on the suspected network while pretending to be extracting a fingerprint. For this reason, ownership resolution should be performed by an impartial agency trusted by both the accuser and the suspect.

6.4 Hybrid Fingerprinting

Paper VI presents a neural network fingerprinting method that combines information domain metrics, namely a model's output, with physical domain metrics, namely a set of power traces collected from a model during inference, for more effective detection of neural network cloning. Analysis of the power side channel is typically used by adversaries in attack scenarios, however, in this paper, it is used for the benign purpose of ownership resolution.

The experimental setup, shown in Fig. 6.1, includes a microcontroller board that runs the inference, a capture board that captures the power traces, and a computer to drive the experiments and analyze the collected information.

Model extraction methods including direct access, numerical analysis, side-channel analysis, and retraining have been considered. See section 6.2 for more information on the existing model extraction attacks. Cloning through numerical analysis and side-channel analysis is simulated by adding noise of various intensities to the model parameters. In the case of retraining, actual retraining is performed on a set of neural networks. Every clone is then compared to the respective neural network it is based on according to the following similarity metrics: output predictions alone, probability score vectors (or confidence values) alone, power traces alone, output predictions together with power traces, and score vectors together with power traces.

The key results are as follows. Power traces are as effective as the information domain metrics for a clone that is either an exact copy of the original network or extracted with high accuracy. For clones that are extracted with low accuracy, power traces, and the two hybrid metrics all perform better than the information domain metrics. For retrained clones, the hybrid metrics resulted in only a small improvement over using information domain metrics alone. In addition, it is demonstrated that using randomly generated inputs instead of samples from the testing dataset of the original model further improves effectiveness at detecting models extracted with low accuracy.

To conclude, Paper VI demonstrates that incorporating power side-channel analysis into neural network fingerprinting improves the effectiveness, especially in the case the clone is extracted with low accuracy, possibly through numerical analysis and side-channel analysis.

6.5 Summary

Model extraction is one of the most significant threats to security in machine learning. Neural network fingerprinting is considered a promising defense against model extraction, although it requires further improvement to be widely adopted. Paper VI improves existing fingerprinting methods by performing power side-channel analysis in tandem with the analysis of information domain metrics.

7

Conclusions

This chapter provides a summary of the whole thesis and discusses future work.

7.1 Recap

The two essential parts of security research are attacks and countermeasures. Security research aims to ultimately develop countermeasures to existing attacks. In the meantime, a thorough understanding of possible attacks and testing new countermeasures via known attacks are prerequisites for developing countermeasures.

The main objectives of this thesis are expressed in two research questions, listed in section 1.2. The research questions reflect the attack and countermeasure parts of security research, respectively.

The contributions, results, and impact of this thesis are summarized as follows, in relation to the research questions.

Among the six papers included in this thesis, Papers II, III, and IV present various attacks, thus they are related to the first research question, while Papers I, V, and VI provide countermeasures, thus are related to the second research question.

Secure key generation and storage on an FPGA are necessary to ensure secure communication with the FPGA. However, they are challenging tasks due to the vulnerabilities against existing physical attacks. PUFs are a technology that enables secure key generation and storage with an improved robustness against physical attacks compared to existing methods. Paper I presents an FPGA implementation of a novel arbiter PUF design with a much smaller resource utilization than the existing PUF implementations, making PUFs available for a larger variety of devices.

Side-channel attacks are a type of non-invasive physical attack that threatens the security offered by cryptography. A side-channel attack exploits side-channel information, such as power consumption patterns and EM radiation, to extract secret information from an electronic device. Paper II explores the capabilities and limitations of an adversary who launches a deep learning-based EM side-channel attack on a single-board computer to extract information in memory

instructions. Attack results are successful, meaning that it should be taken into account that information that is written to or read from the main memory is visible from outside the device in the form of EM radiation, and that secret information should not be stored in the main memory in an unprotected form. Furthermore, by focusing on the memory operations, the results of Paper II open the door to the generalization of the existing side-channel attacks.

Malicious modifications of an electronic device's behavior, so-called hardware Trojans, are a vulnerability for FPGAs. Hardware Trojans can lead to various types of physical attacks. The attack surface of an FPGA significantly increases when deployed in an embedded system or the cloud. This is partly due to the physical access to the device being easier and partly to the shared usage of the device. Papers III and IV take part in exploring a covert transmitter and receiver, respectively, implemented purely as digital logic, without using any peripherals dedicated to wireless communication. Such Trojans that can covertly transmit and receive information are proven to be possible, although with limitations, pointing to the need for academia and industry to become aware of the demonstrated risks and to work toward developing countermeasures.

The major FPGA clouds, today, perform design checks on their clients' unprotected FPGA designs for the detection of malicious circuits. This is a big problem given that FPGA designs typically are valuable IPs, accordingly, sharing the designs with the cloud increases the risk of IP theft. The method proposed in Paper V, called circuit disguise, achieves protection of IP while allowing design checks to be performed by the cloud. Circuit disguise is expected to make FPGA clouds more secure, thereby enabling more to benefit from the advantages of cloud computing.

In machine learning, neural network theft is one of the most significant security threats. A neural network that is deployed in the cloud or the end user's device can be stolen via multiple means, including non-invasive physical attacks. Paper VI proposes a hybrid fingerprinting approach for post-attack identification of cloned neural networks. The hybrid approach incorporates power side-channel measurements into information domain metrics used in the majority of previous fingerprinting methods. The hybrid approach improves the existing methods, especially in the case where the parameters of a cloned neural network are extracted with low accuracy.

7.2 Future Work

The future work, generally speaking, should maintain the ongoing race between attacks and countermeasures by exploring new attacks to existing countermeasures, and, in the meanwhile, increasing the robustness of the existing countermeasures and developing new ones. Some of the future work directions are as follows.

The resistance of arbiter PUF, implemented in Paper I, can be evaluated against machine learning-based modeling attacks and power side-channel attacks. Moreover, its uniqueness may be improved through alternative placement and routing strategies. Implementing and evaluating the PUF on newer FPGA architectures should also be considered.

Generalization of power/EM side-channel analysis is one of the main intentions of Paper II. To this end, the work presented in Paper II can be extended to incorporate complexities of general-purpose computers that have not been addressed in the paper, such as complex cache architectures, virtual memory, single-core concurrency, and multi-core parallelism. Side-channel analysis can be improved to extract the instruction memory, rather than just the data memory. The ultimate goal is to investigate side-channel attacks on personal and server computers. This will not only allow us to develop more robust countermeasures but also make side-channel monitoring against malware on general-purpose computers possible.

Papers III and IV present covert wireless transmitters and receivers, respectively, on FPGA configurable fabric. Although Papers III and IV demonstrate the possibility of such transmitters and receivers separately, they have not been used as the two ends of a single communication channel. An interesting future work is to combine Papers III and IV to explore the possibilities of a covert channel between, e.g., two cloud FPGAs in physical proximity. If such a channel is possible, it would give an adversary the possibility to inject a transmitter into a client's FPGA to transmit confidential information, and meanwhile receive the signal from another FPGA located in the same cloud infrastructure, hence achieving a completely remote covert channel in the cloud.

The implementation of circuit disguise, presented in Paper V, is tested on several benchmark designs, however, a case study on a real cloud would be very interesting to do. In addition, the cell libraries in the circuit disguise framework can be extended to support newer AMD/Xilinx architectures and architectures provided by other vendors.

The hybrid fingerprinting method, proposed in Paper VI, can be improved to work across different neural network architectures and scales. Furthermore, robustness against different kinds of fingerprint removal attacks can be evaluated.

References

[1] `https://upload.wikimedia.org/wikipedia/commons/a/aa/Silicon_chip_3d.png`. accessed: Jan. 8, 2025.

[2] Z. Cui, *Nanoscale Pattern Transfer*, pp. 211–266. Boston, MA: Springer US, 2008.

[3] `https://quantumlibraries.com.au/wp-content/uploads/2018/10/EM-and-RFID-Book-Tags-Category.jpg`. accessed: Feb. 19, 2025.

[4] `https://upload.wikimedia.org/wikipedia/commons/1/10/CanakkaleTrojanHorse.jpg`. accessed: Feb. 28, 2025.

[5] D. Chahal, M. Mishra, S. Palepu, and R. Singhal, "Performance and cost comparison of cloud services for deep learning workload," in *Companion of the ACM/SPEC International Conference on Performance Engineering*, ICPE '21, (New York, NY, USA), p. 49–55, Association for Computing Machinery, 2021.

[6] J. W. Dong, X. N. Kang, J. Peng, X. M. Jiang, and T. J. Zhu, "Trust collaboration for edge computing," in *Proceedings of the 2024 14th International Conference on Communication and Network Security*, ICCNS '24, (New York, NY, USA), p. 35–43, Association for Computing Machinery, 2025.

[7] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction APIs," in *25th USENIX Security Symposium (USENIX Security 16)*, (Austin, TX), pp. 601–618, USENIX Association, Aug. 2016.

[8] T. Nayan, Q. Guo, M. A. Duniawi, M. Botacin, S. Uluagac, and R. Sun, "SoK: All you need to know about On-Device ML model extraction - the gap between research and practice," in *33rd USENIX Security Symposium (USENIX Security 24)*, (Philadelphia, PA), pp. 5233–5250, USENIX Association, Aug. 2024.

[9] H. Touqeer, S. Zaman, R. Amin, M. Hussain, F. Al-Turjman, and M. Bilal, "Smart home security: challenges, issues and solutions at different IoT layers," *The Journal of Supercomputing*, vol. 77, no. 12, pp. 14053–14089, 2021.

[10] C. Li, J. Wang, S. Wang, and Y. Zhang, "A review of IoT applications in healthcare," *Neurocomputing*, vol. 565, p. 127017, 2024.

[11] N. Mouha and M. Dworkin, "Review of the Advanced Encryption Standard," NIST Interagency/Internal Report (NISTIR) 8319, National Institute of Standards and Technology, Gaithersburg, MD, July 2021. Language: en.

[12] C. Bouillaguet, A. Fleury, P.-A. Fouque, and P. Kirchner, "We are on the same side. Alternative sieving strategies for the number field sieve," in *Advances in Cryptology – ASIACRYPT 2023* (J. Guo and R. Steinfeld, eds.), (Singapore), pp. 138–166, Springer Nature Singapore, 2023.

[13] C. O'Flynn and Z. David Chen, "Side channel power analysis of an AES-256 bootloader," in *2015 IEEE 28th Canadian Conference on Electrical and Computer Engineering (CCECE)*, pp. 750–755, 2015.

[14] A. Bauer, E. Jaulmes, V. Lomné, E. Prouff, and T. Roche, "Side-channel attack against RSA key generation algorithms," in *Cryptographic Hardware and Embedded Systems – CHES 2014* (L. Batina and M. Robshaw, eds.), (Berlin, Heidelberg), pp. 223–241, Springer Berlin Heidelberg, 2014.

[15] W. Hu, C.-H. Chang, A. Sengupta, S. Bhunia, R. Kastner, and H. Li, "An overview of hardware security and trust: Threats, countermeasures, and design tools," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, no. 6, pp. 1010–1038, 2021.

[16] M. Lipp, M. Schwarz, D. Gruss, T. Prescher, W. Haas, A. Fogh, J. Horn, S. Mangard, P. Kocher, D. Genkin, Y. Yarom, and M. Hamburg, "Meltdown: Reading kernel memory from user space," in *27th USENIX Security Symposium (USENIX Security 18)*, 2018.

[17] P. Kocher, J. Horn, A. Fogh, D. Genkin, D. Gruss, W. Haas, M. Hamburg, M. Lipp, S. Mangard, T. Prescher, M. Schwarz, and Y. Yarom, "Spectre attacks: Exploiting speculative execution," in *40th IEEE Symposium on Security and Privacy (S&P'19)*, 2019.

[18] M. Moraitis, Y. Ji, M. Brisfors, E. Dubrova, N. Lindskog, and H. Englund, "Securing CRYSTALS-Kyber in FPGA using duplication and clock randomization," *IEEE Design & Test*, vol. 41, no. 5, pp. 7–16, 2024.

[19] M. Tehranipoor, N. Nalla Anandakumar, and F. Farahmandi, *Clock Glitch Fault Attack on FSM in AES Controller*, pp. 199–217. Cham: Springer International Publishing, 2023.

[20] Y.-S. Won, S. Chatterjee, D. Jap, A. Basu, and S. Bhasin, "DeepFreeze: Cold boot attacks and high fidelity model recovery on commercial EdgeML device," in *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pp. 1–9, 2021.

[21] H. Pirayesh and H. Zeng, "Jamming attacks and anti-jamming strategies in wireless networks: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 24, no. 2, pp. 767–809, 2022.

[22] P. C. Kocher, "Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems," in *Advances in Cryptology — CRYPTO '96* (N. Koblitz, ed.), (Berlin, Heidelberg), pp. 104–113, Springer Berlin Heidelberg, 1996.

[23] S. P. Skorobogatov, "Semi-invasive attacks – A new approach to hardware security analysis," Tech. Rep. UCAM-CL-TR-630, University of Cambridge, Computer Laboratory, Apr. 2005.

[24] S. Skorobogatov, "How microprobing can attack encrypted memory," in *2017 Euromicro Conference on Digital System Design (DSD)*, pp. 244–251, 2017.

[25] L. Azriel, J. Speith, N. Albartus, R. Ginosar, A. Mendelson, and C. Paar, "A survey of algorithmic methods in IC reverse engineering," *Journal of Cryptographic Engineering*, vol. 11, no. 3, pp. 299–315, 2021.

[26] D. Petryk, Z. Dyka, R. Sorge, J. Schäffner, and P. Langendörfer, "Optical fault injection attacks against radiation-hard shift registers," in *2021 24th Euromicro Conference on Digital System Design (DSD)*, pp. 371–375, 2021.

[27] Z. Chen, G. Vasilakis, K. Murdock, E. Dean, D. Oswald, and F. D. Garcia, "VoltPillager: Hardware-based fault injection attacks against intel SGX enclaves using the SVID voltage scaling interface," in *30th USENIX Security Symposium (USENIX Security 21)*, pp. 699–716, USENIX Association, Aug. 2021.

[28] K. Golshan, *ASIC Design Implementation Process*. Springer, 2024.

[29] J. Xu, "Comparison and application of FPGA and ASIC in digital system design," in *Proceedings of the International Conference on Decision Science & Management*, ICDSM '24, (New York, NY, USA), p. 216–220, Association for Computing Machinery, 2024.

[30] A. Magyari and Y. Chen, "Review of state-of-the-art FPGA applications in IoT networks," *Sensors*, vol. 22, no. 19, 2022.

[31] F. Yan, A. Koch, and O. Sinnen, "A survey on FPGA-based accelerator for ML models," 2024. arXiv:2412.15666. Available at: `https://arxiv.org/abs/2412.15666`.

[32] J.-Y. Kim, "Chapter five - FPGA based neural network accelerators," in *Hardware Accelerator Systems for Artificial Intelligence and Machine Learning* (S. Kim and G. C. Deka, eds.), vol. 122 of *Advances in Computers*, pp. 135–165, Elsevier, 2021.

[33] "UltraScale architecture configuration user guide (UG570)." `https://www.amd.com/content/dam/xilinx/support/documents/user_guides/ug570-ultrascale-configuration.pdf`. accessed: Jan. 7, 2025.

[34] "IEEE standard for VHDL language reference manual," *IEEE Std 1076-2019*, pp. 1–673, 2019.

[35] "IEEE standard for SystemVerilog–Unified hardware design, specification, and verification language," *IEEE Std 1800-2023 (Revision of IEEE Std 1800-2017)*, pp. 1–1354, 2024.

[36] "AMD Vivado design suite." `https://www.amd.com/en/products/software/adaptive-socs-and-fpgas/vivado.html`. accessed: Mar. 22, 2025.

[37] H. Delfs, H. Knebl, and H. Knebl, *Introduction to cryptography*, vol. 2. Springer, 2002.

[38] National Institute of Standards and Technology (NIST), M. J. Dworkin, E. Barker, J. Nechvatal, J. Foti, L. E. Bassham, E. Roback, and J. D. Jr., "Advanced Encryption Standard (AES)," Federal Information Processing Standards

(FIPS), National Institute of Standards and Technology (NIST), Gaithersburg, MD, 2001.

[39] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, p. 120–126, Feb. 1978.

[40] R. L. Rivest, "The MD5 message-digest algorithm," RFC 1321, Internet Engineering Task Force (IETF), 1992. Available at: `https://www.rfc-editor.org/rfc/rfc1321.txt`. Accessed: May. 5, 2025.

[41] National Institute of Standards and Technology (NIST) and Q. Dang, "Secure Hash Standard," Federal Information Processing Standards (FIPS), National Institute of Standards and Technology (NIST), Gaithersburg, MD, 2015.

[42] D. J. Bernstein, "ChaCha, a variant of Salsa20," in *Workshop Record of SASC 2008: The State of the Art of Stream Ciphers*, 2008.

[43] T. Elgamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Transactions on Information Theory*, vol. 31, no. 4, pp. 469–472, 1985.

[44] C. M. Lonvick and T. Ylonen, "The Secure Shell (SSH) Protocol Architecture," RFC 4251, Internet Engineering Task Force (IETF), 2006. Available at: `https://www.rfc-editor.org/info/rfc4251`. Accessed: May 5, 2025.

[45] J. Johnson, "New directions in cryptography," *IEEE transactions on Information Theory*, vol. 22, no. 6, 1976.

[46] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. Mohamed, and H. Arshad, "State-of-the-art in artificial neural network applications: A survey," *Heliyon*, vol. 4, November 2018.

[47] I. El Naqa and M. J. Murphy, *What Are Machine and Deep Learning?*, pp. 3–15. Cham: Springer International Publishing, 2022.

[48] S. Mssassi, A. Abou El Kalam, and Y. Jabrane, "On FPGA security and bitstream reverse engineering," in *Proceeding of the International Conference on Connected Objects and Artificial Intelligence (COCIA2024)* (Y. Mejdoub and A. Elamri, eds.), (Cham), pp. 8–13, Springer Nature Switzerland, 2024.

[49] K. Sathya, J. Premalatha, and V. Rajasekar, "Investigation of strength and security of pseudo random number generators," *IOP Conference Series: Materials Science and Engineering*, vol. 1055, p. 012076, feb 2021.

[50] T. L. Liao, P. Y. Wan, and J.-J. Yan, "Design and synchronization of chaos-based true random number generators and its FPGA implementation," *IEEE Access*, vol. 10, pp. 8279–8286, 2022.

[51] Y. Lu, Y. Yang, R. Hu, H. Liang, M. Yi, H. Zhengfeng, Y. Ma, T. Chen, and L. Yao, "High-efficiency TRNG design based on multi-bit dual-ring oscillator," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 16, Dec. 2023.

[52] F. Frustaci, F. Spagnolo, S. Perri, and P. Corsonello, "A high-speed FPGA-based true random number generator using metastability with clock managers," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 70, no. 2, pp. 756–760, 2023.

[53] H. Lohrke, S. Tajik, T. Krachenfels, C. Boit, and J.-P. Seifert, "Key extraction using thermal laser stimulation: A case study on Xilinx Ultrascale FPGAs," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2018, p. 573–595, Aug. 2018.

[54] M. Garcia-Bosque, G. Díez-Señorans, C. Sánchez-Azqueta, and S. Celma, "FPGA implementation of a new PUF based on galois ring oscillators," in *2021 IEEE 12th Latin America Symposium on Circuits and System (LASCAS)*, pp. 1–4, 2021.

[55] J. Guajardo, S. S. Kumar, G. J. Schrijen, and P. Tuyls, "FPGA intrinsic PUFs and their use for IP protection," in *CHES*, pp. 63–80, 2007.

[56] S. S. Kumar, J. Guajardo, R. Maes, G.-J. Schrijen, and P. Tuyls, "Extended abstract: The butterfly PUF protecting IP on every FPGA," in *2008 IEEE International Workshop on Hardware-Oriented Security and Trust*, pp. 67–70, 2008.

[57] A. Gohr, "Improving attacks on round-reduced Speck32/64 using deep learning," in *Advances in Cryptology – CRYPTO 2019* (A. Boldyreva and D. Micciancio, eds.), (Cham), pp. 150–179, Springer International Publishing, 2019.

[58] B. Hettwer, S. Leger, D. Fennes, S. Gehrer, and T. Güneysu, "Side-channel analysis of the Xilinx Zynq UltraScale+ encryption engine," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2021, p. 279–304, Dec. 2020.

[59] S. Choi, Y. Shin, and H. Yoo, "Analysis of ring-oscillator-based true random number generator on FPGAs," in *2021 International Conference on Electronics, Information, and Communication (ICEIC)*, pp. 1–3, 2021.

[60] S. Kalanadhabhatta, D. Kumar, K. K. Anumandla, S. A. Reddy, and A. Acharyya, "PUF-based secure chaotic random number generator design methodology," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 7, pp. 1740–1744, 2020.

[61] G. E. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," in *Proceedings of the 44th Annual Design Automation Conference*, DAC '07, (New York, NY, USA), p. 9–14, Association for Computing Machinery, 2007.

[62] Y. Liu, Y. Chen, and X. Cui, "A modeling attack on the sub-threshold current array PUF," in *2022 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pp. 169–172, 2022.

[63] N. Wisiol, B. Thapaliya, K. T. Mursi, J.-P. Seifert, and Y. Zhuang, "Neural network modeling attacks on arbiter-PUF-based designs," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 2719–2731, 2022.

[64] Y. Yu, M. Moraitis, and E. Dubrova, "Why deep learning makes it difficult to keep secrets in FPGAs," in *Proceedings of the 2020 Workshop on DYnamic and*

Novel Advances in Machine Learning and Intelligent Cyber Security, DYNAMICS '20, (New York, NY, USA), Association for Computing Machinery, 2022.

[65] U. S. Inan, A. S. Inan, and R. K. Said, *Engineering electromagnetics and waves*. Pearson, 2nd ed ed., 2015.

[66] V. S. Adhin, A. Maliekkal, K. Mukilan, K. Sanjay, R. Chitra, and A. P. James, "Acoustic side channel attack for device identification using deep learning models," in *2021 IEEE International Midwest Symposium on Circuits and Systems (MWSCAS)*, pp. 857–860, 2021.

[67] G. Hospodar, B. Gierlichs, E. De Mulder, *et al.*, "Machine learning in side-channel analysis: a first study," *Journal of Cryptographic Engineering*, vol. 1, pp. 293–302, 2011.

[68] H. Wang, *Deep Learning Side-Channel Attacks on Advanced Encryption Standard*. PhD thesis, KTH, Electronic and embedded systems, 2023. QC 20230213.

[69] G. Camurati, S. Poeplau, M. Muench, T. Hayes, and A. Francillon, "Screaming channels: When electromagnetic side channels meet radio transceivers," in *Proceedings of the 25th ACM conference on Computer and communications security (CCS)*, CCS '18, ACM, October 2018.

[70] R. Wang, H. Wang, and E. Dubrova, "Far field EM side-channel attack on AES using deep learning," in *Proceedings of the 4th ACM Workshop on Attacks and Solutions in Hardware Security*, ASHES'20, (New York, NY, USA), p. 35–44, Association for Computing Machinery, 2020.

[71] NewAE Technology Inc., "CW305 Artix FPGA target." `https://rtfm.newae.com/Targets/CW305%20Artix%20FPGA`. accessed: Feb. 19, 2025.

[72] NewAE Technology Inc., "CW1173 ChipWhisperer-Lite." `https://rtfm.newae.com/Capture/ChipWhisperer-Lite/`. accessed: Feb. 19, 2025.

[73] H. Wang, M. Brisfors, S. Forsmark, and E. Dubrova, "How diversity affects deep-learning side-channel attacks," in *2019 IEEE Nordic Circuits and Systems Conference (NORCAS): NORCHIP and International Symposium of System-on-Chip (SoC)*, pp. 1–7, 2019.

[74] L. Lathrop, "Differential power analysis attacks on different implementations of AES with the ChipWhisperer Nano." Cryptology ePrint Archive, Paper 2020/1008, 2020.

[75] H. Wang and E. Dubrova, "Tandem deep learning side-channel attack on FPGA implementation of AES," *SN Computer Science*, vol. 2, no. 6, p. 373, 2021.

[76] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Advances in Cryptology — CRYPTO' 99* (M. Wiener, ed.), (Berlin, Heidelberg), pp. 388–397, Springer Berlin Heidelberg, 1999.

[77] A. V. Oppenheim, A. S. Willsky, and S. H. Nawab, *Signals & systems*. Pearson Educación, 1997.

[78] A. Golder, D. Das, J. Danial, S. Ghosh, S. Sen, and A. Raychowdhury, "Practical approaches toward deep-learning-based cross-device power side-channel attack," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 12, pp. 2720–2733, 2019.

[79] T. Schneider and A. Moradi, "Leakage assessment methodology," in *Cryptographic Hardware and Embedded Systems – CHES 2015* (T. Güneysu and H. Handschuh, eds.), (Berlin, Heidelberg), pp. 495–513, Springer Berlin Heidelberg, 2015.

[80] E. Prouff and M. Rivain, "Masking against side-channel attacks: A formal security proof," in *Advances in Cryptology – EUROCRYPT 2013* (T. Johansson and P. Q. Nguyen, eds.), (Berlin, Heidelberg), pp. 142–159, Springer Berlin Heidelberg, 2013.

[81] J.-P. Kaps and R. Velegalati, "DPA resistant AES on FPGA using partial DDL," in *2010 18th IEEE Annual International Symposium on Field-Programmable Custom Computing Machines*, pp. 273–280, 2010.

[82] O. Glamočanin, A. Kostić, S. Kostić, and M. Stojilović, "Active wire fences for multitenant FPGAs," in *2023 26th International Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS)*, pp. 13–20, 2023.

[83] B. Hettwer, K. Das, S. Leger, S. Gehrer, and T. Güneysu, "Lightweight side-channel protection using dynamic clock randomization," in *2020 30th International Conference on Field-Programmable Logic and Applications (FPL)*, pp. 200–207, 2020.

[84] "Field programmable gate array (FPGA) market - forecasts from 2023 to 2028." `https://www.researchandmarkets.com/reports/4986720/field-programmable-gate-array-fpga-market`. accessed: Jun. 21, 2023.

[85] C. Ramesh, S. B. Patil, S. N. Dhanuskodi, G. Provelengios, S. Pillement, D. Holcomb, and R. Tessier, "FPGA side channel attacks without physical access," in *2018 IEEE 26th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pp. 45–52, 2018.

[86] F. E. Potestad-Ordóñez, A. Casado-Galán, and E. Tena-Sánchez, "Protecting FPGA-based cryptohardware implementations from fault attacks using ADCs," *Sensors*, vol. 24, no. 5, 2024.

[87] "Amazon EC2 F1 instances." `https://aws.amazon.com/ec2/instance-types/f1/`. accessed: Jun. 21, 2023.

[88] "NP-series." `https://learn.microsoft.com/en-us/azure/virtual-machines/np-series`. accessed: Jun. 21, 2023.

[89] D. Chiou, " The Microsoft Catapult Project ," in *2017 IEEE International Symposium on Workload Characterization (IISWC)*, (Los Alamitos, CA, USA), pp. 124–124, IEEE Computer Society, Oct. 2017.

[90] H. Shahzad, A. Sanaullah, and M. Herbordt, "Survey and future trends for FPGA cloud architectures," in *2021 IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1–10, 2021.

[91] C. Choppali Sudarshan, A. Arora, and V. A. Chhabria, "GreenFPGA: Evaluating FPGAs as environmentally sustainable computing solutions," in *Proceedings of the 61st ACM/IEEE Design Automation Conference*, DAC '24, (New York, NY, USA), Association for Computing Machinery, 2024.

[92] Advanced Micro Devices Inc., "AMD supports new, long lifecycle FPGA designs through 2040, 2045, and beyond." `https://community.amd.com/t5/adaptive-computing/amd-supports-new-long-lifecycle-fpga-designs-through-2040-2045/ba-p/702533`. accessed: Feb. 24, 2025.

[93] T. Hoque, J. Cruz, P. Chakraborty, and S. Bhunia, "Hardware IP trust validation: Learn (the untrustworthy), and verify," in *2018 IEEE International Test Conference (ITC)*, pp. 1–10, 2018.

[94] S. Moini, S. Tian, D. Holcomb, J. Szefer, and R. Tessier, "Power side-channel attacks on BNN accelerators in remote FPGAs," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 11, no. 2, pp. 357–370, 2021.

[95] I. Giechaskiel, K. Rasmussen, and J. Szefer, "Reading between the dies: Cross-SLR covert channels on multi-tenant cloud FPGAs," in *2019 IEEE 37th International Conference on Computer Design (ICCD)*, pp. 1–10, 2019.

[96] F. Schellenberg, D. R. Gnad, A. Moradi, and M. B. Tahoori, "Remote inter-chip power analysis side-channel attacks at board-level," in *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 1–7, 2018.

[97] I. Giechaskiel, K. B. Rasmussen, and J. Szefer, "C^{3}APSULe: Cross-FPGA covert-channel attacks through power supply unit leakage," in *2020 IEEE Symposium on Security and Privacy (SP)*, pp. 1728–1741, 2020.

[98] D. R. E. Gnad, F. Oboril, and M. B. Tahoori, "Voltage drop-based fault attacks on FPGAs using valid bitstreams," in *2017 27th International Conference on Field Programmable Logic and Applications (FPL)*, pp. 1–7, 2017.

[99] J. Krautter, D. R. E. Gnad, and M. B. Tahoori, "FPGAhammer: Remote voltage fault attacks on shared FPGAs, suitable for DFA on AES," *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, vol. 2018, Issue 3, pp. 44–68, 2018.

[100] I. Giechaskiel and J. Szefer, "Information leakage from FPGA routing and logic elements," in *Proceedings of the 39th International Conference on Computer-Aided Design*, ICCAD '20, (New York, NY, USA), Association for Computing Machinery, 2020.

[101] S. Tian and J. Szefer, "Temporal thermal covert channels in cloud FPGAs," in *Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, FPGA '19, (New York, NY, USA), p. 298–303, Association for Computing Machinery, 2019.

[102] I. Giechaskiel, S. Tian, and J. Szefer, "Cross-VM information leaks in FPGA-accelerated cloud environments," in *2021 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pp. 91–101, 2021.

[103] Q. Wingering, R. Fontaine, M.-A. Tétrault, and A. C. Therrien, "FPGA based time-to-digital converter for time of flight computed tomography," in *2024 IEEE Nuclear Science Symposium (NSS), Medical Imaging Conference (MIC) and Room Temperature Semiconductor Detector Conference (RTSD)*, pp. 1–2, 2024.

[104] T. Sugawara, K. Sakiyama, S. Nashimoto, D. Suzuki, and T. Nagatsuka, "Oscillator without a combinatorial loop and its threat to FPGA in data centre," *Electronics Letters*, vol. 55, no. 11, pp. 640–642, 2019.

[105] T. M. La, K. Matas, N. Grunchevski, K. D. Pham, and D. Koch, "FPGADefender: Malicious self-oscillator scanning for Xilinx UltraScale+ FPGAs," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 13, sep 2020.

[106] R. Elnaggar, J. Chaudhuri, R. Karri, and K. Chakrabarty, "Learning malicious circuits in FPGA bitstreams," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 42, no. 3, pp. 726–739, 2023.

[107] M. Moraitis, "FPGA bitstream modification: Attacks and countermeasures," *IEEE Access*, vol. 11, pp. 127931–127955, 2023.

[108] R. S. Siva Kumar, M. Nyström, J. Lambert, A. Marshall, M. Goertzel, A. Comissoneru, M. Swann, and S. Xia, "Adversarial machine learning-industry perspectives," in *2020 IEEE Security and Privacy Workshops (SPW)*, pp. 69–75, 2020.

[109] S. Wang, R. K. L. Ko, G. Bai, N. Dong, T. Choi, and Y. Zhang, "Evasion attack and defense on machine learning models in cyber-physical systems: A survey," *IEEE Communications Surveys & Tutorials*, vol. 26, no. 2, pp. 930–966, 2024.

[110] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, (New York, NY, USA), p. 1322–1333, Association for Computing Machinery, 2015.

[111] J. Liu, M. Juuti, Y. Lu, and N. Asokan, "Oblivious neural network predictions via MiniONN transformations," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, (New York, NY, USA), p. 619–631, Association for Computing Machinery, 2017.

[112] M. Jagielski, A. Oprea, B. Biggio, C. Liu, C. Nita-Rotaru, and B. Li, "Manipulating machine learning: Poisoning attacks and countermeasures for regression learning," in *2018 IEEE Symposium on Security and Privacy (SP)*, pp. 19–35, 2018.

[113] K. Li, "AI model security: Reverse engineering machine learning models." Talk in HITB2018DXB, 2018.

[114] L. Batina, S. Bhasin, D. Jap, and S. Picek, "CSI NN: Reverse engineering of neural network architectures through electromagnetic side channel," in *28th USENIX Security Symposium (USENIX Security 19)*, (Santa Clara, CA), pp. 515–532, USENIX Association, Aug. 2019.

[115] M. F. Babar and M. Hasan, "Trusted deep neural execution—a survey," *IEEE Access*, vol. 11, pp. 45736–45748, 2023.

[116] J.-W. Lee, H. Kang, Y. Lee, W. Choi, J. Eom, M. Deryabin, E. Lee, J. Lee, D. Yoo, Y.-S. Kim, and J.-S. No, "Privacy-preserving machine learning with fully homomorphic encryption for deep neural network," *IEEE Access*, vol. 10, pp. 30039–30054, 2022.

[117] N. Carlini, M. Jagielski, and I. Mironov, "Cryptanalytic extraction of neural network models," in *Advances in Cryptology – CRYPTO 2020* (D. Micciancio and T. Ristenpart, eds.), (Cham), pp. 189–218, Springer International Publishing, 2020.

[118] B. G. Atli, S. Szyller, M. Juuti, S. Marchal, and N. Asokan, "Extraction of complex DNN models: Real threat or boogeyman?," in *Engineering Dependable and Secure Machine Learning Systems* (O. Shehory, E. Farchi, and G. Barash, eds.), (Cham), pp. 42–57, Springer International Publishing, 2020.

[119] Y. Adi, C. Baum, M. Cisse, B. Pinkas, and J. Keshet, "Turning your weakness into a strength: Watermarking deep neural networks by backdooring," in *27th USENIX Security Symposium (USENIX Security 18)*, (Baltimore, MD), pp. 1615–1631, USENIX Association, Aug. 2018.

[120] S. Szyller, B. G. Atli, S. Marchal, and N. Asokan, "DAWN: Dynamic adversarial watermarking of neural networks," in *Proceedings of the 29th ACM International Conference on Multimedia*, MM '21, (New York, NY, USA), p. 4417–4425, Association for Computing Machinery, 2021.

[121] N. Lukas, E. Jiang, X. Li, and F. Kerschbaum, "SoK: How robust is image classification deep neural network watermarking?," in *2022 IEEE Symposium on Security and Privacy (SP)*, pp. 787–804, 2022.

[122] H. Yao, Z. Li, K. Huang, J. Lou, Z. Qin, and K. Ren, "RemovalNet: DNN fingerprint removal attacks," *IEEE Transactions on Dependable and Secure Computing*, vol. 21, no. 4, pp. 2645–2658, 2024.

[123] A. Waheed, V. Duddu, and N. Asokan, "GrOVe: Ownership verification of graph neural networks using embeddings," in *2024 IEEE Symposium on Security and Privacy (SP)*, pp. 2460–2477, 2024.