Side-Channel Analysis with Deep Learning

An Evergrowing Ally in Hardware Security Evaluation

Weissbart, L.J.A.

DOI
10.4233/uuid:2129b2da-7268-4b71-ad46-68defc4d34e0

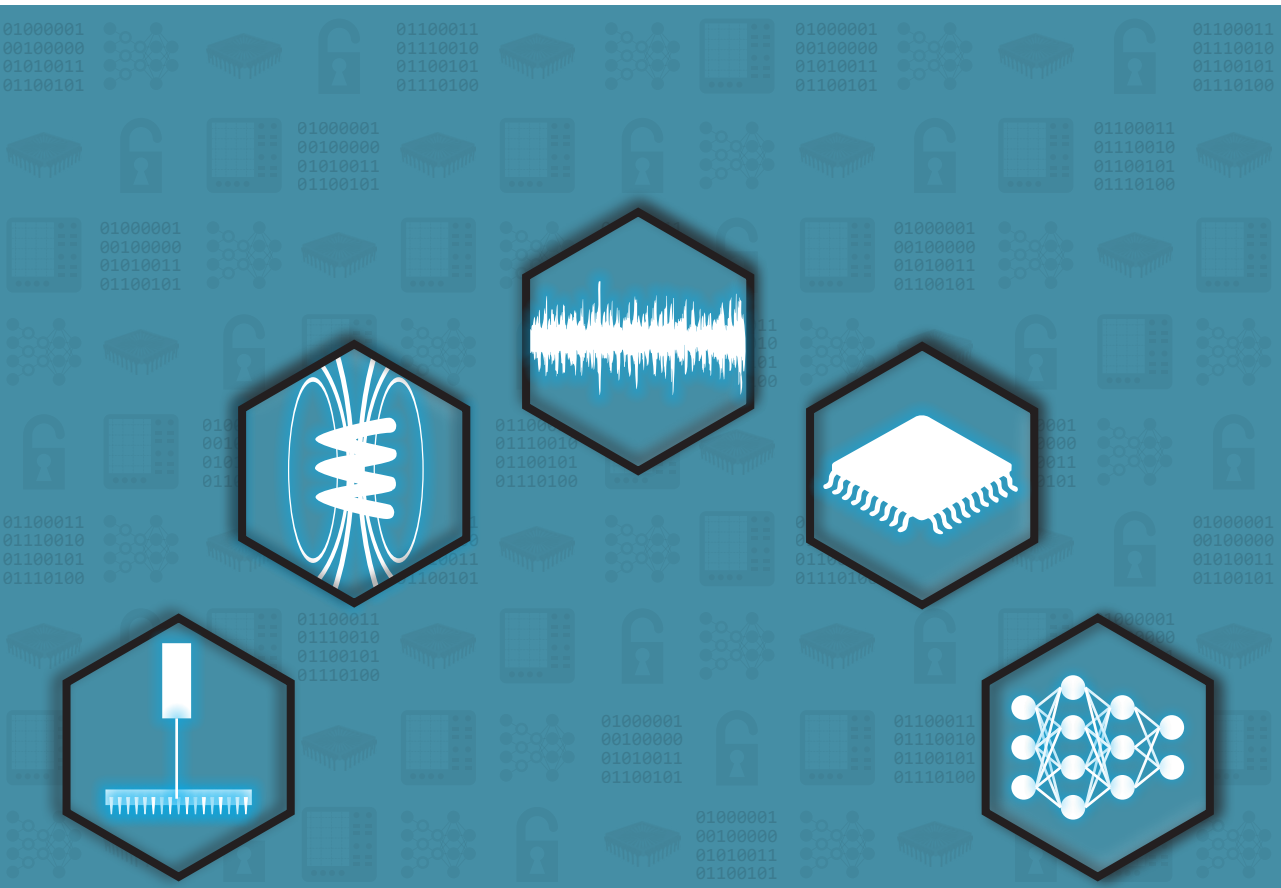
Publication date
2025

Document Version
Final published version

Citation (APA)
Weissbart, L. J. A. (2025). *Side-Channel Analysis with Deep Learning: An Evergrowing Ally in Hardware Security Evaluation*. [Dissertation (TU Delft), Delft University of Technology]. https://doi.org/10.4233/uuid:2129b2da-7268-4b71-ad46-68defc4d34e0

SIDE-CHANNEL ANALYSIS WITH DEEP LEARNING

AN EVERGROWING ALLY IN HARDWARE SECURITY EVALUATION



Léo Weissbart

Propositions

accompanying the dissertation

SIDE-CHANNEL ANALYSIS WITH DEEP LEARNING
AN EVERGROWING ALLY IN HARDWARE SECURITY EVALUATION

by

Léo WEISSBART

1. Cryptographic algorithms will never be invulnerable to side-channel attacks (Chapter 4).
2. Every side-channel analysis method can be outperformed with deep learning features.
3. TEMPEST attacks are going to become the next big thing for hackers.
4. Modern cryptography does not bring online security.
5. Open-source hardware security market expansion calls on a collaboration between academia and industry actors to be successful.
6. Social network is the worst place to promote science.
7. The diversity of open-source frameworks is the single guaranty of reproducible research.
8. An internship is imperative achievement to complete a PhD education.
9. Engineers grow incompetent in a remote working environment.
10. Dutch lunch contains too few nutrients for healthy development of young researchers.

These propositions are regarded as opposable and defendable, and have been approved as such by the promotor Prof.dr.ir. R.L. Lagendijk, promotor Prof.dr. L. Batina, and copromotor Dr. S. Picek.

SIDE-CHANNEL ANALYSIS WITH DEEP LEARNING

AN EVERGROWING ALLY IN HARDWARE SECURITY EVALUATION

SIDE-CHANNEL ANALYSIS WITH DEEP LEARNING

AN EVERGROWING ALLY IN HARDWARE SECURITY EVALUATION

PROEFSCHRIFT

ter verkrijging van der graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus Prof.dr.ir. T.H.J.J. van der Hagen,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op
donderdag 23 oktober 2025 om 15:00 uur.

door

LÉO JOSEPH ALOYSE WEISSBART

Elektrotechnisch ingenieur,
Technische Universiteit Delft, Nederland
geboren te Colmar, Frankrijk.

RECTOR MAGNIFICUS	VOORZITTER
Prof.dr.ir. R.L. Lagendijk	Technische Universiteit Delft, promotor
Prof.dr. L. Batina	Radboud Universiteit, promotor
Dr. S. Picek	Radboud Universiteit, copromotor

ONAFHANKELIJKE LEDEN	
Prof.dr. G. Smaragdakis	Technische Universiteit Delft
Prof.dr. J. van Gemert	Technische Universiteit Delft
Prof.dr.ir. N. Mentens	Universiteit Leiden, Belgium
Prof.dr. E. Dubrova	KTH Royal Institute of Technology, Sweden
Prof.dr. C.M. Jonkers	Technische Universiteit Delft, reservelid

keywords: Side-channel Analysis, Deep Learning, Public-key Cryptography, Symmetric-key Cryptography, Machine Learning, Neural Network Reverse Engineering

ACKNOWLEDGEMENTS

I would like to thank all the people who have accompanied and helped me during this PhD journey.

First, I am deeply grateful to my supervisors, Lejla Batina and Stjepan Picek. They believed in me and were always present to support me in the most important moments. I am very grateful for your guidance and I feel very lucky to had you as my supervisors. I want to thank Inald Lagendijk for his support as my promotor and his valuable feedback on my work.

I thank the reading committee, Jan van Gemert, Georgios Smaragdakis, Nele Mentens, Elena Dubrova and Catholijn Jonker for taking the time to read my thesis, and being part of the defense ceremony.

I am very grateful to the co-authors of the papers presented in this thesis for sharing their knowledge and for the great collaborations. I would like to thank: Łukasz Chmielewski, Zhuoran Liu, Niels Samwel, Zhengyu Zhao, Dirk Lauret, and Martha Larson.

I would like to thank all my colleagues at the Digital Security group at Radboud University for the great time we had together. I am very grateful for the support and the good moments we shared. Especially, I would like to express my gratitude to all who have once shared an office with me: Joost Renes, Ko Stoffelen, Pedro Maat C. Massolino, Niels Samwel, Omid Bazangani, Parisa Amiri Eliasi, Konstantina Miteloudi, Durba Chatterjee, and Silvia Mella.

I want to thank all my friends and colleagues that I met during my time in the Netherlands. I would like to thank: Matthias J. Kannwischer, Thom Wiggers, Joost Rijneveld, Benoit Viguier, Paulus Meessen, Pol Van Aubel, Gabriel Bucur, Louiza Papachristodoulou, Kostas Papagiannopoulis, Ileana Buhan, Unai Rioja, Servio Paguada, Gorka Abad, Lichao Wu, Marina Kreck, Huimin Li, Jing Xu, Luca Mariot, Stefanos Koffas, Hulya Evkan, Xiaoyun Xu, Guilherme Perin, Oğuzan Ersoy, Azade Rezaeezade, Behrad Tajalli, Vahid Jahandideh, Asmita Adhikary, Abraham Basurto, Péter Horváth, Estuardo Alpirez Bock, Christoph Dobraunig, Veelasha Moonsamy, Peter Schwabe, Amber Sprenkels, Yanis Belkheyar, Mario Marhuenda Beltran, Alexandre Bouez, Aldo Gunsing, Jan Schoone, Shahram Rasoolzadeh, Charlotte Lefevre, Solane El Hirch, Jonathan Fuchs, Alireza Mehrdad, Suprita Talnikar, Koustabh Ghosh, Joan Daemen, Bart Mennink, Cristian Daniele, Seyed Behnam Andarzian, Erik Poll, Krijn Reijnders, Lars Ran, Monika Trimoska, Simona Samardjiska, Thijs Heijligenberg, Guido Knips,

Robert Primas, Ján Jančár, Tomáš Balihar, Morten Øygarden, Trevor Yap Hong Eng, Parisa Naseri, Mohanna Hoveyda, Saeid Akbari Bibihayat, Heydar Soudani, Shiva Azizzadeh, Hamid Bostani, Zahra Moti Jeshveghani, Irma Haerkens, Shanley Fijn, Janet Versluys, Tom Janssen-Groesbeek. I am really grateful for all the great moments we shared.

I want to specially thank Ronny and Désirée for being my landlords during all my time in Nijmegen. I am very grateful for your kindness and for the warm environment that you shared in your home.

I am deeply grateful to my friends who supported me in France: Gilles, Célia, Robin, Corentin, Manon, Fiona, Mathieu, Robin, Baptiste, Valentin, Caroline, Camille, Delphine, Jonathan, Julie, Arthur, Laura, Nathan, Régis, Yohan, Thomas, Anirouddh, Cyrille, Germain, Guillaume, Laurie, Hoang, Maxime, Mehdi, Théotime, and Valentin.

Finally, I would like to thank my family for their unconditional support and love. I want to thank my mother and father for always being there for me. Your encouragement and support mean the world to me.

Contents

Part I

INTRODUCTION & PRELIMINARIES

1 INTRODUCTION

This chapter gives a general introduction and motivation of the thesis.

CONTENT OF THIS CHAPTER

1.1 HISTORICAL BACKGROUND

Concealing information to an adversarial party is a very old practice. The act of transforming written messages into a form that is not understandable to an adversary has been known since antiquity [Kah67]. Among the first known methods, people camouflage information in a text by modifying the order of the letters or symbols used (e.g., atbash, Caesar cipher, or Vigenère cipher). Another means of concealing information is by transforming the support of the message, as in the case of the invisible ink, or by wrapping a paper strip around a rod to read the message (i.e., scytale).

All these methods have in common that they are based on the principle of obscurity. The adversary would need to find the concealing method to recover the original message with limited additional effort. While these methods are still used today for games or mere amusement, they cannot be used to secure modern communications because they can easily be defeated by a knowledgeable adversary [GP13]. More advanced methods have been developed to secure communications, and it is still an active field of research to ensure that the security of communication can be guaranteed against any adversary. Cryptographic primitives are algorithms that ensure basic cryptographic properties. These primitives

are often used as building blocks of more complex algorithms, referred to as cryptosystems [MOV96].

The security properties that cryptographic primitives aim to provide are confidentiality, integrity, authenticity, and non-repudiation [MOV96]. Those properties are essential to protecting communications and personal and private data in digital systems and echo individuals' fundamental rights. The confidentiality of the data guarantees that unauthorized parties cannot read the message's content. Integrity of the message guarantees to the receiver that the message was not altered by a third party. The authenticity of the message guarantees to the receiver that the expected sender sent the message. Finally, non-repudiation guarantees that the sender cannot deny having sent the message.

Liberty, safety, property, and resistance to oppression are the founding bases of Human Rights, the foundation of modern society, and the basis of modern democracies. The right to privacy enables freedom of speech, freedom of thought, and freedom of association and is protected in digital communications using cryptography. Threats to this right are an open door to mass surveillance, data collection, and consumer tracking.

While laws exist to govern cybersecurity and data privacy for individuals in front of the justice, cryptography is the tool to protect users' digital data. Encryption has been acknowledged as essential for preserving privacy and security online and safeguarding Human rights [HRAu].

Cryptography goes beyond secure digital communications and expands to protect digital data at rest and digital identities. Among those methods, digital signature can bring authenticity to a message sent with a signature obtained from a private key that anyone can verify with its corresponding public key. Message authentication and integrity checking ensure that a message has not been altered during its transmission. Secure computation can achieve the computation of a function on encrypted data, keeping the data private during the processing of the function [Yao82; SRA81]. Protecting communication on all devices over the internet is a great challenge, as in 2023, the number of digital devices was nearing 16.7 billion and is expected to reach 25 billion by 2026 [Sin23].

1.2 CRYPTOGRAPHY AND CRYPTANALYSIS DUALITY

The study of secure communication methods between a sender and an intended recipient in an adversarial channel is known as *cryptography*, while the study of the means to circumvent the security of a cryptosystem is known as *cryptanalysis* [MOP06]. It is a natural duality to design cryptographic algorithms to bring more security to the used cryptosystems and to prove their security. The analysis

of cryptosystems running on an actual device is known as *side-channel analysis* and aims to break the security of the algorithms by exploiting all the unintended information that can be observed by the physical properties of the device during the execution of the cryptographic functions. Analyzing a cryptosystem's mathematical and physical security is an important step in cryptography, as it ensures the security of existing implementations and helps design future implementations.

Cryptographic schemes commonly have two main categories: symmetric (or secret-key) cryptography and asymmetric (or public-key) cryptography. Secret-key cryptography is commonly used to achieve confidentiality and requires both communicating parties to share a common cryptographic secret (e.g., a generated long and random passkey). One application of secret-key cryptography is the encryption of a message that works with a common shared secret key between the sender and the recipient and ensures that communication is confidential. Algorithms used for this purpose can be block ciphers, stream ciphers, or authenticated encryption schemes, with the most notable examples being the Advanced Encryption Standard (AES) [DR99] and the Data Encryption Standard (DES) [NIS99]. Message authentication codes (MAC) are also typical applications of secret-key cryptography and consist of an added short information to the ciphertext that can be used to verify the integrity of the message by the recipient [Aum17].

On the other side, public-key cryptography is more commonly used to achieve integrity and authenticity, requiring that one party generates a key pair consisting of a private key and a public key. The private key is kept secret, while the public key is shared with the world and can be accessed by any other party. In the context of digital signature, a message is signed by the sender using its private key, and the signature can be verified by any party using the sender's public key. In the context of key establishment, a shared secret key is derived by the two parties by using their respective private key and the public key of the other party.

However, the use of cryptography in a device does not guarantee the security of secured information or personal privacy. The Kerckhoffs's principle states that: *"A cryptosystem should be secure even if everything about the system, except the key, is public knowledge"* [Ker83]. This principle is the basis of modern cryptography, and the reason why the security of a cryptosystem used nowadays relies on the secrecy of the key (and not of the used algorithms). However, the mathematical security of a cryptographic algorithm can have flaws once used in real-world applications. These flaws can be categorized into two types of cryptanalysis: classical cryptanalysis and implementation attacks [Pop09]. Classical cryptanalysis aims to break the mathematical security of a cryptosystem, by exploiting the internal structure of the underlying cryptographic algorithms.

Implementation attacks aim to break the security of a cryptosystem by exploiting its physical implementation. An implementation attack can exploit different physical properties of an implementation and be of a different nature. These attacks are commonly divided into active and passive attacks. Active attacks, such as Fault Injection (FI) attacks, rely on an introduced physical perturbation to the nominal behavior of the implementation to produce a faulty output that can be exploited to recover the secret analytically. In contrast, passive attacks, such as Side-channel Analysis (SCA), rely on the observation of the physical properties of a physical device to recover the secret during a legitimate execution of the implementation. While active attacks require physical access to the device and can sometimes be invasive, passive attacks only require observing the physical properties of the device and can even be performed remotely under some conditions.

1.3 BRINGING ARTIFICIAL INTELLIGENCE TO SIDE-CHANNEL ANALYSIS

Side-channel analysis is based on the measurement of passive physical leakages, like Electromagnetic (EM) emanations or power consumption, during the execution of cryptographic functions. Because a cryptographic implementation runs on an actual hardware device made of electronic components handling bit-logic, it follows the laws of physics of electricity, electromagnetic, and thermodynamics. Thus, it is possible to infer the device's internal state by observing the physical phenomena that result from the computation on the hardware. Exploiting this information for cryptographic implementations falls into SCA.

In the context of SCA, the security analysis of the physical implementation of a cryptosystem is assessed by a human expert who can identify the relevant information in the physical traces using engineering knowledge to test a given implementation for a specific product. The security evaluator's task can be very time-consuming and requires expertise in a specific cryptosystem and target platform. While the specific task of evaluating the security of a given cryptosystem might require different know-hows, the vector of attack in side-channel analysis is conceptually simple. It consists in analyzing a dataset (i.e., traces and corresponding public information) to find a relation to the internal state of the device that relies on the value of a secret.

Exploiting side-channel traces with deep learning methods is a promising approach to automating the analysis of the physical implementation. The reason is that deep learning methods are able to generalize the knowledge from a set of traces and abstract the mathematical relation between the trace features and the sensitive information processed. Such a method provides a possible tool that a

non-expert can use to assert the security of a given implementation. It could also be used during the development of a new product to give guarantees on the security of an implementation before the official standard evaluation by an expert.

The ambition of creating a machine that can think and learn like a human being has stemmed before the invention of the first computers. Augusta Ada Lovelace, the pioneer of computer science and programming, realized that the very first computing machines could be used to do more than just calculations. She reflected on Charles Babbage's work, that the Analytical Engine could be programmed to solve problems of any complexity [FF15]. Ada's vision of computer programming did foresee the potential of advanced computing machines to bring about artificial intelligence and machine learning and imagined the possibilities of countless applications advanced computing machines could bring.

Artificial Intelligence (AI) is an ever-growing field of research and innovations. First, the intention of AI was to solve problems or tasks that are intellectually difficult for humans but conceptually straightforward given a set of rules. Now, the true challenge of AI is to solve problems that are easily solvable by a human but difficult to describe formally. So, instead of describing a formal method to solve such problems, one solution is to allow the machine to learn to solve the problem by itself and provide experiences to let it 'understand' the hierarchy of concepts, leading to a resolution in a divide-and-conquer way. With this method, drawing a graph of the small concepts built on each other that solve a problem would lead to a deep graph with many layers, and for this reason, this method is called Deep Learning (DL).

Deep learning side-channel evaluation is starting to become a required evaluation method for international standardization requirements, such as the Common Criteria (CC) standards [Iso]. For example, the ISO/IEC 17825 requires the evaluators to use any "state-of-the-art" methods appropriate for the product or system being evaluated and includes deep learning side-channel analysis [Iso]. The German Federal Office for Information Security (BSI) has also set guidelines for the evaluation and certification of implementations regarding their side-channel resistance, which includes the evaluation of machine-learning based side-channel attacks [Inf24]. This guide is introduced as a reference guide to be followed in accordance to the common criteria by developers and evaluators of cryptographic implementations for smartcards and similar devices.

1.4 RESEARCH QUESTIONS

This thesis investigates the different aspects of side-channel analysis of cryptographic implementations with deep learning methods and the security aspects of neural networks. The thesis is not about creating new cryptographic schemes or secure implementations. Instead, it concentrates on particular learning algorithms and particular attacks and tries to find the optimal settings for attacking cryptosystems and verifying their security claims. Other algorithms that handle sensitive information and are often protected as valuable Intellectual Property are neural networks and personal display devices of smartphones. This thesis extends the evaluation to enhance attacks in these other privacy-focused applications and uses side-channel analysis to assess their security.

This thesis represents the result of the work conducted to answer the following research questions:

Can model optimization be applied to different AES implementations to efficiently find the best deep-learning attacks?

With the advent of machine learning and deep learning techniques, there is a constant need to evaluate the security of existing cryptographic implementations against these new analysis methods to ensure security against physical attacks. Multilayer perceptrons and convolutional neural networks are the two most common deep learning methods used to achieve good attack results. While their performance is acknowledged, we still lack an understanding of their potential utilization in optimized settings. A systematic evaluation of a given deep learning model, considering all the possible hyperparameters and training considerations, is needed to gain a deeper understanding of the performance of a given algorithm or primitive under attack. Tuning a deep learning model to generalize well on a given dataset is a perpetual challenge. Understanding the process of obtaining such a model can significantly contribute to the design of better models for future attacks. Because exploring every possible configuration and hyperparameter setting is impossible, the exploration should be guided by an algorithm. The most commonly used is grid search, which explores the possible hyperparameters in a pre-configured range by training each model to evaluate its performance. Such a method can provide a good overview of the performance of a given deep learning method but requires a fine-tuned grid to be efficient and find the 'best' model.

Regarding the exploration of the performance of deep learning methods, this thesis clarifies the way for grid search analysis for multilayer perceptron on masked AES implementation to find an efficient yet low parameter count model in Chapter 3.

Can deep learning be used to enhance the performance of SCA on lightweight cryptographic implementations of Ascon?

Significant amount of research has been conducted on the side-channel analysis of implementations of symmetric cryptography. While the use of symmetric ciphers is still prevalent in the industry, concerns about their efficiency in terms of resources and features have led to the development of new algorithms that are fast and lightweight in terms of design and resource used. Ascon is the candidate of the NIST lightweight cryptography competition that have been selected to be standardized for Authenticated Encryption with Associated Data (AEAD). While previous works have shown successful attack on Ascon [SD17], there is a need to evaluate the security of Ascon implementations against DLSCA.

This thesis, in Chapter 4, explores the performance of convolutional neural network in the attack of Ascon AEAD implementation on a 32-bit microcontroller. The results show a successful attack on a reference implementation and a protected implementation.

How far can deep learning-based side-channel analysis improve the performance of side-channel attacks?

While deep learning-based side-channel analysis has shown promising results in previous works, the preciseness of deep learning analysis against different countermeasures and implementations of symmetric and public-key cryptography is still an open question. Moreover, DLSCA enables attacks that could circumvent countermeasures protecting cryptosystems against the state-of-the-art side-channel analysis methods. Protecting a cryptographic implementation against side-channel analysis is a challenging task. Implementers must consider an implementation's weaknesses and applicable attacks to design efficient countermeasures. A common approach is to separate the computation of sensitive information from direct interaction with user inputs. Masking, for example, consists in separating the secret into multiple shares to perform the computation on the shares instead of the secret itself. While this approach is efficient against first-order side-channel attacks, higher-order attacks, while more expensive, can still be effective on weaker implementations. DLSCA also have the advantage to be used for higher-order attacks without exclusive attack considerations.

In the direction of exploring the possibilities of deep learning-based side-channel analysis, we investigate the application of deep learning methods in the attack of different protected implementations. Specifically, we explore the performance of deep learning methods in the attack of the ephemeral key of Ed25519 in Chapter 5 and Chapter 6 and masked implementations of AES in Chapter 3 and Ascon Chapter 4.

How to evaluate the security of devices against TEMPEST attacks in regard to deep-learning methods?

TEMPEST attacks are side-channel attacks exploiting the electromagnetic emanations of a device to recover sensitive information. Declassified documents from the NSA have shown first use of TEMPEST attacks in the 1960s to recover information from a teletype machine [McN]. This type of attack has been later used to recover information displayed on a computer screen from a distance. A limitation of this attack can be the amount of noise in the captured signal that can make the recovery of the information impossible to read. To overcome this limitation, deep learning methods can be used to enhance the recognition of the information in the signal. However, even with of successful recognition by a deep learning model, there is a need for a systematic evaluation of the performance of the attack on different devices and different use cases.

In this thesis, we offer an evaluation testbed to assess the performance of TEMPEST attacks on mobile device displays in Chapter 7. We provide an overview of a deep learning method to enhance the recognition of the information in the signal and evaluate the performance of the attack on different devices and use cases.

Are deep learning implementations secured against side-channel analysis?

An increasing number of applications are using deep learning models as part of their Intellectual Property, and the model architecture and parameters are considered a secret because of the resources needed to obtain the best models for their customers. However, it is unclear if side-channel analysis can threaten the dense and complex implementations of neural networks on hardware.

We propose an analysis of the side-channel leakage of a neural network implementation on a GPU platform in Chapter 8. We evaluate the possibility of recovering the architecture and parameters. The results show that a reference implementation used by standard deep learning libraries can disclose sensitive information about the neural network that an attacker with little knowledge about the model can recover with low effort, including the number of layers and neurons and the type of activation function used. We point out that this information can be used to reverse-engineer the entire model in an advanced black-box attack.

1.5 ORGANIZATION OF THE THESIS AND CONTRIBUTION OF THE AUTHOR

In this thesis, we contribute in deep learning in side-channel analysis. The applications of side-channel analysis can vary between security and privacy concerns. We

explore various problems where deep-learning analysis can be used to enhance the performance of classical analysis methods. In particular, we focus on three main areas of research, which each constitutes a part of the thesis: symmetric cryptography, public-key cryptography, and non-cryptographic applications that deals with privacy.

The content of this thesis is organized in the following parts and chapters:

Chapter 2 provides the necessary background for the rest of the thesis. First, it introduces the concept of side-channel analysis and the different types of side-channel attacks. Second, it provides an overview of machine learning and deep learning methods, and how they can be used in side-channel analysis.

In Part II, we explore the use of deep learning in side-channel analysis of implementations of symmetric cryptography.

Chapter 3 investigates the performance of Multilayer Perceptron (MLP) in profiling attack against a masked AES implementation. We use the ASCAD dataset from [Pro+18], a public dataset of side-channel traces that was introduced together with a successful attack using MLP and CNN. In this work, we perform an exhaustive grid search to visualize the performance of the MLP method with different hyperparameters and compare it to the performance of the best MLP model found in the previous study. The results show that preprocessing of the input features can have a significant impact on the size of a successful model. Furthermore, while behind in performance, MLP represents a simpler and faster-to-train alternative to CNN, with less hyperparameter tuning to find a suitable model.

> The author contributed to the design of the attack and the practical execution of the experiments. This research work has resulted in a paper *"Performance Analysis of Multilayer Perceptron in Profiling Side-channel Analysis"* [Wei] that has been presented at the Artificial Intelligence in Hardware Security (AIHWS) workshop at the 2020 International Conference on Applied Cryptography and Network Security (ACNS).

Chapter 4 investigate the side-channel resistance of Ascon authenticated cipher implementation on a 32-bit microcontroller using Correlation Power Analysis (CPA) and deep learning-based side-channel analysis. This work investigates two possible leakage types susceptible to represent a vulnerability of the S-box operation and considers a masked implementation that stands resistance against both attacks. We demonstrated that in the best results obtained from an attack on a reference and a protected implementation, a partial key can be recovered with only 20 attack traces using a CNN-based attack. Furthermore, we demonstrate the use of a multi-target model for a full-key recovery with 1 000 attack traces on a

reference implementation but remain unsuccessful on the protected implementation.

> The author contributed to the adaptation of the cryptographic implementation to the target board, designed the experiments, and collected the traces. The author also contributed to analyzing the traces and training of the deep learning models. This research has resulted in a paper *"Lightweight but Not Easy: Side-channel Analysis of the Ascon Authenticated Cipher on a 32-bit Microcontroller"* [WP23].

In Part III, we explore the use of deep learning in side-channel analysis of implementations of public-key cryptography.

Chapter 5 proposes a one-trace attack on the ephemeral key on the digital signature algorithm Ed25519 implemented in WolfSSL on an STM32F4 microcontroller. The attack is based on a convolutional neural network, and its result on the collected dataset is compared to the results from other machine learning-based methods, namely Template Attack (TA), Random Forest (RF), and Support Vector Machine (SVM).

> The author contributed both to the design and the practical execution of the attack. This research work has resulted in a paper *"One Trace is All it Takes: Machine Learning-Based Side-Channel Attack on EdDSA"* [WPB] that was presented in 2019 at the International Conference on Security, Privacy and Applied Cryptography Engineering (SPACE).

Chapter 6 extends the previous work and systematizes the use of deep learning for the side-channel analysis of EdDSA. The attack is applied on a different target that implements countermeasures against side-channel analysis. The results are compared to those of the previous work, and the impact of the countermeasure is evaluated. The results show that our deep learning attack is the only method surveyed to recover the secret key with a single trace attack. We also establish that preprocessing the input features, using Principal Component Analysis (PCA), negatively impacts the attack and provide an integrated gradient visualization method that can be used to view the influence of the input features on the predictions of a successful model.

The author contributed to analyzing the publicly available dataset [Chm20] and designing and training the machine learning models. This research work has resulted in the paper: *"Systematic Side-Channel Analysis of Curve25519 with Machine Learning"* [Wei+20b] published in the Journal of Hardware and Systems Security in 2020.

In Part IV, we deal with the use of side-channel analysis and neural networks outside of cryptography, but in the context of security and privacy.

Chapter 7 investigates using deep learning to enhance the TEMPEST attack on mobile device displays. This work establishes a testbed to evaluate the performance of such attacks and provides a broad overview of the use of deep learning to enhance the results from the electromagnetic leakage reconstructed image obtained with the TEMPEST attack. This overview compiles experiments on different devices with two main use cases: reading an eye doctor's letter and recovering a PIN code embedded in a text message. The experiments prove the efficiency of the deep learning methods in recognizing human unreadable digits from the reconstructed images, even when the close-range probe was at a distance from the device.

The author contributed to the elaboration of the testbed, the construction of the experimental setups, and the execution of the experiments. This research work resulted in a paper *"Screen Gleaning: A Screen Reading TEMPEST Attack on Mobile Devices Exploiting an Electromagnetic Side Channel"* [Liu+] presented at the 2021 Network and Distributed System Security (NDSS) symposium.

Chapter 8 presents an evaluation of the leakage resilience of neural networks implemented on a GPU platform against side-channel analysis. The electromagnetic leakage that could be obtained by observing during the execution of the neural network is analyzed to find any unintended information about the neural network's characteristics and parameters. The results show that it is possible to accurately recover the number of neurons and layers of a multilayer perceptron and the type of activation function used.

The author contributed to the execution of the experiments and the analysis of the results. This research work has resulted in a paper *"On Reverse Engineering Neural Network Implementation on GPU"* [CW] that has been presented at the Artificial Intelligence in Hardware Security (AIHWS) workshop at the 2021 International Conference on Applied Cryptography and Network Security (ACNS).

Chapter 9 concludes this thesis by summarizing the main contributions and results and proposing research direction for future works in regard to the output of this thesis.

2 BACKGROUND

This chapter provides general information necessary to understand the context of the thesis. It gives an overview of side-channel analysis with profiled and non-profiled attacks. Next, a general introduction to machine learning with some examples is given, focusing on neural networks and deep learning.

CONTENT OF THIS CHAPTER

2.1 SIDE-CHANNEL ANALYSIS

2.1.1 *Physical Leakage Properties*

Digital circuits consume power by drawing current from a power supply to perform computations. This power consumption is directly related to the number and type of components active in the circuit at a given time. This instantaneous power consumption enables attack vectors in cryptographic devices. Complementary metal-oxide-semiconductor (CMOS) is the most used technology in digital circuits. Any logic can be built with CMOS technology logic cells. The power consumption of a digital circuit with CMOS logic cells is the sum of the power consumption of its logic cells [MOP06; HW10]. Each cell has a static and dynamic power consumption. The static part is due to the leakage current of the transistors, and the dynamic part is due to the switching of the transistors. During switching signals in a circuit, many effects come into play and contribute to power consumption. During a state transition, a transistor charges a load capacitance, and small short-circuits happen because of the imperfect nature of the insulation between the drain and source, and glitches caused by the heterogeneity of signal propagation in the circuit can have a substantial impact on dynamic consumption.

The dynamic consumption contributes the most to the total power consumption, especially for larger chip size technologies.

In a simple model, the instantaneous power consumption of a digital circuit is expressed by the number of transitions of its transistors and is defined as the Hamming Distance (HD) model. The HD model only takes into account the number of transistor transitions. It does consider the transitions from 0 to 1 and from 1 to 0 as equal consumption, disregarding the effect of the signal propagation in the circuit and static consumption. HD model is well known to be well suited to describe the power consumption of data buses in a microcontroller based on Harvard or Von Neumann architectures [RCN02]. Because the data buses are shared between many elements in a microcontroller, they tend to be long and hence have a high capacitive load, producing a high effect of the value of the transferred data on power consumption. However, HD model requires a minimum knowledge of the circuit, or the previous bus state, that sometimes is not known. Hamming Weight (HW) model is simpler by only considering the number of active transistors and only requires one data value to be approximated [Sta10; PSQ07]. Another power consumption model is the Identity (ID) model. This model considers the correspondence of the value of the data being processed with the instantaneous power consumption. Unlike the two previous models, the ID model does not consider the bit representation of the value but the value itself [Bat+11].

2.1.2 *Evaluation Metrics*

First, we will introduce the notations used in the following sections. We denote vectors with bold letters, $\mathbf{a}$. $\mathbb{K}$ is the space of the variable k, and $|\mathbb{K}|$ is the size of the space.

Different metrics have been introduced to measure the performances of a side-channel attack. The most common metrics are the Success Rate (SR) and the Guessing Entropy (GE) [SMY09]. The target of a side-channel attack is to retrieve a secret key k^* (or sometimes part of it) from all possible keys $k \in \mathbb{K}$. A side-channel analysis consists in the interpretation of the results obtained from a collection of challenges to a device under attack. This analysis holds on the sets of input/output data (pt) of the challenges and the corresponding side-channel traces T. All analysis methods boil down to comparing different key candidates k to match a statistical model M using the side-channel samples (T, pt). This comparison creates a probabilistic distribution of the key candidates that can be represented with a guessing vector $\mathbf{g} = [g_1, g_2, \ldots, g_{|\mathbb{K}|}]$ ordered by decreasing probabilities, and the most probable key candidate is the one whose guess value maximizes the probability of the model.

In practice, the guessing vector is obtained from a set of traces $\mathbf{T}$ and corresponding metadata $\mathbf{pt}$. Estimating the minimum number of traces to collect to have a working attack is a crucial question in side-channel analysis. It is usually a challenge to find the point where the attack is guaranteed to succeed with a high probability to obtain the best evaluation [Bat+11; SMY09; Riv08]. In other words, we want to evaluate the number of traces required for the attack to succeed given any other experiment in similar experimental conditions.

In a white-box scenario, the attacker has complete knowledge of the system under attack and can access the code, data, and internal state of the device during the execution of the cryptographic algorithm. In a black-box scenario, the attacker knows nothing about the implementation of the cryptographic algorithm and can only observe the device's physical leakage together with the input/output data of the challenges.

SUCCESS RATE A successful attack is assessed when the recovered key k^* that has the maximum probability from the guessing vector g_k is the correct key: $k^* = argmax_{k \in \mathbb{K}}(\mathbf{g})$. The probability of a successful key retrieval is defined as the Success Rate (SR) and can be expressed as:

$$SR = P_r \left[k^* = argmax_{k \in \mathbb{K}}(\mathbf{g}) \right] \tag{1}$$

GUESSING ENTROPY The Guessing Entropy (GE) is the average number of guesses needed to find the correct key and can be expressed as:

$$GE = E\left(rank_{k^*}(\mathbf{g}) \right) \tag{2}$$

with $rank_{k^*}(\mathbf{g})$ being the position of the correct key k^* in the key guessing vector $\mathbf{g}$, and E is the average of multiple realizations of the key rank, commonly performed by multiple successive attacks using randomly selected traces. The partial guessing entropy is the average position of the correct key in the key guessing vector restricted to a subset of the key (e.g., one key byte).

2.1.3 *Non-profiled Attacks*

In non-profiled attacks, the attacker is assumed to have access to the device under attack and can query the device with chosen plaintexts. The attacker can then collect and analyze the side-channel traces to recover the secret key. Some examples of non-profiled attacks are described in the following paragraphs.

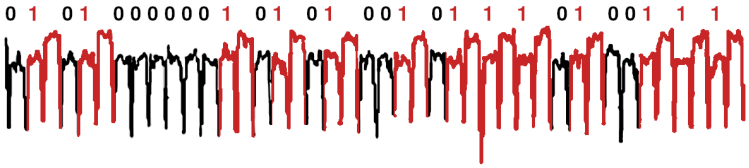


Figure 1: Power consumption traces of the square and multiply operations of RSA, where a key bit of zero is represented by a short pattern and a key bit of one is represented by a long pattern (in red).

SIMPLE POWER ANALYSIS The first attack proposed by Kocher in 1996 [Koc96] is the Simple Power Analysis (SPA). The SPA is a non-profiled attack that exploits the power consumption of a cryptographic device to recover the secret key. This analysis relies on variances in timing or instantaneous power consumption patterns that are visible for a trace of the power consumption and reveals the sequence of instructions executed, and can be used to break cryptographic implementations in which the execution path depends on the data being processed. The SPA is a non-profiled attack because it does not require any knowledge of the target device and can be performed on a single trace. SPA can also be performed on multiple traces of the same operation averaged together to make the leakage visible by increasing the signal-to-noise ratio. The principle of this attack, as proposed by Kocher, is to exploit the timing difference between the square (SQR) and multiply (MUL) operations from RSA during the modular multiplication function. As the sequence of operations between SQR and MUL depends directly on the bit value of the key (i. e.when the key bit is zero, only SQR is performed, and when the key bit is one, SQR is followed by MUL), it is possible to recover each bit by simply looking at the sequence of short and long patterns of the power consumption. The same principle is applied to Elliptic Curve Cryptography (ECC) and its double and addition operations, which follow the same sequence logic.

Simple power analysis can also be used to exploit other non-constant-time implementations of cryptography. Another example of such an attack is the exploitation of memory loads and stores in a software implementation of AES. The substitution operation of AES can be implemented as a lookup table, and the memory access pattern of the lookup table depends on the key value. Observing the memory access pattern makes it possible to recover the key.

DIFFERENTIAL POWER ANALYSIS Differential Power Analysis (DPA) is a type of statistical analysis attack first introduced by Kocher in 1999 [KJJ99]. The DPA relies on the analysis of data-dependent correlations of sets of trace

measurements. The method consists in partitioning a set of traces into subsets, and compute the difference of means between those sets. If the difference of means is significant with increasing number of traces, the choice of which the traces are grouped is correlated to trace measurements.

This method was later improved with the evaluation of different distinguishers for partition-based attacks, such as mutual information analysis [Gie+08], and comparison-based attacks, such as Pearson's correlation coefficient or Bayesian analysis [SGV08].

CORRELATION POWER ANALYSIS Correlation Power Analysis (CPA) is a variation of the DPA introduced by Brier et al. in 2004 [BCO04]. This analysis relies on the comparison of individual traces in the set to a model of the device leakage through a distinguisher. This model can be Hamming Weight, Hamming Distance, or Identity model. The distinguisher of CPA is the Pearson correlation coefficient which measures the linear relationship between the trace samples and the model for a given key value. A greater correlation coefficient between the trace samples and the model determines the most probable key candidate. CPA is considered more reliable attack than DPA, because it is resistant to changes in the mean of traces over the whole traceset, for example due to environmental changes in time during the traceset collection [HGR13].

2.1.4 *Profiling Attack*

In profiling attacks, the attacker is assumed to have access to an identical clone of the device under attack and has complete control over that device. The attacker can then collect and analyze as many side-channel traces as needed to build a model of the leakage of the clone device. In a second phase, the attacker has access to the device under attack and can query the device and collect side-channel traces to recover the secret key using the knowledge built during the profiling phase.

Some examples of profiling attacks are described in the following paragraphs.

TEMPLATE ATTACKS Template Attack (TA) are a type of profiling attack introduced by Chari et al. in 2002 [CRR02], introduced as the strongest statistical attack in side-channel analysis in the information theoretic sense. The principle of template attacks is to build models of an operation on the leakage of a target device, and use these models to categorize traces of a device under attack based on statistical distance to the models, using Bayes' theorem. The templates are built by collecting many traces x of the target device, which can be considered as the realisation of a random variable X. Each trace consisting of F features. The

value of the secret key is known for each trace as a label y, realisation of the random variable Y. For each label's value, a template is built using the average and variance of traces with the same label. The posterior probability for each label value y is given by:

$$P(Y = y|X = x) = \frac{P(X = x|Y = y)P(Y = y)}{P(X = x)} \tag{3}$$

Since X is continuous while Y is discrete, the discrete probability $P(Y = y)$ can be replaced by the sample frequency, where $P(X = x|Y = y)$ represents a density function, and can be assumed to rely on a multivariate Gaussian distribution with mean $\bar{x}_y$ and covariance matrix Σ_y. The previous relation can be written as:

$$P(X = x|Y = y) = \frac{1}{\sqrt{(2\pi)^F|\Sigma_y|}} \exp\left(-\frac{1}{2}(x - \bar{x}_y)^T\Sigma_y^{-1}(x - \bar{x}_y)\right) \tag{4}$$

Pooled templates have been introduced by Choudary and Kuhn in [CK13] because the estimation of the covariance matrices for each class can lead to overfitting when the amount of traces in each class is too low. To overcome this problem, the authors proposed using a pooled covariance matrix for all classes Σ, i.e., using a single covariance matrix formed as a combination of the covariance matrices of each class. The previous equation can be rewritten as:

$$P(X = x|Y = y) = \frac{1}{\sqrt{(2\pi)^F|\Sigma|}} \exp\left(-\frac{1}{2}(x - \bar{x}_y)^T\Sigma^{-1}(x - \bar{x}_y)\right) \tag{5}$$

The templates are then used to categorize traces of the device under attack by calculating the distance between the trace and each template. The highest posterior probability determines the matching label and the most probable key candidate for each trace.

2.1.5 *Countermeasures against Side-channel Analysis*

Countermeasures are the techniques applied to a cryptographic implementation used to reduce the attack surface of known side-channel attacks. The countermeasures can be designed from the algorithmic level down to the physical realizations of the signal emission. For example, shielding can lessen the EM signal accessible by an attacker and consists in adding a metal layer around the source that acts as a Faraday cage to reduce the signal propagation [PSS22]. When shielding is not possible, using jamming noise superposed to the signal can hide that signal,

making it harder to exploit it. Another countermeasure that can be grouped in the same category is the application of random delays to remove correlation of the leaked signal with the timing of the processed algorithm [CK09].

At the algorithmic level, the developed countermeasures mainly aim to remove the dependency of the secret data from the execution path of the algorithm. Against timing attacks, constant-time implementation is the best strategy to prevent an attacker from exploiting the timing difference in the execution of the cryptographic algorithm based on a secret value. For example, the implementation of RSA is commonly prone to timing attacks when the square and multiply algorithm executes a different number of operations based on the value of the key [KJJ99]. A well-known countermeasure is the Montgomery ladder algorithm, which ensures the same number of operations are executed regardless of the value of the handled key [BSS99].

While constant-time implementation is a good coding practice, some operations remain data dependent, such as read operations on fixed pre-computed look-up table on devices with memory cache [Nas+17]. In fact, timing differences can be found by accessing values from the table and revealing which part of the table is accessed. Predictable memory access designs should be preferred to avoid this kind of attack.

Against differential side-channel analysis, some common countermeasures are the use of Boolean masking or threshold implementation.

BOOLEAN MASKING The principle of Boolean masking is to split sensitive values y into a set of shares $y_0, \ldots, y_d$, such that $y = y_1 \oplus \ldots \oplus y_n$, with $\oplus$ the XOR operation. The shares are then processed independently, and the final cryptographic output is obtained by combining the results of the shares. The number of shares n is also called the masking order. This parameter defines the security level of the countermeasure and specifies that no information can be recovered from less than $d < n$ shares. This property is called the d-order probing security [ISW03].

2.2 MACHINE LEARNING AND DEEP LEARNING

2.2.1 *Machine Learning*

Profiling attacks can be seen as a classification problem, and machine learning methods are well suited to solve such problems. Among the different machine learning methods, random forest, support vector machines, and neural networks are particularly promising for side-channel analysis.

Machine learning is a set of statistical methods that can model patterns from data and generalize the analyzed patterns to make predictions on unseen new data [Mur12]. Machine learning algorithms are able to learn a task from given experience data and performance measures [Mit97].

In SCA, data samples are usually a set of vectors of physical measures of a target device during the handling of secret data, e.g., the power consumption during the execution of a cryptographic operation.

To ensure a satisfying result, the data set should be large enough and statistically representative of the task to learn. To monitor the performance of the model during the training phase, a data set is split into three sets: training, validation, and test sets. The *training set* contains the data on which the model will learn the patterns and train its parameters. The *validation set* is used to evaluate the model's ability to generalize to unseen data during the training phase. The *test set* is used on the final model to evaluate whether the model is unbiased. All sets must be disjoint but ideally have the same statistical distribution. These sets are often created by randomly splitting a common data set with a typical train/test-split ratio of 80% to 20%, and the validation set is a subset of the training set.

A commonly used method is the *k-fold cross-validation*, where the training set is randomly split into disjoint k subsets (i.e., folds) that are used iteratively to train the model on $k-1$ folds and test it on the remaining fold [AH98]. The average error of the model across all trails is the cross-validation error. This method provides more certainty on the model generalization when the test set is small.

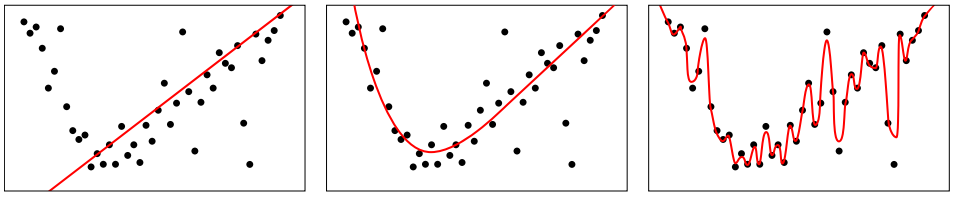


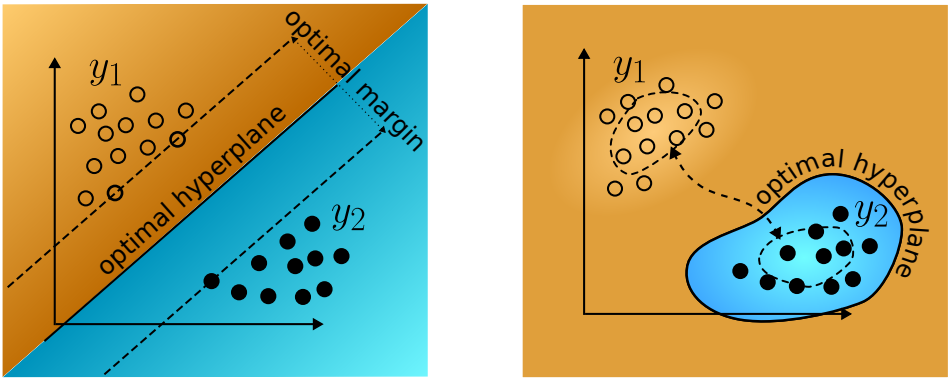
Figure 2: Examples of different learning distributions, with under fitting (left), proper fitting (middle), and over fitting (right). The black points are the training data samples, and the red lines represents the approximated distribution of the ML model.

Two behaviors should be avoided during model training: underfitting happens when the model cannot capture the patterns in the training data, and the model cannot obtain sufficiently low error on the training and test sets. Overfitting happens when the model captures more than the patterns in the training data, performing well on the training set but poorly on the test set. These two behaviors are illustrated in Figure 2.

The following paragraphs give a brief overview of the most important machine learning methods used in SCA.

RANDOM FOREST Random Forest (RF) is a construction of multiple decision trees, where each tree is built using a random subset of the training data and a random subset of the features [Bre01]. A tree is a combination of Boolean decisions on the features, and the output of the tree is a label. In the training phase, the decision rules in each node are tuned to minimize the error rate of the random forest for the training set.

SUPPORT VECTOR MACHINES Support Vector Machine (SVM) represents a family of kernel-based machine learning methods that can be used to classify both linearly separable and linearly inseparable data [CV95]. SVM classifiers are also known as maximum-margin classifiers, as this method transforms each data point (i.e., traces in a traceset) into a higher dimensional space and creates optimal hyperplanes between data points belonging to different classes, as shown in the binary example in Figure 3.



(a) Linear kernel (b) RBF kernel

Figure 3: Binary classification using a SVM

For linearly separable problems, the linear kernel is given by

$$\mathbf{w}^T\mathbf{x} + b, \tag{6}$$

where $\mathbf{w}$ is denoted as the weight vector and b as the bias.

The Radial Basis Function (RBF) is a kernel trick applied as a separation function for SVM [ABR64; BGV92]. This function measures the distance between

data points in infinite dimensions and then estimates a classification by a majority vote. The kernel function is given by:

$$\exp(-\gamma\|\mathbf{w}-\mathbf{w}'\|^2), \tag{7}$$

where $\gamma > 0$ is the kernel coefficient.

The training function used for SVM is the Sequential Minimal Optimization (SMO) algorithm [Pla98; Kee+01], which iterates the margin optimization between classes using the hinge loss [CS01].

For better results, regularization is applied on the input, and its parameter C, which is inversely proportional to the hyperplane's margin, reduces the model's variance and influences the misclassification tolerance.

GRADIENT BOOSTING Gradient boosting for classification is an algorithm that trains several weak learners (i.e., decision trees that perform poorly considering the classification problem) and combines their predictions to make one stronger learner. Gradient boosting differs from the random forest in building the decision trees. While in a random forest classifier, each tree is trained independently using random data samples, decision trees in gradient boosting depend on the previously trained tree's prediction to correct its errors. Gradient tree boosting is composed of a concatenation of several smaller decision trees. We used the eXtreme Gradient Boosting (XGBoost) (or XGB) implementation of gradient boosting, designed by Chen and Guestrin [CG16], which uses a sparsity aware algorithm for handling sparse data and a theoretically justified weighted quantile sketch for approximate learning.

NAIVE BAYES Gaussian Naive Bayes (NB) classifier is one of the classification algorithms that applies Bayes's theorem with the "naive" assumption. The naive assumption describes the conditional independence between every pair of features in a given class sample. The Gaussian assumption is assumed to be the feature probability distribution. The Naive Bayes method is highly scalable with the number of features and requires only a few representative features per class to achieve a satisfying performance.

2.2.2 *Deep Learning*

Among machine learning methods, methods based on neural networks are categorized in deep learning. Artificial Neural Network (ANN) are a distinct class of algorithms that rely on mathematical models of small units processing a set of

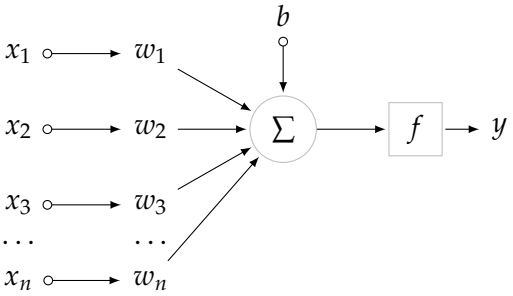


Figure 4: Anatomy of a neuron.

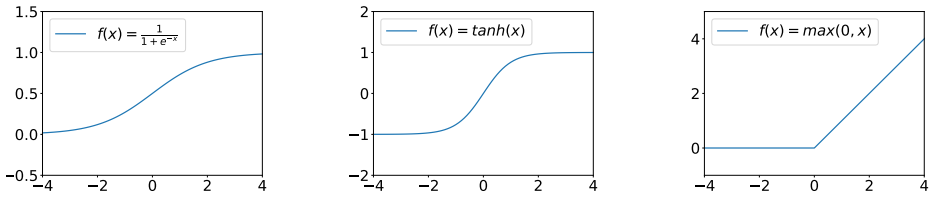
inputs in coordination with each other in a network manner to produce an output. Because of the resemblance of many operators receiving vector inputs from other similar operators and producing a scalar output with its own activation rule to the biological science about neurons, these units are called neurons. The later research on neural networks is, however, driven by mathematical and engineering disciplines and has no ambition to reproduce the workings of the human brain.

PERCEPTRON One type of ANN is the perceptron, often described as the smallest unit of ANN. The perceptron (a.k.a. neuron or node) is a mathematical function that maps a set of inputs to a single real-valued output through a weighted sum and an activation function. The function is represented in Figure 4 and is defined as:

$$y = f(\sum_{i=1}^{n} w_i x_i + b), \tag{8}$$

with x_i the input vector, w_i the weights, b the bias, and f the activation function.

ACTIVATION FUNCTIONS Activation functions are used to introduce non-linearity in the output of a neural network layer. The non-linearity is an important property that enables the network to learn complex patterns by ignoring the contribution of some neurons in the network. The application of the activation function can also range the output of a layer for normalization purposes to prevent the output from exploding or vanishing. The most common activation functions are the sigmoid function, the hyperbolic tangent function (TanH), and the rectified linear unit function (ReLU). These functions are illustrated in Figure 5.



(a) Sigmoid (b) TanH (c) ReLU

Figure 5: Activation functions for neural networks

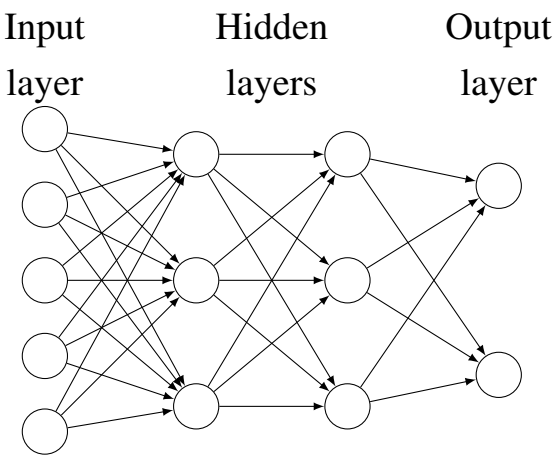


Figure 6: Schematic of a Multilayer perceptron

MULTILAYER PERCEPTRON Ordinary ANN is the Multilayer Perceptron (MLP). It consists of several layers of neurons that form a graph where each layer is fully connected to the next layer, as shown in Figure 6.

The first layer is the input layer, the last layer is the output layer, and the layers between are called hidden layers. This model is also called a feedforward neural network, because information flows from the input layer to the output layer without any feedback loop, in contrast to recurrent neural networks. Deep learning is a terminology taken from the depth of the overall succession of layers in the network. Commonly, a network with more than one hidden layers is considered a deep network.

Each unit in a layer acts in parallel to other units, each representing a vector-to-scalar function, and the output of the layer is a vector of the outputs of each unit. In each unit, when the output is different from zero, we say that the neuron activation feeds the next layer as its input. The number of layers and the number

of units in each layer are adjusted to the complexity of the task to fit the training data. Too many layers or units can lead to overfitting, and too few can lead to underfitting.

CONVOLUTIONAL NEURAL NETWORKS Layers with a convolution function as the network input function are called convolutional layers and are the core building blocks in Convolutional Neural Network (CNN). CNN have been introduced by LeCun et al. in [LeC+90]. Their use is particularly suited for image recognition but also for speech and time series data. CNNs are a great tool to handle large input data by dividing into smaller task-specific mechanisms into their architecture to reduce the overall number of parameters of the model while keeping the ability to learn complex patterns.

CNNs are commonly composed of three main types of layers: convolutional layers, pooling layers, and fully-connected layers.

CONVOLUTION LAYER computes the output of neurons from locally sparse combinations of initial raw input features to reduce the space volume of information into smaller regions of interest. This layer extracts information from small spatial regions of the input data with kernels (also called filters). The result of the convolution is a feature map that is passed to the next layer. It can be mathematically expressed as the dot product of the input data and the kernel, as shown in Equation 9, and illustrated for a 2D convolution example in Figure 7.

$$y_{\mathbf{k}',f} = \mathbf{x}_{\mathbf{k}'} * \mathbf{w}_f = \sum_{k_1,\ldots,k_N}^{r_i-1} x_{\mathbf{k}'+\mathbf{k}} \times w_{\mathbf{k},f} \tag{9}$$

with $\mathbf{k}'$ the coordinates of the output scalar, $x_{\mathbf{k}}$ the input, $\mathbf{w}_f$ a kernel of dimension N and sizes $(\mathbf{r_i})_{i\in N}$ and filter index f.

When the convolution is applied only without padding (as in the example), we call it a *valid* convolution. When the convolution is applied with zero-padding to keep the output size the same as the input size, we call it a *same* convolution. Moreover, the sliding of the kernel over the input data is called the *stride* of the convolution and represents the number of indices to slide for the next computation (in the example, the stride is 1).

POOLING LAYERS are used after a convolution layer to sample down local regions and create spatial regions of interest. The method is called max-pooling, when the maximum value of the region is taken, and average pooling when the

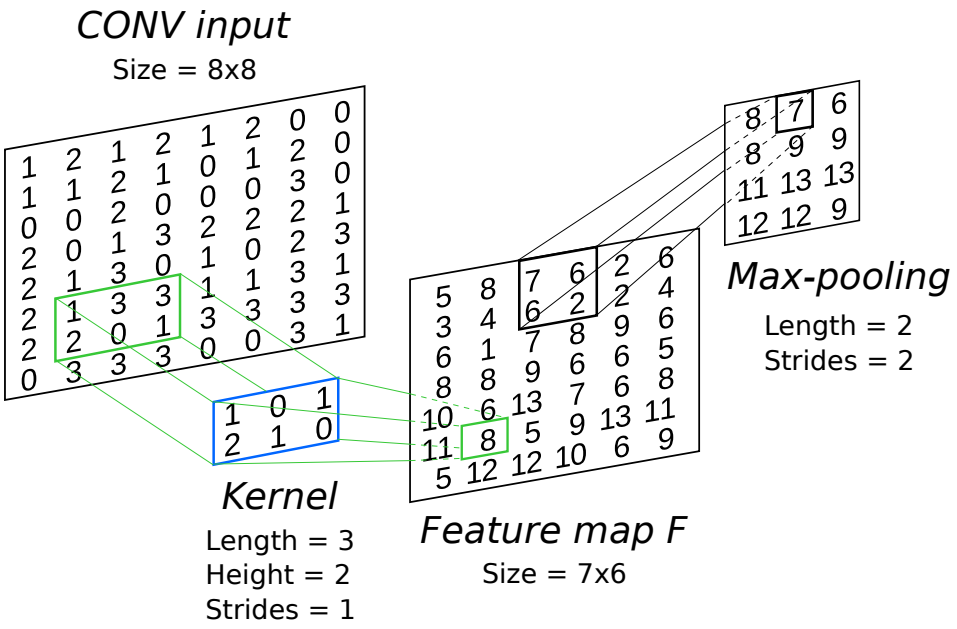


Figure 7: Schematic of a 2D convolutional layer followed by a max-pooling layer.

average value is taken. Pooling layers are important for compressing the output of convolution layers and enabling the network to be invariant to small translations of the input data (e.g., misalignment or temporal deformations). An example of the max-pooling operation is shown in Figure 7.

A fully connected layer at the end of a CNN behaves as a classifier for the extracted feature map.

MODEL TRAINING Training an ANN is an iterative and multi-step process that requires the optimization of the model parameters (i.e., weights and biases) to minimize the error of a dedicated loss function (e.g., the categorical cross-entropy loss in a classification problem) between the labeled output and model prediction.

Gradient descent search is an optimization method used to find a local minimum of a differentiable multivariate function. This iterative method starts with an arbitrary initial weight vector and then modifies it in small steps in the direction that produces the steepest descent along the error surface [Mit97]. Its stochastic application to the neural network is known as Stochastic Gradient Descent (SGD), and is derived in different variations of this optimizer, such as Root Mean Square Propagation (RMSProp) [MSS12], Adaptive Gradient Algorithm (AdaGrad) [DHS11] and Adaptive Moment Estimation (Adam) [KB15].

The hyperparameters are the parameters that define the architecture of a neural network. Examples are the number of hidden layers, the number of neurons per layer, or the type of activation function used in the neurons. According to the *no free lunch theorem* proposed in [Wol96], no machine learning algorithm is universally any better than any other. Thus, to obtain the best results from a network, it should be optimized for the given task to solve. Several methods exist to find the optimal hyperparameters, such as random search [BB12], Bayesian optimization [Tur+21; SLA12; HHL11; JSW98], evolutionary algorithms [BS02], reinforcement learning [LM17], or gradient-based methods [EMH18].

Part II

DEEP LEARNING SIDE-CHANNEL ANALYSIS OF SYMMETRIC CRYPTOGRAPHY

3

PERFORMANCE ANALYSIS OF MULTILAYER PERCEPTRON IN PROFILING SIDE-CHANNEL ANALYSIS

This chapter is based on [Wei], a single author paper, that has been presented during the Artificial Intelligence in Hardware Security (AIHWS) in 2020.

CONTENT OF THIS CHAPTER

3.1 INTRODUCTION

Side-channel analysis (SCA) exploits weaknesses in cryptographic algorithms' physical implementations rather than the algorithms' mathematical properties [MOP06]. There, SCA correlates secret information with unintentional leakages like timing [Koc96], power dissipation [KJJ99], and electromagnetic (EM) radiation [QS01a]. One standard division of SCA is into non-profiling (direct) attacks and profiling (two-stage) attacks. Profiling SCA is the worst-case security analysis as it considers the most powerful side-channel attacker with access to a clone device (where keys can be chosen and known by the attacker). During the past few years, numerous works showed the potential and strength of machine learning in profiling side-channel analysis. Across various targets and scenarios, researchers were able to show that machine learning can outperform other techniques considered state-of-the-art in the SCA community [CDP17; MPP16]. More interestingly, some machine learning techniques are successful, even on implementations protected with countermeasures [CDP17; Kim+19b]. There, in the spotlight are techniques from the neural network family, most notably, multilayer perceptron (MLP) and convolutional neural networks (CNNs).

When considering the attack success, we commonly take into account only the performance as measured by the number of traces needed to obtain the key. While this is an important criterion, it should not be the only one. For instance, attack complexity (complexity of tuning and training a model) and interpretability of the attack are also essential but much less researched. For instance, CNNs are often showed to perform better than MLPs in SCA's context [CDP17; MPP16; Pic+19], as they make the training of a model more versatile and alleviate the feature engineering process. On the other hand, MLP has a more straightforward structure and is probably easier to understand than CNNs, but still, the performance of MLP for SCA raises less attention. Consequently, this raises an interesting dilemma: do we consider profiling SCA as a single-objective problem where the attack performance is the only criterion, or should it be a multi-objective problem where one considers several aspects of "success"? We believe the proper approach is the second one as, without a better understanding of attacks, we cannot make better countermeasures, which is an integral part of the profiling SCA research.

In this paper, we experimentally investigate the performance of MLP when applied to real-world implementations protected with countermeasures and explore the sensitivity of the hyperparameter tuning of a successful MLP architecture. We emphasize that this work does not aim to compare the performance of different techniques, but rather to explore the multilayer perceptron's capabilities. To achieve this, we use two datasets containing different AES implementations protected with random delay countermeasure and masking countermeasure. Our results show that we require larger architectures only if we have enough high-quality data. Hence, one can (to a certain degree) overcome the limitation in the number of hidden layers by providing more perceptrons per layer or vice versa. Finally, while our experiments clearly show the difference in the performance concerning the choice of hyperparameters, we do not notice that MLP is overly sensitive to that choice. This MLP "stability" means it is possible to conduct a relatively short tuning phase and still expect not to miss a hyperparameter combination yielding high performance.

3.1.1 *Datasets*

We consider two datasets presented in previous researches and that we denote as ASCAD and AES_RD. Both datasets are protected with countermeasures: the first one with masking and the second one with the random delay interrupts.

The ASCAD dataset, introduced in the work of Prouff et al. [Pro+18], consists of electromagnetic emanations (EM) measurements from a software implementation of AES-128 protected with first-order Boolean masking running on an 8-bit

AVR microcontroller (ATMega8515). This dataset counts 60 000 traces of 700 samples each and targets the third byte of the key. The SNR for this dataset is around 0.8 if the mask is known and 0 if it is unknown. The trace set is publicly available at https://github.com/ANSSI-FR/ASCAD/tree/master/ATMEGA_AES_v1/ATM_AES_v1_fixed_key.

The AES_RD dataset, introduced in the work of Coron and Kizhvatov [CK09], consists of power traces from a software implementation of AES-128 protected with random delayed interruptions running on an 8-bit AVR microcontroller (ATmega16). This dataset has 50 000 traces with 3 500 samples each, and targets the first byte of the key. The SNR has a maximum value of 0.0556. The trace set is publicly available at https://github.com/ikizhvatov/randomdelays-traces.

3.2 RELATED WORK

The corpus of works on machine learning and SCA so far is substantial, so we concentrate only on works considering multilayer perceptron. Yang et al. considered neural networks and backpropagation as a setting for profiling SCA [Yan+12]. They indicated that "...neural network based power leakage characterization attack can largely improve the effectiveness of the attacks, regardless of the impact of noise and the limited number of power traces.". Zeman and Martinasek investigated MLP for profiling SCA where they mentioned the machine learning algorithm simply as "neural network" [MZ13]. They considered an architecture with only a single hidden layer and experimented with several possible numbers of neurons in that layer. Finally, they only considered a sigmoid for the activation function. After those, there have been several papers using MLP with good results, but usually comparable with other machine learning techniques [GHO15; Heu+16; MHM14]. Still, the hyperparameter tuning was often not sufficiently explored. Despite our attempts, we could not confirm the first paper using MLP in a deep learning paradigm, i.e., with more than a single hidden layer. Interestingly, first papers with MLP were often not clear on the number of layers, as the tuning phase played an even smaller role than today.

In 2016, Maghrebi et al. conducted the first experiments with convolutional neural networks for SCA, and they compared their performance with several other techniques (including MLP) [MPP16]. Their results indicated that, while MLP is powerful, CNNs can perform significantly better. From that moment on, we observe a number of papers where various deep learning techniques have been considered in comparison with MLP, see, e.g., [HGG18; Pic+18a; Pic+19; Pic+18b].

Pfeifer and Haddad considered how to make additional types of layers for MLP to improve the performance of profiling SCA [PH18]. B. Timon investigated the "non-profiled" deep learning paradigm, where he first obtained the measurements in a non-profiled way, which are then fed into MLP or CNN [Tim19]. Interestingly, the author reported better results with MLP than CNNs. Finally, Picek et al. connected the Universal Approximation Theorem and performance of the side-channel attack, where they stated that if the attacker has unlimited power (as it is usually considered), most of the MLP-based attacks could (in theory) succeed in breaking implementation with only a single measurement in the attack phase [PHG19].

3.3 EXPERIMENTAL SETUP

In this section, we present our strategy to evaluate and compare the performance of the different MLP attacks on different datasets. We want to observe the influence of the choice of leakage model, information reduction, and major hyperparameters defining an MLP (i.e., number of layers, number of perceptrons per layers, and activation function).

We provide results with power leakage models of both the S-box output (intermediate value model) and the Hamming Weight (HW) representation of the S-box output.

Besides considering the raw traces (i.e., no pre-processing and feature engineering), we apply the Difference-of-Means (DoM) feature selection method [MOP06]. DoM method selects the samples of a dataset that have the highest variance for a given leakage model. Even though selecting features with high variance is likely to preserve the information about the leakage, it is better to select a number of features with different variance since the features containing the leakage are not always the features with the highest or the lowest variance.

To compare the hyperparameters' influence, we conduct a grid search for hyperparameter optimization and consider each resulting model as a profiling model for an attack. Considering the MLP hyperparameters, we fix some parameters (i.e., number of training epochs and learning rate) and explore the influence of the three following hyperparameters:

- The number of perceptrons, with a fixed number of layers.
- The number of layers, with a fixed number of perceptrons.
- The activation function used for the perceptrons in the hidden layers.

In Table 1, we list all the explored hyperparameters. The total number of models trained per experiment is of $n_{act} * n_l * n_p = 2 * 6 * 10 = 120$, where n_{act}, n_l, n_p, represent the number of activation functions, layers and perceptrons per

layers explored respectively. We run our experiments with Keras [Cho+15], and we use 200 epochs for the training phase, with a learning rate of 0.001. To assess the performance of a profiling model for an attack, we use the guessing entropy (GE) metric [SMY09]. GE defines the average rank position of the correct key candidate in the guessing vector. In other words, when considering N attack traces, each of which results in a guessing vector $\mathbf{g} = [g_0, g_1, \ldots, g_{|K-1|}]$ containing the probabilities of each key candidates in the keyspace K, ordered by decreasing probability. For all experiments, when computing GE, we use the generalized guessing entropy introduced in [Wu+20]. GE equal to 0 means that the first key guess is correct, while GE of 128 indicates a random behavior. GE can also show stability or consistent increase above 200 for the correct key candidate when the computation method for GE don't consider averaging several attacks on different traces. Such behavior indicates that the trained model failed to learn how to classify data.

The metric used during a neural network training phase is training accuracy. Note, this metric can be deceiving for assessing the quality of a side-channel attack because it evaluates the attack one trace at a time, while SCA metrics take several traces into account, giving a more accurate estimation for a real attack scenario [Pic+19].

3.4 EXPERIMENTAL RESULTS

Table 1: List of evaluated hyperparameters.

Hyperparameter	Range
Activation function	$ReLU, Tanh$
Number of layers	$1, 2, 3, 4, 5, 6$
Number of perceptrons per layer	$10, 20, 30, 40, 50, 100, 150, 200, 250, 300$

The results for all experiments on both datasets (ASCAD, AES_RD) and leakage models are given in four figures: the final key ranking from the guessing entropy of each model is represented for the activation functions explored in the first two figures. Next, we depict guessing entropy of the attack for all trained MLP architectures. The last figure presents the integrated gradient of the best-obtained model and the median model with the corresponding color value of its final guessing entropy. By doing so, we depict the differences in important features when comparing the best attack model and average model. The integrated gradient is a method introduced in [STY17], which attributes the prediction of a deep neural

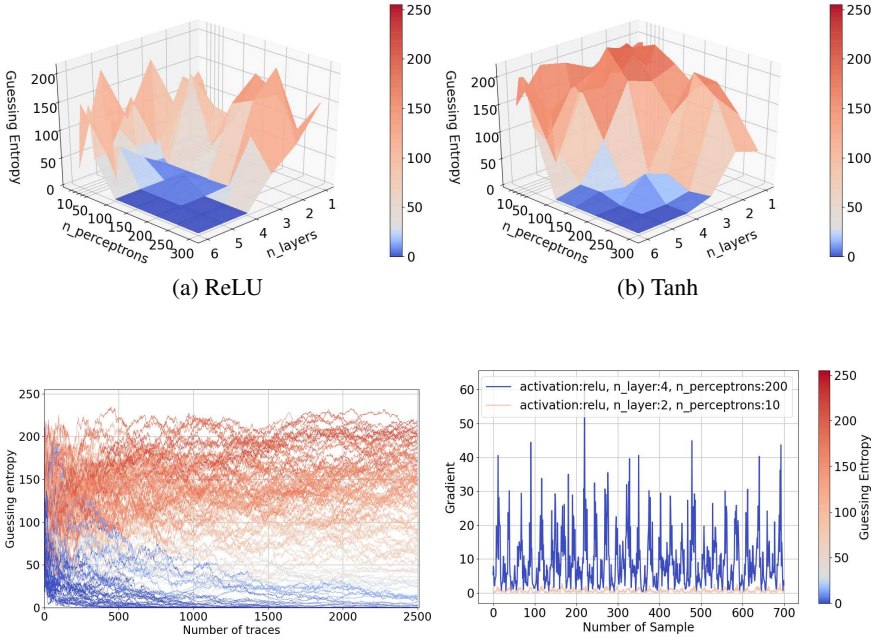
network to its inputs. The integrated gradient can be used in the side-channel analysis to visualize the part of the traces that influence the most a network prediction and understand what trace samples the network evaluates as the leakage.

3.4.1 *ASCAD Results*

INTERMEDIATE LEAKAGE MODEL: In Figure 8, we depict the influence of all combinations of hyperparameter choices for the *ReLU* and *Tanh* activation functions when considering the intermediate value leakage model. For both choices of activation functions, some models reach guessing entropy of 0 within 1 000 attack traces. More models achieve a low guessing entropy with the *ReLU* activation function than with *Tanh*. On the other hand, *Tanh* seems to behave more stable as the resulting GE is more uniform across many explored hyperparameters settings. Several models with *ReLU* activation function and a low number of perceptrons (down to 50) can reach GE near zero.

The authors of the ASCAD dataset report the best performance using an MLP with six layers containing 200 units and *ReLU* activation function trained over 200 epochs. The same hyperparameters are also evaluated and show similarly good results. However, This hyperparameters choice is not unique, and other models show equivalent performances with fewer layers and perceptrons per layers. As represented in Figure 8a, for settings with 200 perceptrons per layer, all MLPs with more than two hidden layers converge approximately equally fast to GE of 0. In Figure 8c, we see that many settings reach GE of 0 and that some have poor performance even after 2 500 attack traces with GE around 200. We interpret this as expected sensitivity to the hyperparameter tuning. Models with too few layers and perceptron per layers failed to properly fit the data because of their poor learnability. Finally, Figure 8d shows that the model that reaches the smaller GE in the attack (in blue) is more sensitive to the various samples of the input than other models that fail to learn the leakage. The leakage seems entirely spread over all samples, which indicates reducing the number of features will reduce the attack performance. The model that reaches a median GE considering all experiments (in orange) has smaller integrated gradients on every data sample, which explains why this model shows poor performance for the attack.

REDUCED NUMBER OF FEATURES: We now reduce the number of features to 50 with the Difference-of-Mean method. We train different MLPs with the traces that have a reduced number of features. We apply the same reduction for the attack dataset and compute guessing entropy, and we show the results in Figure 9.



(a) ReLU

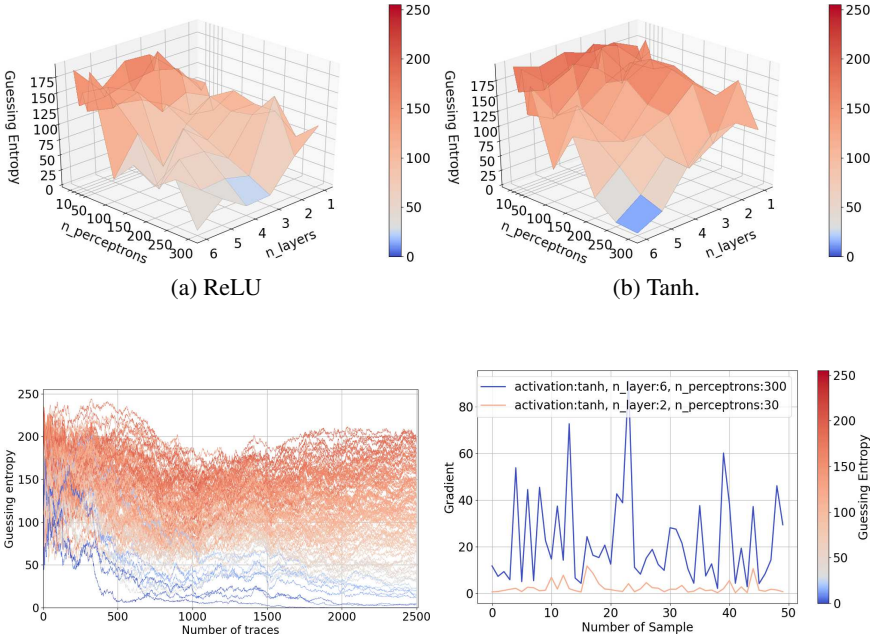
(b) Tanh

(c) Guessing entropy of all MLP architectures versus the increasing number of attack traces.

(d) Integrated Gradient.

Figure 8: ASCAD guessing entropy for the intermediate leakage model.

The area where GE converges toward zero is now smaller. For the *ReLU* activation function, this area is located around three and four layers with 250 and 300 perceptrons per layer. For the *Tanh* activation function, it is located above five layers and 250 perceptrons per layer. Interestingly, the highest score in Figure 9a is not obtained for the highest number of layers. For both activation functions, the hyperparameters leading to a good attack performance are shifted toward larger hyperparameter values. This indicates that when considering features selected with the DoM method (i.e., using less information), we require deeper MLP to reach the same performance level, as the information is still present but more difficult to fit for the model. Figure 9c shows sensitivity to hyperparameter tuning similar to the case with no feature selection. From Figure 9d, the best fitting model has higher gradient values than the median model. Consequently, for the best model, we use most of the available features, while the average models do not manage to combine available features in any way that would indicate influence in the classification process.



(a) ReLU

(b) Tanh.

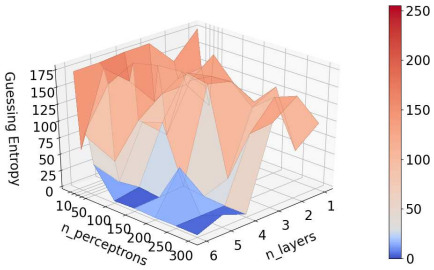
(c) Guessing entropy of all MLP architectures versus the increasing number of attack traces.

(d) Integrated Gradient.

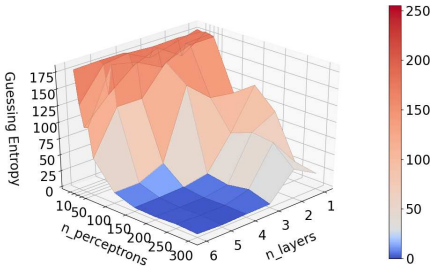
Figure 9: ASCAD guessing entropy with a reduced number of features and the intermediate leakage model.

HW LEAKAGE MODEL: Next, we consider the Hamming Weight (HW) leakage model. From Figure 10, we see similar results when compared to the intermediate value leakage model. Still, in Figure 10b, the number of perceptrons per layer has a more substantial influence on the guessing entropy than the number of layers. We can notice a better behavior for MLP with a small number of layers compared to the intermediate value leakage model scenario. We believe this happens as more perceptrons per layer give more options on how to combine features, while deeper networks would contribute to more complex mappings between input and output, which is not needed for the HW leakage model as the classification task is simpler than when using the intermediate value leakage model. We can also see a stable area for several models with a number of perceptrons above 150 and a number of layers above three. In this area, the hyperparameters choice does not influence the performance of the MLP anymore. Like the intermediate value leakage model, the sensitivity to the hyperparameter tuning (Figure 10c) is as expected, with many settings reaching top performance, but also many performing

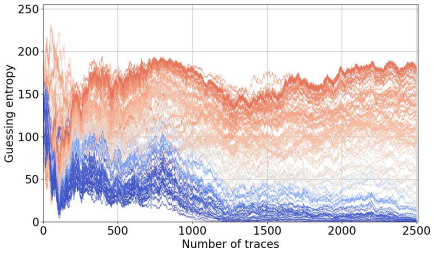
poorly. Interestingly, again we observe a more stable behavior from *Tanh* than the *ReLU* activation function. From Figure 10d, the best fitting model and the median model have similar integrated gradient values. However, the highest peaks are different, showing that the leakage learned by the two models is different, which also accounts for the differences in GE results.



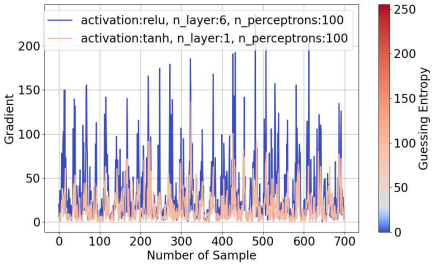
(a) ReLU.



(b) Tanh.



(c) Guessing entropy of all MLP architectures versus increasing number of attack traces.

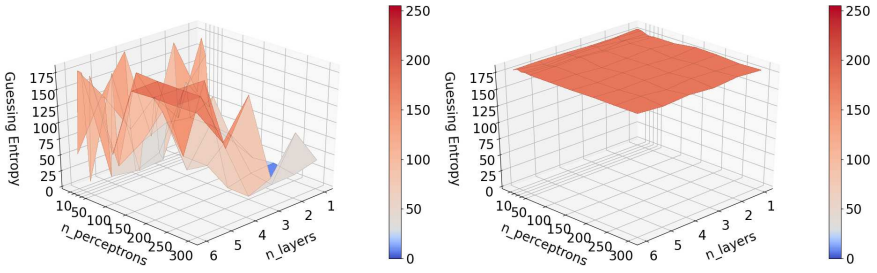


(d) Integrated Gradient.

Figure 10: ASCAD guessing entropy in the Hamming weight leakage model.

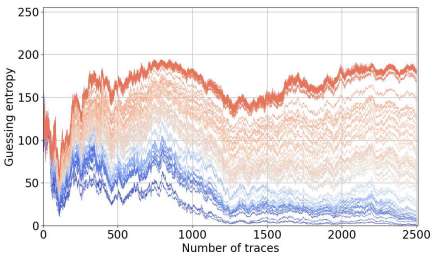
HW LEAKAGE MODEL AND REDUCED NUMBER OF FEATURES: We use the reduced number of feature representation of the dataset and apply the Hamming weight leakage model. We can see in Figure 11c that many MLP architectures differ significantly with a GE spread between 0 and 175. In Figure 11b, no MLP with the *Tanh* activation function succeeds in the attack. Finally, in Figure 11a, MLP with *ReLU* reaching GE of 0 has only one hidden layer, and when the number of layers increases, the performance decreases. Based on the ruggedness of the landscape for *ReLU*, it is clear that the choice of the number of layers/perceptrons plays a significant role. In Figure 11c, slightly differing from previous cases (cf. Figure 10c), we see more groupings in the GE performance.

This indicates that a reduced number of features in the HW leakage model is less expressive, so more architectures reach the same performance. From Figure 11d, the median model presents a higher integrated gradient than the best fitting model. This behavior differs from the previous experiments and shows that a wrong fitting model has high sensitivity on samples that do not correlate with the correct leakage. This also explains the spread of GE results, as there are many subsets of features combinations that result in high GE.

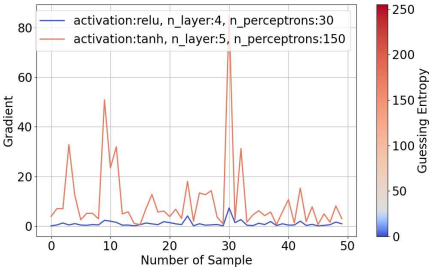


(a) ReLU.

(b) Tanh.



(c) Guessing entropy of all MLP architectures versus increasing number of attack traces.



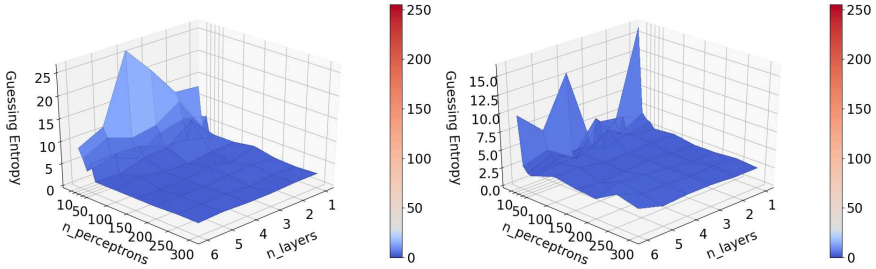
(d) Integrated Gradient.

Figure 11: ASCAD guessing entropy with a reduced number of features and the Hamming weight leakage model.

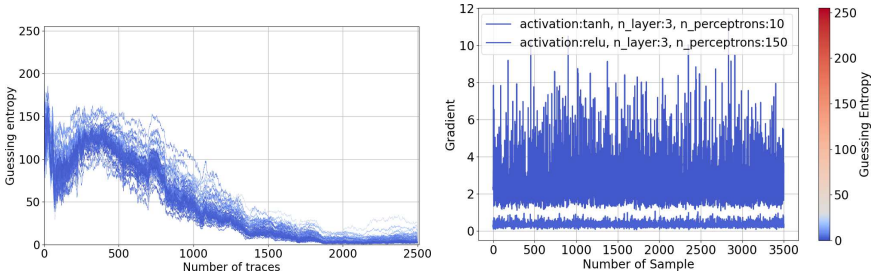
3.4.2 *AES_RD Results*

INTERMEDIATE LEAKAGE MODEL: Given the intermediate value leakage model (Figure 12), all MLP architectures, including the smallest ones (one hidden layer with ten perceptrons), are capable of reaching GE below 30 within 2 500 attack traces. Increasing the number of layers does not have an impact on the *ReLu* activation function. For the *Tanh* activation function, it even seems to

increase GE (thus, decreasing the attack performance). For both activation functions, increasing the number of perceptrons per layer decreases GE. Still, from Figure 12c, regardless of the architecture chosen, all MLP settings converge within the same amount of attack traces. This indicates that there is not enough useful information that larger networks can use, and as such, using them brings no performance gain (consequently, there is not much benefit from detailed hyperparameter tuning). The best-fitting model and the median model are both models that fit the dataset correctly. However, from Figure 12d, the integrated gradient method reveals that the two models have very different sensitivity on the input. Such a result could have been expected as the AES_RD dataset deals with randomly delayed traces, meaning that the leakage is not located in a precise area of the input.



(a) ReLU. (b) Tanh.

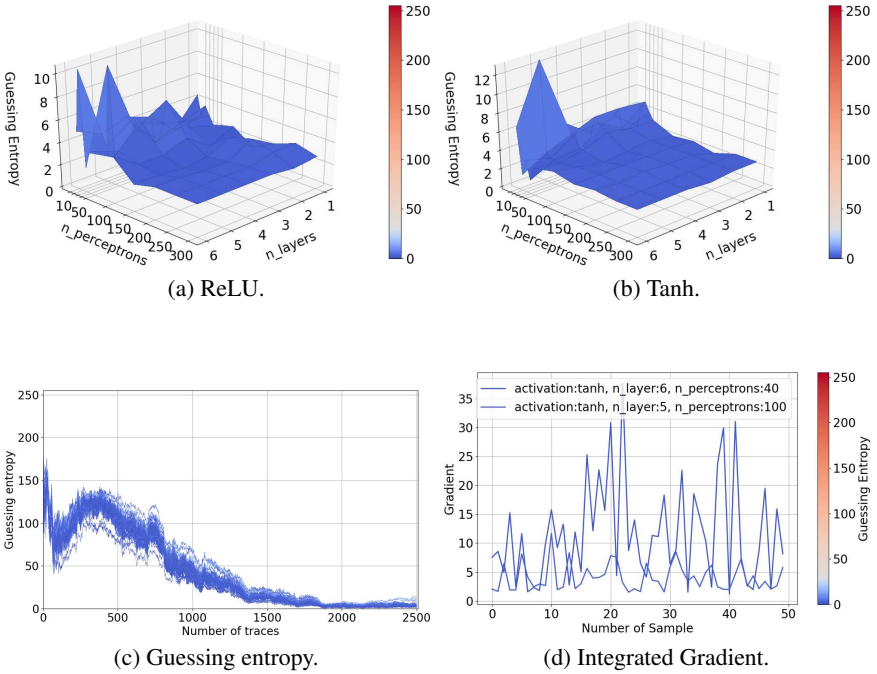


(c) Guessing entropy. (d) Integrated Gradient.

Figure 12: AES_RD guessing entropy for the intermediate leakage model.

REDUCED NUMBER OF FEATURES: In Figure 13, we observe a similar performance when training MLPs with a reduced number of features for the AES_RD dataset and the intermediate leakage model (containing only 50 selected features). Again, this implies there is no useful information in additional features, and that is why MLP cannot perform better even if we use larger/deeper architec-

tures. This is following the expected behavior for the random delay countermeasure as the features are not aligned. Finally, the landscape is smoother for *Tanh* than for *ReLU* (similar to ASCAD but also different from AES_RD with all features). The outcome from Figure 13d is quite similar to the integrated gradient obtained on the raw traces. While the gradient values for the two models have the same levels, no maximum or minimum values are the same, meaning that no samples contribute significantly to network prediction.



(a) ReLU. (b) Tanh.

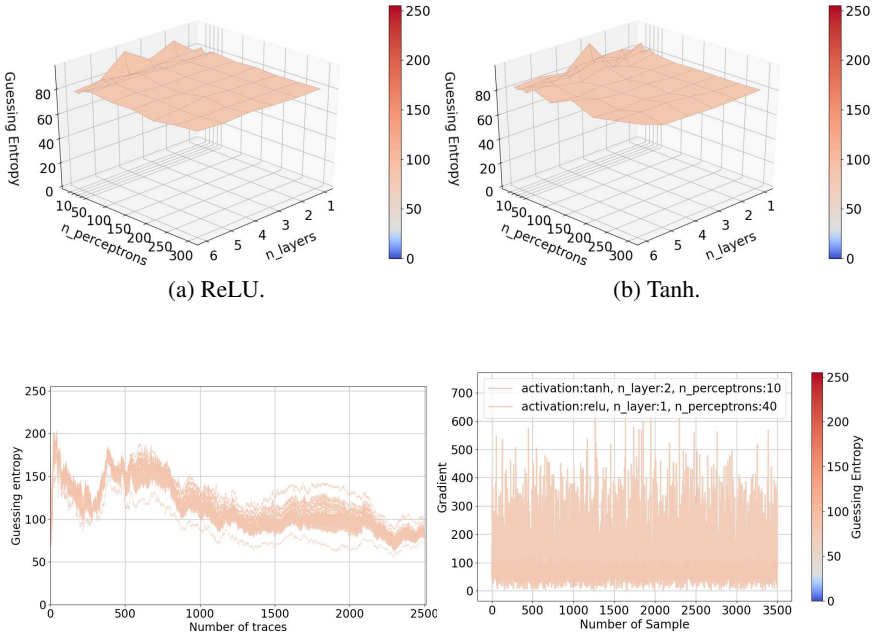
(c) Guessing entropy. (d) Integrated Gradient.

Figure 13: AES_RD guessing entropy with a reduced number of features and the intermediate leakage model.

HAMMING WEIGHT LEAKAGE MODEL: When considering the HW leakage model for the AES_RD dataset, even after 2 500 traces, the attack is still unsuccessful. More precisely, in Figure 14, no hyperparameter setting results in a model that can reach a GE below 60, which is not even close to a successful attack. Note we do not depict results for the reduced number of features as the attack was not successful even with the full number of features. With the intermediate value leakage model, we required around 1 500 traces to succeed in the attack. Now, we use a leakage model with a simpler classification problem and fail with more measurements. This result shows that the HW leakage is either not present or that

the trained models are too simple to fit the leakage. Interestingly, all architectures behave relatively similarly, as visible in Figure 14c. The integrated gradient on Figure 14d shows similar results as obtained for the intermediate value leakage model, but in this case, both models do not fit the dataset correctly, which means it is difficult to talk about features that contribute more to the classification result. No trace samples show a higher sensitivity for the network prediction because of the random delay nature of the dataset.

As no MLP architecture can succeed in the HW leakage model's attack on the AES_RD dataset, we cannot conclude whether more layers or perceptrons would improve the attack performance. The phenomenon preventing MLPs from obtaining good attack performance might be linked to the class imbalance, pointed out by Picek et al. [Pic+19], where they obtain similar results for different architectures of MLP using the HW leakage model. Additionally, they observe increasing performance when balancing the training data among the classes.



(a) ReLU. (b) Tanh.

(c) Guessing entropy. (d) Integrated Gradient.

Figure 14: AES_RD guessing entropy for the Hamming weight leakage model.

3.5 DISCUSSION

MLP can break the masking countermeasure of the ASCAD dataset and the random delay countermeasure of the AES_RD implementation even when training a rather small model. For AES_RD, the smallest models (one layer, 200 perceptrons, and six layers, ten perceptrons) share the best outcome of all the models in the comparison. The same results are observed when using only the most important features. An important leakage of the secret could explain these results if the countermeasure were turned off. Although the random delays shift the first round S-box operation from the start of the encryption execution, a strong leakage of the operation handling the secret information is still present. Consequently, using an MLP is enough to overcome this countermeasure. This result indicates that the current consensus in the SCA community on MLP performance should change. Indeed, CNNs are considered especially good for random delay countermeasure and MLP for masking countermeasure [MPP16; Pic+19]. Our results indicate there is no reason not to consider MLP successful against the random delay countermeasure given the satisfying results obtained on AES_RD with intermediate value. When selecting 50 POIs with a Difference-of-Mean method, the selected points represent only $50/3\,500 \simeq 1\%$ of the original traces in the dataset, and the information about the leakage is reduced. Still, the attack succeeds in the same way, which can be explained because the leakage only comes from the selected POIs. Finally, the integrated gradient is more difficult to interpret as the dataset has randomness in the time domain, which means it becomes difficult to pinpoint a few features with a significant contribution toward the classification result.

For the ASCAD dataset, we observe that the best score obtained for MLP has the following hyperparameters: six layers and 200 perceptrons. Still, we see in Figures 8a and 8b that MLP with similar hyperparameters can perform equally good (where the red point represents the result obtain with the architecture of the best MLP MLP_{best} from the ASCAD paper). When selecting POIs with the Difference-of-Mean method, we can observe that the performance decreases, meaning that the useful information is decreased. This, in turn, results in attacks not able to recover the full secret key. Still, some MLPs can obtain the secret key in the given number of traces, and we observe that both the number of layers and the number of perceptrons influence their performance. Finally, the performance of MLPs with the Hamming weight leakage model gives better performance than for the intermediate value. The range of hyperparameters that can achieve the best results is smaller than for the intermediate value leakage model. From the integrated gradient perspective, we see that many features contribute to a successful attack, but MLP makes slightly different feature selection than DoM, as obviously not all

50 selected features contribute significantly. For the HW leakage model, the integrated gradient is somewhat more aligned, which means that more features in this leakage model contribute similarly. Such behavior is again expected as the HW leakage model forms larger clusters with S-box output values, where the importance of features is more spread within clusters.

To answer the question of how challenging is the tuning of MLP hyperparameters, we observe that there is nearly no influence using a (relatively) big or small MLP for the AES_RD dataset. When considering the ASCAD dataset with the masking countermeasure, depending on the leakage model considered, the size of the MLP can play a significant role. There, either by increasing the number of perceptrons per layer or the number of layers with a fixed number of perceptrons, we can decrease the guessing entropy.

From the activation function perspective, *ReLU* behaves somewhat better for the intermediate leakage model when compared to *Tanh*, i.e., it can reach the top performance with a smaller number of layers/perceptrons. For the Hamming weight leakage model, *Tanh* seems to work better on average, but *ReLU* reaches top performance with smaller architectures than *Tanh*. Finally, *Tanh* gives more stable behavior when averaged over all settings, i.e., with the *Tanh* activation function, the hyperparameter tuning seems to be less sensitive. To conclude, *ReLU* appears to be the preferred option if going for top performance or using smaller architectures. In contrast, *Tanh* should be preferred if stability over more scenarios is required.

MLP is (or, at least, can be) a deep learning algorithm that has a simple architecture and a few hyperparameters but can show good performance in the side-channel analysis. What is more, our results show it can break implementations protected with both masking or hiding countermeasures. If there is no sufficient useful input information (as one would expect when dealing with the random delay countermeasure), a reasonable choice is to go with a relatively small architecture. For masked datasets, the number of perceptrons or the number of layers must be large, but the activation function's choice also plays an important role. Finally, we observe that in all considered scenarios, the MLP architectures are not overly sensitive to the hyperparameter choice, i.e., there does not seem to be a strong motivation to run very fine-grained hyperparameter tuning.

Based on those observations, we list general recommendations for MLP in the profiled SCA context [1]:

1 The recommendations are based on the tested configurations. There is no guarantee that different results could not be achieved with significantly different settings, e.g., having a different number of perceptrons per layer. Still, following our recommendations should provide good performance in most of the scenarios commonly encountered in profiling SCA.

1. Many hyperparameter settings can lead to good performance, which makes the benefit of an exhaustive search very limited.
2. *ReLU* is better for top performance, while *Tanh* is more stable over different hyperparameter combinations.
3. Smaller depth of an MLP can be compensated with wider layers.
4. Integrated gradient is an efficient method for evaluating the influence of features if MLP manages to reach good performance.
5. Simpler leakage models require fewer layers.

3.6 CONCLUSIONS AND FUTURE WORK

In this paper, we considered the behavior of a multilayer perceptron for profiling side-channel analysis. We investigated two datasets protected with countermeasures and a number of different MLP architectures concerning three hyperparameters. Our results clearly show that the input information to the MLP plays a crucial role, and if such information is limited, larger/deeper architectures are not needed. On the other hand, if we can provide high-quality input information to the MLP, we should also use larger architectures. At the same time, our experiments revealed no need for very fine-grained hyperparameter tuning. While the results for MLP maybe cannot compare with state-of-the-art results for CNNs, we note that they are not far apart in many cases. If we additionally factor in that MLP is simpler and faster to train, the choice between those two techniques becomes even more difficult to make and should depend on additional goals and constraints. For example, reaching the top performance is the argument for the usage of CNNs, but if one requires small yet powerful architecture, a more natural choice seems to be MLP.

In this work, we concentrated on scenarios where each hidden layer has the same number of perceptrons. It would be interesting to investigate the performance of MLP when each layer could have a different number of perceptrons. Naturally, this opens a question of what combinations of neurons/layers to consider as one could quickly come to thousands of possible settings to explore. Similarly, for activation functions, we consider only the two most popular ones where all hidden layers use the same function. It would be interesting to allow different layers to have different activation functions. Recent experiments showed that MLP could outperform CNNs when considering different devices for training and testing (i.e., the portability case) [Bha+19]. We plan to explore the influence of the hyperparameter choice in those scenarios. Finally, as we already mentioned, MLP architectures are usually simpler than CNNs, which should mean they are easier to

understand. We aim to explore whether we can design stronger countermeasures against machine learning based-attacks based on MLP inner working.

4

LIGHTWEIGHT BUT NOT EASY: SIDE-CHANNEL ANALYSIS OF THE ASCON AUTHENTICATED CIPHER ON A 32-BIT MICROCONTROLLER

This chapter is based on [WP23], a joint work with Stjepan Picek, that was published on the IACR Cryptology ePrint Archive in 2023.

CONTENT OF THIS CHAPTER

4.1 INTRODUCTION

In the field of symmetric cryptography, the need for lightweight primitives is crucial in the development of secure, fast, and low-consumption designs that can be used for embedded devices in resource-constrained environments but also for high-bandwidth applications. In this effort, the National Institute of Standards and Technology (NIST) initiated a lightweight cryptography standardization process in 2018. The goal was set to decide on a new standard for lightweight cryptography by comparing submitted designs of block ciphers, hash functions, message authentication codes (MACs), authenticated encryption with associated data (AEAD), and pseudorandom functions (PRFs). The process ended on 7 February 2023, with the selection of the Ascon family for standardization. Ascon has been designed to be fast and easy to implement [Dob+21a]. The use of the permutation function in its S-box eases the implementation of countermeasures against side-channel analysis and prevents common pitfalls known from implementations of the AES S-box. During the standardization process, side-channel evaluation has

been performed on every finalist [Moh+23], and protected implementation of Ascon in hardware using domain-oriented masking implementation that has been shown resistant against side-channel analysis. However, protected the protected software implementations of Ascon uses a specific masking countermeasure that is formally verified and have not had an extensive evaluation against side-channel analysis, especially against profiled attacks and DLSCA.

The Ascon team provided implementations for numerous platforms, including a masking implementation using Domain-Oriented Masking (DOM) for hardware and a specific masking countermeasure for software implementations. This implementation allows a configurable number of shares and rotations that can be applied to hardware and software platforms with great flexibility and low overhead, yet is robust against higher-order attacks by randomizing shares with reduced need for fresh randomness compared to a threshold implementation masking scheme. This masking is also the only countermeasure publicly provided for software implementations of Ascon.

The side-channel analysis report obtained from the common effort of several evaluation laboratories and researchers during the last round of the standardization process has evaluated Ascon with classical side-channel analysis methods, namely Test vectors leakage assessment (TVLA), χ^2-test, and correlation power analysis. Those evaluations have been performed on hardware and software implementations and confirmed side-channel resistance even with second-order CPA, considering several millions of traces [Moh+23].

This paper focuses on showing that deep learning side-channel analysis (DLSCA) can be applied to Ascon. Moreover, we aim to enlarge the possible attack surface of Ascon in the context of profiling side-channel analysis. For this purpose, we consider different leakage models that can expose information about the key during the initialization phase of Ascon. We explore different leakage models to learn if DLSCA can be applied to an unprotected implementation of Ascon and if the same leakage model can also be applied to a protected implementation.

The core challenge we try to address in this paper is to conduct a profiling side-channel analysis on Ascon with deep learning and compare the results with the known CPA attack. The contributions of this paper are as follows:

- We provide a dataset for profiling side-channel analysis on Ascon from a reference software implementation and a first-order protected implementation [1].

1 https://zenodo.org/records/10229484

- We show that DLSCA can successfully recover the key using fewer traces than CPA for both unprotected and protected Ascon implementations. The CPA attack on the unprotected dataset can recover the key in 8 000 traces against 1 000 traces for DLSCA. On the protected dataset, a second-order CPA fails to recover any part of the secret key, while DLSCA can recover certain partial keys in 800 traces.
- We show different leakage models that could be used to evaluate the leakage of the Ascon S-box function to recover the key. Namely, the S-box output value and the output state's register value. While the first can be simplified to obtain better results for CPA, the second can result in better results using DLSCA in the presence of side-channel countermeasures.
- With a single task model, we can obtain the partial key under DLSCA with less than 20 traces, but the effort of the attacker to recover the full key can be underestimated with this method. For this reason, we build a multi-task model that can analyze the traces and simultaneously make decisions for every partial key separately.

The rest of the paper is organized as follows. Section 4.2 presents the background on Ascon. Section 4.3 provides the related work applying side-channel analysis as well as known attacks on Ascon. Section 4.4 discusses the leakage models we considered. Section 4.5 presents the experimental results for DLSCA and a comparison with CPA. In Section 4.6, we apply the multi-task learning methodology for a deep learning attack to recover every partial key. Finally, Section 4.7 gives concluding remarks and elaborates on possible future work.

4.2 ASCON

Ascon is a family of authenticated encryption and hashing algorithms standardized by NIST [Dob+21b]. It is a permutation-based block cipher with a 320-bit state divided into five words of 64-bits $(x_0, x_1, x_2, x_3, x_4)$. The permutation is applied iteratively on the state in an SPN-like fashion. The two parameters (a, b) of Ascon are the number of rounds and the number of bits to be rotated in the permutation. The Ascon round transformation consists of a 1-byte addition of a round constant, a 5-bit S-box applied bit-sliced, and a linear diffusion layer. The Ascon's S-box is designed in a bit-sliced fashion that operates on the 5-bit columns of the state, 64 times in parallel.

The linear diffusion layer is applied on each register of the state by XORing two copies of the same registers with different cyclic shifts together with the register.

Protected software implementation of Ascon is specifically designed thanks to the design of the Chi (χ) function of its S-box [Gig+24]. The idea is to introduce masked Toffoli gates for the Chi function inside the S-box with the addition of re-used randomness that is cyclically refreshed at each call. This countermeasure aims to prevent the combination of shares at the instruction level and randomize the operations that, if combined can leak information on the shares. An additional SCA countermeasure is set to rotate the offsets between the shares. This masking countermeasure has been heuristically verified with formal verification [Zai+19].

4.3 RELATED WORK

Side-channel analysis is commonly considered when evaluating the security of symmetric cryptographic primitives. Differential power analysis was first introduced by Kocher et al. and was originally applied to DES [KJJ99]. The authors showed that the leakage of the Hamming weight of the intermediate values of the S-box function can be used to recover the key successfully, using the difference-of-means as a statistical analysis method to compare the hypothetical power consumption values with the recorded traces. Another way to determine the relationship between data is to use the Pearson correlation coefficient. This method represents the pillar of the non-profiled SCA and is commonly referred to as Correlation Power Analysis (CPA) [BCO04; CCD00].

Deep learning side-channel analysis has been a promising technique for profiling side-channel analysis since the work of Maghrebi et al. [MPP16]. Later, Benadjila et al. [Pro+18] introduced a dataset for profiling side-channel analysis on AES protected with Boolean masking and showed that a convolutional neural network (CNN) is an efficient approach to recovering the AES key. Many other research works have also shown that deep learning can be a powerful profiling attack against AES [MPP16; Pic+18b; Kur+21; RK21]. More recently, a number of works consider the multi-task paradigm in DLSCA, showcasing it can be more powerful than the single-task approach [Mag20; MO23b; MO23a; MS23].

Since Ascon was introduced in 2016 for the CAESAR competition, several works have been published on Ascon's security. The authors of [SD17] demonstrated a CPA attack on a hardware implementation of Ascon. The authors showed that bits of the output state of the round function can lead to a key recovery. In [RAD20], the authors attacked the Ascon S-box operation of a hardware implementation that executes operations in a sequential mode. They could separate the different bitwise S-box operations to apply a horizontal attack, and they used reinforcement learning to recover the key from the leakage of the S-box. The authors also employed an autoencoder to perform dimensionality reduction given the

sub-traces of every S-box operation. In [SS23], the authors used transfer learning from a well-fitting model for AES on the ASCAD dataset to improve DLSCA on an unprotected Ascon implementation for RISCV microcontroller. In [You+23], the authors demonstrate the first working template attack with belief-propagation and key enumeration techniques on a software protected implementation of Ascon, where they could obtain a success rate above 90% using 20 traces. In general, the literature on machine learning-based SCAs on ciphers different from AES is relatively sparse, especially considering lightweight symmetric ciphers, see, e.g., [Heu+17; Heu+16; MWM21]. This indicates more work is needed to understand how to mount powerful attacks and how much of the knowledge is transferable from one target to another.

4.4 LEAKAGE MODELS

The target of side-channel attacks against Ascon is the value of the key used in the initialization phase. The attack at this point is possible because we know the content of the state at initialization, except for the key. During the first permutation round, the state is composed of a 64-bit initialization value, the 128-bit key, and a user-defined 128-bit nonce.

The non-linear properties of the Ascon S-box with the controllable nonce enable possible leakage that can be exploited with SCA. One intuitive leakage model that can be applied is the S-box output. The Ascon S-box is a 5-bit S-box that is applied to the columns of the state. It is possible to consider this leakage differently depending on the cipher implementation.

The output state registers $x_0, .., x_5$ can be rewritten in the algebraic normal form (ANF) as follows:

$$y_0 = x_1(x_4 + x_2 + x_0 + 1) + x_3 + x_2 + x_0 \tag{10}$$
$$y_1 = (x_3 + 1)(x_2 + x_1) + x_2x_1 + x_4 + x_0 \tag{11}$$
$$y_2 = x_4(x_3 + 1) + x_2 + x_1 + 1 \tag{12}$$
$$y_3 = (x_0 + 1)(x_4 + x_3) + x_2 + x_1 + x_0 \tag{13}$$
$$y_4 = x_1(x_4 + x_0 + 1) + x_3 + x_4. \tag{14}$$

4.4.1 *Leakage Models for Differential Attacks*

DPA attacks use datasets of many power traces from a cryptographic device operating with the same key to exploit the data dependency of the power consumption. As a consequence, the activity of the computation that has a constant contribution

to the power consumption will be equal for each trace collected and will cancel out in the analysis. Thus, all the variables contributing with a constant amount to the activity can be removed from the previous notation, i.e., the terms with x_0, x_1, x_2, or combinations of those.

$$y_0 = x_4x_1 + x_3 \tag{15}$$
$$y_1 = x_3(x_2 + x_1 + 1) + x_4 \tag{16}$$
$$y_2 = x_4(x_3 + 1) + 1 \tag{17}$$
$$y_3 = (x_4 + x_3)(x_0 + 1) \tag{18}$$
$$y_4 = x_4(x_1 + 1) + x_3 \tag{19}$$

In Eqs. (17) and (18), y_2 and y_3 do not involve computation on the bits of the key, and thus cannot be used as leakage functions. However, in Eqs. (15) and (19), it can be noticed that both y_0 and y_4 have a relation with a bit of state x_1, and can be used interchangeably to recover the first half of the key. By attacking the register y_1 in Eq. (16), the leakage from the value of $x_1 + x_2$ can be learned. This term can be used in conjunction with the information recovered from the previous leakage on x_1 to get the second half of the key related to x_2, as also pointed out in [SD17].

The content of register x_1 or $x_1 + x_2$ is the secret denoted k, and x_3 and x_4 are denoted m and m', respectively.

$$y_0 = km' + m$$
$$y_1 = m(k + 1) + m'$$
$$y_4 = m'(k + 1) + m.$$

The application of the linear diffusion layer on the output of the S-box function permits to obtain the following expressions:

$$S0_i(M, K^*) = k_0^* m'_i + m_i + k_1^* m'_{i+45} + m_{i+45} + k_2^* m'_{i+36} + m_{i+36} \tag{20}$$
$$S1_i(M, K^*) = m_i(k_0^* + 1) + m'_i + m_{i+3}(k_1^* + 1) + m'_{i+3} + m'_i + m_{i+25}(k_2^* + 1) + m'_{i+25} \tag{21}$$
$$S4_i(M, K^*) = m'_i(k_0^* + 1) + m_i + m'_{i+57}(k_1^* + 1) + m_{i+57} + m'_{i+23}(k_2^* + 1) + m_{i+23}. \tag{22}$$

Eqs. (20), (21), and (22) show the leakage functions that can be used to recover the key when targeting the output of the round function. This leakage function directly aims to correlate the power consumption with the S-box output value. Since the S-box operation works on a column of the state, the storage of the out-

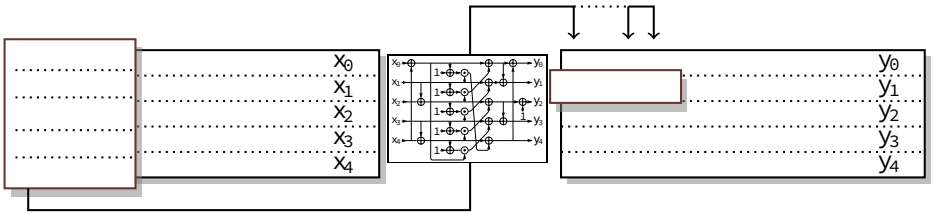


Figure 15: Computation of the leakage value of a given output register.

put should be uncorrelated with the operation and is understandable from the bit-sliced nature of the operation. The leakage is based on the value of only one bit of the output state. Thus, using the Identity or Hamming weight power model makes no difference for a software implementation. The Hamming distance power model can be used when targeting a hardware implementation.

4.4.2 *Leakage Models that Apply Better for Profiled Attacks*

For profiled attacks, the profiling traceset is composed of traces collected from encryption with random keys, and the attack traceset is collected with a fixed key for the attack phase. From Eq. (14), it is possible to exploit the leakage that only depends on the input register x_1, which contains the first half of the key. It is then possible to obtain the second half of the key from the leakage of any other register because y_0, y_1, y_2 and y_3 depend on x_2. This paper focuses on the leakage of y_2 (see Eq. (12)).

While the Ascon S-box is bit-sliced, the output leakage can be stronger for bytes of the output state when considering a software implementation because of the architecture of the microprocessor and leakage during the storing operation. Alternatively, slices of the state can be considered to observe the concatenated leakage for a single byte slice, as shown in Figure 15.

When considering this leakage function, the goal is to observe a correlation between the power consumption and the storage of the output state. Because the architecture of the studied microcontroller is 32-bit, the stored output is large, and it can be difficult to get a correlation based on the full-length variable. With this leakage function, we aim to obtain partial information of the value stored, and the size of the variable can be determined smaller than 32-bit. The partial correlation on a large word can be used to obtain information about the value stored in a register, as shown in [Tun+07]. Repeating this attack on successive

parts of the register makes it possible to recover the secret value using fewer traces and computing power than when using the full register correlation.

4.5 EXPERIMENTAL RESULT

4.5.1 *Implementation*

In this paper, we consider a software implementation of Ascon. The C implementation from the Ascon team can be found on their GitHub repository [Tea]. They provide implementations for every Ascon mode and several platforms. This work considers only the AEAD implementation of Ascon-128 v1.2 optimized for ARMv7m microcontrollers. The Ascon-128 v1.2 is the recommended implementation when using Ascon for AEAD for a key of 128-bit. Instead of using the C-reference implementation, we use the 32-bit optimized implementation they provide with our target device architecture to ensure the closest results to a real scenario.

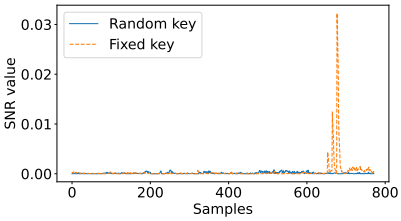
Traces are collected with a ChipWhisperer Lite board, an 8-bit precision oscilloscope, coupled with the STM32F4 target.[2] The target microcontroller is a 32-bit platform running at a default clock frequency of 7.37MHz. Traces are collected in a manner that only contains power samples during the first round of the initialization permutation for both reference and protected implementations.

The unprotected implementation dataset contains 60 000 traces with 772 samples each. In this dataset, 50 000 traces are collected with random keys, and 10 000 traces with a fixed key. The protected implementation dataset contains 560 000 traces with 1 408 samples each. The implementation uses bit-interleaved domain-oriented masking with two shares. In this dataset, 500 000 traces are collected with random keys, and 60 000 traces with a fixed key.

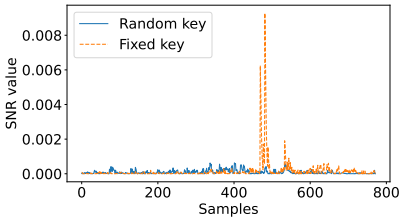
4.5.2 *Signal-to-Noise (SNR) for Leakage Models*

Figure 16 depicts the SNR of the round function leakage model with fixed key and random key datasets. We can observe that for both registers' leakage, only the dataset with a fixed key shows an important leakage. This observation supports the leakage established in Section 4.4 for non-profiled leakage models. We can also observe that leakage from register y_4 is stronger than the leakage from register y_1 by a factor of almost 4. This difference of leakage between y_1 and y_4 can result from the difference of the registers used inside the ARM microcontroller. It is also

2 https://www.newae.com/products/NAE-CWLITE-ARM

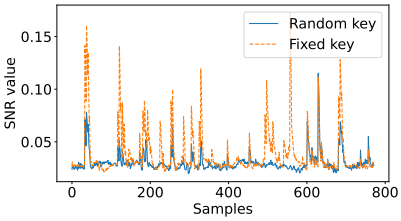


(a) Round function output register y_4.

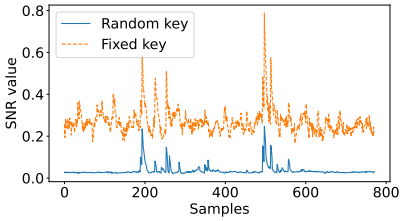


(b) Round function output register y_1.

Figure 16: SNR for unprotected implementation round function leakage model.



(a) 8-bit S-box output register y_4.



(b) 8-bit S-box output register y_2.

Figure 17: SNR for unprotected implementation S-box leakage models.

interesting to notice the leakage position for the two registers. While the leakage of the register y_4 is around sample 700, the register y_1 leaks mostly around sample 500, confirming that the two registers are treated in separate instructions in the considered implementation.

Figure 17 shows the SNRs for the register S-box output leakage models given for 8-bit S-box output as described in Section 4.4. We can see that the traceset with fixed key shows higher leakage than for random keys. While the SNRs suggest a correlation between these intermediate values and power consumption, it is not possible to argue whether the leakage can be exploited by a profiling attack.

Finally, the SNR obtained from the S-box output column shows higher overall leakage across all samples, as shown in Figure 18. It can be noticed that leakages from a fixed key dataset and random keys dataset overlap the most among all previously displayed SNRs. This result could indicate that this leakage model can be the most present one from those considered and can be used for the profiling attack because the samples involved in this leakage are the same for the random and fixed key traces.

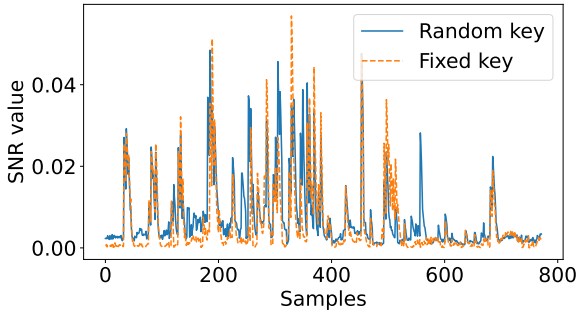
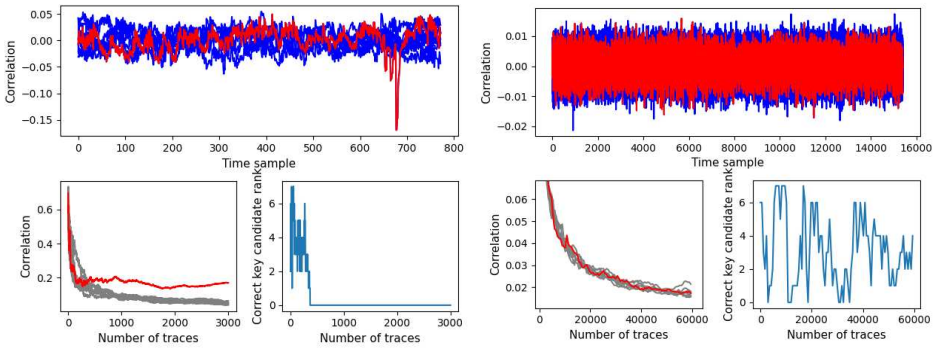


Figure 18: SNR for unprotected implementation S-box leakage on output state column.



(a) Reference implementation

(b) Protected implementation

Figure 19: Correlation power analysis targeting the first bit of the state output.

4.5.3 *Correlation Power Analysis*

For CPA, we use the output of the round function as the leakage function. This leakage was used in previous works [SD17; RAD20; SS23], and is a baseline for non-profiled attacks.

In Figure 19a, we see that CPA can successfully recover one partial key (i.e., 3 bits) with less than 1 000 traces. In the top figure, we display the correlation of all trace samples for every key candidate, with the correct key in red. We can see that the correct key candidate has a significantly higher correlation around sample 690. In the bottom left figure, we display the highest correlation value for all samples against the number of evaluated traces for the CPA. We can observe that the correlation for the correct key candidate becomes higher than that of other key candidates after 400 traces and stays higher the more traces are evaluated.

y_4	0	1	2	7	8	9	14	15	16	20	21	22	28	29	30
	35	36	37	42	43	44	45	49	50	51	52	56	57	58	63
y_1	0	2	3	4	5	6	7	8	9	10	12	13	15	16	17
	18	20	21	22	24	26	34	37	42	46	50	51	52	53	54
	55	56	58												

Table 2: Best set of column indices to recover the key from output state CPA in registers y_4 and y_1.

The bottom right figure shows the rank of the correct key against the number of evaluated traces, and it shows more clearly the convergence of the correct key candidate to zero within 400 traces. To confirm that this attack can be applied to recover every partial key, we describe how to repeat the previous partial key recovery efficiently.

The attack described above targets a sequence of three bits to get halves of the key given the two registers y_4 and y_1. When considering the register y_4, the sequence of bits of the key $[i, i+57, i+23]$ is related to the index of column i of the output state. With one CPA, the attacker can obtain one sequence of three bits of the first half of the key. One can do as many CPAs as there are bits to guess and only consider the first guessed bit, disregarding the rest of the found value. However, the least number of CPAs to perform to guess each bit of the first half of the key is defined by the minimal set of sequences with the least overlapping bits.

Given the distribution of all partial keys in register y_4, the best search can be obtained by walking through the indices in Table 2. This set of indices requires performing 30 CPA attacks to recover the first half of the key. We can do a similar walk with the indices for register y_1 with a total of 33 CPA attacks to recover the second half of the key.

The indices are obtained with the following methodology. First, we build a list of the bit sequences (i.e., the partial keys) and their index in the binary representation of the key for all columns in the state. Then, we remove every second sequence that has an overlapping bit index. From this reduced list, we iteratively append the partial keys for which the sequence contains one bit of the key that is missing from all the sequences in the reduced list. The final list contains the minimum number of key indices for which an attacker should repeat CPA attacks to recover the 128 bits of the key. To recover the complete 128-bit key, we must repeat the same attack 63 times on several bits of the output state.

The result of the full key recovery is described with the success rate metric. With this metric, it is possible to evaluate the minimum number of traces needed to recover every partial key, and a success rate of one translates to a successful

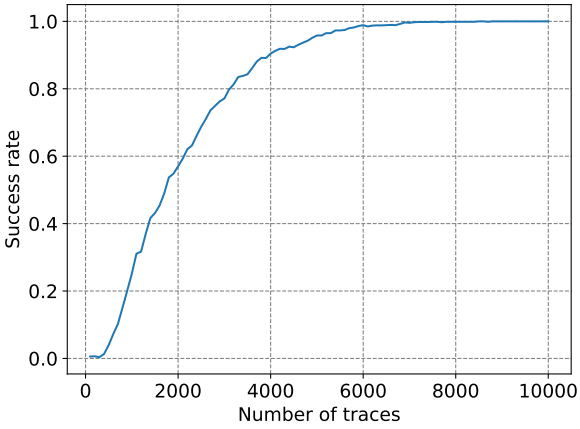


Figure 20: Success rate of the CPA.

attack. The success rate of the full key recovery is shown in Figure 20. With the fixed key dataset, it is possible to correctly guess all the bits of the 128-bit key using around 8 000 traces.

For the protected implementation, the output of the CPA targeting the first bit of the S-box output is shown in Figure 19b. We can see that even after 60 000 traces, the correlation of the first bit of the S-box output is not significant enough to recover the key, as the correlation value of the correct key candidate stays indistinguishable from the other key candidates.

4.5.4 *Deep Learning-based Attack*

For a deep learning attack to be successful, a well-fitting model should be trained. Finding such a model can be challenging, depending on the leakage model and dataset considered. To find such a model, the state-of-the-art Bayesian optimization (BO) methods for hyperparameter tuning [Moc77; RW06; Ngu19; WNS21] are commonly considered to be the best approach for sequential model-based global optimization. Note that such hyperparameter tuning approaches are already used in DLSCA and give good results; see, e.g., [WPP20; Rij+21]. This paper adopted the Tree-structured Parzen Estimator (TPE) approach from [Ber+11]. This approach is based on the Gaussian process for tree-structured configuration hyperparameter spaces and is well suited for a high dimensional model like CNNs with the number of layers as a hyperparameter, where the evaluation of the surrogate function is cheap. The principle of BO is to minimize an objective function using a surrogate function, a probabilistic model of the score obtained with the

objective function given a set of hyperparameters. This method helps to efficiently search the hyperparameter search space by deciding the next most promising step toward the best set of hyperparameters based on the results of previously evaluated sets. This method is also known to obtain better results compared to the grid search and random search methods in fewer evaluations [Tur+21]. The BO search is faster to converge toward a well-fitting model and can be used to make a better decision on when to stop the search for a better-performing model. The TPE approach adds several parameters in the algorithm to scale the exploration of the search space, taking into account the tree-structure base of a deep neural network with a high number of hyperparameters. This algorithm can also estimate the expected improvement direction to explore in the search space.

To define the search space, we first design a hyper architecture (i.e., hypermodel), which represents a guideline to define the directions for our models to grow. The Bayesian optimization method is applied for network architecture search with guessing entropy in the validation set as a surrogate function. The expected behavior of this search method is to explore the hyperparameter search space to maximize the architecture that could lead to a model with the fastest convergence to an attack with a guessing entropy of zero.

We set a few rules to shape the network architecture of a CNN to reduce the hyperparameter search space and match known well-performing designs for the analysis of 1-dimensional signals. The main principle is to use stacked convolutional layers followed by several fully-connected layers. The number of channels of the convolution layers starts from a small number and increases by a factor of 8 for every new convolution block. The architecture of the CNN hypermodel is described in Figure 21. First, the input data goes through a batch normalization layer, followed by n_conv_blocks convolutional blocks constituted by one convolution layer, an activation function layer, a batch normalization layer present every two convolutional blocks, and an average pooling layer. After the convolutional blocks, there is a flatten layer to reshape the data and, finally, n_fc_layers of linear and activation layers. The last linear layer outputs the decision of the neural network.

The training process follows the same procedure for all models. The training is done with a batch size of 128 and the 'Adam' optimizer [KB15]. The loss function is the cross-entropy loss, and the number of epochs is set to 200. These hyperparameters are selected manually to reduce the workload of the BO, and we note we achieve good performance with such a model. During every evaluation step of the model, the validation set is used to compute guessing entropy, together with accuracy and loss. These three metrics are tracked during training to assess the model's performance. In a particular case when the number of convolutional

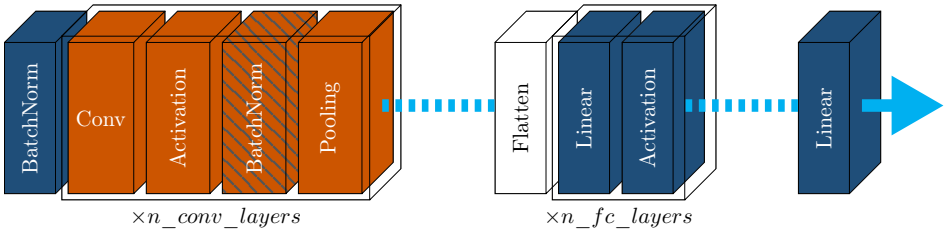


Figure 21: Hypermodel of a Convolutional Neural Network.

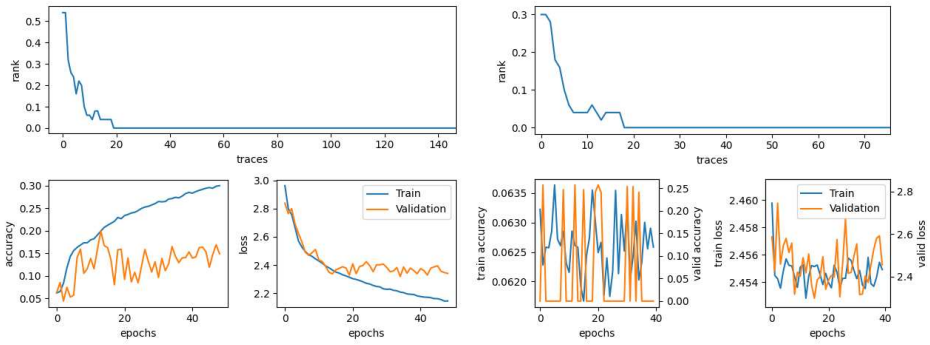
blocks is zero, the network consists only of fully connected layers, and we call it a multilayer perceptron (MLP).

The hyperparameter search space is described according to the network architecture rules in Table 3.

Hyperparameter	**Range**	**Step**
learning_rate	$[1e-5, \ldots, 1e-3]$	Log
activation_function	[relu, selu, tanh]	None
n_conv_layers	$[0, \ldots, 3]$	1
kernel_size	$[3, \ldots, 80]$ (for each conv layer)	1
n_fc_layers	$[1, \ldots, 5]$	1
size_fc_layers	$[10, \ldots, 500]$ (for each fc layer)	10

Table 3: Hyperparameter space explored in the optimization process

We train models on our unprotected Ascon dataset for the S-box output leakage model, as discussed in Section 4.4, for which the leakage function should be more adapted to the use of a random key traceset during the training phase. From the model search, the best-found model has two convolution blocks with kernel sizes of 16 and 11, respectively, and two fully connected layers of 100 and 50 neurons, respectively. The number of epochs to reach the fastest guessing entropy of zero is 50 epochs with a learning rate of $1e-5$, and the attack using this model reaches a guessing entropy of zero after 20 attack traces. In the results shown in Figure 22a, we can see the guessing entropy with an increasing number of traces on top and the accuracy and loss of the model during training for the training and validation sets at the bottom. While the accuracy and loss are sometimes misleading to understand the performance of a model for side-channel analysis [Pic+23], we can observe that the training loss does decrease together with the validation loss, indicating a generalization of the model on validation data.



(a) Reference implementation.

(b) Protected implementation.

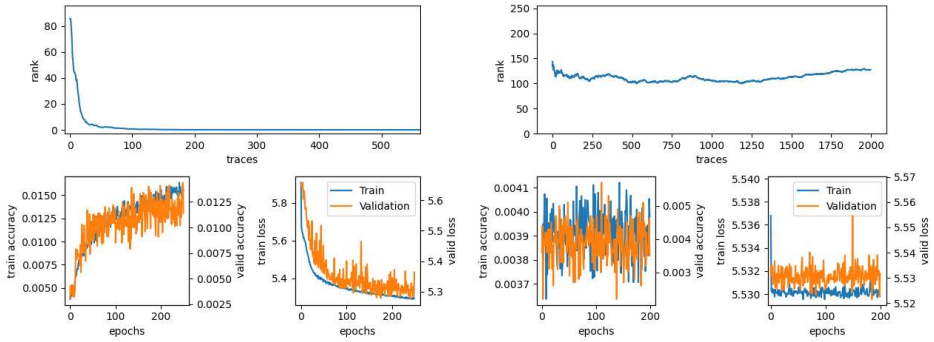
Figure 22: Results of the best-found CNN model with S-box output leakage.

The same training is applied to the first-order masking Ascon dataset using the same S-box output leakage model. The best model found has two convolution blocks with kernel sizes 27 and 18 and five fully connected layers of sizes 110, 230, 20, 140, and 500. The number of epochs to reach the fastest guessing entropy of zero is 20 with a learning rate of $1e-5$, and it reaches a guessing entropy of zero after 15 traces. The results of the model are shown in Figure 22b.

When using the 8-bit register S-box output leakage model, we can find a good model for the unprotected dataset. With this leakage model, the partial key that is targeted is 8-bit long; thus, the guessing entropy we obtain ranges in 2^8. The convergence to GE zero is reached after 200 traces. In Figure 23a, we show the results of the best-found model. The model is composed of two convolution blocks with kernel sizes 30 and 5, and four fully connected layers of sizes 500, 470, 10, and 480. The number of epochs to reach the fastest guessing entropy of zero is 160, with a learning rate of $1e-5$. The accuracy and loss of the model indicate that the model learns the leakage and stabilizes at the latest stage.

However, the same leakage model cannot lead to a satisfying model when training with the protected dataset. Indeed, in Figure 23b, it can be seen that the attack does not lead to a model capable of reducing the guessing entropy of the key. The rank value for the correct key-byte guess stays around the value 128, attesting to a random output prediction from the model. The accuracy and loss of the model show that the model does not learn from leakage at all, as the values remain stable from the first epochs until the last trained epoch. It can be concluded that the masking scheme is effective in protecting this specific leakage, even for DLSCA.

Still, the best-found model for the S-box output leakage model on the unprotected dataset is better than the CPA. The CPA can recover the key after 800



(a) Reference implementation.

(b) Protected implementation.

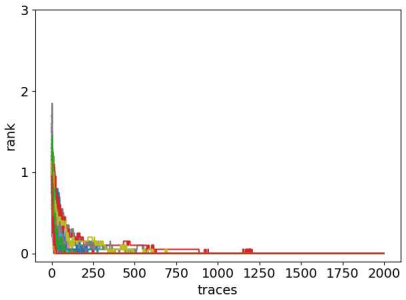
Figure 23: Results of the best-found CNN model with 8-bit S-box register output leakage model.

traces, while the best-found model can recover the key after 20 traces when exploiting leakage of the S-box output state. On the protected implementation, the best-found model for the S-box output leakage model can do partial key recovery with 20 traces, while CPA cannot recover the key, even after 60 000 traces. Our proposed DLSCA is better than CPA for the unprotected dataset, as it can recover the partial keys with 40 times fewer traces and can also do partial key recovery on the protected dataset with the same effort, while CPA is unsuccessful. Still, the question remains whether this result can be improved.

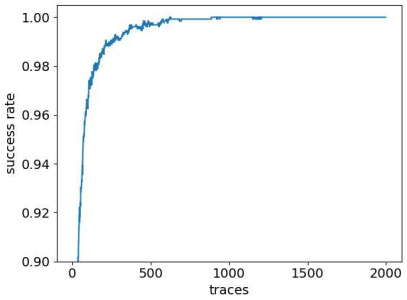
4.6 MULTI-TASK RESULTS

The previous models all considered only a partial key. It is possible to build a model with multiple outputs, where each would estimate a different partial key and thus obtain the full key of the attacked dataset with a single model evaluation. This problem is called multi-task learning. The idea resides in the fact that a CNN can extract features from the learning set and use the different feature maps independently to output probabilities for different tasks with separated MLPs. Some works have successfully applied multi-task models to the well-known masked AES dataset like ASCAD [MO23b; MO23a; MS23] as discussed in Section 4.3.

To construct a multi-task model, we follow the same architecture as in Figure 21, but the output of the flatten layer is connected to multiple independent fully connected models in parallel. Each fully connected model will output the probability for a partial key (as in the previous section) and form an oracle for the full key guess. The training process of a multi-task model is similar to a single-



(a) Guessing entropy of the multi-task CNN model for every partial key.



(b) Success rate of the full key recovery using the multi-task model.

Figure 24: Multi-task results on unprotected dataset.

task model. For each output of the fully connected models, we construct labels corresponding to traces in the dataset for the given partial key, and the loss function is computed as the sum of the categorical cross-entropy of every branch. The backward loop during the training phase will update the weights of each fully connected model and the convolution layers to fit the leakage function for every output simultaneously.

In Figure 24a, we can see the guessing entropy of individual partial keys from the multi-task model trained on the unprotected dataset. Each line in the figure represents the guessing entropy of a partial key. We can observe that all partial keys converge to a guessing entropy of zero but at different speeds. Note that all partial keys should reach a guessing entropy of zero to obtain a successful multi-task attack.

In Figure 24b, we present the attack's success rate. A success rate of one is reached when the model correctly evaluates every partial key correctly for a given test traceset. The success rate reaches one after an average of 1 000 traces. Compared to the results obtained from the CPA attack on the same dataset, the multi-task model can obtain the key with $8\times$ fewer traces.

For the protected dataset, we use a multi-task model with the same hyperparameters obtained for the best single-task model. In Figure 25, we see the guessing entropy of every partial key of the multi-task model. While some partial keys converge to a guessing entropy of zero, others converge to a fixed position with consistent errors. The errors stem from the prediction of the model that does not output random ranking for every trace (as a poorly trained model would) but ranks the correct key candidate at a fixed position for every trace. This position seems to be different for all tasks and ranges between all values of the ranking vector.

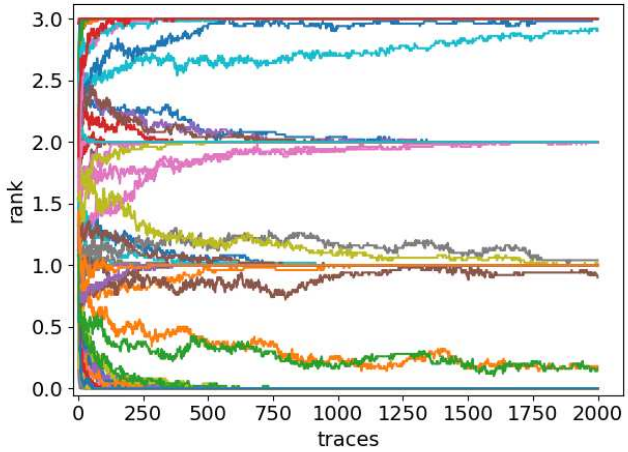


Figure 25: Guessing entropy of the multi-task CNN model for every partial key on the masked implementation at the end of model training.

The value of the error for different partial keys is evenly distributed and leads to a random success rate when aggregating all partial key results. The multi-task model trained on the protected dataset can recover some partial keys but cannot generalize the knowledge for all partial keys.

Compared to DLSCA on AES, the number of output classes is reduced due to the fact that our S-box output leakage function is a value between 0 and 2^3, instead of 2^8 as for the AES S-box. The ranking of partial keys in the guessing vector makes a bigger difference when the difference between the probabilities is smaller (i.e., when the prediction is difficult). When the correct key ranks in the middle of the guessing vector, the attack is not distinguishable from a random guessing attack. This behavior explains why only fewer very good fitting models can lead to a successful attack can be used and where medium fitting models can lead to a random success rate. For AES, a model that systematically ranks the correct key candidate in the first five or ten positions of the guessing vector can lead to a successful attack when considering enough attack traces, while for Ascon, a model that ranks the correct key candidate at the third position will not lead to a better attack than a random guess.

4.7 CONCLUSIONS AND FUTURE WORK

In this paper, we have evaluated the side-channel resistance of Ascon implementations against CPA and DLSCA. We have shown two different leakages that can be

exploited: the S-box output and the register of the state after the S-box operation. Our results show it is possible to obtain successful deep learning attacks for both leakage models on the unprotected Ascon dataset. The best-found model for the S-box output leakage model can recover the key after 20 traces, and the best-found model for the register leakage model can recover the key after 200 traces. Both leakage models are better than the results obtained from the CPA attack. On the protected implementation of Ascon, our results show that the masking scheme is protects against the leakage of the register output, as we cannot obtain a fitting model for the attack on this leakage. However, the masking does not prevent the S-box output leakage model. The best-found model for the S-box output leakage model can recover a partial key after only 20 traces.

This work gives the first example of a DLSCA on a protected software implementation of Ascon. The exploited leakages we present are not specific to software and could be, depending on the implementation, applied to hardware. In future work, we plan to explore protected hardware implementation of Ascon with the presented leakage models and model optimization methods. It could also be interesting to apply non-profiled DLSCA to Ascon.

Part III

DEEP LEARNING SIDE-CHANNEL ANALYSIS OF PUBLIC KEY CRYPTOGRAPHY

5

ONE TRACE IS ALL IT TAKES: MACHINE LEARNING-BASED SIDE-CHANNEL ATTACK ON EDDSA

This chapter is based on [WPB], a joint work with Stjepan Picek and Lejla Batina, that has been presented in 2019 at the International Conference on Security, Privacy and Applied Cryptography Engineering (SPACE).

CONTENT OF THIS CHAPTER

5.1 INTRODUCTION

Cryptographic algorithms ensure the security of a system (e.g., communication on a network or payment with a smartcard), by providing security features (e.g., authenticity and non-repudiation). However, implementations of those algorithms can fail during the engineering process and present flaws, leaking secret information over side-channels, even for the strongest protocols. Side-channel analysis (SCA) designates a set of signal processing techniques targeting the execution of cryptographic implementations, evaluating a system's security.

Since Differential Power Analysis by Kocher et al. [KJJ99], many other powerful SCAs have been successfully used to break all cryptographic algorithms, including recent machine learning approaches, on both symmetric key cryptography [CRR02; Heu+17; Kim+19b; LBM14; MPP16; Pic+19; Pic+17; Pro+18] and public-key cryptography [MO08; PZS17]. Among all SCAs, profiling attacks are the most powerful provided that the attacker has access to a clone device with full control that can be profiled offline, to later use this knowledge on another device

during the attack phase. Template attack [CRR02] has been the most popular instance of profiling attacks, but in recent years, new techniques based on machine learning were able to outperform template attack and break implementations protected with countermeasures. However, most of those results are obtained on block ciphers implementations (and more precisely on AES) and there are almost no results considering machine learning (deep learning) on public-key cryptography.

In this paper, we attack the digital signature algorithm Ed25519 as implemented in WolfSSL on an STM32F4 microcontroller, and we also compare the results obtained from different profiling attacks. To that end, we consider several machine learning techniques (i.e., Random Forest, Support Vector Machines, and Convolutional Neural Network) that have been proved strong in related work (albeit mostly on block ciphers) and template attack, which we consider the standard technique and a baseline setting.

5.1.1 *Related Work*

Template attacks (TAs) have been introduced by Chari et al. in 2003 [CRR02] as the most powerful SCA in the information-theoretic point of view and became a standard tool for profiling SCA. As straightforward implementations of TA can lead to computationally intensive computation, one option for more efficient computation is to use only a single covariance matrix, and is referred as the so-called pooled template attack presented by Choudary and Kuhn [CK13] where they were able to template a LOAD instruction and recover all 8 bits treated with a guessing entropy of 0. Several works applied machine learning methods to SCA of block ciphers because of their resemblance to general profiling techniques. Two methods stand out particularly in profiling SCA, namely Support Vector Machines (see, e.g., [Pic+17; MPP16; SH12; LBM14]) and Random Forest (see, e.g., [Heu+17; Pic+19; SH12]). With the general evolution in the field of deep learning, more and more works deal with neural networks for SCA and often show top performance. There, most of the research concentrated on either multilayer perceptron or convolutional neural networks [MPP16; Pic+18b; CDP17; CJ19].

There is a large portion of works considering profiling techniques for block ciphers, but there is much less for public-key cryptography. Lerman et al. considered template attack and several machine learning techniques to attack RSA. However, the targeted implementation was not secure, which makes the comparison with non-machine learning techniques less favorable [LBM14]. Nascimento et al. applied a horizontal attack on ECC implementation for AVR ATmega microcontroller targeting the side-channel leakage of cmov operation. Their approach to side-channel is similar to ours, but they don't use deep learning in the analy-

sis [Nas+17]. Poussier et al. used horizontal attacks and linear regression to conduct an attack on ECC implementations, but their approach cannot be classified as deep learning [PZS17]. Carbone et al. used deep learning to attack a secure implementation of RSA [Car+19]. Previous work has shown TA to be efficient for attacking SPA-resistant ECDSA with P192 NIST curve on 32-bit microcontroller [MO08].

5.1.2 *Contributions*

There are two main contributions of this paper:

1. We present a comprehensive analysis of several profiling attacks by exploring different sets of hyper-parameters that permit to obtain the best results for each method. This evaluation can be helpful when deciding on an optimal strategy for machine learning and in particular, deep learning attacks on implementations of public-key cryptography.

2. We consider elliptic curve cryptography (actually EdDSA using curve Curve25519) and profiling attacks where we show that such techniques, and especially the convolutional neural networks can be extremely powerful attacks.

Besides those contributions, we also present a publicly available dataset we developed for this work. We aim to make our results more reproducible but also motivate other researchers to publish their datasets for public-key cryptography. Indeed, while the SCA community realizes the lack of publicly available datasets for block ciphers (and tries to improve it), the situation for public-key cryptography seems to attract less attention despite even worse availability of codes, testbeds, and datasets.

5.2 PRELIMINARIES

In this section, we start by introducing the elliptic curve scalar multiplication operation and EdDSA algorithm.

5.2.1 *EdDSA*

In the context of public-key cryptography, one important feature is the authentication of a message between two parties. This feature ensures to party B that

Table 4: Notation for EdDSA

Name	Symbol
Private key	k
Private scalar	a (first part of $H(k)$).
Auxiliary key	b (last part of $H(k)$).
Ephemeral key	r
Message	M

party A has indeed sent a message M and that this message is original and unaltered. Message authentication can be performed by Digital Signature Algorithms (DSA). DSA creates a signature pair (R, S) for proving that a message M was emitted by the known party A, unaltered and that A cannot repudiate. For security reasons and computational speed, public-key cryptography has turned toward Elliptic Curves based cryptography (ECC) as it tends to become the successor of RSA for public-key cryptography because it can meet higher security levels with smaller key lengths. ECC is based on the Elliptic Curve Discrete Logarithm Problem (ECDLP), which states that it is easy and hence efficient to compute $Q = k \cdot P$, but it is difficult to find k knowing Q and P.

EdDSA [Ber+12] is a variant of the Schnorr digital signature scheme [Sch91] using Twisted Edward Curves, a subgroup of elliptic curves that uses unified formulas, enabling speed-ups for specific curve parameters. This algorithm proposes a deterministic generation of the ephemeral key, different for every different message, to prevent flaws from a predictable random number generator. The ephemeral key r is made of the hash value of the message M and the auxiliary key b, generating a unique ephemeral public key R for every message.

EdDSA, when using parameters of Curve25519 is referred to as Ed25519 [Ber16]. EdDSA scheme for signature generation and verification is described in Algorithm 1, where the notation $(x, \ldots, y)$ denotes the concatenation of the elements. The notation used in Algorithm 1 is given in Table 4.

After the signature generation, party A sends (M, R, S), i.e., the message along with the signature pair (R, S) to B. The verification of the signature is done by B with Steps 10 to 11. If the last equation is verified, it represents a point on the elliptic curve and the signature is correct, ensuring that the message can be trusted as an authentic message from A.

Algorithm 1 EdDSA Signature generating and verification

Keypair Generation (k, P)**:** (Used once, first time private key is used.)

1: Hash k such that $H(k) = (h_0, h_1, \ldots, h_{2u-1}) = (a, b)$
2: $a = (h_0, \ldots, h_{u-1})$, interpret as integer in little-endian notation
3: $b = (h_u, \ldots, h_{2u-1})$
4: Compute public key: $P = aB$.

Signature Generation:

5: Compute ephemeral private key $r = H(b, M)$.
6: Compute ephemeral public key $R = rB$.
7: Compute $h = H(R, P, M) \mod l$.
8: Compute: $S = (r + ha) \mod l$.
9: Signature pair (R, S)

Signature Verification:

10: Compute $h = H(R, P, M)$
11: Verify if $8SB = 8R + 8hP$ holds in E

5.2.2 *Elliptic Curve Scalar Multiplication*

The security of ECC algorithms depends on the ability to compute a point multiplication and the presumed inability to reverse the computation to retrieve the multiplicand given the original and product points. This security is strengthened with a greater prime order of the underlying finite field. In our attack, we aim to extract the ephemeral key r from its scalar multiplication with the Elliptic Curve base point B (see step 5 in Algorithm 1). To understand how this attack works, we decompose this computation as implemented in the case of WolfSSL Ed25519.

The implementation of Ed25519 in WolfSSL is based on the work of Bernstein et al. [Ber+12]. The implementation of elliptic curve scalar multiplication is a window-based method with radix-16, making use of a precomputed table containing results of the scalar multiplication of $16^i|r_i| \cdot B$, where $r_i \in [-8, 7] \cap \mathbb{Z}$ and B is the base point of Curve25519 (see Algorithm 2). This method is popular because of its trade-off between memory usage and computation speed, but also because the implementation is time-constant and does not feature any branch condition nor array indices and hence is presumably secure against timing attack. Leaking information from the corresponding value loaded from the memory with a function *ge_select* is used here to recover e and hence can be used to easily connect to the ephemeral key r. More details are given in the remainder of this paper.

5.3 ATTACKER MODEL

The general warning for implementations of ECDSA is to select different ephemeral private keys r for different signature. The flaw of using the same r for different messages happens since the two corresponding signatures would result in two signature pairs (R, S) and (R, S') for messages M and M', respectively. Then, an attacker can use this information to recover r as $r = (z - z')(S - S')^{-1}$ (with z and z', few bits of $H(M)$ and $H(M')$ interpreted as integers). Finally, to recover the private scalar a required to forge signatures, the attacker can trivially compute $a = R^{-1}(Sr - z)$.

Here, the aim of the attacker is the same as for every ECDSA attack: recover the secret scalar a. The difference is that the attacker cannot acquire two signatures with the same random r, but can still recover the secret scalar in two different ways. One method would consist of attacking the implementation of the hash function to recover b from the computation of ephemeral private key [Sam+18]. Another method (developed in this paper) attacks the implementation of the scalar multiplication during the computation of the ephemeral public key. With this method, the attacker collects side-channel traces of each computation since r is different in every message. This paper shows that even with a single attack trace, the attacker can recover private scalar with high confidence where we provide a comparison with different state-of-the-art profiling SCA.

5.4 DATASET GENERATION

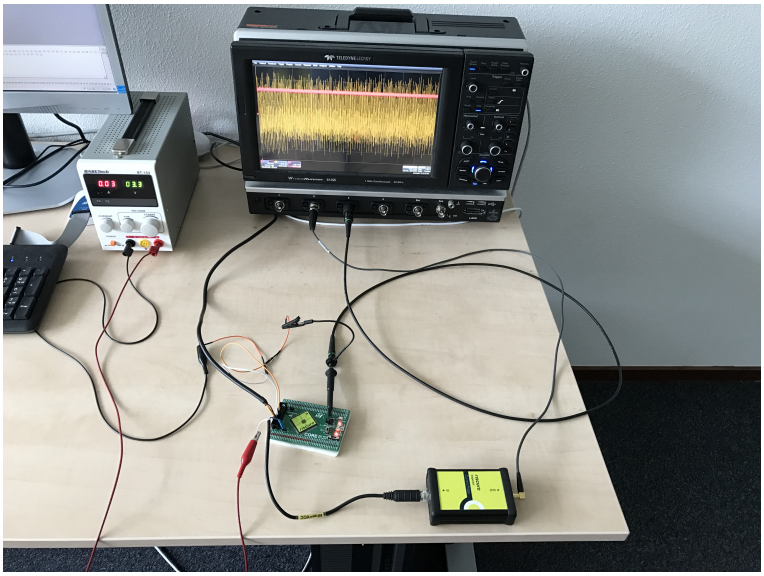
In this section, we first present the measurement setup and explain the methodology for creating a dataset from the power traces obtained with our setup (see Figure 26).

5.4.1 *Measurement Setup*

The device under attack is a Piñata development board developed by Riscure to perform SCA evaluations [1]. The board is based on a 32-bit STM32F4 microcontroller with an ARM-based architecture, running at the clock frequency of 168 MHz. The board is modified to perform SCA through power consumption. The target is Ed25519 implementation of WolfSSL 3.10.2. As WolfSSL is an open-source library written in C, we have a fully transparent and controllable implementation for the profiling phase.

1 Pinata Board: https://www.riscure.com/product/pinata-training-target/

Figure 26: The measurement setup



Power consumption is measured with a current probe [2] placed between the power source and the board's power supply source. Power measurements are obtained with a Lecroy Waverunner z610i oscilloscope. The measures are performed with a sampling frequency of 1.025 GHz and the trigger is implemented with an I/O pin of the board around the *ge_select* function (see Algorithm 2) to retrieve a part of the key **e**.

5.4.2 *Dataset*

To evaluate the attack proposed in this paper and to facilitate reproducible experiments, we present the dataset we built for this purpose [Dat]. We follow the same format for the dataset as in recently presented ASCAD database [Pro+18]. For this attack, we profile the EC scalar multiplication with the ephemeral key with the base point of curve Ed25519. Regarding the implementation of this operation for our target, we focus on the profiling of one function of the operation as it is more challenging by exploiting less information. We focus on the Lookup Table (LUT) operation used to fetch the precomputed chunks of the result in a table stored in memory. For speed reasons, the 256 bits scalar/ephemeral secret key r is interpreted in slices of 4-bits (nibbles) $e[i], i \in [0, 63]$, and to compute $R = rB$, the field multiplication with the base point B, we would have to com-

2 Current Probe: https://www.riscure.com/product/current-probe/

Algorithm 2 Elliptic curve scalar multiplication with base point [BSS99]

Require: R, a with $a = a[0] + 256 * a[1] + ... + 256^{31} a[31]$
Ensure: $H(a, s, m)$

for $i = 0; i < 32; ++i$ **do**
  $e[2i + 0] = (a[i] >> 0\&15);$
  $e[2i + 1] = (a[i] >> 4)\&15;$
end for
$carry = 0;$
for $i = 0; i < 63; ++i$ **do**
  $e[i] += carry;$
  $carry = (e[i] + 8);$
  $carry >>= 4;$
  $e[i] -= carry << 4;$
end for
$e[63] += carry;$ ▷ $\forall i < 64, -8 \leq e[i] \leq 8$
$ge\_p3\_0(h);$
for $i = 1; i < 64; i += 2$ **do**
  $ge\_select(\&t, i/2, e[i]);$ ▷ load from precomputed table $(e[i] \cdot 16^i) \cdot B$ in E.
  $ge\_madd(\&r, R, \&t); ge\_p1p1\_to\_p3(R, \&r);$
end for
$ge\_p3\_dbl(\&r, R); ge\_p1p1\_to\_p2(\&s, \&r);$
$ge\_p2\_dbl(\&r, \&s); ge\_p1p1\_to\_p2(\&s, \&r);$
$ge\_p2\_dbl(\&r, \&s); ge\_p1p1\_to\_p2(\&s, \&r);$
$ge\_p2\_dbl(\&r, \&s); ge\_p1p1\_to\_p3(R, \&r);$
for $i = 0; i < 64; i += 2$ **do**
  $ge\_select(\&t, i/2, e[i]);$ ▷ load from precomputed table $(e[i] \cdot 16^i) \cdot B$ in E.
  $ge\_madd(\&r, R, \&t); ge\_p1p1\_to\_p3(R, \&r);$
end for

Table 5: Organization of the database.

DATABASE					
	ATTACK_TRACES		PROFILING_TRACES		
	TRACES	trace_1[1 000] ... trace_n_a[1 000]	TRACES	trace_1[1 000] ... trace_n_p[1 000]	
	LABELS	label_1[1] ... label_n_a[1]	LABELS	label_1[1] ... label_n_p[1]	

pute $\sum_{i=0}^{63} e[i]16^i B$. As multiplication is resource consuming, the implementation stores the results for every nibble number i and nibble value $e[i]$ in a precomputed LUT and loads corresponding chunks when needed.

Each trace in the database is represented by a tuple composed of one power trace and its corresponding label (class). The database is composed of two groups: the first group is PROFILING_TRACES, which contains n_p tuples. The second group is ATTACK_TRACES, which contains n_a tuples (see Table 5). In total, there are 6 400 labeled traces. We divide the traces in 80/20 ratio for profiling/attacking groups, and consequently, have $n_p = 5\,120$ and $n_a = 1\,280$. The profiling group is additionally divided in 80/20 ratio for training and validation sets.

A group contains two datasets: TRACES and LABELS. The dataset TRACES contains the raw traces recorded from different nibbles during the encryption. Each trace contains 1 000 samples and represents the relevant information of one nibble encryption. The dataset LABELS contains the correct subkey candidate for the corresponding trace. In total, there are 16 classes since we consider all possible nibble values.

To the best of our knowledge, besides the dataset we presented here, there is only one publicly available dataset for SCA on public-key cryptography on elliptic curves. Tuveri et al. conducted a side-channel analysis of SM2 (a digital signature algorithm) public-key cryptography suite where they consider various side channels [Tuv+18]. Additionally, the authors published EM side-channel measurements of elliptic curve point multiplication[1]. We note that due to the choice of the suite (SM2 is not an international standard), this dataset is difficult to compare with ours.

1 available at https://zenodo.org/record/1436828#.XRhmfY-xWrw

5.5 EXPERIMENTAL SETTING AND RESULTS

To examine the feasibility and performance of our attack, we present different settings for power analysis and use two different metrics. We first compare the performance by using the accuracy metric since it is a standard metric in machine learning. The second metric we use is the success rate as it is an SCA metric that gives a more concrete idea on the power of the attacker [SMY09]. Note that we assume the attacker who can collect as many power traces as she wants and that the profiling phase is nearly-perfect as also suggested by Lerman et al. [Ler+15].

5.5.1 *Hyperparameters Choice*

Here we discuss the choice of hyperparameters for each method we consider in this paper.

TA: Classical Template Attack is applied with pooled covariance [CK13]. Profiling phase is repeated for a different choice of points of interest (POI).

RF: Hyper-parameter optimization is applied to tune the number of decision trees used in Random Forest. We consider the following number of trees: 50, 100, 500. The best number of decision trees is 100 with no PCA and 500 when PCA is applied for 10 and 656 POI.

SVM: For the linear kernel, the hyperparameter to optimize is the penalty parameter C. We search for the best C among a range of $[1, 10^5]$ in logarithmic space. In the case of the radial basis function (RBF) kernel, we have two hyperparameters to tune: the penalty C and the kernel coefficient γ. The search for best hyperparameters is done within $C = [1, 10^5]$ and $\gamma = [-5, 2]$ in logarithmic spaces. We consider only those hyperparameters that give the best scores for each choice of POI (see Table 6).

CNN: The chosen hyperparameters for *VGG-16* follows several rules that have been adapted for SCA in [Kim+19b] or [Pro+18] and that we describe here:

1. The model is composed of several convolution blocks and ends with a dropout layer followed by a fully connected layer and an output layer with the Softmax activation function.

2. Convolutional and fully-connected layers use the ReLU activation function.

Table 6: Chosen hyperparameters for SVM

Number of features	Kernel	C	γ
1 000	linear	1 000	–
	rbf	1 000	1
656	linear	1 000	–
	rbf	1 000	1
10	linear	1 333	–
	rbf	1 000	1.23

Table 7: Architecture of the CNN

Hyper-parameter	Value
Input shape	$(1000, 1)$
Convolution layers	$(8, 16, 32, 64, 128, 256, 512, 512, 512)$
Pooling type	Max
Fully-connected layers	512
Dropout rate	0.5

3. A convolution block is composed of one convolution layer followed by a pooling layer.

4. An additional batch normalization layer is applied for every odd-numbered convolution block and is preceding the pooling layer.

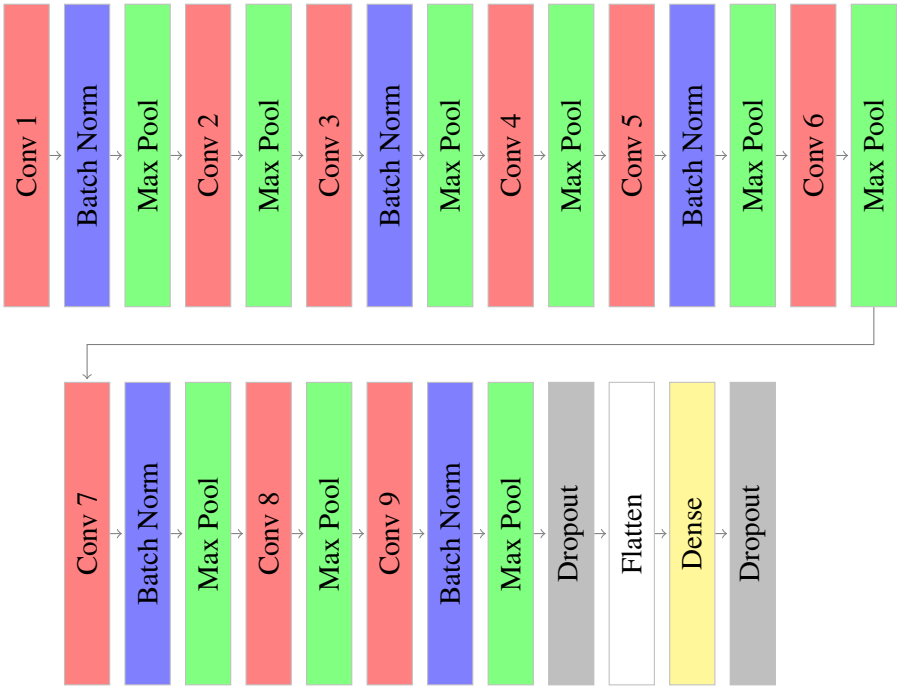
5. The chosen filter size for convolution layers is fixed on size 3.

6. The number of filters $n_{filters,i}$ in a convolution block i keeps increasing according to the following rule: $n_{filters,i} = max(2^i \cdot n_{filters,1}, 512)$ for every layer $i \geq 0$ and we choose $n_{filters,1} = 8$

7. The stride of the pooling layers is of size 2 and halves the input data for each block.

8. Convolution blocks follow each other until the size of the input data is reduced to 1.

Figure 27: CNN architecture as implemented in Keras. This architecture consists of 9 convolutional layers followed by max pooling layers. For each odd convolutional layer, there is a batch normalization layer before the pooling layer. At the end of the network, there is one fully connected layer.



5.5.2 *Dimensionality Reduction*

For computational reasons, one may want to select points of interest (POI) and consequently, we explore several different setting where we either use all the features in a trace or we conduct dimensionality reduction. Here, for dimensionality reduction, we use Principal Component Analysis (PCA) [Boh+03]. Principal component analysis (PCA) is a well-known linear dimensionality reduction method that may use Singular Value Decomposition (SVD) of the data matrix to project it to a lower dimensional space. PCA creates a new set of features (called principal components) that are linearly uncorrelated, orthogonal, and form a new coordinate system. The number of components equals the number of original features. The components are arranged in a way that the first component covers the largest variance by a projection of the original data and the subsequent components cover less and less of the remaining data variance. The projection contains (weighted) contributions from all the original features. Not all principal components need to be kept in the transformed dataset. Since the components are sorted by the variance covered, the number of kept components, designated with L, maximizes the variance in the original data and minimizes the reconstruction error of the data transformation.

Note, while PCA is meant to select the principal information from a data, there is no guarantee that the reduced data form will give better results for profiling attacks than its complete form. We apply PCA to have the least possible number of points of interest that maximize the score from TA (10 points of interest) and the number of POI using a Bayesian model selection that estimates the dimensionality of the data based on heuristics (see [Min01]). After an automatic selection of the number of components to use, we have 656 points of interest.

5.5.3 *Results*

In Table 8, we give results for different profiling methods when considering recovery of a single nibble of the key. We can see that all profiling techniques reach very good performance with all accuracy scores above 95%. Still, some differences can be noted. When considering all available features (1 000), CNN performs the best and has the accuracy of 100%. Both linear and rbf SVM and RF have the same accuracy. The performance of SVM is interesting since the same value for linear and rbf kernel indicates that there is no advantage of going into higher dimensional space, which means that the classes are linearly separable. Finally, TA performs the worst of all considered techniques.

Algorithm	1 000 features	656 PCA components	10 PCA components
TA	0.9977	0.9984	0.9830
RF	0.9992	0.9914	0.9937
SVM (linear)	0.9992	0.9992	0.995
SVM (rbf)	0.9992	0.9992	0.995
CNN	1.00	0.95	0.96

Table 8: Accuracy for the different methods obtained on the attacking dataset.

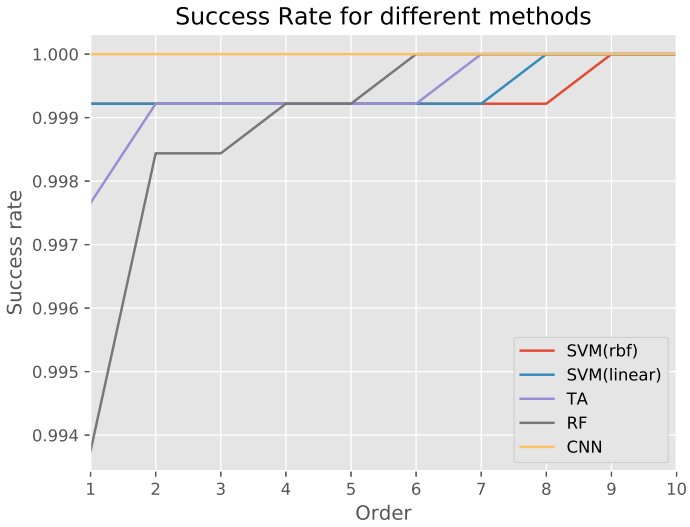
Applying PCA to the dataset results in lower accuracy scores. More precisely, when considering the results with PCA that uses an optimal number of components (656), we see that the results for TA slightly improve while the results for RF and CNN decrease. While the drop in the performance for RF is small, CNN has a significant performance drop and becomes the worst performing technique. SVM with both kernels retains the same accuracy level as for the full number of features. Finally, when considering the scenario where we take only 10 most important components from PCA, all the results deteriorate when compared with the results with 1 000 features. Interestingly, CNN performs better with only 10 most important components than with 656 components but is still the worst performing technique from all the considered ones.

To conclude, all techniques exhibit very good performance, but CNN is the best if no dimensionality reduction is done. There, the maximum accuracy is obtained after only a few epochs (see Figures 29 and 30). If there is dimensionality reduction, CNN shows a quick performance deterioration. This behavior should not come as a surprise since CNNs are usually used with the raw features (i.e., no pre-processing). In fact, applying such techniques could reduce the performance due to a loss of information and changes in the spatial representation of features. Interestingly, TA is never the best technique while SVM and RF show good and stable behavior for all feature set sizes.

In Figure 28, we give the success rate with orders up to 10 for all profiling methods on the dataset without applying PCA. Note, a success rate of order o is the probability that the correct subkey is ranked among the firsts o candidates of the guessing vector. While CNN has a hundred percent success rate of order 1, other methods achieve the perfect score only for orders greater than 6.

The results for all methods are similar in the recovery of a single nibble from the key. If we want to have an idea of how good these methods are for the recovery of a full 256-bit key, we must apply the classification on the successive 64 nibbles. We can have an intuitive glimpse of the resulting accuracy P_c with the cumulative

Figure 28: Success rate results.



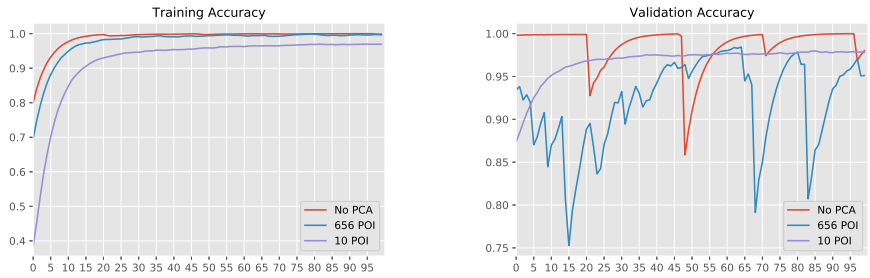
probability of the probability of one nibble $P_s : P_c = \Pi_{64} P_s$ (see Table 9). The cumulative accuracy obtained in such a way can be interpreted as the predictive first-order success rate of a full key for the different methods in terms of a security metric.

From these results, we can observe that the best result is obtained with CNN when there is no dimensionality reduction. Other machine learning methods and TA are nonetheless powerful profiling attacks with up to 95 and 90% performance to recover the full key on the first guess with the best choice of hyperparameters and dimensionality reduction. Note the low accuracy value for CNN when using 656 PCA components: this result is obtained as the accuracy of CNN for a single nibble raised to the power of 64 (since now we consider 64 nibbles). When considering the results after dimensionality reduction, we see that SVM is the best performing technique, which is especially apparent when using only 10 PCA components. Finally, we observe again that TA is never the best performing technique.

As it can be observed from Figures 29 and 30, both scenarios without dimensionality reduction and dimensionality reduction to 656 components, reach the maximal performance very fast. On the other hand, the scenario with 10 PCA components does not seem to reach the maximal performance within 100 epochs since we see that the validation accuracy does not start to decrease. Still, even longer experiments do not show further improvement in the performance, which

Algorithm	1 000 features	656 PCA components	10 PCA components
TA	0.86	0.90	0.33
RF	0.95	0.57	0.66
SVM (linear)	0.95	0.95	0.72
SVM (rbf)	0.95	0.95	0.72
CNN	1.00	0.03	0.07

Table 9: Cumulative probabilities of the profiling methods.



(a) Training Accuracy (b) Validation Accuracy

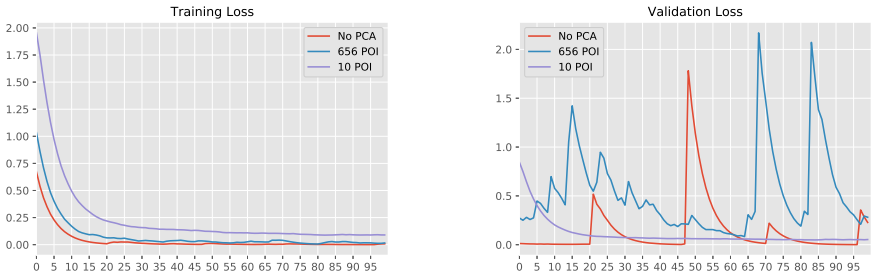
Figure 29: Accuracy of the CNN method over 100 epochs

indicates that the network simply learned all that is possible and that there is no more information that can be used to further increase the performance.

5.5.3.1 *Choosing the Minimum Number of Traces for Training on CNN.*

As it is possible to obtain a perfect profiling phase on our dataset using CNN, we focus here on finding the smallest training set that gives a success rate of 1. More precisely, we evaluate the attacker in a more restricted setting [PHG19]. To do so, we first reduce the size of the training set to k number of traces per class (to always have a balanced distribution of the traces) and then we gradually increase it to find out when the success rate reaches 1. In Table 10, we give the results obtained after one hundred epochs.

Interestingly, it turns out that 30 traces per class for training the CNN is enough to reach the perfect profiling of this dataset. At the same time, the additional experiments did not show good enough behavior with a lower number of traces per class. Note the scenario with only 10 traces per class where the validation accuracy is lower than the testing accuracy. This happens since we use only 20% of



(a) Training Loss

(b) Validation Loss

Figure 30: Loss of the CNN method over 100 epochs.

Table 10: Validation and test accuracy of CNN with an increasing number of training traces.

Number of traces per class k	10	20	30	50	100	300
Validation accuracy	0.937	1.0	1.0	1.0	1.0	1.0
Testing accuracy	0.992	0.992	1.0	1.0	1.0	1.0

the training set for the validation, which results in an extremely small validation set and consequently, less reliable results.

5.6 CONCLUSIONS AND FUTURE WORK

In this paper, we consider a number of profiling techniques to attack the Ed25519 implementation in WolfSSL. The results show that although several techniques perform well, convolutional neural networks are the best if no dimensionality reduction is done. In fact, in such a scenario, we can obtain the accuracy of 100%, which means that the attack is perfect in the sense that we obtain the full information with only a single trace in the attack phase. What is especially interesting is the fact that CNN used here is taken from related work (more precisely, CNN used for profiling SCA on AES) and is not further adapted to the scenario here. This indicates that CNNs can perform well over various scenarios in SCA. Finally, to obtain such results, we require only 30 measurements per class, which results in less than 500 measurements to reach a success rate of 1 with CNN.

The implementation of Ed25519 we attack in this work does not feature any countermeasure for SCA (that is, beyond constant-time implementation). In future

work, we plan to evaluate CNN for SCA on Ed25519 with different countermeasures to test the limits of CNN in the side-channel analysis.

6

SYSTEMATIC SIDE-CHANNEL ANALYSIS OF CURVE25519 WITH MACHINE LEARNING

This chapter is based on [Wei+20b], a joint work with Łukasz Chmielewski, Stjepan Picek and Lejla Batina, and was published in the journal of hardware and system security in 2020.

CONTENT OF THIS CHAPTER

6.1 INTRODUCTION

Various cyber-physical devices have become integral parts of our lives. They provide basic services, and as such, also need to fulfill appropriate security requirements. Designing such secure devices is not easy due to limited resources available for implementations, and the need to provide resilience against various attacks. In the last decades, implementation attacks emerged as real threats and the most potent attacks. In implementation attacks, the attacker does not aim at the weaknesses of an algorithm, but the weaknesses in implementations [MOP06]. One powerful category of implementations attacks is the profiled side-channel analysis (SCA) where the attacker has access to a profiling device she uses to learn about the leakage from the device under attack. Profiled SCA uses a broad set of methods to conduct the attack.

In the last few years, attacks based on the machine learning classification task have proved to be very successful when attacking symmetric-key cryptography [Kim+19b; LBM14; MPP16; Pic+19; Pro+18]. On the other hand, profiled

SCAs on public-key cryptography implementations are much more scarce [MO08; PZS17; Car+19].

While the current state-of-the-art results on profiled SCA and public-key cryptography suggest breaking targets with a relatively small effort, many questions remain unanswered. For instance, it is not yet clear what are the benefits of countermeasures against machine learning-based attacks. What is more, public-key cryptography has different use cases and parameters that also result in classification problems with significantly different number of classes one commonly encounters when attacking, e.g., block ciphers. Finally, in profiled SCA on symmetric ciphers, we are slowly moving away from scenarios where the only interesting aspect is the attack performance. Indeed, the SCA community is now becoming interested in questions like interpretability [MDP19; PEC19; VP19] and explainability [VPB19] of deep learning attacks, but also building methodologies [Zai+19] and frameworks [Pic+18a; PHG19] for objective analysis.

This paper considers profiled side-channel attacks on two implementations of scalar multiplication on one of the most popular elliptic curves for applications, i.e., Curve25519. The first implementation is the baseline implementation with the complete formulae as used for EdDSA in WolfSSl. The second implementation also includes several countermeasures. To evaluate the security of those implementations, we consider seven different profiled methods. Additionally, we investigate the influence of the dimensionality reduction technique. By doing this, we aim at filling the knowledge gap and give insights into the performance of different profiled methods. Finally, we compare the differences in the attack performance when considering protected and non-protected implementations.

This paper is based on the work "One Trace Is All It Takes: Machine Learning-Based Side-Channel Attack on EdDSA" [WPB19]. The main differences are:

1. We provide results for an additional target, protected with countermeasures.
2. We provide results for several more profiled methods and different dimensionality reduction steps.
3. We investigate the applicability of one visualization technique for deep learning when attacking public-key implementations.

The rest of this paper is organized as follows. In Section 6.2, we give details about EdDSA and scalar multiplication procedure. Afterwards, we discuss the profiled methods we use in our experiments. Section 6.3 provides details about the attacker model, the datasets we use, hyperparameter tuning, and dimensionality reduction. In Section 6.4, we provide experimental results for both targets. In Section 6.5, we discuss related works. Finally, in Section 6.6, we conclude the paper and offer some potential future research directions.

6.2 PRELIMINARIES

In this section, we start by introducing the elliptic curve scalar multiplication operation and the EdDSA algorithm. After that, we discuss profiling attacks that we use in our experiments.

6.2.1 *Elliptic Curve Digital Signature Algorithm*

In the context of public-key cryptography, one important feature is the (entity) authentication between two parties. This feature ensures to party B that party A has sent a message M and that this message is original and unaltered. Authentication can be performed by the Digital Signature Algorithm (DSA). Nowadays, public-key cryptography for constrained devices typically implies Elliptic Curves cryptography (ECC) as the successor of RSA because it achieves a higher security level with smaller key lengths saving the resources such as memory, power, and energy. The security of ECC algorithms is based on the difficulty of Elliptic Curve Discrete Logarithm Problem (ECDLP), which states that while it is easy and efficient to compute $Q = k \cdot P$, it is "difficult" to find k with knowledge of Q and P.

EdDSA [Ber+12] is a variant of the Schnorr digital signature scheme [Sch91] using Twisted Edward Curves, a subgroup of elliptic curves that uses unified formulas, enabling speed-ups for specific curve parameters. This algorithm proposes a deterministic generation of the ephemeral key, different for every message, to prevent flaws from a biased random number generator. The ephemeral key r is made of the hash value of the message M and the auxiliary key b, generating a unique ephemeral public key R for every message.

EdDSA, with the parameters of Curve25519, is referred to as Ed25519 [Ber16]. EdDSA scheme for signature generation and verification is described in Algorithm 3, where the notation $(x, \ldots, y)$ denotes the concatenation of the elements. The hash function H is SHA-512 [NIS15]. The key length is of size $u = 256$. We denote the private key with k, the private scalar a is the first part of the private key's hashed value, and the auxiliary key b is the second part. We denote the ephemeral key with r and M is the message.

After the signature generation, party A sends (M, R, S), i.e., the message along with the signature pair (R, S) to B. The verification of the signature is done by B with Steps 10 to 11. If the last equation is verified, it represents a point on the elliptic curve, and the signature is correct, ensuring that the message can be trusted as an authentic message from A.

Algorithm 3 EdDSA Signature generating and verification

Keypair Generation: (Used once, first time private key is used.)
Input: k, **Output:** a, b, P
1: Hash k such that $H(k) = (h_0, h_1, \ldots, h_{2u-1}) = (a, b)$
2: $a = (h_0, \ldots, h_{u-1})$, interpret as integer in little-endian notation
3: $b = (h_u, \ldots, h_{2u-1})$
4: Compute public key: $P = aG$.

Signature Generation:
Input: M, a, b, P **Output:** R, S
5: Compute ephemeral private key $r = H(b, M)$.
6: Compute ephemeral public key $R = rG$.
7: Compute $h = H(R, P, M) \mod l$.
8: Compute: $S = (r + ha) \mod l$.
9: Signature pair (R, S)

Signature Verification:
Input: M, P, R, S, **Output:** {True, False}
10: Compute $h = H(R, P, M)$
11: Verify if $8SG = 8R + 8hP$ holds in E

6.2.2 *Elliptic Curve Scalar Multiplication*

We focus on two types of implementations of EC scalar multiplication. The first implementation is of EdDSA using Ed25519 as in WolfSSL. This implementation is based on the work of Bernstein et al. [Ber+12] and is a window-based method with radix-16, making use of a precomputed table containing results of the scalar multiplication of $16^i|r_i| \cdot G$, where $r_i \in [-8, 7] \cap \mathbb{Z}$ and G is the base point of Curve25519. This method is popular because of its trade-off between memory usage and computation speed, but also because the implementation is time-constant and does not feature any branch condition nor array indices and hence is presumably secure against timing attacks.

Leaking information from the corresponding value loaded from memory with a function *ge_select* is here used to recover e and hence can be used to connect to the ephemeral key r easily. More details are given in the remainder of this paper. We can attack this implementation and extract the ephemeral key r from Step 5 in Algorithm 3.

The second implementation we focus on is the Montgomery Ladder scalar multiplication as used in μNaCl [D+15]. The implementation employs arithmetic-

based conditional swap and is additionally protected with projective coordinate re-randomization and scalar randomization. The traces used to analyze this implementation are obtained from a publicly available dataset [Chm20]. All details on this implementation, including the additional countermeasures, are described in [NC17].

6.2.3 *Profiling Attacks*

6.2.3.1 *Random Forest - RF*

Random Forest is an ensemble learning method that consists of a number of decision trees [Bre01]. Decision trees consist of combinations of Boolean decisions on a different random subset of attributes of input data (called bootstrap sampling). For each node of each tree, the best split is taken among these randomly chosen attributes. Random forest is a stochastic algorithm since it has two sources of randomness: bootstrap sampling and attribute selection at node splitting. While the random forest has several hyperparameters to tune, we investigate the influence of the number of trees in the forest, where we do not pose any limits on the tree size.

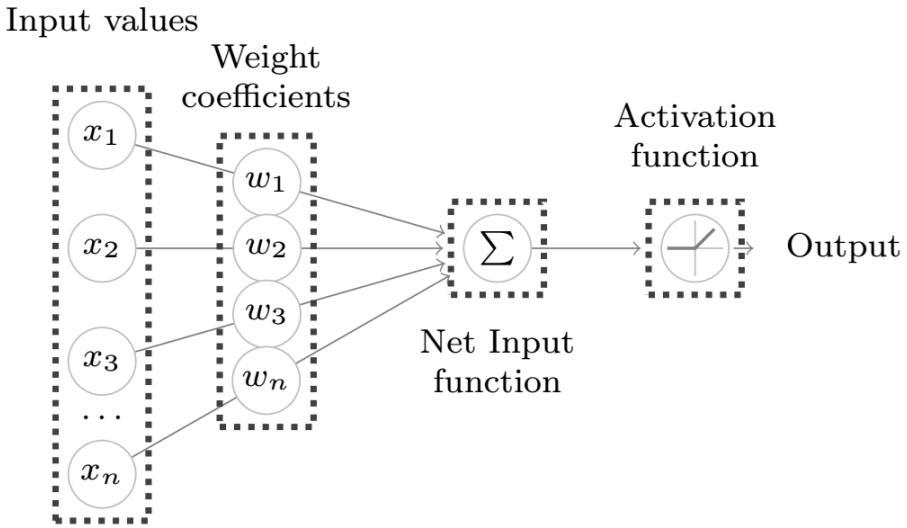
6.2.3.2 *Support Vector Machines - SVM*

Support Vector Machines is a kernel-based machine learning family of methods used to classify linearly separable and linearly inseparable data [Vap95]. The idea for linearly inseparable data is to transform them into a higher dimensional space using a kernel function, wherein the data can usually be classified with higher accuracy. The scikit-learn implementation we use considers libsvm's C-SVC classifier [Ped+11] that implements SMO-type algorithm [FCL05]. This implementation of SVM learning is widely used because it is simpler and faster compared to older methods. The multi-class support is handled according to a one-vs-one scheme. We investigate two variations of SVM: with a linear kernel and with a radial kernel. Linear kernel-based SVM has the penalty hyperparameter C of the error term. Radial kernel-based SVM has two significant hyperparameters to tune: the cost of the margin C and the kernel γ.

6.2.3.3 *Convolutional Neural Networks - CNNs*

CNNs, like other types of neural networks, have several layers where each layer is made up of neurons, as depicted in Figure 31. Every neuron in a layer computes a weighted combination of an input set by a net input function (e.g., the sum function in neurons of a fully-connected layer) from which a nonlinear activation

Figure 31: Anatomy of a neuron.



function produces an output. When the output is different from zero, we say that the neuron activation feeds the next layer as its input. Layers with a convolution function as the net input function are referred to as convolutional layers and are the core building blocks in a CNN. Pooling layers are commonly used after a convolution layer to sample down local regions and create spatial regions of interest. The last fully-connected layers of a CNN behave as a classifier for the extracted features from the inputs.

In this work, we start from the VGG-16 architecture introduced in [SZ14] for image recognition. This architecture was also recently applied for SCA on AES [Kim+19b] and EdDSA [WPB19]. This CNN architecture also uses the following elements:

1. Batch normalization to normalize the input layer by applying standard scaling on the activations of the previous layer.
2. Flatten layer to transform input data of rank greater than two into a one-dimensional feature vector used in the fully-connected layer.
3. Dropout (randomly dropping out units (both hidden and visible) in a neural network with a certain probability at each batch) as a regularization technique for reducing overfitting by preventing complex co-adaptations on the training data.

The architecture of a CNN depends on a large number of hyperparameters, so choosing hyperparameters for each different application is an engineering challenge. The choices made in this paper are discussed in Section 6.4.

6.2.3.4 *Gradient Boosting - XGB*

Gradient boosting for classification is an algorithm that trains several weak learners (i.e., decision trees that perform poorly considering the classification problem) and combines their predictions to make one stronger learner. Gradient boosting differs from the random forest in the way the decision trees are built. While in random forest classifier, each tree is trained independently using random samples of the data, decisions trees in gradient boosting depend on the previously trained tree's prediction to correct its errors. Gradient tree boosting is composed of a concatenation of several smaller decision trees. We used the eXtreme Gradient Boosting (XGB) implementation of gradient boosting, designed by Chen and Guestrin [CG16], which use a sparsity aware algorithm for handling sparse data and a theoretically justified weighted quantile sketch for approximate learning.

6.2.3.5 *Naive Bayes - NB*

Gaussian Naive Bayes classifier is one of the classification algorithms that applies Bayes's theorem with the "naive" assumption. The naive assumption describes the conditional independence between every pair of features in a given class sample. The Gaussian assumption is assumed as the features' probability distribution. The Naive Bayes method is highly scalable with the number of features and requires only a few representative features per class to achieve a satisfying performance.

6.2.3.6 *Template Attack - TA*

The template attack relies on the Bayes theorem and considers the features to be dependent. Commonly, template attack relies on a normal distribution [CRR02], and it assumes that each $P(\vec{X} = \vec{x}|Y = y)$ follows a (multivariate) Gaussian distribution parameterized by its mean and covariance matrix for each class Y. Choudary and Kuhn proposed using one pooled covariance matrix averaged over all classes Y to cope with statistical difficulties and thus lower efficiency [CK13]. In our experiments, we use this version of the attack.

6.3 EXPERIMENTAL SETUP

6.3.1 *Attacker Model*

The general recommendation for EdDSA, as well as other ECDSA implementations, is to select different ephemeral private keys r for each different signature. When this is not applied and the same r is used for different messages, the two resulting signature pairs (R, S) and (R, S') for messages M and M', respectively can be used to recover r as $r = (z - z')(S - S')^{-1}$, where z and z' represent a majority of leftmost bits of $H(M)$ and $H(M')$ interpreted as integers [1]. Finally, the private scalar a is exposed as $a = R^{-1}(Sr - z)$ and can be misused by the attacker to forge new signatures [2].

The attacker's aim is the same as for every ECDSA attack: recover the secret scalar a. The difference is that the attacker cannot acquire two signatures with the same random r, but can still recover the secret scalar in two different ways. The first method consists of attacking the hash function's implementation to recover b from the computation of ephemeral private key [Sam+18]. The second one attacks the implementation of the scalar multiplication during the ephemeral public key's computation to infer it in a single trace [WPB19]. In this paper, we consider only the profiled attacks, i.e., those based on the supervised machine learning paradigm, where the task is the classification (learning how to assign a class label to examples). As side-channels, we consider the power and electromagnetic (EM) leakage.

6.3.2 *SCA Datasets*

We analyze two publicly available datasets targeting elliptic curve scalar multiplication on Curve25519 for microcontrollers. The first dataset consists of power traces of a baseline implementation, and the second dataset consists of electromagnetic traces of a more protected implementation.

6.3.2.1 *Baseline Implementation Dataset*

We consider a dataset of scalar multiplication on Curve25519. The implementation follows the baseline implementation of the scalar multiplication algorithm as in [WPB19]. The traces contain power measurements collected from a Piñata

1 To be precise: z and z' correspond to l leftmost bits of $H(M)$ and $H(M')$ respectively, where l denotes the bit length of the group order.

2 For details we refer the reader to the presentation about a real-world application of this attack: https://wikileaks.org/sony/docs/05/docs/Hacks/PS3%20timeline.pdf

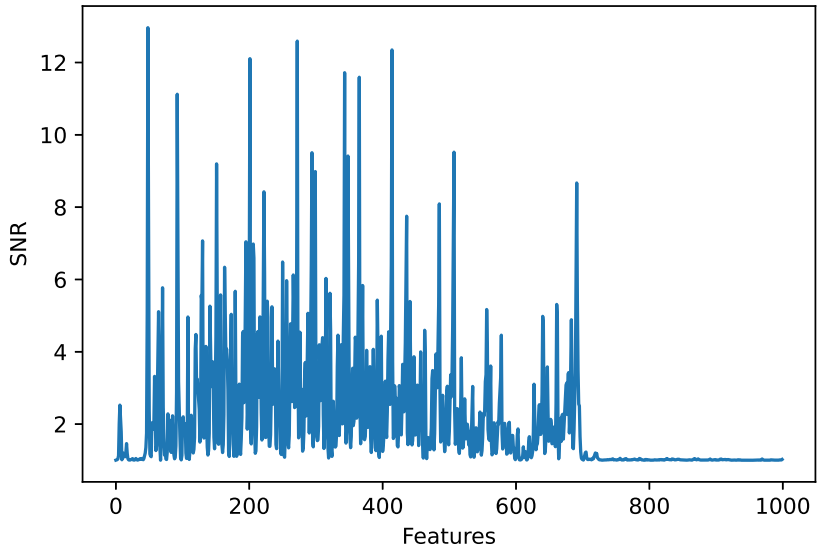


Figure 32: Signal-to-noise ratio for the baseline implementation dataset.

development board[1] based on a 32-bit STM32F4 microcontroller with an ARM-based architecture, running at the clock frequency of 168 MHz. The device is running the Ed25519 implementation of WolfSSL 3.10.2. The target is the EC scalar multiplication of the ephemeral key and the base point of curve Ed25519 (as explained in Section 6.3.1). Because of the chosen implementation, it is possible to profile the full scalar by nibble in a horizontal fashion. The dataset is thus composed of multiple separate nibble computations.

The dataset has 6 400 labeled traces of 1 000 features each, with associated nibble value. In Figure 32, we give the signal-to-noise ratio of this dataset. The SNR is high and reaches a maximum value of 12.9. Such a high SNR is the consequence of dealing with power leakages that are less noisy than usual EM leakages. The leakage is essentially located between points 50 and 700, where several features seem to leak information about the handled nibble.

6.3.2.2 *Protected Implementation Dataset*

The traces in the protected dataset are taken from a publicly available dataset [Chm20]. This set contains electromagnetic traces coming from 5 997 executions of

1 Pinata Board: https://www.riscure.com/product/pinata-training-target/

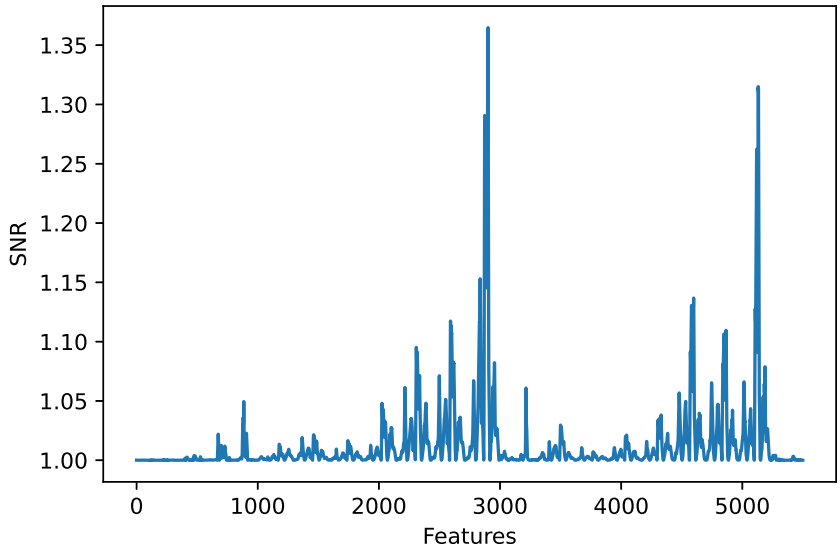


Figure 33: Signal-to-noise ratio for the protected implementation dataset.

Curve25519 μNaCl Montgomery Ladder scalar multiplication [3] running on the Piñata target, the same as in Section 6.3.2.1. The implementation employs an arithmetic-based conditional swap and is additionally protected with the projective coordinate re-randomization and scalar randomization. Each trace from the dataset represents a single iteration of the Montgomery Ladder scalar multiplication that is cut from the whole execution trace; such trace is labeled with the corresponding cswap condition bit [4]. Furthermore, all these cut traces ($5\,997 * 255 = 1\,529\,235$) are aligned to exploit the leakage efficiently. Details about the implementation and how the traces are aligned are in [NC17].

Figure 33 represents the SNR of the dataset for the bit model. This SNR is relatively flat except for two peaks where the leakage of the data is stronger. One is located before feature 3 000 and the second after feature 5 000. The noise level is high for an EM dataset but is smaller than the other dataset based on power traces.

3 http://munacl.cryptojedi.org/curve25519-cortexm0.shtml

4 Observe that a full scalar can be trivially recovered from the cswap condition bits used in the 255 Montgomery Ladder iterations.

6.3.3 *Evaluation Metrics*

To examine the feasibility and performance of our attack, we use two different metrics. We first compare the performance using the accuracy metric since it is a standard metric in machine learning. The accuracy metric represents the fraction of the measurements that are classified correctly. The second metric we use is the success rate as it is an SCA metric that gives a more concrete idea on the power of the attacker [SMY09]. Let us consider the settings where we have A attack traces. As the result of an attack, we output a key guessing vector $v = [v_1, v_2, \ldots, v_{|\mathcal{K}|}]$ in decreasing order of probability with $|\mathcal{K}|$ being the size of the keyspace. Then, the success rate is the average empirical probability that v_1 is equal to the correct key.

6.3.4 *Dimensionality Reduction*

For computational reasons, one may want to analyze only the most informative features from the dataset's traces. Consequently, we explore several different settings where we use all the features in a trace or conduct dimensionality reduction. For dimensionality reduction, we use a method called principal component analysis. Principal component analysis (PCA) is a linear dimensionality reduction method that uses Singular Value Decomposition (SVD) of the data matrix to project it to a lower-dimensional space [Boh+03]. PCA creates a new set of features (called principal components) that form a new orthogonal coordinate system that is linearly uncorrelated. The number of components is the same as the number of original features. The components are arranged so that the first component covers the largest variance by a projection of the original data, and the following components cover less and less of the remaining data variance. The projection contains (weighted) contributions from all the original features. Not all principal components need to be kept in the transformed dataset. Since the components are sorted by decreasing covered variance, the number of kept components, designated by L, maximizes the original data variance and minimizes the data transformation's reconstruction error. While PCA is meant to select the principal information from data, there is no guarantee that the reduced data form will give better results for profiling attacks than its complete form.

6.3.5 *Hyperparameter Tuning*

Most machine learning methods are parametric and require some hyperparameters to be tuned before the training phase. Depending on this pre-tuning, the trained

classifier will potentially have a different outcome. The different classification methods we used are trained with a wide set of hyperparameters as detailed in this section. The exact used hyperparameters are listed in Tables 11 and 14.

TA. We use the Template Attack with a pooled covariance matrix [CK13]. This method has no hyperparameters to tune.

NB. We do not conduct hyperparameter tuning as the method is non-parametric (i.e., there are no hyperparameters to tune).

RF. We tune the number of decision trees. We consider the following number of trees: 50, 100, 500.

SVM. For the linear kernel, the hyperparameter to optimize is the penalty parameter C. We search for the best C in the range $[1, 10^5]$ in logarithmic space. For the radial basis function (RBF) kernel, we have two hyperparameters to tune: the penalty C and the kernel coefficient γ. The search for best hyperparameters is done within $C = [1, 10^5]$ and $\gamma = [-5, 2]$ in logarithmic spaces.

XGB. In the same fashion as the random forest classifier, we set the hyperparameters exploration for the number of trees to 50, 100, and 300. We impose a maximum depth for each tree from 1 to 3 nodes, to force each tree to be a weak learner.

CNN. The chosen hyperparameters for *VGG-16* follows several rules that have been adapted for SCA in [Kim+19b] or [Pro+18] and that we describe here:

1. The model is composed of several convolution blocks and ends with a dropout layer followed by a fully-connected layer and an output layer with the Softmax activation function.
2. Convolutional and fully-connected layers use the ReLU activation function $(max(0, x))$.
3. A convolution block is composed of one convolution layer followed by a pooling layer.
4. An additional batch normalization layer is applied for every odd-numbered convolution block and is preceding the pooling layer.
5. The chosen filter size for convolution layers is set to the size 3.
6. The number of filters $n_{filters,i}$ in a convolution block i increases according to the following rule: $n_{filters,i} = max(2^i \cdot n_{filters,1}, 512)$ for every layer $i \geq 0$ and we choose $n_{filters,1} = 8$.

Table 11: Best hyperparameters found for the baseline implementation dataset.

Algorithm	Number of features	Best hyperparameters
SVM linear	1 000	C=1 000
	500	C=23.1
	100	C=284.8
	10	C=1 333
SVM rbf	1 000	C=1 000, γ=1
	500	C=12.3, γ=0.65
	100	C=81.1, γ=0.65
	10	C=1 000, γ=1.23
RF	1 000, 500, 100, 10	n_tree=500
XGB	1 000, 500, 100, 10	n_tree=300, max_depth=3

7. The stride of the pooling layers equals two and halves the input data for each block.
8. Convolution blocks follow each other until the size of the input data is reduced to 1.

6.4 RESULTS

In this section, we first present results for the baseline implementation and the protected implementation afterwards. We finish the section with results on visualization and discussion. The best results in Tables 12 and 15 are given in bold.

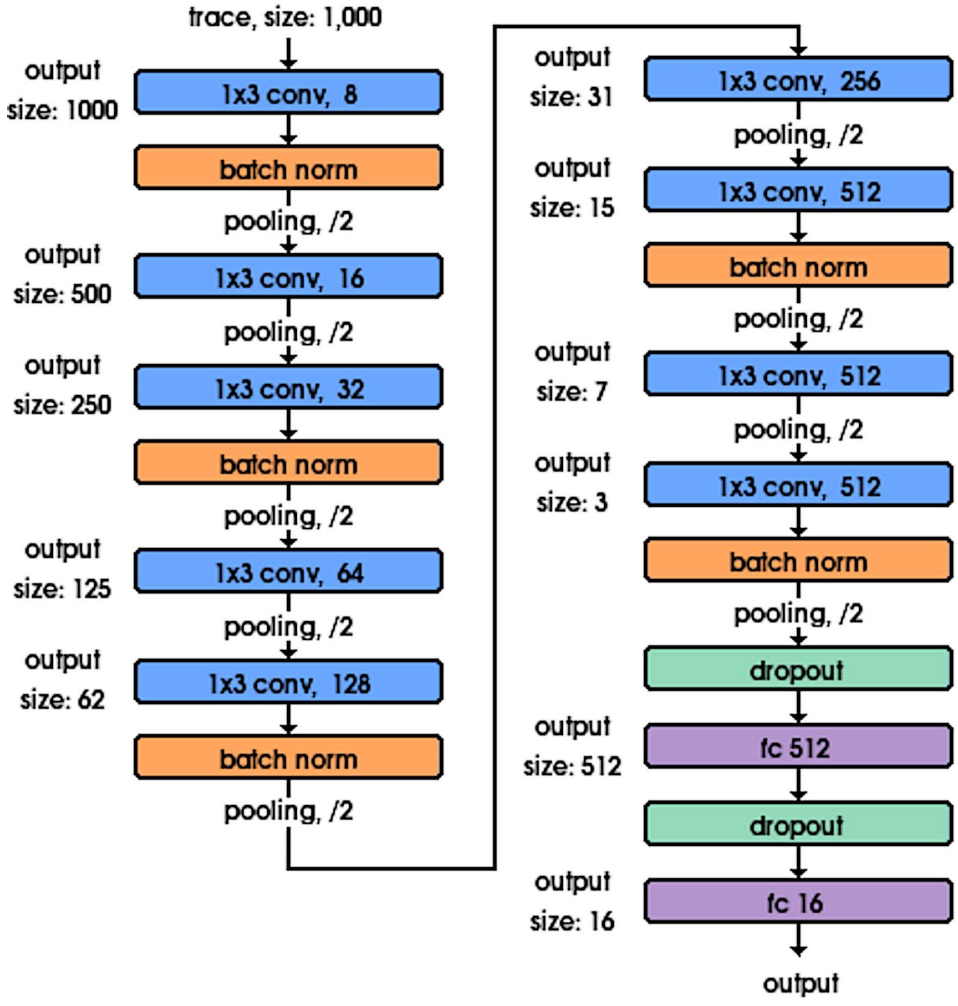
6.4.1 *Baseline implementation*

After the conducted training phase of all the different classifiers with their hyperparameters, we list in Table 11 the best hyperparameters combinations for each machine learning model.

The resulting CNN architecture for a 1 000 features input is depicted in Figure 34. Other architectures will have a different number of convolutional blocks and a number of weights depending on the number of features of the input.

In Table 12, we give the accuracy score for different profiling methods when considering the recovery of a single nibble of the key. We can see that all profiling techniques reach excellent performance with accuracy above 95%. When

Figure 34: CNN architecture, as implemented in Keras. This architecture takes a 1 000 features input and consists of nine convolutional layers followed by max pooling layers. For each odd convolutional layer, there is a batch normalization layer before the pooling layer. At the end of the network, there is one fully-connected layer.



considering all available features (1 000), CNN performs the best and achieves an accuracy of 100%. Both SVM (linear and RBF) and RF have the same accuracy. SVM's performance is interesting since the same value for linear and RBF kernel indicates there is no advantage of using higher-dimensional space, which means that the classes are linearly separable. Finally, NB, XGB, and TA still perform well, but we conclude they reach the worst results compared to other methods.

PCA results in lower accuracy scores for most of the considered techniques. When considering 500 or 100 PCA components, the TA's results slightly improve, while RF and CNN results slightly decrease. SVM with both kernels can reach minimally higher accuracy when considering 500 PCA components. When considering the scenario with only the ten most important PCA components, all the results deteriorate compared with the results with 1 000 features, and SVM performs the best.

To conclude, all techniques exhibit strong performance, but CNN is the best if no dimensionality reduction is applied. There, the maximum accuracy is obtained after only a few epochs (see Figures 36 and 37). If dimensionality reduction is applied, CNN shows a progressive performance deterioration. This behavior should not come as a surprise since CNNs are usually used with the raw features (i.e., no pre-processing). Applying such techniques could reduce the performance due to a loss of information and changes in the spatial representation of features. Interestingly, TA and SVM are very stable methods, regardless of the number of used features (components), and those methods show the best performance for a reduced number of features settings.

In Figure 35, we present a success rate with orders up to 10 for all profiling methods on the dataset without applying PCA. Recall, a success rate of order o is the probability that the correct subkey is ranked among the first o candidates of the guessing vector. While CNN has a 100% success rate of order 1, other methods achieve the perfect score only for orders greater than 6.

The results for all methods are similar in the recovery of a single nibble from the key. To have an idea of how good these methods perform for the recovery of a full 256-bit key, we apply classification on the successive 64 nibbles. We obtain an intuition of the resulting accuracy by considering the cumulative probability P_c of the probabilities of recovery of one nibble $P_s : P_c = \Pi_{64} P_s$ (see Table 13). The cumulative accuracy obtained in such a way can be interpreted as the predictive first-order success rate of a full key for the different methods in terms of a security metric.

From these results, the best result is obtained with CNN when no dimensionality reduction is applied. Other methods are nonetheless powerful profiling attacks with up to 95% performance to recover the full key on the first guess with the

Table 12: Accuracy results for the baseline implementation dataset.

Algorithm	1 000 features	500 PCA	100 PCA	10 PCA
TA	0.9977	0.9992	**0.9992**	0.9830
RF	0.9992	0.9909	0.9921	0.9937
SVM (linear)	0.9992	0.9995	0.9990	**0.995**
SVM (rbf)	0.9992	**0.9996**	0.9989	**0.995**
CNN	**1.00**	0.9796	0.9968	0.96
XGB	0.9965	0.9794	0.9807	0.9901
NB	0.9837	0.9475	0.9731	0.9823

Figure 35: Success rate results for the baseline implementation dataset.

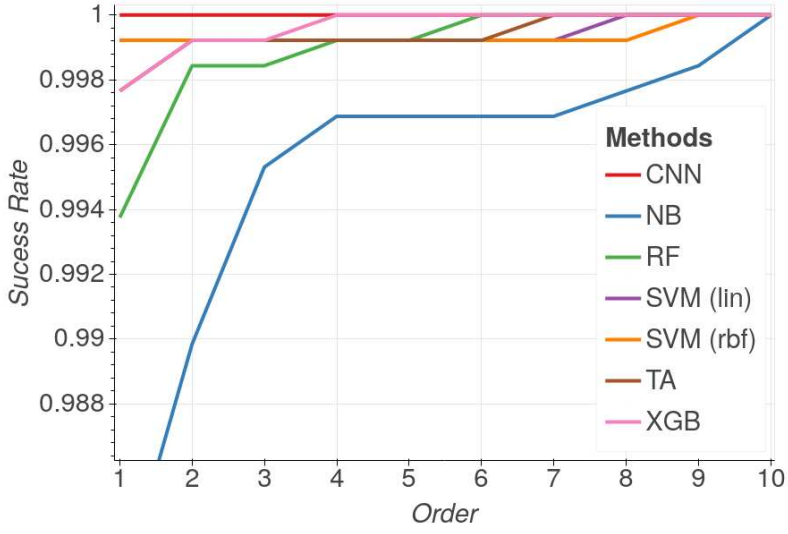
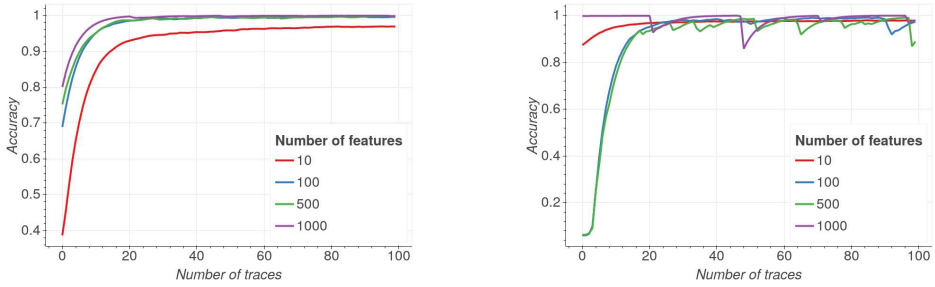


Table 13: Cumulative probabilities for the profiling methods.

Algorithm	1 000 features	500 PCA	100 PCA	10 PCA
TA	0.86	0.95	0.95	0.33
RF	0.95	0.56	0.61	0.67
SVM (linear)	0.95	0.97	0.94	0.73
SVM (rbf)	0.95	0.98	0.93	0.73
CNN	1.00	0.27	0.82	0.04
XGB	0.80	0.27	0.29	0.53
NB	0.35	0.03	0.18	0.32



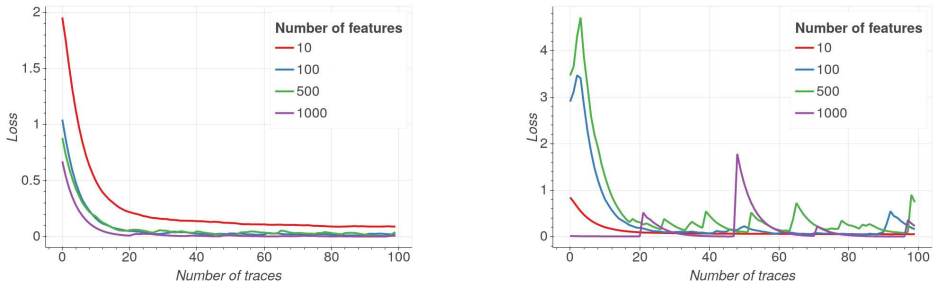
(a) Training Accuracy

(b) Validation Accuracy

Figure 36: Accuracy of the CNN method over 100 epochs for the baseline implementation dataset.

best choice of hyperparameters and dimensionality reduction. When considering the results after dimensionality reduction, SVM is the best performing technique when using 500 PCA components.

As it can be observed from Figures 36 and 37, both the scenarios without dimensionality reduction and dimensionality reduction to 100 and 500 components reach the maximal performance very fast. On the other hand, the scenario with 10 PCA components does not reach the maximal performance within 100 epochs since the validation accuracy does not start to decrease. Still, even longer experiments do not show further improvement in the performance, which indicates that the network simply learned all that is possible and that there is no more information that can be used to increase the performance further. Finally, the fast increase in training and validation accuracy, and the stable behavior of profiling methods clearly indicate that attacking the implementation without countermeasures is easy.



(a) Training Loss (b) Validation Loss

Figure 37: Loss of the CNN method over 100 epochs for the baseline implementation dataset.

Table 14: Best hyperparameters found for the protected implementation dataset.

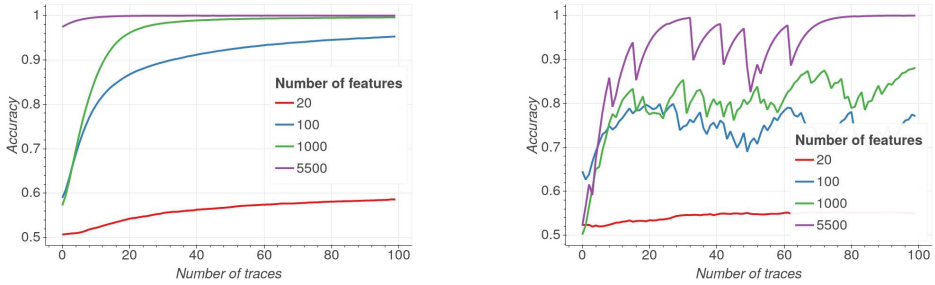
Algorithm	Number of features	Best hyperparameters
RF	5 500, 1 000, 10	*n_tree*=500
XGB	5 500, 1 000	*n_tree*=300, *max_depth*=3
	10	*n_tree*=300, *max_depth*=2

6.4.2 *Protected Implementation*

We list the selected hyperparameters for the protected implementation in Table 14. The protected implementation dataset contains more features per trace than the other dataset. Therefore, the number of trainable parameters for machine learning methods greatly increases, increasing the models' training load. We experimented with RF, NB, and XGB and left out SVM (both with linear and RBF kernel) as this method's training becomes too expensive.

Algorithm	5 500 features	1 000 PCA	10 PCA
RF	0.9903	0.5022	0.5023
NB	0.6058	0.4971	0.5018
XGB	0.6058	0.4945	0.5019
TA	0.9908	**0.8954**	0.5238
CNN	**0.9999**	0.5014	**0.5572**

Table 15: Accuracy results for the protected implementation dataset.



(a) Training Accuracy

(b) Validation Accuracy

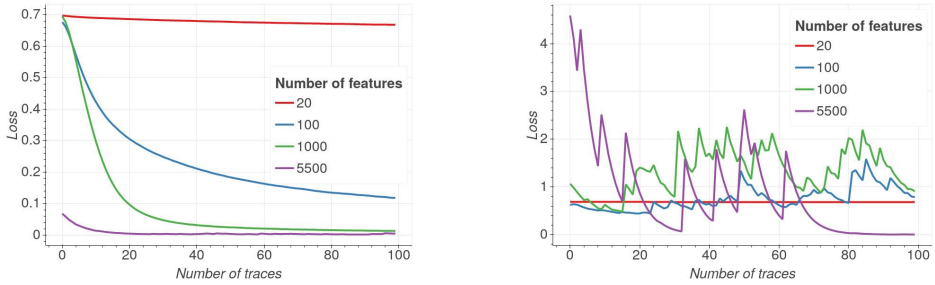
Figure 38: Accuracy of the CNN method over 100 epochs on the protected implementation dataset.

We show the accuracy results for all tested methods on the protected implementation dataset in Table 15. Notice that, contrary to the previously considered dataset, not all profiling techniques have good performance, and most of them are even close to random guessing. Still, some profiling methods can reach above 99% accuracy, where the best results are obtained with CNN. When PCA is applied, random forest performs poorly with 50.2% accuracy for ten and 1 000 components, which is not better than one could expect from random guessing. However, this method turns out to be quite efficient on the raw features and reaches an accuracy of 93% for one bit recovery.

Naive Bayes and XGB perform poorly regardless of the hyperparameters explored and if dimensionality reduction is applied. The accuracy stays around random guessing when PCA is applied with ten and 1 000 components, and does not go above 60% in the best case. Naive Bayes and XGB are simple classifiers and, considering their accuracy score on this dataset, are not powerful enough to defeat a protected EC scalar multiplication implementation.

The template attack is performing well, where the more features are taken, the better the results. The best accuracy score for template attack is obtained when all features are kept, and it reaches 99% accuracy. When PCA is applied and 1 000 components are selected, the accuracy falls to 89% (which is, in fact, the best results for all considered techniques). Finally, when the number of selected components is reduced to 10, the accuracy falls to 52%.

CNN is a highly efficient method only when considering the dataset without applying the PCA method, where it reaches an accuracy above 99%. As we can see in Figures 38 and 39, when PCA is applied, while the training loss and accuracy seems to fit the training set, the model fails to generalize and converge on the validation set given the chosen number of traces and epochs.



(a) Training Loss (b) Validation Loss

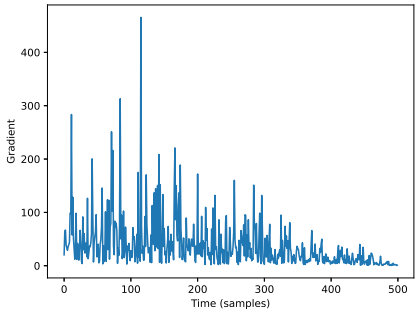
Figure 39: Loss of the CNN method over 100 epochs on the protected implementation dataset.

We can evaluate the accuracy of the different methods to predict a 256 bits scalar by computing the cumulative probability of success of a single bit over 256 attempts. The cumulative probability p_c for a 256 bits key considering a single bit probability recovery P_s is: $P_c = \prod_{256} P_s$. Here, only the methods with a single accuracy above 99% are worth considering as the other methods have a cumulative probability close to zero. For example, the cumulative accuracy for the random forest with 5 500 features is 8%, and CNN with 5 500 features is 98%.

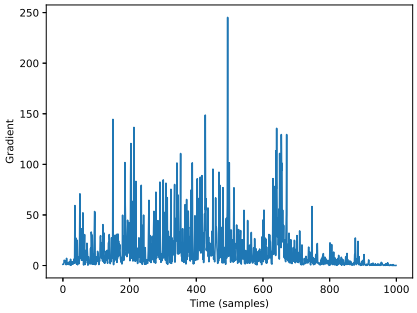
6.4.3 *Visualization of the Integrated Gradient*

For CNNs, various visualization techniques have been developed to help researchers understand what input features influence the neural network predictions. These tools are interesting in side-channel analysis to evaluate if a network bases its prediction on the part of the trace where the leakage is the strongest. We note that visualization techniques proved to be a helpful tool when considering profiled SCA and block ciphers [HGG20; MDP19]. We use here the Integrated gradient method [OPB16]. In this method, the higher is the gradient value, the more important the feature is for the model's prediction.

From Figures 40 and 41, we can notice that when we apply principal component analysis, the network tends to rely more on the first features. After applying PCA, the features are reorganized and ranked from the most important to the least important feature. When considering the dataset without applying PCA, the features' order is the same as those sampled with the oscilloscope. We can notice interesting similarities between the SNR of the unprotected implementation (Figure 32) and the integrated gradient of the CNN. The interpretation of the integrated gradient obtained for the CNN trained on the protected implementation dataset is

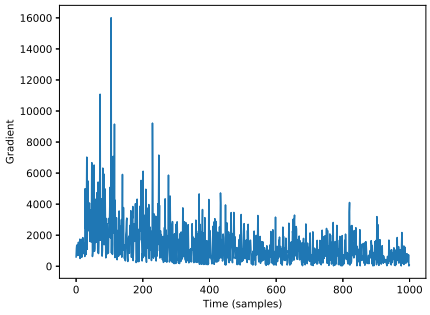


(a) 500 POI

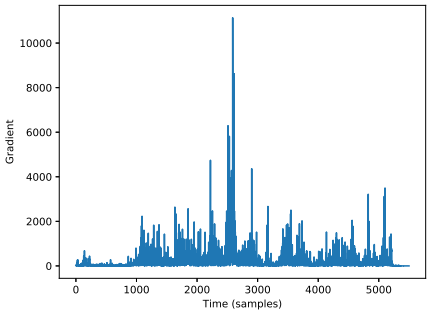


(b) 1 000 features

Figure 40: Integrated gradient method applied to CNN trained on the baseline implementation dataset.



(a) 1 000 features



(b) 5 500 features

Figure 41: Integrated gradient method applied to CNN trained on the protected implementation dataset.

less evident as the high peaks do not correspond to the leaking features indicated by the SNR (see Figure 33). When comparing the visualization results for both datasets, the similarity between the baseline results for the full number of features and after dimensionality reduction indicates that the performance should be similar, which is confirmed by the accuracy results. On the other hand, we see striking differences between two visualizations for the protected implementation, where the one with 1 000 features cannot concentrate on the most important elements, which is again evident from the accuracy results.

6.4.4 *General Remarks*

The obtained results allow us to infer some more general recommendations one could follow one attacking ECC with profiled SCAs:

1. When attacking unprotected implementations, most of the considered methods work well. While CNN performs the best, computationally simpler methods represent an interesting alternative.
2. For protected implementation, deep learning performs significantly better than other considered methods.
3. For the protected implementation, all the methods perform worse when Principal component analysis is applied to reduce the number of features.
4. Template attack should be an interesting option in cases when one cannot use all the features.
5. There is not much difference in the attack performance concerning hyperparameter tuning, which indicates that coarse-grained tuning should be enough.
6. Visualization techniques offer good indication in the performance of CNNs, as they show on what features CNN concentrates. If CNN cannot concentrate on a smaller number of features, this results in a poor attack performance.

6.5 RELATED WORK

In 2003 Chari et al. [CRR02] introduced a template attack (TA) as a powerful SCA method in the information-theoretic point of view, which became a standard tool for profiling SCA. As TA's straightforward implementations can lead to computationally intensive computation, one option for more efficient computation is to use only a single covariance matrix, which is referred to as the so-called pooled template attack presented by Choudary and Kuhn [CK13]. There, the authors were

able to template a LOAD instruction and recover all 8 bits treated with a guessing entropy equal to zero.

Several works applied machine learning methods to SCA of block ciphers because they resemble general profiling techniques. Two methods stand out particularly in profiling SCA, namely Support Vector Machines ([Pic+17; MPP16; SH12; LBM14]) and Random Forest ([Heu+17; Pic+19; SH12]). Few other works also experienced SCA with naive Bayes [Pic+17] and Gradient boosting methods [Pic+18b; XWZ18]. With the general evolution in the field of deep learning, more and more works deal with neural networks for SCA and often show top performance. Most of the research concentrated on either multilayer perceptron or convolutional neural networks [MPP16; Pic+18b; CDP17; CJ19].

There is a large portion of works considering profiling techniques for symmetric-key ciphers, but there is less for public-key cryptography [5], especially ECC. Template attacks on ECC trace back to an attack on ECDSA, as demonstrated by Medwed and Oswald 2009 [MO08]. That work showed TA to be efficient for attacking SPA-resistant ECDSA with the P192 NIST curve on a 32-bit microcontroller [MO08]. Heyszl presented another template attack on ECC in [Hey+12]. That attack exploited register location-based leakage using a high-resolution inductive EM probe. Another approach to attack ECC is the so-called online template attacks [Bat+14; Dug+16; Bat+17; OPB16]. The first three approaches [Bat+14; Dug+16; Bat+17] use correlation to match the template traces to the whole attacked traces while the fourth attack [OPB16] employs instead several machine learning distinguishers.

Lerman et al. considered a template attack and several machine learning techniques to attack RSA. However, the targeted implementation was not secure, making the comparison with non-machine learning techniques less favorable [LBM14]. Nascimento et al. applied a horizontal attack on ECC implementation for AVR ATmega microcontroller targeting the side-channel leakage of cmov operation. Their approach to side-channel is similar to ours, but they do not use deep learning in the analysis [Nas+17]. Note, that approach was extended to unsupervised settings using clustering [NC17]. Poussier et al. used horizontal attacks and linear regression to conduct an attack on ECC implementations, but their approach cannot be classified as deep learning [PZS17]. Carbone et al. used deep learning to attack a secure implementation of RSA [Car+19]. The results from that paper show that deep learning can reach strong performance against secure implementations of RSA.

5 We do not consider here the post-quantum schemes, because although they belong to public-key cryptography, they differ significantly from ECC or RSA.

6.6 CONCLUSIONS

In this paper, we consider several profiling methods to attack Curve25519 in both unprotected and protected settings. The results show that unprotected implementation is easy to attack with many techniques, where good results are achieved even after dimensionality reduction. We observe a significantly different behavior for the protected dataset, where only CNN can easily break the target implementation. What is more, most of the other methods perform on the level of random guessing. For this dataset, we also see a strong negative influence of dimensionality reduction. Finally, our results with the integrated gradient visualization indicate such methods useful in evaluating CNN's behavior. Indeed, when there are clear peaks for the integrated gradient, this maps to a simple classification task, and consequently, powerful attack performance.

We plan to investigate whether standard machine learning metrics like accuracy have fewer issues for public-key cryptography implementations than are reported for symmetric-key ciphers. As this gap between machine learning and side-channel metrics represents one of the most significant challenges in the SCA community today, insights about public-key particularities are needed.

Part IV

SIDE-CHANNELS ENHANCED BY NEURAL NETWORKS AND THE OPPOSITE

7

SCREEN GLEANING: RECOVERING INFORMATION FROM MOBILE PHONE SCREENS VIA ELECTROMAGNETIC SIDE-CHANNEL ANALYSIS

This chapter is based on [Liu+], a joint work with Zhuoran Liu, Niels Samwel, Zhengyu Zhao, Dirk Lauret, Lejla Batina and Martha A. Larson, that was presented in 2022 during the 29th Annual Network and Distributed System Security Symposium (NDSS).

CONTENT OF THIS CHAPTER

7.1 INTRODUCTION

Most of our daily business relies on the devices we carry on us. A great deal of sensitive information is exchanged through these devices, and the security and privacy of our data is constantly at stake. Even the task of authenticating ourselves (or our data) has been shifted to our phones, where two-factor authentication, a common approach, requires successfully presenting two or more pieces of evidence to confirm our identity.

To protect our data, mobile devices typically use secret (cryptographic) keys that are not accessible from the outside. Getting a hold of the key allows a hacker to steal our data. The majority of real-world attacks on security implementations on small devices today use side-channel analysis (SCA), i.e., they measure and

process physical quantities, like the power consumption or electromagnetic emanations of a chip, or reaction time of a process. Moreover, thanks to computing power becoming ever cheaper nowadays, modern adversaries have started using state-of-the-art machine and deep learning algorithms for SCA. Securing (embedded) systems against SCA remains a great challenge.

In certain cases, the security implementation is not the target of an attack. Instead, the target is the sensitive information displayed on the screen. For example, here, we can think of secret security codes sent from banks or credit card companies, giving secure access to a user who is the only one able to read the code. SCA can take advantage of the fact that information is exposed in this way in order to mount an attack. Since we can expect adversaries will always target the weakest link, such attacks are more feasible than cryptographic attacks i.e. cryptanalysis.

In this paper, we investigate the problem of sensitive information on mobile phone screens. Until now, the study of side-channel analysis attacks that aim to recover the screen content of a mobile phone has focused on visible-spectrum signals. This focus is consistent with people's general belief that protecting information on their mobile phone screen means hiding it from the line of sight of a person or a camera. However, SCA can go beyond visible-spectrum information displayed on the screen. In this paper, we present a low-cost SCA attack that can recover information displayed on a mobile device's screen by capturing the electromagnetic signal sent to the phone screen. Our work introduces an attack, which we call *screen gleaning*, that uses an antenna and a basic software-defined radio (SDR). Our attack demonstrates the security threat posed by emanations leaking from mobile devices. We release an implementation of our attacks that allows for further testing and extension.[1]

The side-channel analysis that we consider in this work is a type of TEMPEST technique. TEMPEST techniques exploit vulnerabilities of communication and other types of emanations from electrical equipment that contain sensitive data [McN]. From our experiments with a simple TEMPEST setup using an SDR receiver, we were able to successfully capture the phone screen content without a visible-spectrum line of sight. The signal recovered from the screen can be visualized as a gray-scale image, which we refer to as an *emage*. A challenge faced by our attack is that the emage is often not interpretable, meaning that it cannot be read by way of human eyesight. We propose a machine learning-based approach capable of processing an emage that is not interpretable to the human eye in order to recover secret information, such as a security code in two-factor authentication.

This simple attack story illustrates the potential danger of our attack:

1 Code available at: https://github.com/cescalab/screen_gleaning

Alice keeps her mobile phone on a stack of magazines on top of her desk. She lays the phone face down because she receives security codes and she believes that blocking the visual line of sight to the phone screen will keep the codes secure. Eve has access to Alice's desk and has hidden an antenna under the top magazine. The antenna can read the security code via electromagnetic emanations of the phone.

In sum, this paper makes five contributions:

- We present a novel side-channel technique called screen gleaning, an attack that can be used to recover information such as a security code communicated by text message. The attack does not require a visual line of sight nor the readability of the signal by a human. In fact, the signal we observe is, in most cases, not interpretable to the human eye, so the information in the leakage is not obvious.
- We show that this kind of challenge can be tackled using machine learning, and specifically, using a deep learning classifier we are able to attain very high accuracy (of close to 90%) when guessing the digits of a security code.
- We quantitatively demonstrate that our attack is effective for three representative phone models under various environmental conditions. In particular, our attack is applicable in the context of cross-device, through-magazine, and noisy environments.
- We define and validate a new testbed applicable for further research on screen gleaning. The testbed includes a parameterized attacker model, which will guide future research to systematically explore and exploit the threat of screen gleaning.
- Finally, we propose and discuss possible countermeasures against screen gleaning attacks on mobile devices.

The remainder of this paper is organized as follows: In Section 7.2, we discuss related work. Section 7.3 describes the attacker model. In Section 7.4, we describe our measurement and machine learning setup. In Section 7.5, we explain the experiments we conducted together with the results. Section 7.6 introduces a testbed. Section 7.7 discusses the results of the paper and describes different countermeasures. Section 7.8 discusses different formulations of the screen gleaning problem. Finally, Section 7.9 concludes the paper.

7.2 RELATED WORK

7.2.1 *Side-Channel Attacks*

A security attack exploiting unintentional physical leakage is called a *side-channel attack*. For example, an adversary might be able to monitor the power

consumed by a device while it performs secret key operations [KJJ99; Koc+11]. Other sources of side-channel information, such as electromagnetic emanations from a chip [GMO01; QS01b; Agr+02] and timings for different operations performed [Koc96], were also shown to be exploitable (for an overview see [MOP06]).

Side-channel attacks pose a real threat to the security of mobile and embedded devices and since their invention many countermeasures have been proposed. The goal of countermeasures is to remove the dependence between the (secret) data and the side channel such as power consumed during the computation. An extensive study of the power side channel from mobile devices was presented in [Yan+15]. One approach for countermeasures aims to break the link between the actual data processed by the device and the data on which the computation is performed. Such a countermeasure is usually called *masking* and is exploiting the principle of secret sharing [Cha+99]. A second approach aims at breaking the link between the data computed by the device and the power consumed by the computations. This approach is called *hiding*, and one way to achieve it is by flattening the power consumption of a device by, for example, using special logic styles that are more robust against SCA attacks such as WDDL [Tir+05].

SCA attacks belong to the most serious threats to embedded crypto devices and often target the secret (cryptographic) key in a device that keeps personal data and communications secure [Bel+16; Gen+16] or even white-box implementations [Bos+16]. There are many examples of SCA attacks in the real-world such as [OP11; Bal+12; Eis+08] and more recent ones [Mog+20; Coh+20; JSS].

TEMPEST is another side-channel technique that has been known for decades. TEMPEST refers to spying on computer systems through leaking emanations, including unintentional radio or electrical signals, sounds, and vibrations [KA98]. For example, through TEMPEST, one could easily detect a user's keystrokes using the motion sensor inside smartphones or recover the content from a computer or other screens remotely. In 1985 van Eck published the first unclassified analysis of the feasibility and security risks of emanations from computer monitors. Previously, such monitoring was believed to be a highly sophisticated attack available only to governments. However, van Eck successfully eavesdropped on a real system, at a range of hundreds of meters, by measuring electromagnetic emanations using just $15 worth of equipment plus a CRT television set [Eck85]. Later, Kuhn performed a comprehensive study on a range of flat-screen monitors and eavesdropping devices [Kuh02a; Kuh02b; KA98]. Other side channels can also convey the screen's content in the frequency range of the visible spectrum [Kuh02b; BDU08; Bac+09; Xu+13] or through acoustic channel [Gen+19] but can sometimes even require an expensive telescope.

More recently, Backes et al. [BDU08; Bac+09] improved TEMPEST further and argue that the requirement on a direct line of sight is not necessary as they exploit reflections between the target screen and the observer. Xu et al. [Xu+13] broadened the scope of the attacks by relaxing the previous requirements and showing that breaches of privacy are possible even when the adversary is "around a corner". A new technique is presented for reconstructing the text typed on a mobile device, including password recovery via analysis of finger motions over the keyboard and language model. The main distinction from the works by Backes et al. is that they use "repeated" reflections, i.e., reflections of reflections in nearby objects, but always originating from the surface of a person's eyeball. Nevertheless, all those papers use direct or indirect reflections from the screen, which makes their research line very different from ours. More specifically, those papers focus on recovering text and images from the screen while being typed and being captured by a camera from an eyeball, which implies rather special assumptions on the setup and attacker model.

Hayashi et al. performed a comprehensive evaluation of electromagnetic emanations from a chip including countermeasures [Hay+12b; Hay+12a; Hay+14; Hay+16]. However, their focus is on recovering secret information from "inside" such as cryptographic keys and not the screen content.

As a follow-up, the work of Kinugawa et al. [KFH19] demonstrates that it is possible to amplify the electromagnetic leakage with cheap hardware modification added on potentially any device and spread the attack to a broader distance. They demonstrate that this additional circuitry, a so-called interceptor, enlarges the amount of leakage and even forces leakage in devices that do not suffer potential electromagnetic leakage.

Goller and Sigl proposed to use standard radio equipment when performing side-channel attacks on smartphones [GS15]. They also aimed their attack at cryptographic operations inside the chip as they demonstrate the ability to distinguish between squaring and multiplications. This observation could lead to the full RSA key recovery, assuming that the modular exponentiation is implemented with a basic square-and-multiply algorithm. Their setup used an Android phone to collect electromagnetic leakages from (albeit they modified the hardware, which makes their attacker's model different).

There exist many papers considering finger movements on the screen or other traces from typing on a smartphone. For example, Cai et al. developed an Android application called TouchLogger, which extracts features from device orientation data to infer keystrokes [CC11]. Aviv et al. used the embedded accelerometer sensor to learn user tapping and gesturing to unlock smartphones [Avi+12]. In another work, they introduce smudge attacks as a method that relies on detecting

the oily smudges left behind by the user's fingers when operating the device using simple cameras and image processing software [Avi+10].

As another two examples of recent work, we also mention the papers of Genkin et al. [GPT15; Gen+19]. In [GPT15], the authors use various side channels like power and electromagnetic radiation to extract cryptographic keys, i.e., RSA and ElGamal keys from laptops, but do not discuss the possibility to perform the attacks on a phone. On the other hand, in [Gen+19] the authors show how to extract the screen content by using the acoustic side channel. They demonstrate how the sound can be picked up by microphones from webcams or screens and transmitted during a video conference call or archived recordings. It can also be recorded by a smartphone or other device with a microphone placed in the screen proximity. These two examples are different from our work because they use either another kind of emanation or have different attack goals (or both).

Other work using acoustic side channels is from Berger et al. [BWY06], which demonstrated a dictionary attack using keyboard acoustic emanations. Backes et al. [Bac+10] investigated acoustic side channel on printers, and Asonov and Agrawal [AA04] used the sound emanated by different keys to recover information typed on a keyboard.

In sum, the uniqueness of our contribution is a side-channel analysis attack that exploits the electromagnetic emanations of the display cable from a mobile phone. These emanations are less accessible and may be substantially weaker than the signals analyzed in more traditional TEMPEST technique attacks. To the best of our knowledge, the most recent work, which bears superficially similarity to ours, is [Lem+20]. This work applied deep learning to recognition on TEMPEST signals, but does so with the goal of automation and enhancement. In other words, [Lem+20] targeted a captured signal in which the content is clearly interpretable to the human eye (cf. Figure 2 in [Lem+20]). In our work, machine learning is used for the purpose of identification. We face the challenge of an uninterpretable emage derived from a mobile phone.

7.2.2 *Deep Learning and Side-channel Analysis*

Several side-channel analysis techniques are based on profiling a physical device and are commonly known as *template attacks* and refer to the first such attack presented by Chari et al. [CRR02]. Profiling attacks estimate a power profile of a cryptographic device for each possible secret key from their resulting power traces (also known as the training phase) and predicting the corresponding key of an unknown trace. From this very similar approach to machine learning, several methods have been inspired by machine learning and neural networks [Car+19;

Kim+19a; CDP17; MPP16]. These methods have raised much attention as they provide more powerful attacks than the state-of-the-art. In our work, we will discuss the usability of deep learning, specifically Convolutional Neural Networks (CNNs) [LeC+98; KSH12], for classifying the emages that are reconstructed from the screen content.

Image classification is the task of predicting a class for a given image according to its content. In the context of machine learning, it can be automated by modeling a transformation from an image to its corresponding class. Early research [SGS09; GPK02; Low04] tackled this problem via a two-step process: manually extracting features from the images and then training a discriminative model for classification.

Deep learning algorithms, such as CNNs [LeC+98; KSH12], automatically learn image features simultaneously with learning the classification by making use of a large number of filters in an end-to-end manner. Deep learning has lead to breakthrough success in general image classification. Large-scale training and diverse data augmentation techniques make an important contribution. In particular, it has been demonstrated that deep learning can achieve superhuman performance in specific domains where the discriminative visual patterns are hard to distinguish by the human eye, e.g., image forensics and steganalysis [YNY17; BS18; Zho+18]. In our work, since the emage content is hardly recognizable to the human eye, we use CNNs to capture the subtle differences between various classes, rather than relying on human-interpretable features.

7.3 ATTACKER MODEL

The attacker's goal is to recover the information (e.g., security code, password, or message) displayed on the target display. We start from the general attack story presented in the introduction: an antenna is planted that can read a security code from a mobile phone screen without a visible-spectrum line of sight. This story is the basis for the attacker model, which is illustrated in Figure 42 and characterized in detail in Table 16. In this section, we provide an explanation of the attacker model and its motivation.

Our attacker model makes the following assumptions:

- The set of symbols displayed on the phone is finite and known (i.e., digits 0-9). This assumption holds true of any information expressed as alphanumeric characters.
- The attacker has access to a profiling device sufficiently similar to the target device, which is used to collect training data for the machine learning classifier.

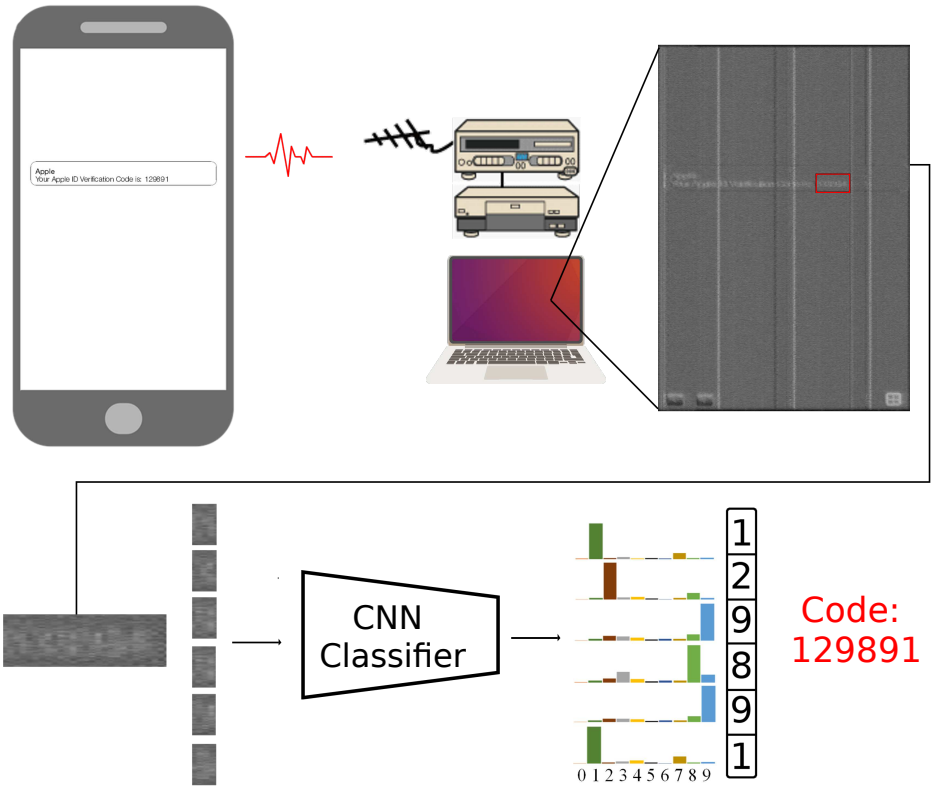


Figure 42: Screen gleaning attack. The target emits electromagnetic side-band intercepted by an antenna connected to a software-defined radio (SDR). The leaked information is collected and reconstructed as a gray-scale image (emage). From emage, the 6-digit security code is cropped and fed into a CNN classifier for recognition.

- The context for the attack is a side-channel analysis setup for a passive adversary, featuring an antenna that has been positioned to collect electromagnetic emanations and an SDR device for signal processing. The antenna picks up the signal from close range.
- During the attack, the attacker can collect electromagnetic traces from the target device representing the image displayed on the screen. The traces are analyzed for the appearance and identification of the pincode.

We now explain the attack in more detail. The device under attack (Figure 42 upper left) is assumed to be a standard device (e.g. a phone) and comply with the standards imposed by EMC regulations laws. The attacker can only rely on unintentional electromagnetic leakage of the device under attack to reconstruct the image displayed on the victim's screen. The leaked electromagnetic signal is

characterized by several physical properties of the screen (e.g., resolution, refresh rate) and by the technology used for the image rendering (e.g., CRT, TFT-LCD). The work of Marinov [Mar14] led to the development of a software toolkit (Figure 42 upper middle) capable of reconstructing the image from emanations of a video monitor. This tool, TempestSDR, is publicly available [Mar] and used as a starting block of our work.

It is important to understand that the challenges involved with the capture and interpretation of electromagnetic emanations from the display cable of a mobile phone are different from those with other devices considered in conventional TEMPEST studies. Given the advance in video display technology, modern screens now use less energy and their circuitry is getting smaller. The resulting electromagnetic coupling is lowered and the carrying frequency of the electromagnetic emanation is increased. Additionally, basic design compliance to guarantee the electromagnetic compatibility of the products helps to reduce unintentional leakages. These factors make the exploitation of this signal more complex and degrade the intercepted signal of electromagnetic emanation of cables.

For completeness, we discuss the future implications of the choices made in our setup. Here, we choose to work in close range and use a near-field magnetic probe. We note that in the future, additional effort can be invested in order to design the antenna that takes into account the electromagnetic properties of the leaking device. A broad description of these characteristics and how to select a matching antenna to the electromagnetic leakage is discussed in [Kuh02a]. We assume that better antennas will relax the constraints of our attacker model in the future. Some relevant work about designing antennas for a better electromagnetic setup was done in [SS13].

We next turn to discuss the "profiling stage" of the attack in more detail. As previously mentioned, if an emage has a low signal-to-noise ratio (SNR), it is impossible for the attacker to read the emage with the naked eye. In this case, in order to interpret the image and recover the screen content, the attacker must use machine learning to analyze and interpret the emage. To realize the machine learning classifier, it is necessary to train it on examples of the signal of the antenna, which is the "profiling" part of our attack. The attacker uses the profiling device to display specific images with known content and captures the emages that correspond to these emages. The collected emages are labeled with the image content and constitutes the training data set.

Once the model is trained, the attacker will be able to record emages from the device under attack to derive the secret information displayed. The process is illustrated in Figure 42. The success of the attack is measured as the classification

Dimension	Description
Message	A six digit security code; each content digit can be 0-9 with equal probability.
Message appearance	The standard size, position, and font with which a security code appears as a push message during a conventional authentication procedure. Plain background and standard brightness are used.
Attack hardware	Close field antenna and standard SDR; we assume immediate proximity of the antenna.
Device profiling	We assume full access to the profiling device for the purpose of collecting training data; We can display an image on the device. We have sufficient time to collect data from several sessions. (2-3 hours.)
Computational resources	About 24 hours on a standard laptop, or 1 hour on a laptop with a GPU for training. For recovery, once the emage has been captured, a matter of seconds.

Table 16: Five-dimensional attacker model: Specifications of the attacker model used in our security code attack

accuracy, which quantifies the ability of the classifier to recover the six-digit security code.

In our experiments, we first set up our attack using the same device at the profiling device and the target device. Considering the same target for profiling and attack phases allows us to understand the danger of the attack under best-case conditions for training data collection. Later, we extend the attack to using two different devices. We consider a device of the same make and model to collect data, and also the situation in which the profiling device is another phone altogether.

We close this section by explicitly summarizing the difference between our attack model and those previously studied in the literature. Because of the specific challenges of mobile phones discussed above, the types of attacks that are successful are not the same as the attacks previously discussed in the literature for other devices. While the TEMPEST technique has been known for decades, there have been no demonstrations of it on mobile devices. The attack model for mobile phones until now has assumed the exploitation of reflections of a visible-spectrum signal, which means that the information is supposed to be visually accessible to

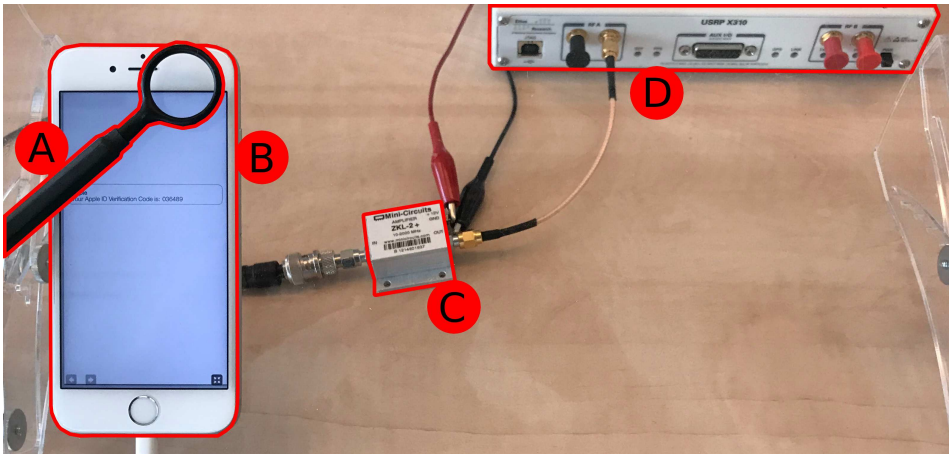


Figure 43: Measurement setup. (A) near-field probe, (B) targeted phone displaying a security code, (C) power amplifier, and (D) the software-defined radio.

humans [BDU08; Bac+09]. Other attack setups exploiting electromagnetic sideband have the goal to do key recovery from cryptographic implementations running on the phone [GS15]. Our work is different as it shows for the first time the threat of TEMPEST on a range of mobile phones for a (machine learning-assisted) adversary that can extract the screen content that could appear incomprehensible to humans.

In Section 7.6, we will provide additional discussion of the attacker model, describing how future work can build on and extend it. We emphasize that the attack that we present in this section is important because it reveals the danger in anticipation of the development of more sophisticated attackers.

7.4 ATTACK SETUP

7.4.1 *Measurement Setup*

7.4.1.1 *Target*

A TEMPEST attack can potentially be performed on any communication device, whether mechanical or electrical, as long as the signal involved for the communication can be intercepted by a third party using unconventional means. It is nontrivial to define such a means, and also the cause of the communication leakage, because this leak has not been designed. Leaks have been shown in the literature to be of several forms linked to the physically inherent properties of the communication signal.

Phone	Leakage Center Frequency	SNR	Screen Technology	OS
iPhone 6s	295 MHz	33.4dB	IPS LCD	IOS 10.2.1
iPhone 6-A	105 MHz	25.0dB	IPS LCD	IOS 12.4.8
iPhone 6-B	105 MHz	26.8dB	IPS LCD	IOS 12.4.8
iPhone 6-C	105 MHz	24.9dB	IPS LCD	IOS 12.4.8
Honor 6X	465 MHz	36.6dB	IPS LCD	Android 7.0
Samsung Galaxy A3	295 MHz	25.9dB	AMOLED	Android 5.0

Table 17: Screen specification of the targets

Our work focuses on electronic personal mobile devices leaking an analogue video signal as electromagnetic emanation. The signal leaks from the ribbon cable that connects the graphical computing unit to the screen. Note that the attack we studied here would be blocked in the case that video encoding is applied to the video signal. The vulnerability of encoded signals needs to be investigated in future work.

The cable, which conveys the electric information, acts as an undesired antenna and transmits the video signal in the electromagnetic spectrum in the surrounding area. An impedance mismatch between the cable and socket on both the motherboard and the display can enhance the ribbon cable's leakage. The difference of impedance is possibly caused by a dimension mismatch between the socket and the ribbon cable. The connecting cable is often designed to be smaller than the socket to avoid possible interference between neighboring connectors. Since each manufacturer is free to use a different offset for these cables, different phones radiate with varying signal strengths. Future research should prove the hypothesis that different phones have different signal strengths radiated, by means of quantifying the radiated signal. According to [Mar14], the frequency of the leaked signal is dependent on several screen properties and can be estimated at a specific frequency (and its harmonics) with the following relation: $f_v = x_t \times y_t \times f_r$, where x_t and y_t are respectively height and with of the screen in pixels and f_r is the screen refresh rate in Hertz (Hz).

The principal target in the experiment section is an Apple iPhone 6s with an IPS LCD screen of size 1334×750 pixels. We also present results using different targets to prove the portability of the attack. The different targets used are listed in Table 17 with the center frequency of the strongest video signal leakage, the SNR of the leakage as well as relevant information about the targets (screen size,

technology and Operating System version). The SNR is computed at the center frequency of the signal with a bandwidth of 50 MHz and a resolution of 25 kHz.

7.4.1.2 *Equipment*

Figure 43 shows an overview of the setup with the elements labeled as follows. The antenna we use is a passive Langer RF-R 400 magnetic probe (A). The target is an iPhone 6s (B). The signal from the probe is amplified with a Minicircuits ZKL-2 amplifier (C) and digitized with a Software-Defined Radio (SDR), an Ettus X310 (D) with a UBX-160 daughter-board. The signal acquired by the SDR is then interpreted with TempestSDR [Mar], an open-source tool capable of reconstructing an image from the display by the obtained sequence of electromagnetic leakages [Mar14].

7.4.1.3 *Positioning and Parameters*

We use SCA equipment to show a proof of concept of this attack because the parameters and positioning settings are close in the two contexts. Nonetheless, using more specialized equipment for TEMPEST attacks may achieve better results. The magnetic probe is placed on top of the target, at a close distance ($<$ 1cm). The best position and distance of the probe from the target is manually optimized to observe the best possible signal to noise ratio (SNR).

TempestSDR has a number of parameters to configure the SDR and to recover the image from the signal. The SDR has the following parameters: center frequency, bandwidth, and sampling rate. The bandwidth and sampling rate are fixed to 12.5 MHz and 25 M samples per second respectively. The SDR captures a bandwidth of 12.5 MHz around the adjustable center frequency. We adjust the center frequency to determine the best SNR. The parameters to recover an image from a signal are: height and width in pixels and refresh rate in frames per second. There are also sliders to adjust the gain and low pass filter of the SDR. The values for the width and the height do not necessarily correspond to the dimensions of the screen as more pixels may be transmitted than those that are displayed. The selected refresh rate should be the closest possible to the actual refresh rate and can be configured with high precision in the software. The parameters require high precision and differ among devices, they should be determined following the description in [Mar14, Section 4.2].

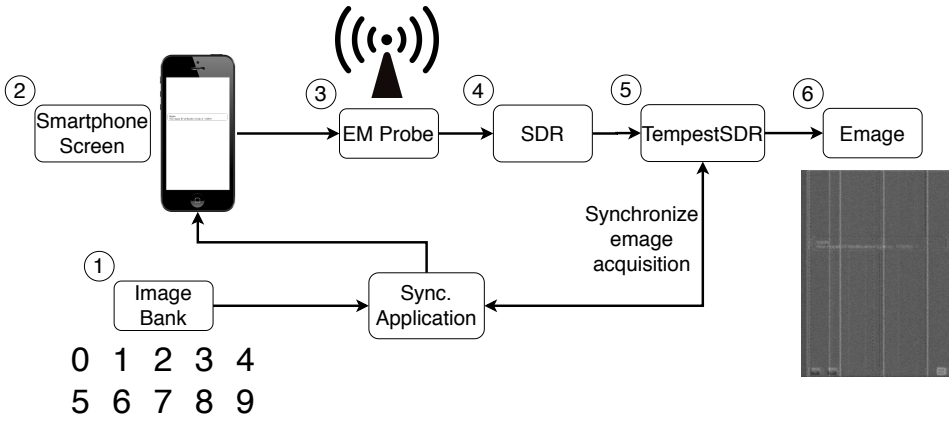


Figure 44: Automation workflow

7.4.1.4 *Automation*

The TempestSDR software contains a built-in function to store a processed frame. The image captured from the reconstruction of the frame is called the emage. For timing efficiency and reliability of the capturing process, we use an automated approach to emage acquisition. Specifically, we set up an application that synchronizes the selection of an image in the image bank, displays it on the screen and saves the emage (see Figure 44). This application consists of a Javascript server and a simple website. Additionally, a small modification to the TempestSDR software was made to automatically save images and communicate with the server. The TempestSDR sends a signal to the server to display an image from the image bank. The server communicates this to the webpage loaded on the phone and the webpage reports back when the image is changed. The TempestSDR captures a parametrizable number of emages and asks for a new image.

7.4.2 *Machine Learning Setup*

Here, we describe the collecting process of emage data sets used to train our security code classifier. Given an emage from the device under attack, the classifier can produce a prediction of the message, which contains a six-digit security code, displayed on the smartphone screen.

It is important to note that the attacks we investigate here can be formulated within a discrimination scenario. This means that the goal of the attack is to discriminate between a set of messages about which the attacker has full information. For example, in the security code scenario, the attacker knows that the security

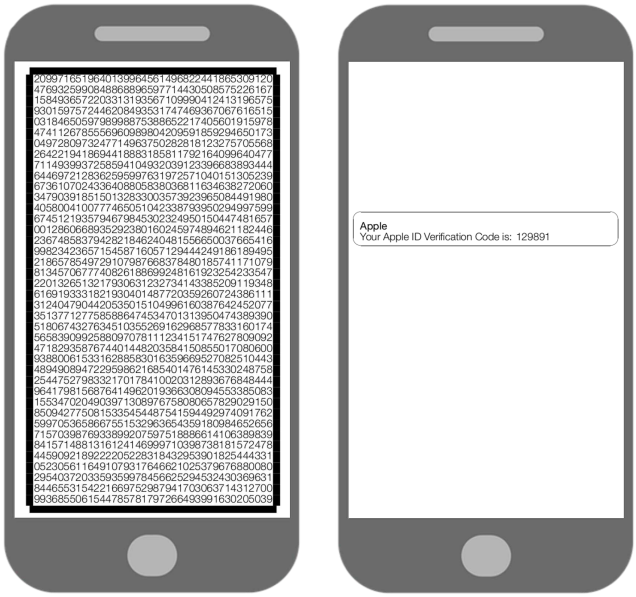


Figure 45: Screen display used to collect digits from a multi-crop grid for training our classifier (left) and from an automated text message containing a security code for testing (right).

code consists of six places and the symbol in each of those places is a digit from 0-9. It is important to contrast the discrimination scenario with a *reconstruction scenario*. The scenarios differ in the amount of information about the content of the screen available to the attacker. It is also possible to formulate screen gleaning attacks within a reconstruction scenario. Here, the goal is to recover the content of the screen exactly as displayed on the screen without using any prior knowledge of what content might be displayed. The reconstruction problem will be discussed further as an outlook onto future work in Section 7.8.

7.4.2.1 *Data Collection*

To train a classifier, the attacker needs to collect training data from the same distribution as the practical data shown on the target device or from similar types of data from other devices. Practically, collecting security code data directly from text messages needs a large amount of annotation effort, since people have to inspect each message and crop the code one by one. Considering such inconvenience, we propose to generate images depicting different numbers (0-9) over the whole image, and collect data using a multi-crop approach. Specifically, each single image is split into $40 \times 40 = 1600$ cells of digits, as shown in Figure 45 (left).

Data	Displayed Content	iPhone 6s	iPhone 6-A	iPhone 6-B	Honor 6X
Train/Val/Test	Multi-Crop Grid	10	5	N/A	5
Single-device Test	Security Code	2	2	N/A	2
Cross-device Test	Security Code	N/A	N/A	1	N/A
Magazine Test	Security Code	1	1	N/A	1
Noise Test	Security Code	1	1	N/A	1

Table 18: The list of collected data sessions for different phones in the security code attack. Multi-crop grid data represents the data collected in the case of multi-crop, and security code data represents the simulated text message with the security code.

Digits	0	1	2	3	4	5	6	7	8	9	All
Acc. (%)	87.2	86.8	97.4	75.8	99.1	97.4	95.1	93.1	82.5	86.1	89.8

Table 19: Accuracy with respect to different digits (0-9) and overall accuracy in our security code attack.

Accordingly, after each trial of emage generation, we can get 1600 emages of different digits. We crop the instances with a certain human inspection to guarantee the data quality. We conduct multiple sessions of emage generation to alleviate the influence of distribution shift, which is validated effective as in Section 7.5.2.

7.4.2.2 *CNN Architecture and Model Training*

For the model architecture, we adopt the simple LeNet [LeC+98], which was initially proposed for handwritten digit recognition. We slightly adapt the LeNet to fit our input emage size of 31×21 (for Honor 6X the input size is 45×21, and for iPhone 6 is 31×20). The details of the architecture is shown in Figure 47. The PyTorch is used for our implementation and the experiments are run on a workstation with a 16-core CPU and a GTX1080Ti GPU. In all cases, 80% of multi-crop grid data are used for training, 10% for validation and 10% for testing. Each round of training can be finished within one hour when using the Adam optimizer [KB15] with a learning rate of 0.001. We conduct the training over 100 epochs with a batch size of 256, and select the optimal model based on the validation accuracy.

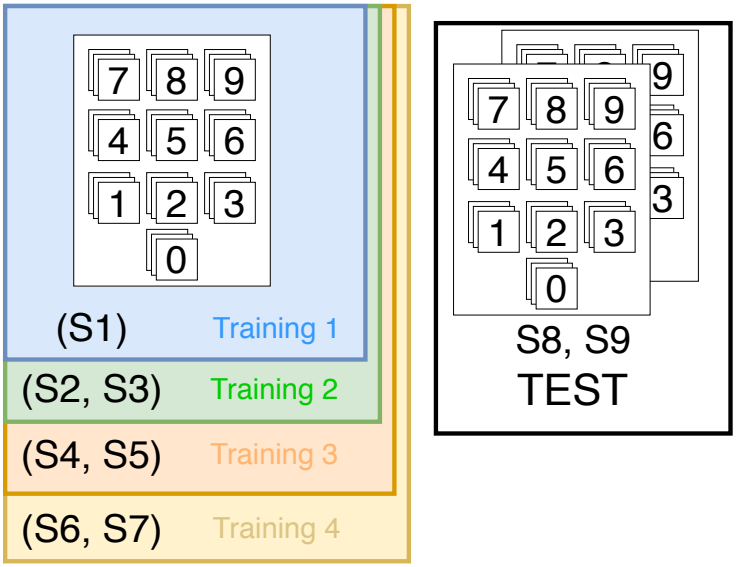


Figure 46: Train/test splits specified in the case of multi-crop grid, where the training set is gradually enlarged by including more sessions, and the test set is fixed with two sessions.

	6 digits	$\geq$ **5 digits**	$\geq$ **4 digits**
Acc. (%)	50.5	89.5	99.0

Table 20: Accuracy of predicting partial security code correctly with the CNN classifier in our security code attack.

7.5 EXPERIMENTS

In this section, we first conduct experiments on iPhone 6s to analyze the properties of our attack on the basic single-device scenario. Specifically, we look into the dimensions that can potentially impact the classification performance, such as size and heterogeneity of the training data (Section 7.5.2), for further analyzing the attacker's capability in various attack settings.

Then, we test our attack using more phones (iPhone 6-A, iPhone 6-B, and Honor 6X) to validate the effectiveness of our attack in more challenging scenarios, such as cross-device attack, magazine occlusion, and interference from environmental signal noise. The specifications of different phones can be found in Table 17, and the detailed data collection settings are shown in Table 18.

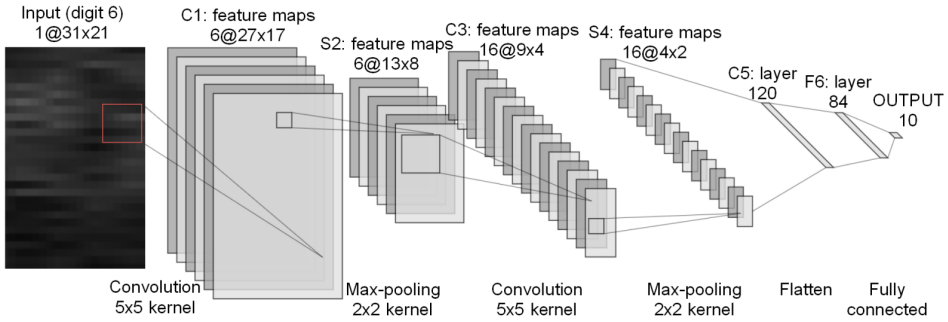


Figure 47: CNN architecture used in our security code attack.

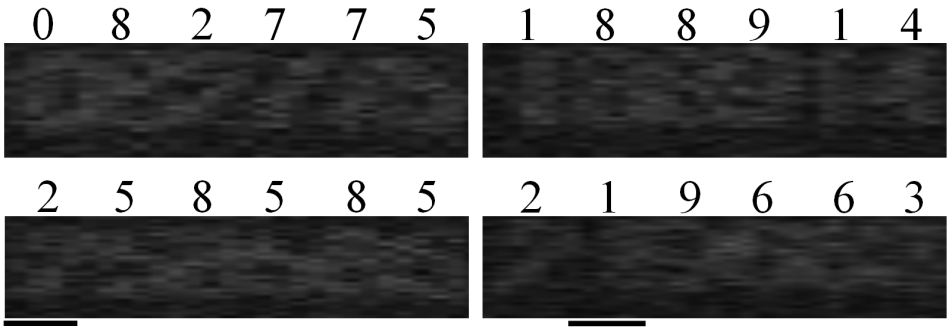


Figure 48: Examples of cropped 6-digit security code. Ground truth labels are shown above each strip, with the underline highlighting the wrong prediction of digits by our classifier.

7.5.1 *Security Code Attack*

In our practical security code attack, we use an Apple iPhone 6s as the target device. We collect 10 sessions of grid data, each of which contains 32000 emage examples. Through human inspection, we drop one session due to an obvious data quality issue. For inter-session evaluation on the grid data, 2 of the remaining 9 valid sessions are fixed as the test set for all the experiments, where session 8 represents a well-positioned antenna scenario and session 9 is for badly-positioned antenna scenario. The training set is gradually enlarged by adding more of the remaining sessions. Specifically, we try four sizes of training set, which respectively consist of 1, 3, 5 and 7 sessions, denoted as Training 1, Training 2, Training 3 and Training 4, as illustrated in Figure 46. Each resulting digit emage will be fed as input to train our CNN classifier, following the principles in Section 7.5.2.

We simulate 200 text messages, each of which contains a 6-digit security code, making sure they look very close to the real case, as shown in Figure 45 (right). In

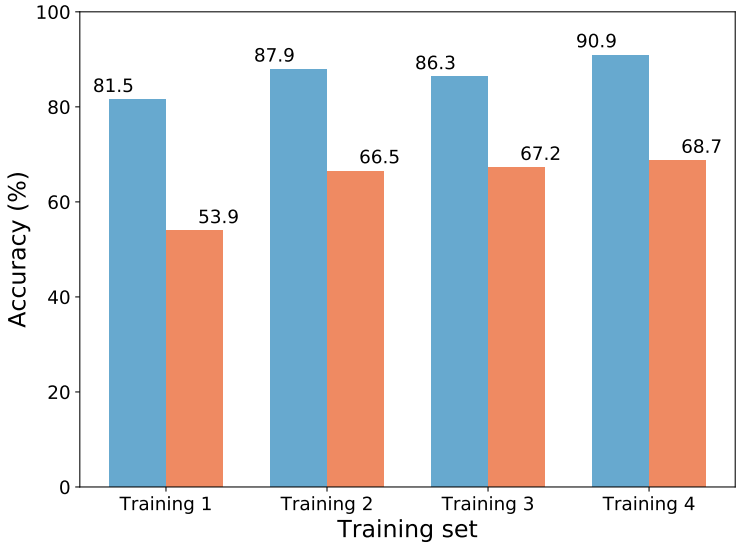


Figure 49: Inter-session accuracy (grid data) of our security code attack for different training sets with gradually increased size. The two bars for each training set represent two different test sessions (session 8 and 9).

this case, each emage of the security code, with a size of 126×31, (see Figure 48 for some examples) is evenly divided into six.

The best overall accuracy (89.8% in Table 19) with respect to all $200 \times 6 = 1200$ individual digits is achieved when using all of the 7 training sessions (more details about the impact of training data amount will be discussed in Section 7.5.2). As can be seen in Table 19, the accuracy differs for different digits, with the highest (99.1%) achieved for digit 4, and lowest (75.8%) for digit 3. Figure 48 shows some examples for the security code along with the ground truth and prediction results. It demonstrates that our approach can correctly predict the digits with high accuracy, although the digits are hardly recognizable to the human eye.

In practice, attackers may have various query budgets for fully uncovering the security code (with all the 6 digits being correct). So, in Table 20, we present the accuracy results when four security code digits or more can be correctly predicted by our classifier. It can be observed that, with one attempt, the attacker can fully recognize the security code at 50% of the cases. The probability of recognizing four or more digits can reach 99%, showing that our approach can present a serious threat in practice.

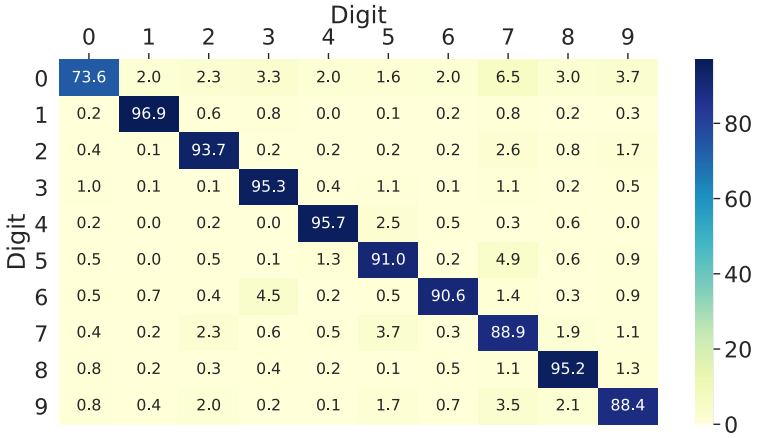


Figure 50: Confusion matrix of the inter-session accuracy (grid data) in our security code task. Results are from the classifier trained on Training 4 and tested on session 8.

7.5.2 *Data Analysis on Grid Data*

We first consider the scenario where the attacker can train the classifier on data sampled from the same distribution as the attacked security code. This can be regarded as the best-case scenario, although almost impossible in most practical cases. Specifically, we achieve an accuracy of 86.5% within the session used in Training 1.

Inter-session evaluation represents a more realistic attack scenario, where the training data from the same session of the target is not accessible, but the attacker can simulate similar data using the same settings. Figure 49 shows the inter-session accuracy of the four classifiers trained on different training sets: training 1, 2, 3 and 4. It can be observed that the accuracy improves as we increase the number of training sessions. We can also observe that inter-session accuracy with only one training session is lower than the multi-crop grid case. However, using more training data with multiple sessions could alleviate this issue, leading to a high accuracy of 90.9% for Training 4 (with seven training sessions). This validates our assumption that incorporating heterogeneous sessions could help alleviate the impact of the random noise introduced to the emage generation. One detailed classification result with respect to different classes are shown in the confusion matrix in Figure 50. We also notice that there is a difference between the prediction performance between two test sessions, which might be explained by their different data quality.

Figure 51: Pictures of the Magazine setting (left), with the phone in between the magazine pages and the probe on top, and the With Noise setting (right).

Acc. (%)		iPhone 6-A	Honor 6X
Test (Grid)		73.42	94.38
Single-device-1 (Security code)		41.42	74.00
Single-device-2 (Security code)		47.08	74.00
Magazine (Security code)	70 pages	14.38	-
	200 pages	-	65.79
With Noise (Security code)		63.29	64.25
iPhone 6-B (Security code)		61.54	-

Table 21: Inter-session classification of our security code attack for different phones and different test settings. Grid means the multi-crop grid test data is used, and Security Code means the simulated text message test data is used. The training set stays the same in all test settings for each device. Single-device-1 and Single-device-2 refer to two different test sessions.

7.5.3 *Experiments on Other Phones*

In this section, we conduct experiments on different phones to further validate the general effectiveness of our security code recognition on different devices. We

show the potential of the recognition in more challenging and realistic scenarios, including cross-device attack, antenna occlusion by a magazine, and interference from the signal noise generated by surrounding phones (cf. Figure 51). The cross-device attack consists of training the recognition algorithm on the data from one device and testing the model on data from another unit of the same model. Specifically, we use two iPhone 6, namely, iPhone 6-A and iPhone 6-B, and make sure that they have the same version of the iOS system, and not refurbished. Five sessions of data are collected for training the recognition model on iPhone 6-A, and two test sessions of security code data are collected for testing. Additionally, we collect a session of testing data with the antenna occluded by a magazine, another test session from iPhone 6-B and a test session with background noise. The measurement setups for occluding the antenna and simulating the background noise are shown respectively in Figure 51. Each of the above four testing sessions contains 200 different security codes and for each code, we repeat the frame twice for a more stable recognition. Our attack can also work on a refurbished iPhone (iPhone 6-C, see Table 17), but no quantitative results are reported in order to maintain fair comparison.

Table 21 summarizes our experimental results under different test settings corresponding to the data descriptions in Table 18, i.e., Multi-crop, Single-device, Magazine, Noise, and Cross-device. As can be seen, our model achieves high accuracy for the original multi-crop data. For other settings, as expected, the performance drops due to the generalization gap but still being effective enough in most cases. Specifically, the high cross-device accuracy suggests the effectiveness of our attack in a more realistic scenario, where the device used for collecting the training data is not necessarily the target device. The results on an Android phone, Honor 6X, with four sessions of training data, verify that the effectiveness of our attack is not limited to the specific phone type, iPhone. We can also observe that the single-device and cross-device sessions of the security code yield different prediction performance, which might be explained by their different data quality, as also reported for iPhone 6s (cf. Section 7.5.1).

For the magazine setting, the accuracy drop can be explained by the signal strength of the antenna. The magnetic probe can be considered as a magnetic dipole, for which it holds that the power density is dependent on a factor r^{-5}, for which r is the distance between the probe and the origin of radiation [Ida00]. Therefore, placing the magnetic probe a little bit further away from the origin of radiation, already has some significant consequences on the quality of the received signal. Specifically, we find the performance of iPhone 6-A drops dramatically with 70 pages, but for Honor 6X, the performance is better maintained even with a thicker magazine of 200 pages because of its higher leakage of signals. The high

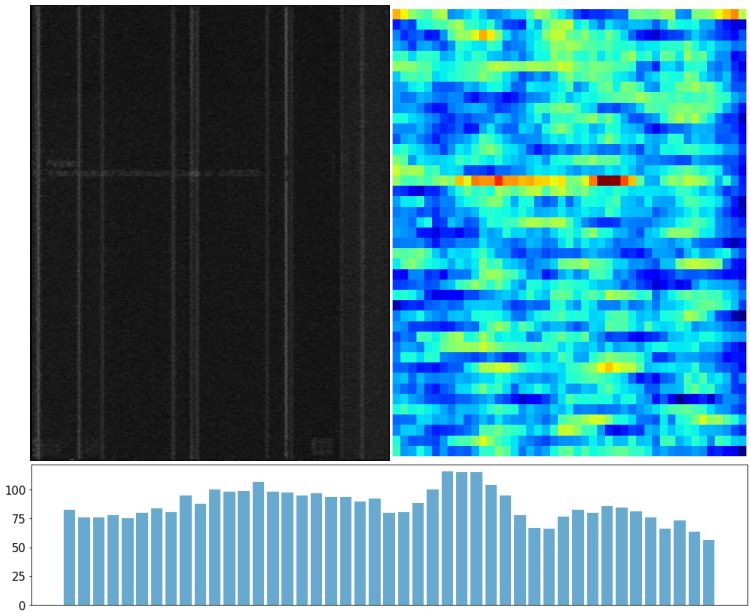


Figure 52: Top: An emage and its predicted activation map by the pre-trained model on iPhone 6s. Warmer color represents higher prediction confidence. Bottom: Activation responses in the row of the text message

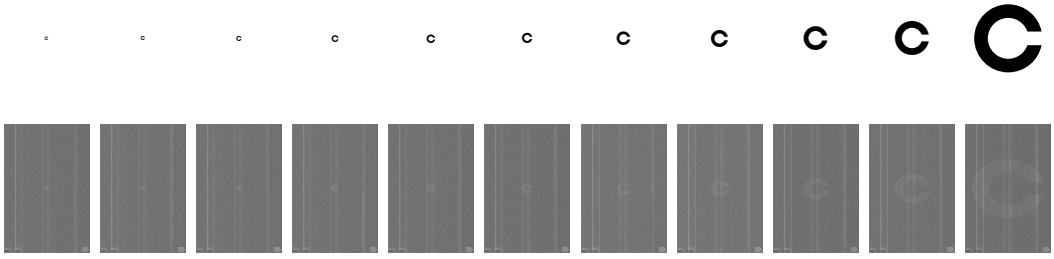
single-device accuracy (74% for both) also confirms this higher signal leakage of this Honor 6X phone than iPhone 6.

We also find that our attack can work on the OLED screen by conducting a preliminary exploration of Samsung Galaxy A3 (2015). However, since this phone is disassembled, we do not go further for quantitative details.

7.5.4 *Discussion*

In practice, the localization of security code patterns in either time or space dimensions is crucial. Here we discuss how a simple sliding window technique can tackle both. When monitoring the target phone in real-time, we can also integrate our recognition model with a simple sliding window operation to identify the key frame(s) that are most likely to contain a text message of the security code. Specifically, we set the height of the sliding window as the height of each digit, and the width as the total width of 6 digits. The horizontal and vertical strides are equal to the height and width of each digit.

As shown in Figure 52, the message area is activated much more than the plain area, indicating that our recognition model can be used to identify the most likely



(a) 1x (b) 1.2x (c) 1.5x (d) 2x (e) 2.5x (f) 3x (g) 4x (h) 5x (i) 7x (j) 10x (k) 20x

Figure 53: Images (top row) and their corresponding emages (bottom row) of letter C displayed at 11 different scales. It can be seen that the scales span from uninterpretable to the human eye to easily interpretable.

frame(s). Furthermore, within the specific row of the text message, the highest activation responses are concentrated on the security code region. This suggests that the textual background will not interfere in our security code recognition. It is also worth noting that, in practice, the attacker could also leverage off-the-shelf language models or visual detection models. Such models would provide a straightforward way to boost the localization performance. We also mention that in our experiments, we use a maximum contrast between the background and the text. Reducing the contrast leads to a less easily readable screen for the human eye, but does not necessarily result in an emage that is more difficult to interpret. Exploratory experiments confirmed that our choice of background represents a challenging setting, and that, if the attacker is lucky, the contrast between the background and the message on the display of the phone might actually make the attack easier.

7.6 TESTBED

So far, we have introduced the screen gleaning attack and shown its effectiveness in recovering a security code displayed as a push message on the screen of a mobile phone. The attack was carried out with technology representative of the current state of the art. However, with time, we expect the quality of the antenna and SDR to improve. Also, additional training data and algorithmic advances will increase the accuracy of the deep learning classifier. These advances mean that screen gleaning attacks can be expected to become increasingly dangerous, and future work will be necessary to understand them and develop countermeasures. To support this future work, we have developed a testbed that enables the systematic test of screen gleaning attacks under incrementally more challenging attacker models. In this section, we describe the testbed, which has also been released,

so that can be directly used by the scientific community. The testbed consists of two parts, first, a definition of a set of images and a set of scales, and, second, a specification of the attacker model, in terms of the model dimensions and the parameterization of the dimensions. We also report the results of experiments validating the testbed.

7.6.1 *Testbed Images*

We base our testbed on the eye chart used by eye doctors to test vision acuity [Bos], and for this reason, we call it the *eye chart testbed*. Most people are familiar with the experience of a vision test. The eye chart measures someone's vision by determining the minimum level of detail that the person's eyes can distinguish at a given distance. Likewise, our testbed uses eye chart letters to determine the minimum level of visual detail that a screen gleaning attack can recover given a particular attack setup.

The testbed is deployed by first specifying an attacker model and creating an attack setup based on that model. Then, different scales are tested until it is possible to determine at which scale the identity of the letter can no longer be recovered by the attack.

The testbed defines 11 different scales. For the largest scale (20×), the size of the letter is the maximum size that can be fit on the screen, with still leaving 10% of the letter width as margins on the side. For the smallest scale, the letter appears with a width of 1/20 of the largest scale. The relative sizes of the testbed scales are illustrated on the top line of Figure 53, using the letter C as an example. The font is the Sloan font used for eye charts. We used the Creative Commons licensed version, which is available on GitHub.[2] The full letter set in the testbed release is C, D, E, F, L, N, O, P, T, Z. The full set of letters is tested as each scale.

The letters in an eye chart are chosen so that all the letters in the set are equally easy to read. This ensures that for each scale, the ability of the person to read the letters is related to the scale, and not to the specific letters. By choosing to use eye chart letters, we extend this property to our test set. Different eye charts use different fonts and different letter sets. We choose our testbed based on the fact that this set is currently in widespread use.

It is natural to wonder why we use the limited set of characters used in an eye chart instead of using a larger set of alphanumeric characters. The answer is that the testbed is designed to detect the ability of an attack to discriminate and recover visual detail. Using eye-chart characters means that the results of the testbed re-

2 https://github.com/denispelli/Eye-Chart-Fonts/blob/master/README.md

Dimension	Description
Message	The symbol set (e.g., 0-9, a-z) must be defined. If the symbols are not all equiprobable, the prior probability of each symbol must be defined.
Message appearance	Any constraints that will be imposed on the scale of the message or on font types must be defined. Assumptions about the pattern of the background and the brightness of the screen must be defined.
Attack hardware	The antenna and the SDR must be specified. Any assumptions on the position of the antenna must be defined (positions range from touching the phone, to under the table, to across the room).
Device profiling	The conditions on device access must be defined (attacker has access to the device to be attacked, to devices of an identical model, to devices of the same make). The ability of the attacker to cause a certain image to appear on the accessible devices must be defined, along with the amount of time that the attacker can count on having access. After the number and nature of devices at the attackers' disposal is defined, the number and length of the sessions that the attacker can record on each device must also be specified.
Computational resources	Define the amount of time and computational resources available for training, and also for the attack itself (i.e., after the model is trained recovering the message from the emage).

Table 22: Five-dimensional attacker model: Parameter settings to specify when designing an attacker model for testing with the testbed.

flect the discernability and interpretability of other forms of visual information as well, for example, symbols or images displayed on the phone screen, and not just text.

Figure 53 depicts emages that were captured with the setup described in Section 7.4. It can be seen that they move from being uninterpretable to the human eye on the left to interpretable on the right. This property of the testbed has the goal of ensuring that the testbed can measure interpretability with other attack setups. We are especially interested in supporting the investigation of attack setups where the signal might be very weak, for example, as the antenna is moved further from the phone. For a very weak signal, the larger letters will become uninterpretable to the human eye. This will allow researchers to quantify the effectiveness of a machine-

learning attack under the conditions of a weak signal. If researchers adopt the same standard testbed, the measurements made can be more easily compared in a fair manner.

Again, it is important to note that although our testbed consists of letters, it does not specifically assess the ability of an attack to recover written text consisting of letters. Instead, it assesses the ability of the attack to recover a message that has a certain level of visual detail. Just like the eye chart tests general visual acuity, and not just reading, our testbed tests the acuity of a particular attack to recover information in the visual form displayed on the phone screen, and not just letters.

7.6.2 *Parameterization of the Attacker Model*

Here, we describe the parameterized attacker model. It contains five dimensions, message, message appearance, attack hardware, device profiling, and computational resources. Each of these dimensions has several parameters. In order to have a fully specified attack mode, specifications must be made for each of the parameters. The parameters can be considered to correspond to the values of design decisions. The five dimensional model along with the parameters for each dimension are described in Table 22. Note that in the security code attack we present in Section 7.3, we use the same five dimensions in the attacker model (Table 16).

This parameterized attacker model forms the basis for the attack setup. It has two purposes. First, it ensures that when the testbed is being applied, the attacker model that is being assumed is fully described, i.e., no detail is left out. Second, it allows researchers to systematically make the attack stronger. The attack strength can be increased by increasing the values of any or all the parameters. In this way, the attacker model guides researchers in discovering increasingly strong attacks. The dimensions of the attacker model can be also used to guide the development of countermeasures.

7.6.3 *Validating the Eye Chart Testbed*

In this section, we validate the eye chart testbed with the demonstration of an attack. The attack uses the same Attack hardware and Computational resources as the Security Code attack demonstrated in Section 7.3. The Message and the Message appearance are derived from the eye chart testbed. The Device Profiling is also the same, and the specifics of data collection are explained in the next section.

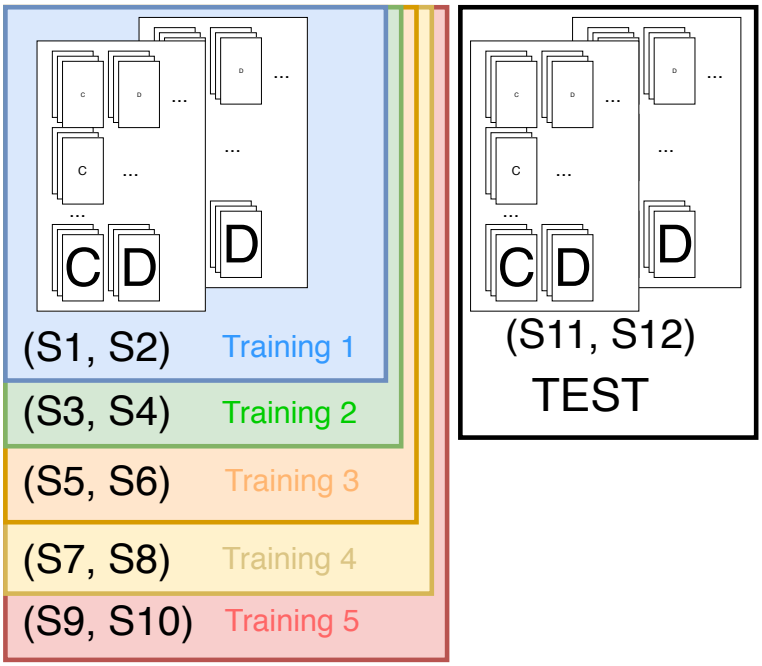


Figure 54: Train/test splits specified in the inter-session case of our eye chart letter classification task, where the training set is gradually enlarged by including more sessions, and the test set is fixed with two sessions.

7.6.3.1 *Data Collection*

We collect a total of 12 sessions, among which two sessions (sessions 1 and 2) have 50 samples for each of 11 classes of letters at each of 10 scales, and the rest 10 sessions have 15 per class per scale. For inter-session evaluation, we use sessions 11 and 12 as testing sets for all the experiments. Session 1 plus 2 are used as the initial training set, and are gradually enlarged by including two more sessions each time, resulting in five different training sets with increased size, denoted as Training 1, 2, 3, 4 and 5, as illustrated in Figure 54. Training 5 has the most data with 24200 samples.

7.6.3.2 *Experiments*

Similar to the security code attack, we use the following partitions: 80% training, 10% validation and 10% testing. We use the ResNet-18 model [He+16] as our classification model, and we train on five training sets individually until convergence. Figure 55 shows that including more training sessions generally lead to performance improvement in the inter-session case. For the second session, we

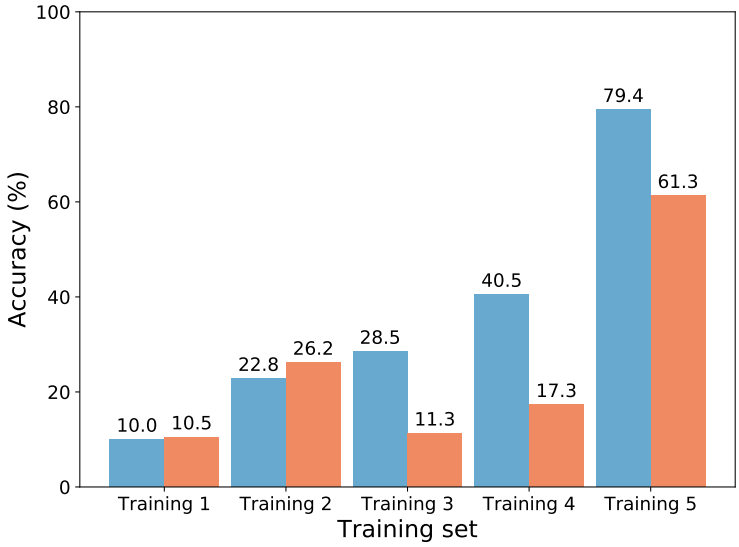


Figure 55: Inter-session accuracy in our eye chart letter classification task. The two bars for each training set represent two different test sessions.

Scale	1	1.2	1.5	2	2.5	3	4	5	7	10	20
Acc. (%)	66.7	48.7	82.0	87.3	86.7	82.0	88.7	89.3	97.3	98.7	46.0

Table 23: Accuracy with respect to 11 different scales in our eye chart letter classification task.

notice an accuracy drop when including more data from Training 2 to Training 3, which can also be explained by the fact that the data quality of different sessions of data could impact the performance.

Figure 56 shows the confusion matrix of the classification accuracy with respect to different classes. We can observe that accuracy differs for different letters. Table 23 shows the results at 11 different scales. We could observe that the accuracy of the letters at moderate scales (e.g, 7, 8 and 9) is comparatively higher than the others. Without surprise, the smallest scale has the lowest accuracy. However, what we found also interesting is that accuracy with respect to scale 1 is also low. We suspect that it is because of the receptive field of the model we chose. More detailed results per class per scale can be found in Figure 57.

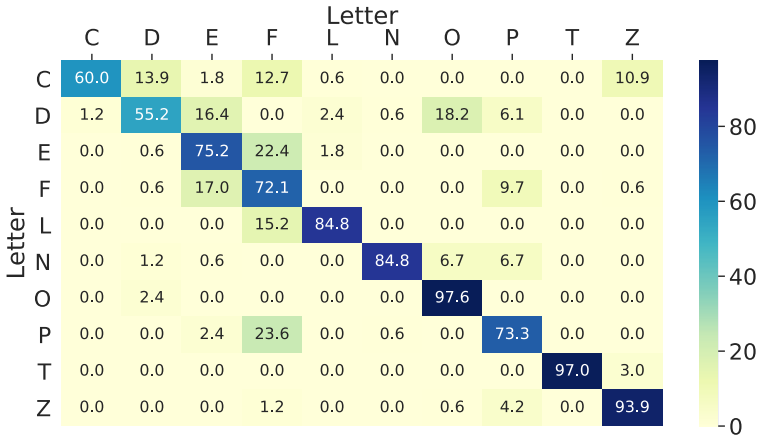


Figure 56: Confusion matrix of the classification in our eye chart letter task.

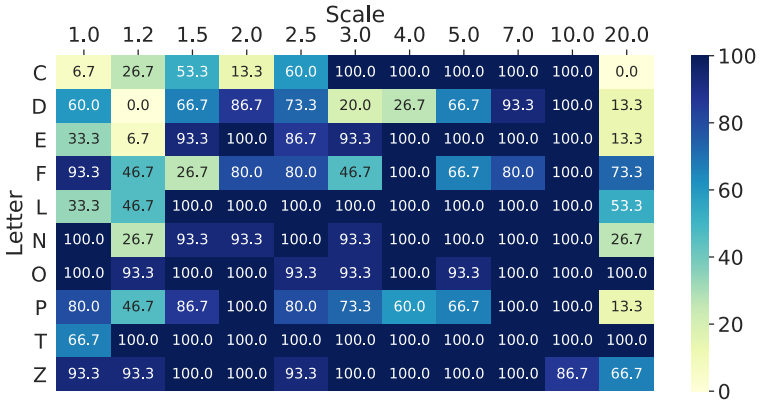


Figure 57: Classification results of our eye chart letter task with respect to different classes and scales.

7.7 COUNTERMEASURES

In our setup, the target device has no extra protection beyond the common design features of commercial devices. As a step towards improving the protection of the device, we discuss possible countermeasures that could possibly mitigate the danger of a potential screen gleaning attack.

7.7.1 *Hardware-Based Approaches*

Screen gleaning attacks would be made difficult by using a shielding technique. Shielding a cable consists of wrapping the center core of the cable that transmits

an electric signal by a common conductive layer. The shield acts as a Faraday cage inside the cable, blocking electromagnetic waves. The resulting electromagnetic leakage is lowered, decreasing the SNR of the signal. Several standard cables (e.g., coaxial cable, twisted pair cable) are shielded to reduce its electromagnetic perturbations and emanations. However, this technique comes at an extra cost and increases the cable dimension. For this reason, flexible flat cables inside small electronics with a display often lack a protective shield, and it is not trivial to add one.

A metallic protective case would also act as a shield for electromagnetic radiation, preventing attacks that measure the signal emitted from the back of the phone, but every telecommunication signal would also be perturbed.

7.7.2 *Communication-Based Approaches*

Another countermeasure against screen gleaning, similar to the method used for pay-TV, could be to encrypt the signal between the graphical unit and the screen. The core idea is to share a cryptographic key between the two entities and encode the video stream using a cipher. As a result, the leaked information by the transmitted signal will become more difficult to interpret by the attacker, who does not have the key. This solution comes at a cost. Although some stream ciphers could meet requirements for throughput and latency, both the screen and the graphical unit would need extra logic for encryption and decryption of the cipher and implement a key establishment protocol to create a shared key when paired together. Moreover, this countermeasure would be ineffective against an attack targeting the screen itself during the rendering (although this is a different attack, see [Gen+19]).

7.7.3 *Graphics-Based Approaches*

M. G. Kuhn in [KA98] introduces a cheap and efficient countermeasure against electromagnetic TEMPEST that consists of a special font where the transmitted signal has been filtered to reduce the strength of the top peaks of its Fourier transform. The resulting font appears visually quite blurry for a high-resolution representation rendered on the screen but makes the side-channel silent.

Another method that can be used as a countermeasure is obfuscation. This obfuscation can either be introduced into the background of the image using confusing patterns and colors behind the text or by using a font with visually difficult to differentiate letters. However, obfuscation is often ineffective against distinguish-

ing methods based on machine learning and may introduce difficulties for humans to read the original image from the screen.

7.8 FROM TEXT TO IMAGE

Here we return to the discussion of different formulations of the screen gleaning problem. As we stated earlier, in the discrimination scenario, the attacker knows a finite set of messages that are possible and attempts to determine which one actually occurred on the phone screen. The security code recovery attack belongs to the discrimination scenario.

As the work on screen gleaning moves forward, it is interesting to look at problems beyond recovering messages built from symbol sets, such as security codes and written words, but also at images. Screen gleaning of images can be addressed within the reconstruction scenario, mentioned above. In this scenario, the attacker has no prior knowledge of the screen contents and attempts to reconstruct the screen exactly as it appears to the human eye. The following is an example of the reconstruction scenario: If the screen was displaying a photo of a person, the goal of the attack would be to recover that photo completely. Complete recovery requires that the features of the person in the photo are clear, as needed for a human viewer to identify the person, but also that the recovered photo looks exactly like the original one including details of the background and the lighting and coloring of the photo.

Screen gleaning of images can also be addressed within a more general classification scenario than the discrimination scenario. The discrimination scenario is a type of classification scenario in which the attacker has access to information about the complete set of possible messages. There exists another classification scenario, which we call the *generalization scenario*, in which the attacker only has some information about the possible content of the screen. Pornography detection is an example of a problem that needs to be addressed in the generalization scenario. We discuss it in more detail here because of its societal relevance, cf. the issue of people looking at porn on their devices on an airplane [Con11; Cur20].

For pornography detection, the attack goal is to determine whether or not a phone display pornography without a direct line of sight to the phone. Here, we assume, it is not possible for the attacker to have complete information in advance about all possible images displayed on the phone. Even if it is possible to access a complete database of all pornographic images, it is not possible to know which non-pornographic images will be displayed. To mount a screen gleaning attack in this case, we must collect representative training data of the different types of phones we expect, similarly to the discrimination case, and different levels of fa-

vorability for antenna positioning. We also, however, must collect representative data of all the different types of pornographic and non-pornographic images that could be relevant to the problem. The data collection task is clearly not trivial. However, this type of scenario is clearly important, so we recommend that future work on screen gleaning focuses not only on discrimination scenarios (as with the security codes) but also on more general classification scenarios (as with pornography detection).

We have based our proposed testbed on a test used for visual acuity, and not specifically for reading. We have made sure that our testbed is not limited to letters and numbers, since we hope that, moving forward, the testbed will be useful for testing screen gleaning in classification scenarios involving generalization and reconstruction. However, assessing the true capacity of our testbed will require validation tests in addition to those carried out here.

7.9 CONCLUSION AND OUTLOOK

In this paper, we have introduced screen gleaning, a new TEMPEST attack that uses an antenna and software-defined radio (SDR) to capture an electromagnetic side channel, i.e., emanations leaking from a mobile phone. We demonstrate the effectiveness of the new attack on three different phones with an example of the recovery of a security code sent in a text message by using machine learning techniques, as the message is not comprehensible to the human eye.

In addition, we propose a testbed that provides a standard setup in which screen gleaning can be tested further with different attacker models. Finally, we provide ideas for possible countermeasures for the screen gleaning threat and discuss their potential.

Future work will involve testing increasingly sophisticated attacker models that can be built by extending the five dimensions of the parameterized model that we propose as part of our testbed framework. As already mentioned, such an extension will involve moving to more sophisticated attack hardware, as hardware continues to develop. We have already identified special electromagnetic near-field scanners [Ems], which are basically arrays of loop antennas that allow the attacker to identify the ‘hot spot’ of the device. The attacker is then able to aim the antenna at this particular spot. These near-field scanners also identify all resonating frequencies within a band of 15 kHz to 80 GHz. These frequencies could then be used for the design of antennas that extend the setup such that attacks on greater distance can be performed.

Further, we will consider a wider range of other devices, including other screens from devices like tablets, laptops and smart displays (such as Google Nest Hub).

For example, the work of Enev et al. [Ene+11] suggests that our conclusion should remain valid for most of the screens, including TV screens.

Finally, we are interested in moving from discrimination scenarios to generalization scenarios, and finally to reconstruction scenarios. In other words, content that the attack recovers from the phone will become increasingly unpredictable, and increasingly challenging. The testbed we presented here has the potential to be further developed to also cover the full range of possible scenarios.

ACKNOWLEDGMENTS

Part of this work was carried out on the Dutch national e-infrastructure with the support of SURF Cooperative. We thank Peter Dolron and Daniel Szálas-Motesiczky of the TechnoCentrum at Radboud University for their support with the measurement setup. A special word of appreciation to Frits, Henan, Jan, Maikel, and Mia, who contributed time with their phones, so that we could carry out screen gleaning attacks.

8

ON REVERSE ENGINEERING NEURAL NETWORK IMPLEMENTATION ON GPU

This chapter is based on [CW], a joint work with Łukasz Chmielewski, that has been presented during the Artificial Intelligence in Hardware Security (AIHWS) in 2021.

CONTENT OF THIS CHAPTER

8.1 INTRODUCTION

Deep learning is more and more deployed in many research and industry areas ranging from image processing and recognition [KSH12], image recognition for autonomous vehicles [FHY19], robotics [KBP13], and natural language processing [TPL10], medical applications [LL19], IoT speech recognition [TSL19] to security [Ku17; Wei+18]. This rapid deployment is caused by the increased computational capabilities of computers and huge amounts of data available for machine learning. Additionally, it leads to more and more complex machine learning architectures.

In this paper, we focus on the analysis of Multilayer Perceptron (MLP) and Convolutional Neural Network (CNN) implemented using GPU accelerators, as they are the most commonly used feed-forward neural networks architectures.

Designing and finding parameters for neural networks has become an increasingly hard task since the NN architectures become more complex. From the industrial point of view, we can observe an increase in the number of intellectual property (IP) of NNs. Such IPs of commercial interest need to be kept secret.

Moreover, in the medical context, the privacy aspects of NNs can also become a threat if revealed.

Additionally, EMVCo, an entity formed by MasterCard and Visa to manage specifications for payment systems, requires deep learning techniques for security evaluations [Ris18]. Due to the above reasons, hackers might want to reverse neural networks to learn secret information.

There exist potentially easier ways to recover a network than using complex side-channels like EM or power consumption. For example, physical access to the device might be sufficient for an attacker to access the NN firmware and to reverse engineer it using binary analysis. As a countermeasure, those devices are equipped with standard protections like blocking binary access, blocking JTAG access, or code obfuscation. Furthermore, the IP vendors usually forbid users to access architectural side-channel information, such as memory and cache due to security and privacy concerns. Additionally, they implement countermeasures in software and hardware against logical attacks that would allow hackers to obtain run-time control on the device.

Therefore, for such protected implementations, side-channel attacks become viable for reverse engineering NNs. Side-channel analysis (SCA) has been widely studied for the last 20 years due to its capability to break otherwise secure algorithms and recover secret information. In 2019, Batina et al. [Bat+19] presented the first SCA attack to extract architecture and weights from a multilayer perceptron implemented on a microcontroller; this attack employed both timing and EM side-channels. This attack has shown that SCA is a serious threat to NNs.

However, there has been little work done on SCA against GPU-based neural networks[1]. To the best of our knowledge, there has been no power or EM side-channel attack presented that targets GPU-based neural networks, while GPU is the platform of choice to train and deploy neural networks.

In this work, we aim at evaluating the security of a setup that is as close to a real-world application as possible, and therefore, we target NNs running on GPU. Therefore, we target the Nvidia Jetson Nano, a module computer embedding a Tegra X1 SoC combining an ARM Cortex-A57 CPU and a 128-core GPU with Maxwell architecture. This hardware accelerator is relatively complex in comparison to a simple microcontroller. In particular, our setup employs the PyTorch python framework running on the full Linux operating system (on the ARM CPU) to instruct the GPU accelerator to execute NN computations. This complexity poses several technical difficulties for our analysis due to a large amount of noise

1 The only SCA against GPU-based NN that we have found is presented in [Wei+20a]. However, it works in a different context to ours and is based on software context-switching timing side-channel; see Subsection 8.1.1 for details.

and misalignment. Because of these challenges, we limit our analysis to so-called simple EM analysis[2], and we recover numbers of layers and neurons as well as the types of activation function being executed. Our experiments show that all this secret information can be recovered using a dozen of EM traces independent of inputs even when significant noise and misalignment are present when sufficient signal processing techniques are used.

Note that we need to analyze a GPU implementation as black-box since the low-level details of the implementation could not be public. Moreover, due to the parallel nature of GPUs, we cannot simply replicate existing attacks for other architectures, but adjust them adequately.

We leave recovering neuron weights and CNN hyperparameters using more complex side-channel attacks to be future work.

8.1.1 *Related Works*

Any computation running on a platform might result in physical leakages. Those leakages form a physical signature from the reaction time, power consumption, and EM emanations released while the device is manipulating data. Side-channel analysis (SCA) exploits those physical signatures to reveal secret information about the running program or processed data In its basic form, SCA was proposed to perform key recovery attacks on cryptographic implementations [Koc96; KJJ99].

The application of SCA is not limited to the type of processing unit and can be applied to microcontrollers as well as other platforms. In [Luo+15; JFK16; JFK17; GZC18], EM and power side-channel attacks are performed on GPU-based AES implementations.

In [Nag+19] SCA is used to break isolation between different applications concurrently using a GPU. Essentially this work identify different ways to measure leakage using software means from any shared component.

Side-channel attacks can be applied to extract the information of a neural network. Batina et al. [Bat+19] presented the first EM side-channel attack to extract the complete architecture and weights from an MLP network implemented on a CPU. Subsequently, Honggang Yu et al. [Yu+20] combined simple EM analysis with adversarial active learning to recover a large-scale Binarized Neural Network, which can be seen as a subset of CNNs, implemented on a field-programmable gate array (FPGA). In this attack, the recovery of the weights is not done through EM analysis, but using a margin-based adversarial learning method. This method

2 Simple EM analysis involves visually interpreting EM traces over time in order to recover the secret.

can be seen as a cryptanalysis against NNs where weights are treated as an attacked secret key. Takatoi et al. [Tak+20] show how to use simple EM analysis to retrieve an activation function from a NN implemented on an Arduino Uno microcontroller. In [Yos+20], correlation power analysis is used to reveal neuron weights from the matrix multiplication implemented with systolic array units on an FPGA. Another relevant attack [Xia+20] uses power SCA together with machine learning classifier to reveal internal network architecture, including its detailed parameters, on an ARM Cortex embedded device. Maji et al. [MBC21] demonstrated a timing attack combined with a simple power analysis of microcontroller-based NN to recover hyperparameters and the inputs of the network.

The only previous attack that targets GPU-based NN [Wei+20a], to the best of our knowledge works in a different setting to ours; a model developer and an adversary share the same GPU when training a network and the adversary aims to break the isolation to learn the trained model. This attack is not applicable to an edge accelerator platform as the training phase is always performed in a controlled environment with more capable resources. This attack also relies on the presence of an adversary sharing GPU resources while our attack does not make such a requirement.

A recent survey of existing SCA methods for architecture extraction of neural networks implementations is presented in [Cha+21] and an overview of hardware attacks against NN is given in [XAQ21].

8.1.2 *Contributions*

In this paper, for the first time, to the best of our knowledge, we investigate using simple EM analysis to break side-channel security of NN (namely Multilayer Perceptron and Convolutional Neural Networks) running on a GPU. We present how to successfully recover the number of layers and neurons as well as the types of activation functions. Most importantly, our results show the importance of side-channel protections for NN accelerators in real-world applications.

We leave recovering neuron weights and CNN hyperparameters using more complex side-channel attacks, like DPA or template attack to be future work.

Our experimental results are obtained on the setup that is as close as possible to a real-world device setup in order to properly assess the applicability and extendability of our methods.

8.1.3 *Organization of the paper*

Section 8.2 presents the employed GPU architecture. Subsequently, our threat model is described in Section 8.3 and the target and NN implementation in Section 8.4. We present our reverse engineering methods and experimental results in Section 8.5. Finally, conclusions and future work are presented in Section 8.6.

8.2 GPU ARCHITECTURE

GPU is a specialized computer hardware designed to accelerate parallel computing for image processing. Deep learning algorithms can benefit from GPU high parallelization to boost their performances, especially when dealing with visual data. A GPU groups several GPU cores into a Streaming Multiprocessor (SM). The specific SM of Maxwell GPU architecture is shown in Figure 58. All GPU cores within a SM can handle floating-point operations in a Single Instruction Multiple Data (SIMD) paradigm. This way, the exact same processing can be applied to a large volume of data to reach a higher throughput than for a CPU.

CUDA is the Software Development Kit (SDK) introduced by NVIDIA that gives direct access to the GPU's instruction set and facilitates general-purpose programming. From the programming perspective, a program that runs on a GPU is divided into parallel threads groups into wraps of 32 threads partitioned into blocks within grids executed on the SM [Nic+08]. When the number of blocks in a SM is less than the number of blocks assigned for the operation, the blocks are queued and scheduled to be executed at a later time. This method allows programs to be scalable for the hardware it is executed on and offers speed up for devices with more blocks per SM. Higher-level programming languages such as python frameworks relies on CUDA to call computation every low level function on GPU. We will see that it is possible to exploit this feature to perform side-channel analysis of the size of data processed by the GPU.

8.3 THREAT MODEL

The main goal of this attack is to reverse engineer the neural network architecture using only side-channel information. In this scenario, we consider an attacker with no insight of the inputs type, source or the implementation of the machine learning algorithm. Currently, to the best of our knowledge, there is no public implementation deploying side-channel countermeasure. We consider a passive and non-invasive attacker who can only acquire side-channel measurement while operating "normally" the target device and cannot control the flow of operation.

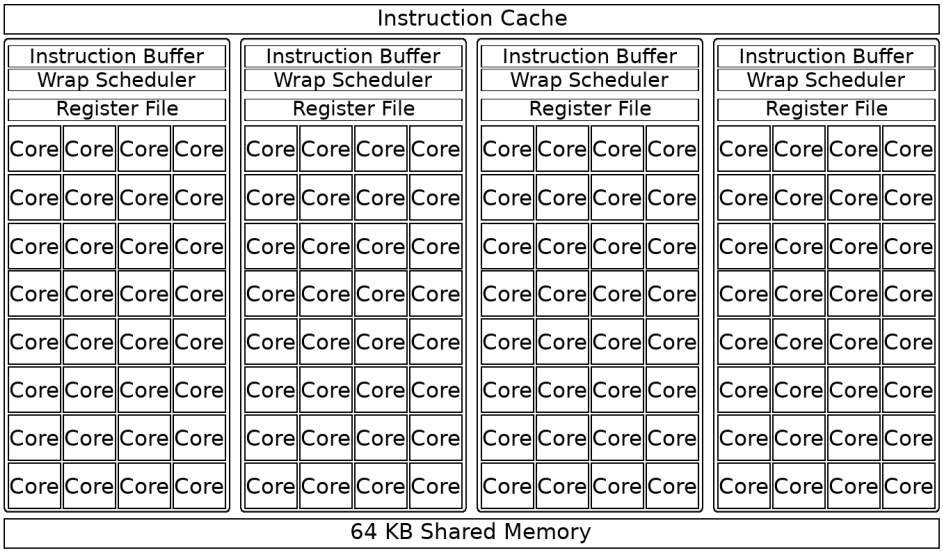


Figure 58: Maxwell Streaming Multiprocessor Architecture

A suitable use case for this attack is considering an attacker who acquired a legal copy of the network in a black-box setting and aims to recover its internal details for IP theft. The attacker controls the inputs and performs side-channel measurement during the inference phase of the neural network. The goal is to reverse engineer the following information about the neural network architecture: number of layers, number of outputs, and activation functions in the network.

If successful, this attack can have severe monetary repercussions for companies investing significant resources to develop customized machine-learning models to create highly valuable IPs [Pap+18]. A successful attacker that is able to steal such models can offer similar services at much lower cost than the investing companies.

8.4 THE TARGET AND NETWORK IMPLEMENTATION

The target is an Nvidia Jetson Nano [Nan], a module computer embedding a Tegra X1 SoC [Teg] combining an ARM Cortex-A57 CPU and a 128-core GPU with Maxwell architecture suitable for AI applications such as image classification, object detection, segmentation, and speech processing. Specifically, modules similar to this one are used for real application in automotive visual computing, namely for Nvidia drive CX and PX computer platforms. The Jetson Nano Tegra X1 SoC contains a GPU with one Maxwell Streaming Multiprocessor (SMM) (see Figure 58). The SMM is partitioned into four distinct 32-CUDA core processing

blocks (128 CUDA cores total), each with its own dedicated resources for scheduling and instruction buffering.

The neural network is a convolutional neural network (CNN) implemented using the PyTorch python framework [Pas+19]. The dataset used to train the network is the CIFAR10 dataset [Kri09], a 60 000 32x32 colored image dataset representing 10 classes. The reference CNN architecture consists of two convolutional layers (of 6 and 16 filters of size 5) with max-pooling and three linear Fully Connected (FC) layers, all regulated with the ReLU activation function, and the final FC output layer. The input is a three-channel image of size $32x32$, and the output is a 10-sized vector of each class of the classification problem. This architecture, together with the corresponding SPA, is presented in Figure 59.

To better measure the execution of the neural network, we use a power trigger. The Jetson Nano handles General Purpose Input/Outputs (GPIOs). We use one GPIO pin to implement a trigger around the forward loop of the neural network to be sure only to measure while the GPU is active. It is to be noticed that, the neural network is already trained and the gradient operation is disabled to prevent the backward loop from happening.

To record the EM traces we removed the heatsink of the target and placed a Riscure Low Sensitivity (LS) EM probe [3] above the main chip package. The best position of the probe is empirically chosen to maximize the leaking signal. We manually searched the position with a grid scanning above the chip for multiple locations and chose the most promising position based on visual inspection of the traces. This best location is presented in Figure 60.

The oscilloscope in our experiment is the Teledyne Lecroy WaveRunner 8404M. We used it in two configurations, one for characterization with 5×10^9 samples/s and at most 32×10^6 samples and the second one for simple EM with 10^9 samples/s and at most 10^7. We used greater sampling rate in the first configuration because we performed frequency analysis, and we needed to be able to record a signal up to the GPU maximum clock frequency, namely 900MHz, in good quality; for simple EM we do not need that high accuracy. The oscilloscope has TCP/IP support for both controlling and downloading measurements, which helps to automatize the entire process.

We acquired and analyzed using Riscure's Inspector software package[4].

The goal of the neural network used in this paper is not meant for high efficiency or presenting a challenging classification task, but rather to show the methods and principles side-channel analysis can bring in to extract information from a neural network closed implementation.

3 https://www.riscure.com/uploads/2017/07/inspector_brochure.pdf
4 https://www.riscure.com/security-tools/inspector-sca

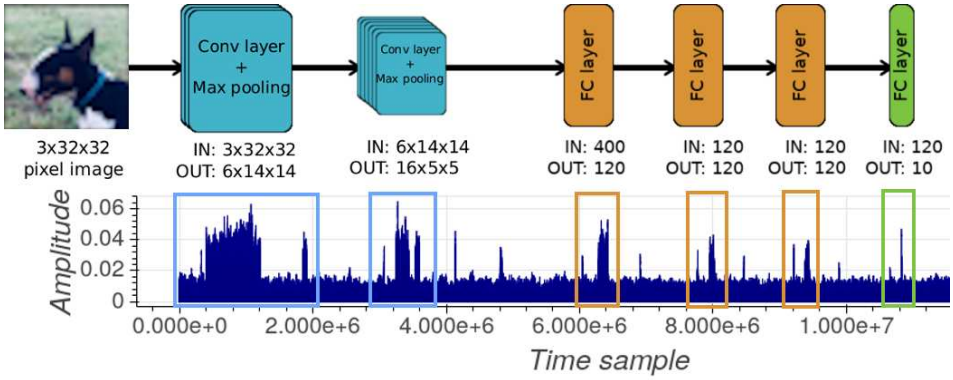


Figure 59: SPA of CIFAR10 Convolutional Neural Network

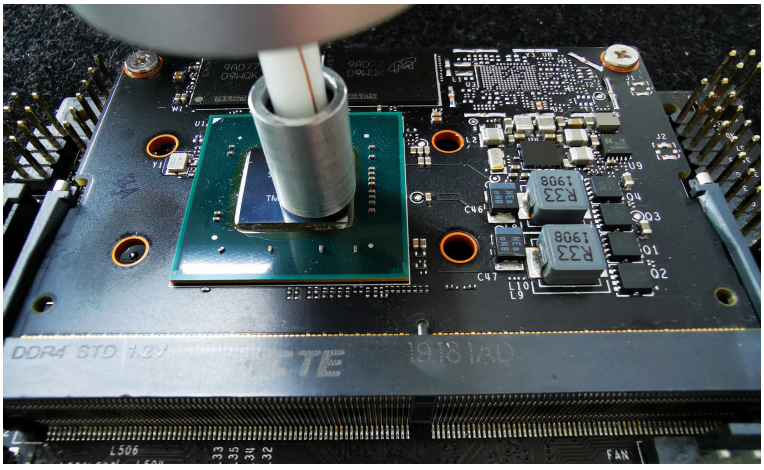


Figure 60: Experimental setup: the EM probe location

8.5 REVERSE ENGINEERING

8.5.1 *Characterization*

In Figure 59, the architecture of the neural network is showed next to an EM trace measured during its execution on the target. From the EM trace, we can observe that every different step of the forward loop of the NN is distinguishable. The two convolutional blocks are identifiable by a first activity corresponding to the convolutional operation followed by a smaller activity corresponding to the pooling operation. The layers of the MLP, namely, the FC layers, are also detectable by single peaks.

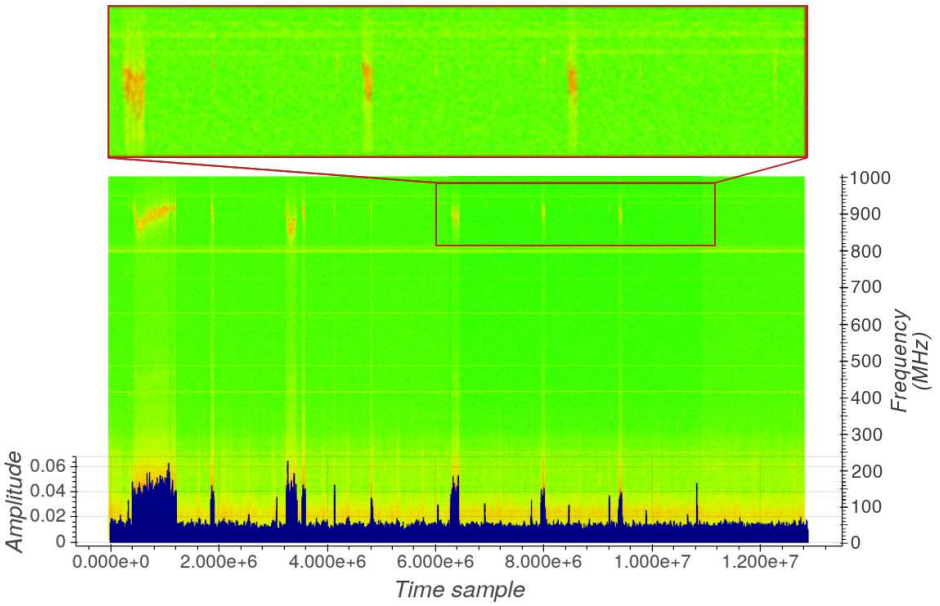


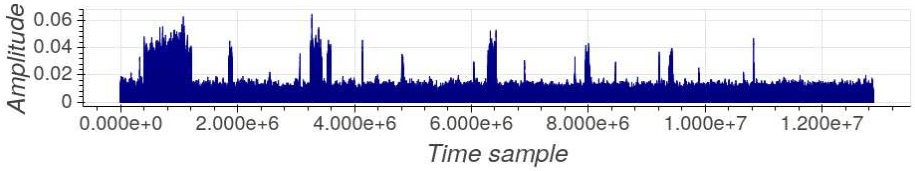
Figure 61: A single EM trace (in the blue color at the bottom) and the corresponding spectrogram (middle) with the MLP activation zoomed-in (top)

It is possible to verify whether the leakage is effectively coming from the GPU activity by observing the leakage in the frequency domain. Because the GPU maximum clock frequency is 900MHz, the computation made on the device will emit leakage in the same range of frequencies. In Figure 61, we represented the absolute value of the leakage together with the spectrogram plot of the leakage. We can see that the detected leakage correspond to the frequency range of the GPU. It would be possible to continue the same analysis using this frequency signal, but in this paper, we will only focus on EM analysis in the time domain.

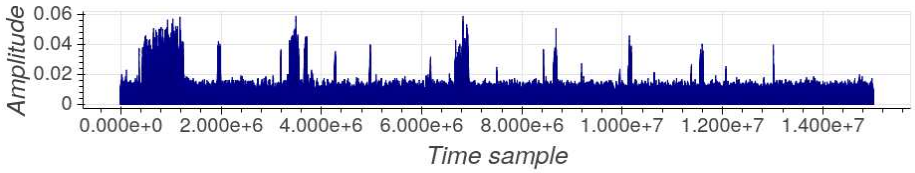
In the following analyses, we either use raw traces or apply an averaged window resampling method on the absolute value of the signal. The averaged window resampling reduces the number of features of the trace by averaging samples in a fixed-size window shifted without overlapping across all samples of the trace. This processing makes alignments on specific patterns faster and easier.

8.5.2 *Reverse Engineering the Number of Layers*

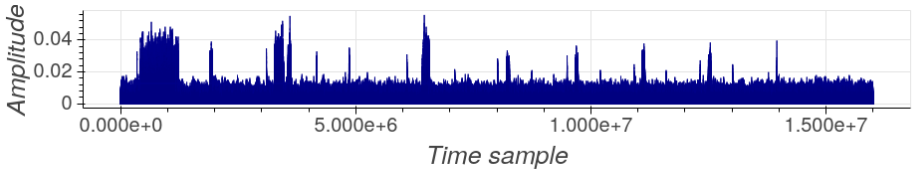
In this section we investigate how to recover the number of hidden layers in the MLP from the SCA during the inference phase.



(a) 4 layers



(b) 5 layers



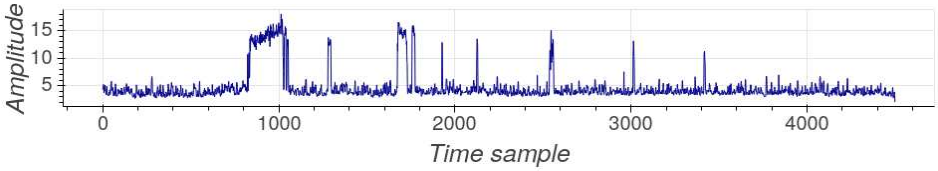
(c) 6 layers

Figure 62: Differences in the number of fully connected layers

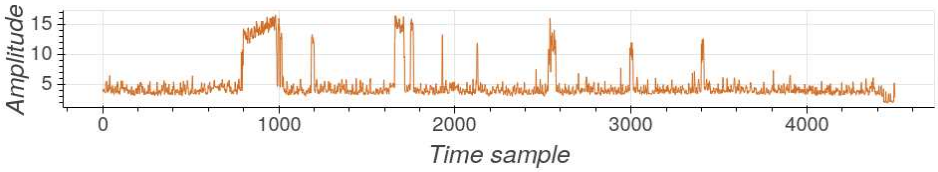
Since the dataflow of a NN is such that layers are processed sequentially, the analysis of different number of fully connected layer is trivial from the EM traces. We measured three different implementations of a neural network, with a different number of fully connected layers and observe their leakage. From the reference neural network model, we change the number of fully connected layers from 4 to 6. The number of neurons in each of the additional layers is the same as the second fully connected layer from the reference model (i.e., 120 neurons). The resulting EM measurements are represented in Figure 62. From the three plots, the two first convolutional blocks and the fully connected layers are clearly identifiable. While the plots are aligned according to the first convolutional layer, the timing of the execution is not consistent. Many process interruptions occur during the computation, leading to misalignments in the traces. However, the additional layers do appear in the EM measurement and are easily identifiable.

8.5.3 *Reverse Engineering the Number of Neurons*

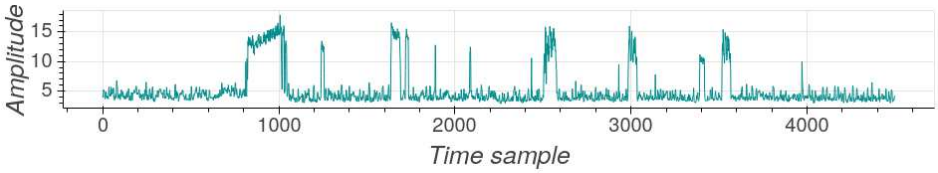
Now we investigate how to recover the number of neurons in a hidden MLP layer.



(a) 60 neurons



(b) 120 neurons



(c) 255 neurons

Figure 63: Difference in number of perceptrons inside fully connected layers

In a GPU implementation, every neuron operation is processed in parallel. However, the parallelization degree depends on the size of the inputs and number of neurons, as there is a limit on the number of floating-point operation that can be computed in parallel. For example, given N GPU threads, each capable of computing one floating-point operation per clock cycle, the GPU scheduler can compute N operations per clock cycle. If $n_{inputs} \times n_{neurons} > N$ then the number of neurons will partially leak, and if $n_{inputs} \geq N$ then the number of neurons will entirely leak as every neuron computation would require more than one cycle.

From the execution of the linear operation in the fully connected layers, it is possible to recover the number of perceptrons per layer using timing side-channel. In Figure 63 different models are analyzed. Here, we control the number of neurons within the hidden layers. We can see that when the number of neurons in the layers increases, the execution time of each layer also increases.

Recovering the exact number of neurons in a layer would require to be capable to distinguish a single neuron difference. In Figure 64, the timing of the first fully connected layer activity with an increasing number of perceptrons from 30 to 100 is represented. For every number of perceptron, we averaged fifty measurements and align the traces according to the desired pattern to measure the execution time

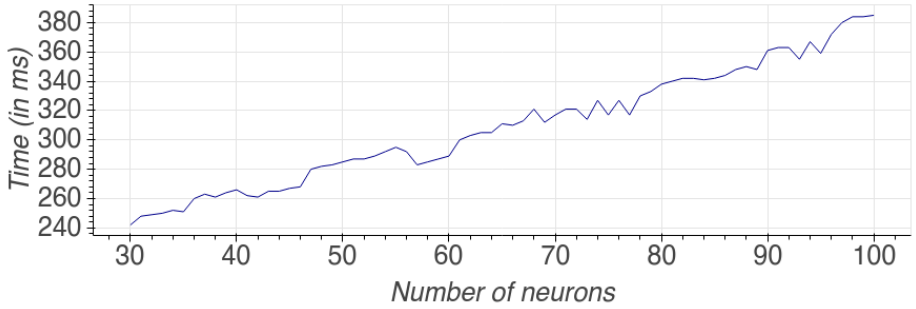


Figure 64: Differences in the number of perceptrons (from 30 to 100 units)

of the specific layer. While the relation shows a linear behavior, the measurements noise and re-alignment, still make it difficult to distinguish a single perceptron change. However, approximate recovery is possible with a relatively low error margin.

8.5.4 *Reverse Engineering the Type of Activation Function*

Nonlinear functions are essential to approximate a linearly non-separable problem. The use of these functions also helps to reduce the number of network nodes. With the knowledge of the type of nonlinear function used in a layer, an attacker can deduce the behavior of the entire neural network using the input values.

We analyze the side-channel leakage from different commonly used activation functions, namely ReLU, Sigmoid, Tanh, and Softmax [NH10; HN04]. The activation function applied to the first convolutional layer of the network is changed in different measurements. We measured the EM leakage of the execution of the layer computation and the activation function for random input and is represented in Figure 65. We identify the execution of the convolutional layer from 0 to 520 time samples, and it is identical for all sub-figures. The execution of the activation function presents differences among all different activation functions. We can notice for example that the execution of the ReLU activation function is the shortest and that the Softmax function is the longest by far. The timing differences between ReLU, Tanh, and Sigmoid activation functions are smaller.

The computation time of the activation function does depend on the size of the input. Therefore, to identify the type of activation function, one should first recover the number of inputs. We measured fifty executions of the activation function after the first convolutional layer. The input of this activation function is of the size of the output of the convolutional layer before the pooling layer and is of size $6 \times 28 \times 28 = 4704$. All measurements are done on random data, and we

Table 24: Statistical analysis on computation time (in μs).

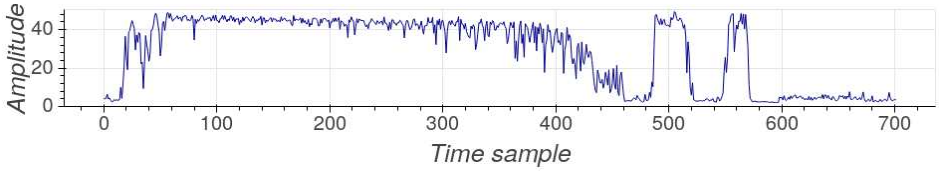
Activation function	Mean	Maximum	Minimum
ReLU	33.5	34	33
Tanh	36.0	37	34
Sigmoid	43.3	46	41
Softmax	124.5	127	123

draw a statistical analysis of the timing pattern for all types of activation functions considered in Table 24. It can be observed that each activation function stands out, and thus it is possible to recover trivially the type of activation function from a neural network implementation.

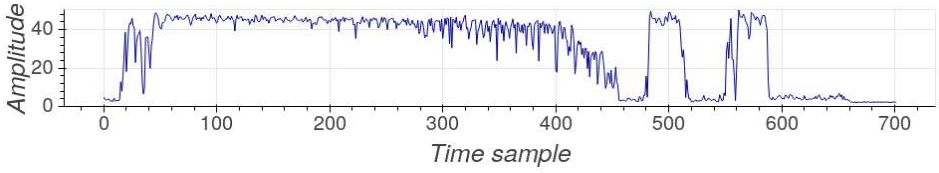
8.6 CONCLUSIONS AND FUTURE WORK

Side-channel analysis have already been proven capable to reverse engineering a neural network implemented on a microcontroller architecture. While microcontrollers can be the hardware of choice for some small edge computing applications, GPU stays the most popular platform for deep learning. In this paper, we show the possibility to recover key parameters of a GPU implementation of a neural network. We can recover the number of layers and number of neurons per layer of a multilayer perceptron with simple power analysis. We can also identify different types of activation functions with single power analysis for a given number of inputs. We can conclude that we have managed to recover all the secret information that can be achieved using only simple EM analysis.

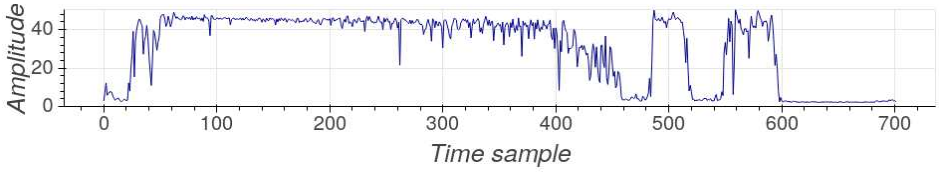
For the reverse engineering of a complete neural network, the weights of all layers for both MLP and CNN networks and network hyperparameters for CNNs should be recovered too. We consider these tasks to be future work, but we envision that the weights can be recovered using correlation or differential power analysis on float vector multiplication similarly to [Bat+19]. However, the main challenges, besides noise and misalignment, would be the lack of information on how the multiplication is performed and the parallel aspect of GPU computation (i.e., multiple intermediate values might be computed at the same time). Recovery of CNN hyperparameters would be probably also a hard task, and we suspect that it would require a template attack.



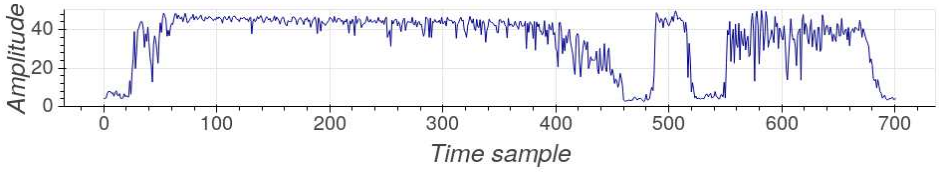
(a) ReLu



(b) Tanh



(c) Sigmoid



(d) Softmax

Figure 65: Differences in the type of activation function applied on layer output

Part V

DISCUSSION

9 DISCUSSION AND FUTURE WORK

This chapter concludes the thesis and provides a discussion for future work.

CONTENT OF THIS CHAPTER

9.1 SUMMARY OF CONTRIBUTIONS

In this section, we list a summary of the results presented in this thesis.

- In Chapter 3, we give solutions to the first research question, *What deep learning methods can be used to automate side-channel analysis of cryptographic implementations?* We presented a study on MLP to attack implementations of AES with side-channel analysis from different datasets in Chapter 3. The preprocessing of the input data significantly impacts the performance of the MLP and does decrease the need for a large network to achieve good results. Conversely, a larger network can better generalize on data with more features. The analysis focuses on the impact of the number of layers and the number of neurons per layer on the network's capacity to fit the leakage in the data. While different configurations can achieve good results, we show that the best configuration is often a trade-off between the two parameters, and it is not always the largest network that achieves the best result. For the AES_RD dataset that implements AES with a random delay countermeasure, we show that an MLP with a single layer of 200 perceptrons and a six-layer network with ten perceptrons per layer can achieve the best performances. For the ASCAD dataset, we also show that there exists a trade-off between the number of layers and the number of perceptrons

per layer that makes an efficient model with even fewer parameters than the best model introduced in previous work introducing the dataset. We demonstrate that MLP can be as efficient as more performant networks like CNN for this task while being simpler and faster to train.

- In Chapter 4, we answer the second researcher question *Can deep learning be used to enhance the performance of SCA on lightweight cryptographic implementations of ASCON?* We present an application of Deep learning Side-Channel Analysis (DLSCA) on the lightweight authenticated encryption algorithm Ascon. We offer a comparison between different leakage models of the S-box function of Ascon that can apply for the fixed-key attack as CPA and for the random key attack as DLSCA enables. We show that DLSCA can successfully recover a partial secret key after 20 attack traces, which is fewer than the presented CPA attack that requires 200 traces. The results are presented on a reference and a protected implementation, and we show that a CNN can be used to attack both implementations with the same number of traces in both cases. At the same time, CPA is only successful on the reference implementation. We also show a multi-target model that can be used to attack all the partial keys of the implementations at once and show successful results on the reference implementation using only a thousand traces.

- In Chapter 5 and 6, we give insight for the question, *How far can deep learning-based side-channel analysis improve the performance of side-channel attacks?* We presented a one-trace attack on EdDSA using the curve Curve25519 as implemented in WolfSSL in Chapter 5. This attack targets the scalar multiplication operation of the ephemeral key generation. With a single trace attack, it is possible to compute the secret scalar from the public information. We collected a dataset of multiple scalar multiplications as implemented in the WolfSSL library. We separated each trace into the single table look-up operation from the nibbles of the scalar. This dataset is used to obtain secret information about the differences in the power consumption of the LUT operation. We trained and compared several machine learning techniques for this attack, namely TA, RF, SVM, and CNN, that have also been shown in the literature to be well suited for attacking implementations of the AES cipher. Each method is trained with and without feature reduction using PCA, and while all methods show accuracy above 95% when targeting a single nibble, only the CNN achieves perfect accuracy and can recover the secret scalar with a single trace.

- We extended the results of the previous chapter in Chapter 6 to an implementation of EdDSA on Curve25519 with countermeasure from the μNaCL library. This implementation uses Montgomery ladder scalar multiplication with an arithmetic-based conditional swap protected with projective coordinate re-randomization and scalar randomization. In this case, we show that the CNN is also a powerful tool to attack the protected implementation with 98% cumulative accuracy on raw traces. In contrast, other machine learning techniques show poor results as the second-best method, random forest, only reaches 8%. We observed that the prospected dimensionality reduction methods have a negative impact on the performance of CNN.

- A TEMPEST attack on mobile devices is presented in Chapter 7. This chapter answers the third research question, *How to evaluate the security devices against TEMPEST attacks in regard to deep-learning methods?* We introduce a new testbed to evaluate the performance of EM side-channel attack to recover the displayed content on the digital screen of a smartphone, using two different setups: a single letter and a PIN code embedded in a text message. The measurement setup is constituted of a near-field probe and a software-defined radio that transmits the collected signal to a software capable of reconstructing a gray-scaled image from the intensity of the EM signal. The quality of the recovered image depends on the distance between the probe and the device and the size of the displayed text. We showed that the attack can be performed even at a few centimeters distance using a CNN model. Using a CNN classifier to enhance digit recognition, we demonstrate an average digit recognition accuracy of 89.8% and a 5-digit PIN code recovery accuracy of 89.5%, far better than human performance. The collected data from six devices was used to demonstrate the reproducibility across different devices of the same model (with three among those) and the extensibility to other different models (with four different devices).

- We presented a side-channel analysis of neural networks implemented on GPU in Chapter 8. This work is the first step in the investigation of the security of neural networks implemented on GPUs, and gives an answer to the research question *Are deep learning implementations secured against side-channel analysis?* We showed that the power consumption of a GPU can leak information on the neural network inference that can be used to reverse-engineer the model's hyperparameters. The analysis of the EM measurements collected above the GPU package during neural network inference shows a distinct frequency pattern revealing the activation of the neural network layers. We demonstrated that this leakage reveals the type and number

of layers from the activation pattern and the number of neurons per layer from the time of the activation of a layer.

In the light of the individual projects presented in the thesis, we can generalize our contributions to design, engineer, and understand deep learning methods for improving the security of sensitive applications.

First, we covered side-channel analysis of cryptography for several symmetric and public-key implementations. Because understanding a security vulnerability is the first step to designing more secure applications, the effort of this contribution is put on the development of attacks around existing vulnerabilities of cryptographic implementations. We use deep learning to achieve better supervised attacks than the state-of-the-art methods and bring security analysis closer to a real-world evaluation of what is possible by an advanced attacker. The work presented in Chapter 4 and Chapter 5 are good examples for enhanced side-channel attacks by deep learning and represents a guideline of evaluation helpful for driving the development of what is considered the criteria for secure cryptographic implementation.

Secondly, we marked the systematization of deep learning training to design more performant side-channel analysis methods. Finding in the least number of observations the secret information present in the physical leakage of the device is important in SCA, as the feasibility of an attack is directly driven by the time an attacker has for getting to the secret. The performance of a predictive deep-learning model is dependent, not only on the dataset used for training, but also on the effort put into engineering the model and hyperparameters. In the thesis, we focus on studying supervised methods like MLP and CNN and their integration in DLSCA methods, as presented notably in Chapter 3 and Chapter 6. We provide keys for understanding the conditions of application of a deep learning model for side-channel analysis, as well as systematic approaches for choosing hyperparameters of a model and for training a successful model given a side-channel dataset.

Finally, since the security of a cryptosystem is not limited to the leakage of its cryptographic implementations, we put forward the analysis of the security of two applications that do not necessarily use cryptography, while possibly subject to side-channel attacks. With the TEMPEST attack presented in Chapter 7 and the reverse engineering of a neural network in Chapter 8, we step outside of usual cryptographic applications and emphasize the link of security in the presence of deep learning. By examining the side-channel leakage of these applications, we highlight unforeseen vulnerabilities. Our work aims to help to build more efficient countermeasures against TEMPEST attacks for designing future secure displays for mobile devices. We have shown the first side-channel attack to recover partial

neural network assets from a GPU device, questioning the security mechanisms set to protect AI models in distributed devices.

We can conclude that deep learning is a valuable asset for side-channel analysis and the security of secure applications on physical devices. While deep learning-based methods will not replace the statistical methods resulting from all the research in SCA, the combination of SCA techniques with deep learning may be a window to lead to the best results.

9.2 FUTURE WORK

In Chapter 3, we presented a detailed analysis of the training of MLP for side-channel analysis of AES. We explored the impact of the number of layers, the number of perceptrons per layer, and the preprocessing of the input data. This work can be extended by exploring more complex hyperparameter configurations where the number of perceptrons per layer is different for each layer. While grid-like searches are exhaustive over the search space, such a method is limited by the number of models it can train in a reasonable time. Optimizing a neural network can be viewed as a Bayesian optimization problem where the function to optimize is the neural network architecture and hyperparameters that give the highest accuracy for the attack. Methods to solve such problems are known, such as Tree Parzen Estimator or Hyperband. They can cover a broader range of hyperparameter configurations with less training time than a grid or random searches. However, the resulting model can be complex, and there is no guarantee that the configuration with the best performance or the least number of parameters that achieve honorable results will be covered. To help reduce the effort on neural network training for DLSCA, we should work on a hyperparameter optimization method that can provably output the best candidates with the least computational cost.

Chapter 4 can be extended further evaluating DLSCA against implementations with countermeasures. The dataset we collected consists of traces from a single device. Collecting traces from copies of the same device can extend the dataset and improve the generalization of the leakage. Moreover, the additional data could resolve the multi-target model behavior for the incorrect partial keys when applied to the attack of the protected implementation. It would also be interesting to consider modification of that model with individual corrections to the branches that show deviant behavior, with the application of adaptive dropout and custom loss functions to improve the performance of individual models. One could also consider if the feature extraction block should be transferred from the single target model or retrained on the multi-target model to understand if the information of

one partial key from the single target model is efficiently reused in the multi-target model for all partial keys, thereby improving the whole performance.

As discussed in Chapter 5 and Chapter 6, our work could be extended by investigating further the use of machine learning metrics in the evaluation of SCA on public-key cryptography. Machine learning metrics such as accuracy and error, while minimizing the cross entropy, which is linked to maximizing the mutual information, can be unsuited to solve SCA problems, as observed for SCA on symmetric cryptography. The major problem of such a metric is that it only tracks the correct key candidate but does not consider the performance of related key candidates. We could imagine that the error made by a model that ranks the correct key candidate in the second position differs from the one that ranks it in the last. Hence, the error produced by the model should be weighted by the rank of the correct key candidate. Similarly, if the best key candidate is one bit off from the correct key, we could imagine that the error should also be less than if more bits are off. Investigations on learning metrics that integrate relations with the key candidates and their distance to the correct key could be a promising direction for future research.

We have shown that CNN can easily break a masked implementation of EdDSA. It would be interesting to investigate the use of different countermeasures and whether the CNN can be used in the presence of multiple countermeasures. Using multiple countermeasures for public-key cryptography is a common practice to secure an implementation and would thus be closer to a real-world scenario. Another question to answer would be the type of deep learning model that can be used to defeat different countermeasures. Would the same model be efficient in defeating all countermeasures, or should the model consider the order of the countermeasure to be efficient?

Another interesting direction is exploring different dimensionality reduction methods, as the methods we investigated have shown to have a negative impact on the performance of the CNN. The dimensionality reduction method used in this work is the PCA, which linearly transforms the data to represent the highest variance as the first component. If this method is destructive to the performance of CNN, we could consider different projection methods, such as wavelet transform or frequency analysis, or use non-linear dimensionality reduction methods like t-SNE.

To improve our contribution in Chapter 7, a wider range of devices should be considered, including tablets and mobile smart displays. The technology used by displays can be different and impact the communication between the display and the graphical processing unit. The use of HDCP (High-bandwidth Digital Content Protection) or other encryption protocols for the communication between the

display and the processor is an effective countermeasure to prevent side-channel attacks and other man-in-the-middle attacks. We did not consider this countermeasure as it is not implemented in the devices we could access. Bypassing the encryption protocol is an entirely different problem that is the subject of research. To further improve the distance of the attack, the use of a longer-range antenna should be preferred, and there might be a need to develop a custom design for the antenna to improve the signal quality of given frequencies evaluated for a specific display type.

Chapter 8 can be extended by reverse-engineering the weights of the neural network with CPA attacks as used in [Bat+19]. The difficulty of such an attack resides in the parallel execution of nodes in the neural network on GPU. This parallelism creates noise that makes the recovery of a single weight harder. To address this problem, an attacker can control the network's input to force the activation of a single node and proceed to a CPA attack to recover the weights of the nodes one by one. After recovering the weights of the first layer, the attacker can proceed to the next layer by finding the inputs that induce zero values for all intermediate inputs of the second layer, and so on. However, this approach can be heavy for complex models. To secure the neural network against this kind of attack, research on countermeasures against SCA on neural networks should be considered. Well-known countermeasures used for cryptographic implementations, like masking or shuffling, can be applied. However, these countermeasures should have limited performance overhead to represent a promising candidate for future research.

9.3 LIMITATIONS

The various applications we presented in this thesis show the application of DLSCA on symmetric and asymmetric cryptography and other perspectives of deep learning in side-channel analysis. The presented DLSCA attacks need to have an advanced knowledge of the target implementation and deep learning techniques. Finding the best configuration for the neural network is a challenging task, as well as the preparation for successful data collection. These problems are still open and need to be addressed in future research.

The use of Bayesian optimization for hyperparameter tuning to reduce the need for manual exploration, in most cases a task requiring high knowledge about neural network training, is a promising direction as presented in Chapter 4. As presented in the same chapter, the multi-target model enables the attack of all partial keys of a block-cipher-like implementation, which can improve the generalization during the training phase. Optimizing this technique with a good dataset and a well-designed neural network can improve the performance of SCA.

While DLSCA attacks can be very efficient, training a performant neural network can require a lot of training traces to fit a leakage, and the knowledge can be unusable for other implementations. This problem can be addressed using transfer learning techniques, but the exact performance of these techniques on DLSCA is still an open question.

BIBLIOGRAPHY

[Agr+02] Dakshi Agrawal, Bruce Archambeault, Josyula Rao, and Pankaj Rohatgi. "The EM Side-Channel(s)." In: *Annual Conference on Cryptographic Hardware and Embedded Systems*. 2002, pp. 29–45.

[ABR64] M. A. Aizerman, E. M. Braverman, and L. I. Rozonoer. "Theoretical foundation of potential functions method in pattern recognition." In: *Automation and Remote Control*. 1964, pp. 821–837.

[AH98] Martin Anthony and Sean B. Holden. "Cross-Validation for Binary Classification by Real-Valued Functions: Theoretical Analysis." In: *Proceedings of the Eleventh Annual Conference on Computational Learning Theory, COLT 1998, Madison, Wisconsin, USA, July 24-26, 1998*. 1998, pp. 218–229.

[AA04] Dmitri Asonov and Rakesh Agrawal. "Keyboard acoustic emanations." In: *IEEE Symposium on Security and Privacy*. 2004, pp. 3–11.

[Aum17] Jean-Philippe Aumasson. *Serious cryptography: a practical introduction to modern encryption*. 2017.

[Avi+10] Adam Aviv, Katherine Gibson, Evan Mossop, Matt Blaze, and Jonathan Smith. "Smudge attacks on smartphone touch screens." In: *USENIX Conference on Offensive Technologies* (2010), pp. 1–7.

[Avi+12] Adam Aviv, Benjamin Sapp, Matt Blaze, and Jonathan Smith. "Practicality of accelerometer side channels on smartphones." In: *Annual Computer Security Applications Conference*. 2012, pp. 41–50.

[Bac+09] Michael Backes, Tongbo Chen, Markus Duermuth, Hendrik Lensch, and Martin Welk. "Tempest in a teapot: Compromising reflections revisited." In: *IEEE Symposium on Security and Privacy*. 2009, pp. 315–327.

[Bac+10] Michael Backes, Markus Dürmuth, Sebastian Gerling, Manfred Pinkal, and Caroline Sporleder. "Acoustic Side-Channel Attacks on Printers." In: *USENIX Security Symposium*. 2010, pp. 307–322.

[BDU08] Michael Backes, Markus Dürmuth, and Dominique Unruh. "Compromising reflections-or-how to read LCD monitors around the corner." In: *IEEE Symposium on Security and Privacy*. 2008, pp. 158–169.

[Bal+12] Josep Balasch, Benedikt Gierlichs, Roel Verdult, Lejla Batina, and Ingrid Verbauwhede. "Power analysis of Atmel CryptoMemory - recovering keys from secure EEPROMs." In: *The Cryptographers' Track at the RSA Conference*. Springer. 2012, pp. 9–34.

[Bat+19] Lejla Batina, Shivam Bhasin, Dirmanto Jap, and Stjepan Picek. "CSI NN: Reverse Engineering of Neural Network Architectures Through Electromagnetic Side Channel." In: *28th USENIX Security Symposium (USENIX Security 19)*. 2019, pp. 515–532.

[Bat+14] Lejla Batina, Łukasz Chmielewski, Louiza Papachristodoulou, Peter Schwabe, and Michael Tunstall. "Online template attacks." In: *Progress in Cryptology - INDOCRYPT 2014 - 15th International Conference on Cryptology in India, New Delhi, India, December 14-17, 2014, Proceedings*. Vol. 8885. 2014, pp. 21–36.

[Bat+17] Lejla Batina, Łukasz Chmielewski, Louiza Papachristodoulou, Peter Schwabe, and Michael Tunstall. "Online template attacks." In: *Journal of Cryptographic Engineering* (Aug. 2017).

[Bat+11] Lejla Batina, Benedikt Gierlichs, Emmanuel Prouff, Matthieu Rivain, François-Xavier Standaert, and Nicolas Veyrat-Charvillon. "Mutual Information Analysis: a Comprehensive Study." In: *J. Cryptol.* 24.2 (2011), pp. 269–291.

[BS18] Belhassen Bayar and Matthew Stamm. "Constrained convolutional neural networks: A new approach towards general purpose image manipulation detection." In: *IEEE Transactions on Information Forensics and Security* 13.11 (2018), pp. 2691–2706.

[Bel+16] Pierre Belgarric, Pierre-Alain Fouque, Gilles Macario-Rat, and Mehdi Tibouchi. "Side-channel analysis of Weierstrass and Koblitz curve ECDSA on Android smartphones." In: *Cryptographers' Track at the RSA Conference*. Springer. 2016, pp. 236–252.

[BWY06] Yigael Berger, Avishai Wool, and Arie Yeredor. "Dictionary attacks using keyboard acoustic emanations." In: *ACM SIGSAC Conference on Computer and Communications Security*. 2006, pp. 245–254.

[Ber+11] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. "Algorithms for Hyper-Parameter Optimization." In: *Advances in Neural Information Processing Systems*. Vol. 24. 2011.

[BB12] James Bergstra and Yoshua Bengio. "Random Search for Hyper-Parameter Optimization." In: *J. Mach. Learn. Res.* 13 (2012), pp. 281–305.

[Ber16] Daniel J Bernstein. "Curve25519: new Diffie-Hellman speed records (2006)." In: *URL: http://cr.yp.to/papers.html#curve25519. Citations in this document* 1.5 (2016).

[Ber+12] Daniel J Bernstein, Niels Duif, Tanja Lange, Peter Schwabe, and Bo-Yin Yang. "High-speed high-security signatures." In: *Journal of Cryptographic Engineering* 2.2 (2012), pp. 77–89.

[BS02] Hans-Georg Beyer and Hans-Paul Schwefel. "Evolution strategies - A comprehensive introduction." In: *Nat. Comput.* 1.1 (2002), pp. 3–52.

[Bha+19] Shivam Bhasin, Anupam Chattopadhyay, Annelie Heuser, Dirmanto Jap, Stjepan Picek, and Ritu Ranjan Shrivastwa. *Mind the Portability: A Warriors Guide through Realistic Profiled Side-channel Analysis*. Cryptology ePrint Archive, Report 2019/661. 2019.

[BSS99] Ian Blake, Gadiel Seroussi, and Nigel Smart. *Elliptic curves in cryptography*. Vol. 265. 1999.

[Boh+03] Lilian Bohy, Michael Neve, David Samyde, and Jean-Jacques Quisquater. "Principal and Independent Component Analysis for Crypto-systems with Hardware Unmasked Units." In: *Proceedings of e-Smart 2003*. Cannes, France. Jan. 2003.

[Bos+16] Joppe Bos, Charles Hubain, Wil Michiels, and Philippe Teuwen. "Differential computation analysis: Hiding your white-box designs is not enough." In: *International Conference on Cryptographic Hardware and Embedded Systems*. Springer. 2016, pp. 215–236.

[BGV92] Bernhard E. Boser, Isabelle Guyon, and Vladimir Vapnik. "A Training Algorithm for Optimal Margin Classifiers." In: *Proceedings of the Fifth Annual ACM Conference on Computational Learning Theory, COLT 1992, Pittsburgh, PA, USA, July 27-29, 1992*. 1992, pp. 144–152.

[Bos] Sarah Boslaugh. *Snellen chart*. https://www.britannica.com/science/Snellen-chart. Accessed: 18-11-2020.

[Bre01] Leo Breiman. “Random Forests.” In: *Machine Learning* 45.1 (2001), pp. 5–32.

[BCO04] Eric Brier, Christophe Clavier, and Francis Olivier. “Correlation Power Analysis with a Leakage Model.” In: *Cryptographic Hardware and Embedded Systems - CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11-13, 2004. Proceedings*. Vol. 3156. 2004, pp. 16–29.

[CDP17] Eleonora Cagli, Cécile Dumas, and Emmanuel Prouff. “Convolutional Neural Networks with Data Augmentation Against Jitter-Based Countermeasures - Profiling Attacks Without Preprocessing.” In: *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*. 2017, pp. 45–68.

[CC11] Liang Cai and Hao Chen. “TouchLogger: Inferring Keystrokes on Touch Screen from Smartphone Motion.” In: *USENIX Summit on Hot Topics in Security* (2011), pp. 9–15.

[Car+19] Mathieu Carbone, Vincent Conin, Marie-Angela Cornélie, François Dassance, Guillaume Dufresne, Cécile Dumas, Emmanuel Prouff, and Alexandre Venelli. “Deep Learning to Evaluate Secure RSA Implementations.” In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2019.2 (Feb. 2019), pp. 132–161.

[Cha+21] Hervé Chabanne, Jean-Luc Danger, Linda Guiga, and Ulrich Kühne. “Side channel attacks for architecture extraction of neural networks.” In: *CAAI Transactions on Intelligence Technology* 6.1 (2021), pp. 3–16.

[Cha+99] Suresh Chari, Charanjit Jutla, Josyula Rao, and Pankaj Rohatgi. “Towards sound approaches to counteract power-analysis attacks.” In: *Annual International Cryptology Conference*. 1999, pp. 398–412.

[CRR02] Suresh Chari, Josyula R Rao, and Pankaj Rohatgi. “Template attacks.” In: *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer. 2002, pp. 13–28.

[CG16] Tianqi Chen and Carlos Guestrin. “XGBoost: A Scalable Tree Boosting System.” In: *CoRR* abs/1603.02754 (2016). arXiv: 1603.02754.

[Chm20] Łukasz Chmielewski. *REASSURE (H2020 731591) ECC Dataset*. Version V1.0. Jan. 2020.

[CW] Lukasz Chmielewski and Léo Weissbart. "On Reverse Engineering Neural Network Implementation on GPU." In: *Applied Cryptography and Network Security Workshops - ACNS 2021, Kamakura, Japan, June 21-24, 2021, Proceedings*. Vol. 12809, pp. 96–113.

[Cho+15] François Chollet et al. *Keras*. https://github.com/fchollet/keras. 2015.

[CK13] Omar Choudary and Markus G. Kuhn. "Efficient Template Attacks." In: *Smart Card Research and Advanced Applications - 12th International Conference, CARDIS 2013, Berlin, Germany, November 27-29, 2013. Revised Selected Papers*. 2013, pp. 253–270.

[CJ19] *Selected Areas in Cryptography - SAC 2018 - 25th International Conference, Calgary, AB, Canada, August 15-17, 2018, Revised Selected Papers*. Vol. 11349. 2019.

[CCD00] Christophe Clavier, Jean-Sébastien Coron, and Nora Dabbous. "Differential Power Analysis in the Presence of Hardware Countermeasures." In: *Cryptographic Hardware and Embedded Systems - CHES 2000, Second International Workshop, Worcester, MA, USA, August 17-18, 2000, Proceedings*. Vol. 1965. 2000, pp. 252–263.

[Coh+20] Shaanan Cohney, Andrew Kwong, Shahar Paz, Daniel Genkin, Nadia Heninger, Eyal Ronen, and Yuval Yarom. "Pseudorandom Black Swans: Cache Attacks on CTR DRBG." In: *IEEE Symposium on Security and Privacy*. 2020, pp. 750–767.

[Con11] Austin Considine. *Pornography on airplanes, where you can't look away, The New York Times*. https://www.nytimes.com/2011/11/20/fashion/pornography-on-airplanes-where-you-cant-look-away.html. Accessed: 18-11-2020. Nov. 2011.

[CK09] Jean-Sébastien Coron and Ilya Kizhvatov. "An efficient method for random delay generation in embedded software." In: *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer. 2009, pp. 156–170.

[CV95] Corinna Cortes and Vladimir Vapnik. "Support-Vector Networks." In: *Mach. Learn.* 20.3 (1995), pp. 273–297.

[CS01] Koby Crammer and Yoram Singer. "On the Algorithmic Implementation of Multiclass Kernel-based Vector Machines." In: *J. Mach. Learn. Res.* 2 (2001), pp. 265–292.

[Cur20] Andrew Curran. *United airlines arains crew to stop in-flight porn use, Simple Flying*. https://simpleflying.com/united-airlines-stop-inflight-porn-use/. Accessed: 18-11-2020. Feb. 2020.

[DR99] Joan Daemen and Vincent Rijmen. "AES proposal: Rijndael." In: (1999).

[Dat] "Database for EdDSA." In: *URL: https://github.com/leoweissbart/MachineLearningBasedSideChannelAttackonEdDSA* (2019).

[Dob+21a] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. "Ascon v1.2: Lightweight Authenticated Encryption and Hashing." In: *J. Cryptol.* 34.3 (2021), p. 33.

[Dob+21b] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. *Ascon PRF, MAC, and Short-Input MAC*. Cryptology ePrint Archive, Paper 2021/1574. 2021.

[DHS11] John C. Duchi, Elad Hazan, and Yoram Singer. "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization." In: *J. Mach. Learn. Res.* 12 (2011), pp. 2121–2159.

[Dug+16] Margaux Dugardin, Louiza Papachristodoulou, Zakaria Najm, Lejla Batina, Jean-Luc Danger, and Sylvain Guilley. "Dismantling real-world ECC with Horizontal and Vertical Template Attacks." In: *Constructive Side-Channel Analysis and Secure Design - 7th International Workshop, COSADE 2016, Graz, Austria, April 14-15, 2016.* 2016.

[D+15] Michael Düll, Björn Haase, Gesine Hinterwälder, Michael Hutter, Christof Paar, Ana Helena Sánchez, and Peter Schwabe. "High-speed Curve25519 on 8-bit, 16-bit, and 32-bit microcontrollers." In: *Des. Codes Cryptogr.* 77.2-3 (2015), pp. 493–514.

[Ems] *EMSCAN EHX EMC Scanner*. https://www.atecorp.com/products/emscan/ehx. Accessed: 18-11-2020. Advanced Test Equipment Corp.

[Eck85] Wim van Eck. "Electromagnetic radiation from video display units: An eavesdropping risk?" In: *Computers & Security* 4.4 (1985), pp. 269–286.

[Eis+08] Thomas Eisenbarth, Timo Kasper, Amir Moradi, Christof Paar, Mahmoud Salmasizadeh, and Mohammad Shalmani. "On the Power of Power Analysis in the Real World: A Complete Break of the KeeLoq Code Hopping Scheme." In: *International Cryptology Conference*. 2008, pp. 203–220.

[EMH18] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. "Neural Architecture Search: A Survey." In: *CoRR* abs/1808.05377 (2018). arXiv: 1808.05377.

[Ene+11] Miro Enev, Sidhant Gupta, Tadayoshi Kohno, and Shwetak Patel. "Televisions, video privacy, and powerline electromagnetic interference." In: *ACM SIGSAC Conference on Computer and Communications Security*. 2011, pp. 537–550.

[FCL05] Rong-En Fan, Pai-Hsuen Chen, and Chih-Jen Lin. "Working Set Selection Using Second Order Information for Training Support Vector Machines." In: *J. Mach. Learn. Res.* 6 (Dec. 2005), pp. 1889–1918.

[FF15] John Fuegi and Jo Francis. "Lovelace & Babbage and the creation of the 1843 'notes'." In: *Inroads* 6.3 (2015), pp. 78–86.

[FHY19] Hironobu Fujiyoshi, Tsubasa Hirakawa, and Takayoshi Yamashita. "Deep learning-based image recognition for autonomous driving." In: *IATSS Research* 43.4 (2019), pp. 244–252.

[GMO01] K. Gandolfi, C. Mourtel, and F. Olivier. "Electromagnetic analysis: Concrete results." In: *International Workshop on Cryptographic Hardware and Embedded Systems*. 2001, pp. 255–265.

[GZC18] Yiwen Gao, Yongbin Zhou, and Wei Cheng. "How Does Strict Parallelism Affect Security? A Case Study on the Side-Channel Attacks against GPU-based Bitsliced AES Implementation." In: *IACR Cryptol. ePrint Arch.* 2018 (2018), p. 1080.

[Gen+16] Daniel Genkin, Lev Pachmanov, Itamar Pipman, Eran Tromer, and Yuval Yarom. "ECDSA key extraction from mobile devices via nonintrusive physical side channels." In: *ACM SIGSAC Conference on Computer and Communications Security*. 2016, pp. 1626–1638.

[Gen+19] Daniel Genkin, Mihir Pattani, Roei Schuster, and Eran Tromer. "Synesthesia: Detecting screen content via remote acoustic side channels." In: *IEEE Symposium on Security and Privacy*. 2019, pp. 853–869.

[GPT15] Daniel Genkin, Itamar Pipman, and Eran Tromer. "Get your hands off my laptop: Physical side-channel key-extraction attacks on PCs." In: *Journal of Cryptographic Engineering* 5.2 (2015), pp. 95–112.

[Gie+08] Benedikt Gierlichs, Lejla Batina, Pim Tuyls, and Bart Preneel. "Mutual Information Analysis." In: *Cryptographic Hardware and Embedded Systems - CHES 2008, 10th International Workshop, Washington, D.C., USA, August 10-13, 2008. Proceedings*. Vol. 5154. 2008, pp. 426–442.

[Gig+24] Barbara Gigerl, Florian Mendel, Martin Schläffer, and Robert Primas. "Efficient Second-Order Masked Software Implementations of Ascon in Theory and Practice." In: *IACR Cryptol. ePrint Arch.* (2024), p. 755.

[GHO15] R. Gilmore, N. Hanley, and M. O'Neill. "Neural network based attack on a masked implementation of AES." In: *2015 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. May 2015, pp. 106–111.

[GS15] Gabriel Goller and Georg Sigl. "Side channel attacks on smartphones and embedded devices using standard radio equipment." In: *International Workshop on Constructive Side-Channel Analysis and Secure Design*. Springer. 2015, pp. 255–270.

[GP13] José Luis Gómez Pardo. "Classical Ciphers and Their Cryptanalysis." In: *Introduction to Cryptography with Maple*. Berlin, Heidelberg, 2013, pp. 1–33.

[GPK02] Simona Grigorescu, Nicolai Petkov, and Peter Kruizinga. "Comparison of texture features based on Gabor filters." In: *IEEE Transactions on Image Processing* 11.10 (2002), pp. 1160–1167.

[HW10] David Harris and N Weste. "CMOS VLSI Design." In: *ed: Pearson Education, Inc* (2010).

[Hay+14] Yuichi Hayashi, Naofumi Homma, Mamoru Miura, Takafumi Aoki, and Hideaki Sone. "A threat for tablet PCs in public space: Remote visualization of screen images using EM emanation." In: *ACM SIGSAC Conference on Computer and Communications Security*. 2014, pp. 954–965.

[Hay+12a] Yuichi Hayashi, Naofumi Homma, Takaaki Mizuki, Takafumi Aoki, Hideaki Sone, Laurent Sauvage, and Jean-Luc Danger. "Analysis of electromagnetic information leakage from cryptographic devices with different physical structures." In: *IEEE Transactions on Electromagnetic Compatibility* 55.3 (2012), pp. 571–580.

[Hay+12b] Yuichi Hayashi, Naofumi Homma, Takaaki Mizuki, Haruki Shimada, Takafumi Aoki, Hideaki Sone, Laurent Sauvage, and Jean-Luc Danger. "Efficient evaluation of EM radiation associated with information leakage from cryptographic devices." In: *IEEE Transactions on Electromagnetic Compatibility* 55.3 (2012), pp. 555–563.

[Hay+16] Yuichi Hayashi, Naofumi Homma, Yohei Toriumi, Kazuhiro Takaya, and Takafumi Aoki. "Remote visualization of screen images using a pseudo-antenna that blends into the mobile environment." In: *IEEE Transactions on Electromagnetic Compatibility* 59.1 (2016), pp. 24–33.

[HN04] Simon Haykin and N Network. "A comprehensive foundation." In: *Neural networks* 2.2004 (2004), p. 41.

[He+16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition." In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 770–778.

[HGG18] Benjamin Hettwer, Stefan Gehrer, and Tim Güneysu. "Profiled Power Analysis Attacks Using Convolutional Neural Networks with Domain Knowledge." In: *Selected Areas in Cryptography - SAC 2018 - 25th International Conference, Calgary, AB, Canada, August 15-17, 2018, Revised Selected Papers*. 2018, pp. 479–498.

[HGG20] Benjamin Hettwer, Stefan Gehrer, and Tim Güneysu. "Deep Neural Network Attribution Methods for Leakage Analysis and Symmetric Key Recovery." In: *Selected Areas in Cryptography – SAC 2019*. Cham, 2020, pp. 645–666.

[Heu+17] A. Heuser, S. Picek, S. Guilley, and N. Mentens. "Lightweight Ciphers and their Side-channel Resilience." In: *IEEE Transactions on Computers* PP.99 (2017), pp. 1–1.

[HGR13] Annelie Heuser, Sylvain Guilley, and Olivier Rioul. "Practical vs. theoretical evaluation of DPA and CPA." In: *3rd International Workshop on Cryptography, Robustness, and Provably Secure Schemes for Female Young Researchers (CrossFyre'13)*. June 2013.

[Heu+16] Annelie Heuser, Stjepan Picek, Sylvain Guilley, and Nele Mentens. "Side-Channel Analysis of Lightweight Ciphers: Does Lightweight Equal Easy?" In: *Radio Frequency Identification and IoT Security - 12th International Workshop, RFIDSec 2016, Hong Kong, China, November 30 - December 2, 2016, Revised Selected Papers*. 2016, pp. 91–104.

[Hey+12] Johann Heyszl, Stefan Mangard, Benedikt Heinz, Frederic Stumpf, and Georg Sigl. "Localized Electromagnetic Analysis of Cryptographic Implementations." In: *Topics in Cryptology – CT-RSA 2012*. Vol. 7178. 2012, pp. 231–244.

[HRAu] UN. Office of the High Commissioner for Human Rights. *The right to privacy in the digital age : report of the Office of the United Nations High Commissioner for Human Rights*. 4 Aug. 2022.

[HHL11] Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. "Sequential Model-Based Optimization for General Algorithm Configuration." In: *Learning and Intelligent Optimization - 5th International Conference, LION 5, Rome, Italy, January 17-21, 2011. Selected Papers*. Vol. 6683. 2011, pp. 507–523.

[Ida00] Nathan Ida. *Engineering electromagnetics*. 2000.

[Iso] *Information security, cybersecurity and privacy protection — Evaluation criteria for IT security*. Standard. Geneva, CH: International Organization for Standardization, Aug. 2022.

[Inf24] Bundesamt für Sicherheit in der Informationstechnik. *Minimum Requirements for Evaluating Machine-learning based Side-Channel Attack Resistance*. 2024.

[ISW03] Yuval Ishai, Amit Sahai, and David A. Wagner. "Private Circuits: Securing Hardware against Probing Attacks." In: *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*. Vol. 2729. 2003, pp. 463–481.

[JSS] Jan Jancar, Petr Svenda, and Vladimir Sedlacek. *Minerva*. https://minerva.crocs.fi.muni.cz/. Accessed: 2020-02-13.

[JFK16] Zhen Hang Jiang, Yunsi Fei, and David Kaeli. "A complete key recovery timing attack on a GPU." In: *2016 IEEE International symposium on high performance computer architecture (HPCA)*. IEEE. 2016, pp. 394–405.

[JFK17] Zhen Hang Jiang, Yunsi Fei, and David Kaeli. "A novel side-channel timing attack on GPUs." In: *Proceedings of the on Great Lakes Symposium on VLSI 2017*. 2017, pp. 167–172.

[JSW98] Donald R. Jones, Matthias Schonlau, and William J. Welch. "Efficient Global Optimization of Expensive Black-Box Functions." In: *J. Glob. Optim.* 13.4 (1998), pp. 455–492.

[Kah67] D. Kahn. *The Codebreakers: The Story of Secret Writing*. 1967.

[Kee+01] S. Sathiya Keerthi, Shirish K. Shevade, Chiranjib Bhattacharyya, and K. R. K. Murthy. "Improvements to Platt's SMO Algorithm for SVM Classifier Design." In: *Neural Comput.* 13.3 (2001), pp. 637–649.

[Ker83] Auguste Kerckhoffs. *La cryptographie militaire, ou, Des chiffres usités en temps de guerre: avec un nouveau procédé de déchiffrement applicable aux systèmes à double clef*. 1883.

[Kim+19a] Jaehun Kim, Stjepan Picek, Annelie Heuser, Shivam Bhasin, and Alan Hanjalic. "Make Some Noise: Unleashing the Power of Convolutional Neural Networks for Profiled Side-channel Analysis." In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* 3 (2019), pp. 148–179.

[Kim+19b] Jaehun Kim, Stjepan Picek, Annelie Heuser, Shivam Bhasin, and Alan Hanjalic. "Make Some Noise. Unleashing the Power of Convolutional Neural Networks for Profiled Side-channel Analysis." In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2019.3 (May 2019), pp. 148–179.

[KB15] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization." In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. 2015.

[KFH19] Masahiro Kinugawa, Daisuke Fujimoto, and Yuichi Hayashi. "Electromagnetic information extortion from electronic devices using interceptor and its countermeasure." In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2019.4 (2019), pp. 62–90.

[KBP13] Jens Kober, J. Andrew Bagnell, and Jan Peters. "Reinforcement Learning in Robotics: A Survey." In: *Int. J. Rob. Res.* 32.11 (Sept. 2013), 1238–1274.

[Koc96] Paul C. Kocher. "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems." In: *Proceedings of CRYPTO'96*. Vol. 1109. 1996, pp. 104–113.

[KJJ99] Paul Kocher, Joshua Jaffe, and Benjamin Jun. "Differential power analysis." In: *Annual International Cryptology Conference*. Springer. London, UK, UK, 1999, pp. 388–397.

[Koc+11] Paul Kocher, Joshua Jaffe, Benjamin Jun, and Pankaj Rohatgi. "Introduction to differential power analysis." In: *Journal of Cryptographic Engineering* 1.1 (2011), pp. 5–27.

[Kri09] Alex Krizhevsky. *Learning Multiple Layers of Features from Tiny Images*. Tech. rep. University of Toronto, 2009.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. "Imagenet classification with deep convolutional neural networks." In: *Advances in Neural Information Processing Systems*. 2012, pp. 1097–1105.

[Kuh02a] Markus Kuhn. "Compromising emanations: eavesdropping risks of computer displays." PhD thesis. University of Cambridge, 2002.

[Kuh02b] Markus Kuhn. "Optical time-domain eavesdropping risks of CRT displays." In: *IEEE Symposium on Security and Privacy*. 2002, pp. 3–18.

[KA98] Markus Kuhn and Ross Anderson. "Soft tempest: Hidden data transmission using electromagnetic emanations." In: *International Workshop on Information Hiding*. Springer. 1998, pp. 124–142.

[Kur+21] Kunihiro Kuroda, Yuta Fukuda, Kota Yoshida, and Takeshi Fujino. "Practical Aspects on Non-profiled Deep-learning Side-channel Attacks against AES Software Implementation with Two Types of Masking Countermeasures including RSM." In: *ASHES@CCS 2021: Proceedings of the 5th Workshop on Attacks and Solutions in Hardware Security, Virtual Event, Republic of Korea, 19 November 2021*. 2021, pp. 29–40.

[Ku17] Martin Kučera, Petar Tsankov, Timon Gehr, Marco Guarnieri, and Martin Vechev. "Synthesis of Probabilistic Privacy Enforcement." In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. New York, NY, USA, 2017, 391–408.

[LeC+98] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. "Gradient-based learning applied to document recognition." In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[LeC+90] Yann LeCun, Ofer Matan, Bernhard E. Boser, John S. Denker, Don Henderson, Richard E. Howard, Wayne E. Hubbard, L. D. Jacket, and Henry S. Baird. "Handwritten zip code recognition with multilayer networks." In: *10th IAPR International Conference on Pattern Recognition, Conference C: image, speech, and signal processing, and Conference D: computer architecture for vision in pattern recognition, ICPR 1990, Atlantic City, NJ, USA, 16-21 June, 1990, Volume 2*. 1990, pp. 35–40.

[Lem+20] Florian Lemarchand, Cyril Marlin, Florent Montreuil, Erwan Nogues, and Maxime Pelcat. "Electro-Magnetic Side-Channel Attack Through Learned Denoising and Classification." In: *IEEE International Conference on Acoustics, Speech and Signal Processing*. 2020, pp. 2882–2886.

[LBM14] Liran Lerman, Gianluca Bontempi, and Olivier Markowitch. "Power Analysis Attack: An Approach Based on Machine Learning." In: *Int. J. Appl. Cryptol.* 3.2 (June 2014), pp. 97–115.

[Ler+15] Liran Lerman, Romain Poussier, Gianluca Bontempi, Olivier Markowitch, and François-Xavier Standaert. "Template Attacks vs. Machine Learning Revisited (and the Curse of Dimensionality in Side-Channel Analysis)." In: *COSADE 2015, Berlin, Germany, 2015. Revised Selected Papers*. 2015, pp. 20–33.

[LM17] Ke Li and Jitendra Malik. "Learning to Optimize." In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. 2017.

[Liu+] Zhuoran Liu, Niels Samwel, Léo Weissbart, Zhengyu Zhao, Dirk Lauret, Lejla Batina, and Martha A. Larson. "Screen Gleaning: A Screen Reading TEMPEST Attack on Mobile Devices Exploiting an Electromagnetic Side Channel." In: *28th Annual Network and Distributed System Security Symposium, NDSS 2021, virtually, February 21-25, 2021*.

[Low04] David Lowe. "Distinctive image features from scale-invariant keypoints." In: *International Journal of Computer Vision* 60.2 (2004), pp. 91–110.

[LL19] Alexander Selvikvåg Lundervold and Arvid Lundervold. "An overview of deep learning in medical imaging focusing on MRI." In: *Zeitschrift für Medizinische Physik* 29.2 (2019). Special Issue: Deep Learning in Medical Physics, pp. 102–127.

[Luo+15] Chao Luo, Yunsi Fei, Pei Luo, Saoni Mukherjee, and David Kaeli. "Side-channel power analysis of a GPU AES implementation." In: *2015 33rd IEEE International Conference on Computer Design (ICCD)*. IEEE. 2015, pp. 281–288.

[Mag20] Houssem Maghrebi. *Deep Learning based Side-Channel Attack: a New Profiling Methodology based on Multi-Label Classification*. Cryptology ePrint Archive, Paper 2020/436. 2020.

[MPP16] Houssem Maghrebi, Thibault Portigliatti, and Emmanuel Prouff. "Breaking Cryptographic Implementations Using Deep Learning Techniques." In: *Security, Privacy, and Applied Cryptography Engineering - 6th International Conference, SPACE 2016, Hyderabad, India, December 14-18, 2016, Proceedings*. 2016, pp. 3–26.

[MBC21] S. Maji, U. Banerjee, and A. P. Chandrakasan. "Leaky Nets: Recovering Embedded Neural Network Models and Inputs through Simple Power and Timing Side-Channels – Attacks and Defenses." In: *IEEE Internet of Things Journal* (2021), pp. 1–1.

[MOP06] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power Analysis Attacks: Revealing the Secrets of Smart Cards*. Dec. 2006, p. 338.

[Mar] Martin Marinov. *TempestSDR*. https://github.com/martinmarinov/TempestSDR. Accessed: 18-11-2020.

[Mar14] Martin Marinov. "Remote video eavesdropping using a software-defined radio platform." MA thesis. University of Cambridge, 2014.

[MO23a] Thomas Marquet and Elisabeth Oswald. "A Comparison of Multi-task learning and Single-task learning Approaches." In: *IACR Cryptol. ePrint Arch.* (2023), p. 611.

[MO23b] Thomas Marquet and Elisabeth Oswald. "Exploring multi-task learning in the context of two masked AES implementations." In: *IACR Cryptol. ePrint Arch.* (2023), p. 6.

[MSS12] James Martens, Ilya Sutskever, and Kevin Swersky. "Estimating the Hessian by Back-propagating Curvature." In: *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*. 2012.

[MZ13] Z. Martinasek and V. Zeman. "Innovative Method of the Power Analysis." In: *Radioengineering* 22.2 (2013).

[MHM14] Zdenek Martinasek, Jan Hajny, and Lukas Malina. "Optimization of Power Analysis Using Neural Network." In: *Smart Card Research and Advanced Applications*. Cham, 2014, pp. 94–107.

[MDP19] Loïc Masure, Cécile Dumas, and Emmanuel Prouff. "Gradient Visualization for General Characterization in Profiling Attacks." In: *Constructive Side-Channel Analysis and Secure Design - 10th International Workshop, COSADE 2019, Darmstadt, Germany, April 3-5, 2019, Proceedings*. 2019, pp. 145–167.

[MS23] Loïc Masure and Rémi Strullu. "Side-channel analysis against ANSSI's protected AES implementation on ARM: end-to-end attacks with multi-task learning." In: *J. Cryptogr. Eng.* 13.2 (2023), pp. 129–147.

[McN] Joel McNamara. *The Complete, Unofficial TEMPEST Information Page*. http://www.kubieziel.de/blog/uploads/complete_unofficial_tempest_page.pdf. Accessed: 18-11-2020.

[MO08] Marcel Medwed and Elisabeth Oswald. "Template attacks on ECDSA." In: *International Workshop on Information Security Applications*. Springer. 2008, pp. 14–27.

[MOV96] Alfred Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. 1996.

[Min01] Thomas P Minka. "Automatic choice of dimensionality for PCA." In: *Advances in neural information processing systems*. 2001, pp. 598–604.

[Mit97] Tom M. Mitchell. *Machine learning, International Edition*. 1997.

[Moc77] Jonas Mockus. "On Bayesian Methods for Seeking the Extremum and their Application." In: *Information Processing, Proceedings of the 7th IFIP Congress 1977, Toronto, Canada, August 8-12, 1977*. 1977, pp. 195–200.

[Mog+20] Daniel Moghimi, Berk Sunar, Thomas Eisenbarth, and Nadia Heninger. "TPM-FAIL: TPM meets Timing and Lattice Attacks." In: *USENIX Security Symposium*. 2020, pp. 2057–2073.

[Moh+23] Kamyar Mohajerani, Luke Beckwith, Abubakr Abdulgadir, Eduardo Ferrufino, Jens-Peter Kaps, and Kris Gaj. "SCA Evaluation and Benchmarking of Finalists in the NIST Lightweight Cryptography Standardization Process." In: *IACR Cryptol. ePrint Arch.* (2023), p. 484.

[MWM21] Thorben Moos, Felix Wegener, and Amir Moradi. "DL-LA: Deep Learning Leakage Assessment: A modern roadmap for SCA evaluations." In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2021.3 (July 2021), 552–598.

[Mur12] Kevin P. Murphy. *Machine learning - a probabilistic perspective*. 2012.

[NIS99] FIPS PUB NIST. "Data encryption standard (des)." In: *National Institute of Standards and Technology* (1999), pp. 46–3.

[NIS15] FIPS PUB NIST. "180-4 Secure Hash Standard (SHS)." In: *National Institute of Standards and Technology* (2015).

[Nan] *NVIDIA Jetson Nano module Datasheet, April 2021*. https://developer.nvidia.com/embedded/dlc/jetson-nano-system-module-datasheet.

[Teg] *NVIDIA Tegra X1 White Paper, April 2021*. http://international.download.nvidia.com/pdf/tegra/Tegra-X1-whitepaper-v1.0.pdf.

[Nag+19] Hoda Naghibijouybari, Ajaya Neupane, Zhiyun Qian, and Nael Abu Ghazaleh. "Side channel attacks on GPUs." In: *IEEE Transactions on Dependable and Secure Computing* (2019).

[NH10] Vinod Nair and Geoffrey E Hinton. "Rectified linear units improve restricted boltzmann machines." In: *Icml*. 2010.

[NC17] Erick Nascimento and Łukasz Chmielewski. *Horizontal Clustering Side-Channel Attacks on Embedded ECC Implementations (Extended Version)*. Cryptology ePrint Archive, Report 2017/1204. 2017.

[Nas+17] Erick Nascimento, Łukasz Chmielewski, David Oswald, and Peter Schwabe. "Attacking Embedded ECC Implementations Through cmov Side Channels." In: *Selected Areas in Cryptography – SAC 2016*. Cham, 2017, pp. 99–119.

[Ngu19] Vu Nguyen. “Bayesian Optimization for Accelerating Hyper-Parameter Tuning.” In: *2nd IEEE International Conference on Artificial Intelligence and Knowledge Engineering, AIKE 2019, Sardinia, Italy, June 3-5, 2019*. 2019, pp. 302–305.

[Nic+08] John Nickolls, Ian Buck, Michael Garland, and Kevin Skadron. “Scalable parallel programming with cuda: Is cuda the parallel programming model that application developers have been waiting for?” In: *Queue* 6.2 (2008), pp. 40–53.

[OP11] David Oswald and Christof Paar. “Breaking Mifare DESFire MF3ICD40: Power analysis and templates in the real world.” In: *International Workshop on Cryptographic Hardware and Embedded Systems*. 2011, pp. 207–222.

[OPB16] Elif Özgen, Louiza Papachristodoulou, and Lejla Batina. “Classification Algorithms for Template Matching.” In: *IEEE International Symposium on Hardware Oriented Security and Trust, HOST 2016, McLean, VA, USA, 2016 (to appear)*. 2016.

[Pap+18] Nicolas Papernot, Patrick McDaniel, Arunesh Sinha, and Michael P. Wellman. “SoK: Security and Privacy in Machine Learning.” In: *2018 IEEE European Symposium on Security and Privacy (EuroS P)*. 2018, pp. 399–414.

[Pas+19] Adam Paszke et al. “PyTorch: An Imperative Style, High-Performance Deep Learning Library.” In: *Advances in Neural Information Processing Systems 32*. 2019, pp. 8024–8035.

[PSS22] Clayton R Paul, Robert C Scully, and Mark A Steffka. *Introduction to electromagnetic compatibility*. 2022.

[Ped+11] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python.” In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[PSQ07] Eric Peeters, François-Xavier Standaert, and Jean-Jacques Quisquater. “Power and electromagnetic analysis: Improved model, consequences and comparisons.” In: *Integr.* 40.1 (2007), pp. 52–60.

[PEC19] Guilherme Perin, Baris Ege, and Łukasz Chmielewski. “Neural Network Model Assessment for Side-Channel Analysis.” In: *IACR Cryptology ePrint Archive* 2019 (2019), p. 722.

[PH18] Christophe Pfeifer and Patrick Haddad. *Spread: a new layer for profiled deep-learning side-channel attacks*. Cryptology ePrint Archive, Report 2018/880. 2018.

[Pic+18a] Stjepan Picek, Annelie Heuser, Cesare Alippi, and Francesco Regazzoni. *When Theory Meets Practice: A Framework for Robust Profiled Side-channel Analysis*. Cryptology ePrint Archive, Report 2018/1123. 2018.

[PHG19] Stjepan Picek, Annelie Heuser, and Sylvain Guilley. *Profiling Side-channel Analysis in the Restricted Attacker Framework*. Cryptology ePrint Archive, Report 2019/168. 2019.

[Pic+19] Stjepan Picek, Annelie Heuser, Alan Jovic, Shivam Bhasin, and Francesco Regazzoni. "The Curse of Class Imbalance and Conflicting Metrics with Machine Learning for Side-channel Evaluations." In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2019.1 (2019), pp. 209–237.

[Pic+17] Stjepan Picek, Annelie Heuser, Alan Jovic, Simone A. Ludwig, Sylvain Guilley, Domagoj Jakobovic, and Nele Mentens. "Side-channel analysis and machine learning: A practical perspective." In: *2017 International Joint Conference on Neural Networks, IJCNN 2017, Anchorage, AK, USA, May 14-19, 2017*. 2017, pp. 4095–4102.

[Pic+23] Stjepan Picek, Guilherme Perin, Luca Mariot, Lichao Wu, and Lejla Batina. "SoK: Deep Learning-based Physical Side-channel Analysis." In: *ACM Comput. Surv.* 55.11 (2023), 227:1–227:35.

[Pic+18b] Stjepan Picek, Ioannis Petros Samiotis, Jaehun Kim, Annelie Heuser, Shivam Bhasin, and Axel Legay. "On the Performance of Convolutional Neural Networks for Side-Channel Analysis." In: *Security, Privacy, and Applied Cryptography Engineering*. Cham, 2018, pp. 157–176.

[Pla98] John Platt. "Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines." In: *Advances in Kernel Methods-Support Vector Learning* 208 (July 1998).

[Pop09] Thomas Popp. "An introduction to implementation attacks and countermeasures." In: *7th ACM/IEEE International Conference on Formal Methods and Models for Codesign (MEMOCODE 2009), July 13-15, 2009, Cambridge, Massachusetts, USA*. 2009, pp. 108–115.

[PZS17] Romain Poussier, Yuanyuan Zhou, and François-Xavier Standaert. "A Systematic Approach to the Side-Channel Analysis of ECC Implementations with Worst-Case Horizontal Attacks." In: *Crypto-*

graphic Hardware and Embedded Systems – CHES 2017. Cham, 2017, pp. 534–554.

[Pro+18] Emmanuel Prouff, Remi Strullu, Ryad Benadjila, Eleonora Cagli, and Cécile Dumas. "Study of Deep Learning Techniques for Side-Channel Analysis and Introduction to ASCAD Database." In: *IACR Cryptology ePrint Archive* 2018 (2018), p. 53.

[QS01a] Jean-Jacques Quisquater and David Samyde. "ElectroMagnetic Analysis (EMA): Measures and Counter-measures for Smart Cards." In: *Smart Card Programming and Security*. Berlin, Heidelberg, 2001, pp. 200–210.

[QS01b] Jean-Jacques Quisquater and David Samyde. "ElectroMagnetic Analysis (EMA): Measures and counter-measures for smard cards." In: *International Conference on Research in Smart Cards*. 2001, pp. 200–210.

[RCN02] Jan M Rabaey, Anantha Chandrakasan, and Borivoje Nikolic. *Digital integrated circuits*. Vol. 2. 2002.

[RAD20] Keyvan Ramezanpour, Paul Ampadu, and William Diehl. "SCARL: Side-Channel Analysis with Reinforcement Learning on the Ascon Authenticated Cipher." In: *CoRR* abs/2006.03995 (2020). arXiv: 2006.03995.

[RW06] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian processes for machine learning*. 2006.

[RK21] Tanu Shree Rastogi and Elif Bilge Kavun. "Deep Learning Techniques for Side-Channel Analysis on AES Datasets Collected from Hardware and Software Platforms." In: *Embedded Computer Systems: Architectures, Modeling, and Simulation - 21st International Conference, SAMOS 2021, Virtual Event, July 4-8, 2021, Proceedings*. Vol. 13227. 2021, pp. 300–316.

[Rij+21] Jorai Rijsdijk, Lichao Wu, Guilherme Perin, and Stjepan Picek. "Reinforcement Learning for Hyperparameter Tuning in Deep Learning-based Side-channel Analysis." In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2021.3 (July 2021), pp. 677–707.

[Ris18] Riscure. *Automated Neural Network construction with Genetic Algorithm (blog post)*. https://www.riscure.com/blog/automated-neural-network-construction-genetic-algorithm. 2018.

[Riv08] Matthieu Rivain. “On the Exact Success Rate of Side Channel Analysis in the Gaussian Model.” In: *Selected Areas in Cryptography, 15th International Workshop, SAC 2008, Sackville, New Brunswick, Canada, August 14-15, Revised Selected Papers*. Vol. 5381. 2008, pp. 165–183.

[Sam+18] Niels Samwel, Lejla Batina, Guido Bertoni, Joan Daemen, and Ruggero Susella. “Breaking ed25519 in WolfSSL.” In: *Cryptographers’ Track at the RSA Conference*. Springer. 2018, pp. 1–20.

[SD17] Niels Samwel and Joan Daemen. “DPA on hardware implementations of Ascon and Keyak.” In: *Proceedings of the Computing Frontiers Conference, CF’17, Siena, Italy, May 15-17, 2017*. 2017, pp. 415–424.

[SGS09] Koen van de Sande, Theo Gevers, and Cees Snoek. “Evaluating color descriptors for object and scene recognition.” In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32.9 (2009), pp. 1582–1596.

[SH12] *Constructive Side-Channel Analysis and Secure Design - Third International Workshop, COSADE 2012, Darmstadt, Germany, May 3-4, 2012. Proceedings*. Vol. 7275. 2012.

[Sch91] Claus-Peter Schnorr. “Efficient signature generation by smart cards.” In: *Journal of cryptology* 4.3 (1991), pp. 161–174.

[SS13] Hidenori Sekiguchi and Shinji Seto. “Study on maximum receivable distance for radiated emission of information technology equipment causing information leakage.” In: *IEEE Transactions on Electromagnetic Compatibility* 55.3 (2013), pp. 547–554.

[SRA81] Adi Shamir, Ronald L Rivest, and Leonard M Adleman. *Mental poker*. 1981.

[SS23] Dillibabu Shanmugam and Patrick Schaumont. “Improving Side-channel Leakage Assessment Using Pre-silicon Leakage Models.” In: *Constructive Side-Channel Analysis and Secure Design - 14th International Workshop, COSADE 2023, Munich, Germany, April 3-4, 2023, Proceedings*. Vol. 13979. 2023, pp. 105–124.

[SZ14] Karen Simonyan and Andrew Zisserman. “Very deep convolutional networks for large-scale image recognition.” In: *arXiv preprint arXiv:1409.1556* (2014).

[Sin23] Satyajit Sinha. *State of IoT 2023: Number of connected IoT devices growing 16% to 16.7 billion globally*. May 2023.

[SLA12] Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. "Practical Bayesian Optimization of Machine Learning Algorithms." In: *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*. 2012, pp. 2960–2968.

[SMY09] François-Xavier Standaert, Tal Malkin, and Moti Yung. "A Unified Framework for the Analysis of Side-Channel Key Recovery Attacks." In: *EUROCRYPT*. Vol. 5479. Cologne, Germany. Apr. 2009, pp. 443–461.

[Sta10] François-Xavier Standaert. "Introduction to Side-Channel Attacks." In: *Secure Integrated Circuits and Systems*. 2010, pp. 27–42.

[SGV08] François-Xavier Standaert, Benedikt Gierlichs, and Ingrid Verbauwhede. "Partition vs. Comparison Side-Channel Distinguishers: An Empirical Evaluation of Statistical Tests for Univariate Side-Channel Attacks against Two Unprotected CMOS Devices." In: *Information Security and Cryptology - ICISC 2008, 11th International Conference, Seoul, Korea, December 3-5, 2008, Revised Selected Papers*. Vol. 5461. 2008, pp. 253–267.

[STY17] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. "Axiomatic attribution for deep networks." In: *arXiv preprint arXiv:1703.01365* (2017).

[Tak+20] Go Takatoi, Takeshi Sugawara, Kazuo Sakiyama, and Yang Li. "Simple Electromagnetic Analysis Against Activation Functions of Deep Neural Networks." In: *Applied Cryptography and Network Security Workshops - ACNS, Rome, Italy, October 19-22, 2020, Proceedings*. Vol. 12418. 2020, pp. 181–197.

[TSL19] Zeenat Tariq, Sayed Khushal Shah, and Yugyung Lee. "Speech Emotion Detection using IoT based Deep Learning for Health Care." In: *2019 IEEE International Conference on Big Data (Big Data), Los Angeles, CA, USA, December 9-12, 2019*. 2019, pp. 4191–4196.

[Tea] ASCON Team. *ASCON C repository*.

[TPL10] Peter Teufl, Udo Payer, and Guenter Lackner. "From NLP (Natural Language Processing) to MLP (Machine Language Processing)." In: *Computer Network Security*. Berlin, Heidelberg, 2010, pp. 256–269.

[Tim19] Benjamin Timon. "Non-Profiled Deep Learning-based Side-Channel attacks with Sensitivity Analysis." In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2019.2 (Feb. 2019), pp. 107–131.

[Tir+05] Kris Tiri, Davis Hwang, Alireza Hodjat, Bo-Cheng Lai, Shenglin Yang, Patrick Schaumont, and Ingrid Verbauwhede. "Prototype IC with WDDL and differential routing – DPA resistance assessment." In: *Annual Conference on Cryptographic Hardware and Embedded Systems*. 2005, pp. 354–365.

[Tun+07] Michael Tunstall, Neil Hanley, Robert McEvoy, Claire Whelan, Colin Murphy, and William Marnane. "Correlation power analysis of large word sizes." In: *IET Irish Signals and Systems Conference (ISSC)*. Sept. 2007, pp. 145–150.

[Tur+21] Ryan Turner, David Eriksson, Michael McCourt, Juha Kiili, Eero Laaksonen, Zhen Xu, and Isabelle Guyon. "Bayesian Optimization is Superior to Random Search for Machine Learning Hyperparameter Tuning: Analysis of the Black-Box Optimization Challenge 2020." In: *CoRR* abs/2104.10201 (2021). arXiv: 2104.10201.

[Tuv+18] Nicola Tuveri, Sohaib ul Hassan, Cesar Pereida Garcia, and Billy Bob Brumley. "Side-Channel Analysis of SM2: A Late-Stage Featurization Case Study." In: *Proceedings of the 34th Annual Computer Security Applications Conference*. New York, NY, USA, 2018, pp. 147–160.

[VP19] Daan van der Valk and Stjepan Picek. *Bias-variance Decomposition in Machine Learning-based Side-channel Analysis*. Cryptology ePrint Archive, Report 2019/570. 2019.

[VPB19] Daan van der Valk, Stjepan Picek, and Shivam Bhasin. *Kilroy was here: The First Step Towards Explainability of Neural Networks in Profiled Side-channel Analysis*. Cryptology ePrint Archive, Report 2019/1477. 2019.

[Vap95] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. New York, NY, USA, 1995.

[Wei+20a] Junyi Wei, Yicheng Zhang, Zhe Zhou, Zhou Li, and Mohammad Abdullah Al Faruque. "Leaky DNN: Stealing Deep-Learning Model Secret with GPU Context-Switching Side-Channel." In: *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE. 2020, pp. 125–137.

[Wei+18] Lingxiao Wei, Bo Luo, Yu Li, Yannan Liu, and Qiang Xu. "I Know What You See: Power Side-Channel Attack on Convolutional Neural Network Accelerators." In: *Proceedings of the 34th Annual Computer Security Applications Conference*. New York, NY, USA, 2018, 393–406.

[WPB19] Léo Weissbart, Stjepan Picek, and Lejla Batina. "One Trace Is All It Takes: Machine Learning-Based Side-Channel Attack on EdDSA." In: *Security, Privacy, and Applied Cryptography Engineering*. Cham, 2019, pp. 86–105.

[Wei] Léo Weissbart. "Performance Analysis of Multilayer Perceptron in Profiling Side-Channel Analysis." In: *Applied Cryptography and Network Security Workshops - ACNS, Rome, Italy, October 19-22, 2020, Proceedings*. Vol. 12418, pp. 198–216.

[Wei+20b] Léo Weissbart, Lukasz Chmielewski, Stjepan Picek, and Lejla Batina. "Systematic Side-Channel Analysis of Curve25519 with Machine Learning." In: *J. Hardw. Syst. Secur.* 4.4 (2020), pp. 314–328.

[WP23] Léo Weissbart and Stjepan Picek. *Lightweight but Not Easy: Side-channel Analysis of the Ascon Authenticated Cipher on a 32-bit Microcontroller*. Cryptology ePrint Archive, Paper 2023/1598. 2023.

[WPB] Léo Weissbart, Stjepan Picek, and Lejla Batina. "One Trace Is All It Takes: Machine Learning-Based Side-Channel Attack on EdDSA." In: *Security, Privacy, and Applied Cryptography Engineering - 9th International Conference, SPACE 2019, Gandhinagar, India, December 3-7, 2019, Proceedings*. Vol. 11947, pp. 86–105.

[WNS21] Colin White, Willie Neiswanger, and Yash Savani. "BANANAS: Bayesian Optimization with Neural Architectures for Neural Architecture Search." In: *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*. 2021, pp. 10293–10301.

[Wol96] David H. Wolpert. "The Lack of A Priori Distinctions Between Learning Algorithms." In: *Neural Comput.* 8.7 (1996), pp. 1341–1390.

[WPP20] Lichao Wu, Guilherme Perin, and Stjepan Picek. “I Choose You: Automated Hyperparameter Tuning for Deep Learning-based Side-channel Analysis.” In: *IACR Cryptol. ePrint Arch.* (2020), p. 1293.

[Wu+20] Lichao Wu, Léo Weissbart, Marina Krček, Huimin Li, Guilherme Perin, Lejla Batina, and Stjepan Picek. *Everything is Connected: From Model Learnability to Guessing Entropy*. Cryptology ePrint Archive, Report 2020/899. 2020.

[Xia+20] Yun Xiang, Zhuangzhi Chen, Zuohui Chen, Zebin Fang, Haiyang Hao, Jinyin Chen, Yi Liu, Zhefu Wu, Qi Xuan, and Xiaoniu Yang. “Open DNN Box by Power Side-Channel Attack.” In: *IEEE Transactions on Circuits and Systems II: Express Briefs* 67.11 (2020), pp. 2717–2721.

[XWZ18] Mengmeng Xu, Liji Wu, and Xiangmin Zhang. “Power Analysis on SM4 with Boosting Methods.” In: *2018 12th IEEE International Conference on Anti-counterfeiting, Security, and Identification (ASID)*. IEEE. 2018, pp. 188–191.

[XAQ21] Qian Xu, Md Tanvir Arafin, and Gang Qu. “Security of Neural Networks from Hardware Perspective: A Survey and Beyond.” In: *Proceedings of the 26th Asia and South Pacific Design Automation Conference*. Tokyo, Japan, 2021, 449–454.

[Xu+13] Yi Xu, Jared Heinly, Andrew White, Fabian Monrose, and Jan-Michael Frahm. “Seeing double: Reconstructing obscured typed input from repeated compromising reflections.” In: *ACM SIGSAC conference on Computer & communications security*. 2013, pp. 1063–1074.

[Yan+15] Lin Yan, Yao Guo, Xiangqun Chen, and Hong Mei. “A study on power side channels on mobile devices.” In: *Asia-Pacific Symposium on Internetware*. 2015, pp. 30–38.

[Yan+12] Shuguo Yang, Yongbin Zhou, Jiye Liu, and Danyang Chen. “Back Propagation Neural Network Based Leakage Characterization for Practical Security Analysis of Cryptographic Implementations.” In: *Information Security and Cryptology - ICISC 2011*. Berlin, Heidelberg, 2012, pp. 169–185.

[Yao82] Andrew Chi-Chih Yao. “Protocols for Secure Computations (Extended Abstract).” In: *23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3-5 November 1982*. 1982, pp. 160–164.

[YNY17] Jian Ye, Jiangqun Ni, and Yang Yi. "Deep learning hierarchical representations for image steganalysis." In: *IEEE Transactions on Information Forensics and Security* 12.11 (2017), pp. 2545–2557.

[Yos+20] Kota Yoshida, Takaya Kubota, Shunsuke Okura, Mitsuru Shiozaki, and Takeshi Fujino. "Model Reverse-Engineering Attack using Correlation Power Analysis against Systolic Array Based Neural Network Accelerator." In: *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*. 2020, pp. 1–5.

[You+23] Shih-Chun You, Markus G. Kuhn, Sumanta Sarkar, and Feng Hao. "Low Trace-Count Template Attacks on 32-bit Implementations of ASCON AEAD." In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2023.4 (2023), pp. 344–366.

[Yu+20] Honggang Yu, Haocheng Ma, Kaichen Yang, Yiqiang Zhao, and Yier Jin. "DeepEM: Deep Neural Networks Model Recovery through EM Side-Channel Information Leakage." In: *2020 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. IEEE. 2020, pp. 209–218.

[Zai+19] Gabriel Zaid, Lilian Bossuet, Amaury Habrard, and Alexandre Venelli. "Methodology for Efficient CNN Architectures in Profiling Attacks." In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2020.1 (Nov. 2019), pp. 1–36.

[Zho+18] Peng Zhou, Xintong Han, Vlad Morariu, and Larry Davis. "Learning rich features for image manipulation detection." In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 1053–1061.

ACRONYMS

CC	Common Criteria
AI	Artificial Intelligence
DL	Deep Learning
SCA	Side-channel Analysis
EM	Electromagnetic
FI	Fault Injection
HD	Hamming Distance
HW	Hamming Weight
ID	Identity
CMOS	Complementary metal-oxide-semiconductor
ECC	Elliptic Curve Cryptography
SPA	Simple Power Analysis
DPA	Differential Power Analysis
CPA	Correlation Power Analysis
TA	Template Attack
SR	Success Rate
GE	Guessing Entropy
ANN	Artificial Neural Network
MLP	Multilayer Perceptron
CNN	Convolutional Neural Network
PCA	Principal Component Analysis
SVM	Support Vector Machine

RF Random Forest

XGBoost eXtreme Gradient Boosting

NB Gaussian Naive Bayes

SMO Sequential Minimal Optimization

RBF Radial Basis Function

RMSProp Root Mean Square Propagation

SGD Stochastic Gradient Descent

AdaGrad Adaptive Gradient Algorithm

Adam Adaptive Moment Estimation

DLSCA Deep learning Side-Channel Analysis

SUMMARY

To ensure that no information unintentionally leaks through side channels during the execution of cryptographic operations, the physical security of a device must be evaluated. Nowadays, a security analysis must show security not only against traditional Side-Channel Analysis (SCA) attacks (e.g., Differential Power Analysis (DPA)) involving classical statistical analysis but also against machine learning and deep learning attacks. If not protected against these attacks, symmetric and public-key cryptographic implementations can be at risk.

While traditional SCA attacks rely on a cryptanalyst's expertise to extract features from the leakages of one or multiple traces and analyze their observations through statistical methods to recover the secret key. Deep Learning-based Side-Channel Analysis (DLSCA) attacks bring a new perspective to the field. DLSCA attacks rely on automating feature extraction using a task-specific algorithm. For most DLSCA attacks, an expert is still needed, but the expert's work is shifted to training this algorithm. Among the different deep learning architectures, the most used in DLSCA are the Multilayer Perceptron (MLP) and the Convolutional Neural Networks (CNN). Those methods are Neural Networks (NN) trained to find patterns in a collected dataset of side-channel traces to recover the secret key given a proper tuning of their hyperparameters and a successful training process.

This thesis investigates the use of deep learning in side-channel analysis of symmetric and public-key cryptography and other applications of side-channel analysis. We go through the application of DLSCA for implementations of AES and ASCON in symmetric cryptography and EdDSA in public-key cryptography. We also explore the use of deep learning to enhance TEMPEST-like side-channel analysis and the use of side-channel analysis to reverse engineer neural networks.

The main contributions of this thesis are as follows. First, we show the performances that can reach a MLP on a dataset of an AES implementation protected with a masking countermeasure. We demonstrate that MLP can defeat the masking countermeasure and recover the secret key with a high success rate for many configurations of hyperparameters and power intermediate models and even with very few parameters.

Second, we present an application of CNN in the side-channel analysis of the lightweight authenticated encryption algorithm ASCON on a 32-bit microcontroller. We demonstrate that the reference implementation is vulnerable to DLSCA

attacks and that the same attack can be applied to a masked implementation but cannot completely recover the secret key.

Third, we propose a single-trace attack on the ephemeral key of EdDSA on the elliptic curve 25519. We show that the attack can recover the secret key from a single execution of an implementation on a 32-bit microcontroller. This attack is based on a CNN, and we demonstrate that, of the other profiling methods explored, the CNN is the most efficient for this attack. Furthermore, we systematize this attack and show that it can be applied to a different target and implement countermeasures against side-channel analysis.

Finally, we demonstrate the use of side-channel analysis and deep learning in different applications than cryptographic implementations. We present a methodology to evaluate TEMPEST attacks using deep learning. We focus the analysis of the electromagnetic emanations of mobile devices without visual line of sight, to build a testbed with a standard setup that can be used to test different attacker models. A second application is the use of side-channel analysis to reverse engineer neural networks on GPU. We show that side-channel analysis of the electromagnetic emanations of a GPU can be used to recover several hyperparameters of a neural network during the inference phase.

Our main research goal is to apply deep learning to side-channel analysis to develop new attacks for existing implementations and countermeasures, and we believe that this thesis is a step in that direction regarding the aforementioned contributions. We also believe that the reading of this thesis will shine the light on the potential of deep learning in side-channel analysis and inspire future research in this field to help to secure the electronics of tomorrow.

SAMMENVATING

Om ervoor te zorgen dat er geen informatie onbedoeld weglekt via nevenkanalen tijdens de uitvoering van cryptografische bewerkingen, moet de fysieke beveiliging van een apparaat worden geëvalueerd. Tegenwoordig moet een beveiligingsanalyse niet alleen beveiliging aantonen tegen traditionele Side-Channel Analysis (SCA)-aanvallen (bijv. Differential Power Analysis (DPA)) met klassieke statistische analyse, maar ook tegen machine learning- en deep learning-aanvallen. Als ze niet beschermd zijn tegen deze aanvallen, kunnen symmetrische en publieke-sleutel cryptografische implementaties gevaar lopen.

Terwijl traditionele SCA-aanvallen vertrouwen op de expertise van een cryptoanalist om kenmerken te extraheren uit de lekken van één of meerdere sporen en de waarnemingen te analyseren met statistische methoden om de geheime sleutel te achterhalen. Deep Learning-gebaseerde Side-Channel Analysis (DLSCA)-aanvallen brengen een nieuw perspectief in het veld. DLSCA-aanvallen vertrouwen op het automatiseren van kenmerkextractie met behulp van een taakspecifiek algoritme. Voor de meeste DLSCA-aanvallen is nog steeds een expert nodig, maar het werk van de expert wordt verschoven naar het trainen van dit algoritme. Van de verschillende deep learning-architecturen zijn de meest gebruikte in DLSCA de Multilayer Perceptron (MLP) en de Convolutional Neural Networks (CNN). Deze methoden zijn neurale netwerken (NN) die zijn getraind om patronen te vinden in een verzamelde dataset van nevenkanaalsporen om de geheime sleutel te achterhalen als hun hyperparameters goed zijn ingesteld en het trainingsproces succesvol verloopt.

Deze dissertatie onderzoekt het gebruik van deep learning in side-channel analyse van symmetrische en publieke-sleutel cryptografie en andere toepassingen van side-channel analyse. We doorlopen de toepassing van DLSCA voor implementaties van AES en ASCON in symmetrische cryptografie en EdDSA in publieke-sleutel cryptografie. We verkennen ook het gebruik van deep learning om TEMPEST-achtige zijkanaalanalyse te verbeteren en het gebruik van zijkanaalanalyse om neurale netwerken te reverse engineeren.

De belangrijkste bijdragen van dit proefschrift zijn als volgt. Ten eerste laten we de prestaties zien die een MLP kan bereiken op een dataset van een AES-implementatie die is beveiligd met een maskerende tegenmaatregel. We tonen aan dat MLP de maskerende tegenmaatregel kan verslaan en de geheime sleutel kan

achterhalen met een hoog succespercentage voor veel configuraties van hyperparameters en tussenliggende modellen en zelfs met zeer weinig parameters.

Ten tweede presenteren we een toepassing van CNN in de analyse van nevenkanalen van het lichtgewicht geauthenticeerde versleutelingsalgoritme ASCON op een 32-bits microcontroller. We tonen aan dat de referentie-implementatie kwetsbaar is voor DLSCA-aanvallen en dat dezelfde aanval kan worden toegepast op een gemaskeerde implementatie, maar de geheime sleutel niet volledig kan achterhalen.

Ten derde stellen we een single-trace aanval voor op de efemere sleutel van EdDSA op de elliptische curve 25519. We laten zien dat de aanval de geheime sleutel kan achterhalen uit een enkele uitvoering van een implementatie op een 32-bits microcontroller. Deze aanval is gebaseerd op een CNN en we tonen aan dat, van de andere onderzochte profileringsmethoden, de CNN het meest efficiënt is voor deze aanval. Verder systematiseren we deze aanval en laten we zien dat deze kan worden toegepast op een ander doelwit en implementeren we tegenmaatregelen tegen side-channel analyse.

Tot slot tonen we het gebruik van side-channel analyse en deep learning in andere toepassingen dan cryptografische implementaties. We presenteren een methodologie om TEMPEST-aanvallen te evalueren met behulp van deep learning. We richten ons op de analyse van de elektromagnetische straling van mobiele apparaten zonder visuele zichtlijn, om een testbed te bouwen met een standaardopstelling die gebruikt kan worden om verschillende aanvalsmodellen te testen. Een tweede toepassing is het gebruik van side-channel analyse voor reverse-engineering van neurale netwerken op GPU. We laten zien dat side-channel analyse van de elektromagnetische straling van een GPU gebruikt kan worden om verschillende hyperparameters van een neuraal netwerk te achterhalen tijdens de inferentiefase.

Ons belangrijkste onderzoeksdoel is om deep learning toe te passen op side-channel analyse om nieuwe aanvallen te ontwikkelen voor bestaande implementaties en tegenmaatregelen, en we geloven dat dit proefschrift een stap in die richting is met betrekking tot de eerder genoemde bijdragen. We geloven ook dat het lezen van dit proefschrift het licht zal schijnen op het potentieel van deep learning in zijkanaalanalyse en toekomstig onderzoek op dit gebied zal inspireren om de elektronica van morgen te helpen beveiligen.

RÉSUMÉ

Pour s'assurer qu'aucune information ne soit involontairement divulguée par l'intermédiaire d'un canal auxiliaire pendant l'exécution d'opérations cryptographiques, la sécurité physique d'un appareil doit être évaluée. De nos jours, une analyse de sécurité doit prouver une résistance non seulement contre les attaques par canal auxiliaire (SCA) traditionnelles (par exemple, analyse de consommation (DPA)) impliquant une analyse statistique classique, mais aussi contre les attaques avec apprentissage automatique et d'apprentissage profond. Si elles ne sont pas protégées contre ces attaques, les implémentations cryptographiques symétriques et à clé publique peuvent être menacées.

Alors que les attaques SCA traditionnelles s'appuient sur l'expertise d'un cryptanalyste pour extraire les caractéristiques des fuites d'une ou plusieurs traces et analysent leurs observations par des méthodes statistiques pour récupérer la clé secrète. Les attaques par canal auxiliaire basées sur l'apprentissage profond (DLSCA) apportent une nouvelle perspective dans ce domaine. Les attaques DLSCA reposent sur l'automatisation de l'extraction des caractéristiques à l'aide d'un algorithme spécifique à une tâche. Pour la plupart des attaques DLSCA, un expert est toujours nécessaire, mais son travail est déplacé vers l'entraînement de cet algorithme. Parmi les différentes architectures d'apprentissage profond, les plus utilisées dans la DLSCA sont le perceptron multicouche (MLP) et les réseaux neuronaux convolutifs (CNN). Ces deux méthodes de réseaux neuronaux (NN) entraînés pour trouver des modèles dans un ensemble de données collectées de traces de canaux auxiliaires qui permettent de récupérer la clé secrète à condition que leurs hyperparamètres soient correctement réglés et que le processus d'entraînement soit suffisamment efficace.

Cette thèse étudie l'utilisation de l'apprentissage profond dans l'analyse des canaux auxiliaires de la cryptographie symétrique et à clé publique et d'autres applications de l'analyse des canaux auxiliaires. Nous examinons l'application de DLSCA pour les implémentations d'AES et d'ASCON en cryptographie symétrique et d'EdDSA en cryptographie à clé publique. Nous explorons également l'utilisation de l'apprentissage profond pour améliorer les attaques de type TEMPEST et l'utilisation d'attaques par canal auxiliaire pour l'ingénierie inverse des réseaux neuronaux.

Les principales contributions de cette thèse sont les suivantes. Premièrement, nous montrons les performances que peut atteindre un MLP sur un ensemble de

données d'une implémentation AES protégée par une contre-mesure de masquage. Nous démontrons que le MLP peut vaincre la contre-mesure de masquage et récupérer la clé secrète avec un taux de réussite élevé pour de nombreuses configurations d'hyperparamètres et de models de consommation intermédiaires et même avec très peu de paramètres.

Deuxièmement, nous présentons une application de CNN dans l'analyse des canaux auxiliaires de l'algorithme de cryptographie légère ASCON sur un microcontrôleur 32 bits. Nous démontrons que l'implémentation de référence est vulnérable aux attaques DLSCA et que la même attaque peut être appliquée à une implémentation masquée, mais ne peut pas récupérer complètement la clé secrète.

Troisièmement, nous proposons une attaque à trace unique sur la clé éphémère d'EdDSA sur la courbe elliptique 25519. Nous montrons que l'attaque peut récupérer la clé secrète avec une seule exécution d'une implémentation sur un microcontrôleur 32 bits. Cette attaque est basée sur un CNN, et nous démontrons que, parmi les autres méthodes de profilage explorées, le CNN est le plus efficace pour cette attaque. En outre, nous systématisons cette attaque et montrons qu'elle peut être appliquée à une cible différente et mettons en œuvre des contre-mesures contre l'analyse des canaux auxiliaires.

Enfin, nous démontrons l'utilisation de l'analyse par canal auxiliaire et de l'apprentissage profond dans des applications autres que les implémentations cryptographiques. Nous présentons une méthodologie pour évaluer les attaques TEMPEST à l'aide de l'apprentissage profond. Nous nous concentrons sur l'analyse des émanations électromagnétiques des appareils mobiles sans ligne de vue directe pour construire un banc d'essai avec une configuration standard qui peut être utilisée pour tester différents modèles d'attaquants. Une deuxième application est l'utilisation de l'analyse des canaux auxiliaires pour l'ingénierie inverse des réseaux neuronaux sur GPU. Nous montrons que l'analyse par canal auxiliaire des émanations électromagnétiques d'un GPU peut être utilisée pour récupérer plusieurs hyperparamètres d'un réseau neuronal pendant sa phase d'inférence.

Notre principal objectif de recherche est d'appliquer l'apprentissage profond à l'analyse des canaux auxiliaires afin de développer de nouvelles attaques pour les implémentations existantes et leurs contre-mesures, et nous pensons que cette thèse est un pas dans cette direction en ce qui concerne les contributions susmentionnées. Nous pensons également que la lecture de cette thèse mettra en lumière le potentiel de l'apprentissage profond dans l'analyse des canaux auxiliaires et inspirera de futures recherches dans ce domaine pour sécuriser l'électronique de demain.

LIST OF PUBLICATIONS

Publications that are the basis of this thesis:

[CW] Lukasz Chmielewski and Léo Weissbart. "On Reverse Engineering Neural Network Implementation on GPU." In: *Applied Cryptography and Network Security Workshops - ACNS 2021, Kamakura, Japan, June 21-24, 2021, Proceedings*. Vol. 12809, pp. 96–113.

[Liu+] Zhuoran Liu, Niels Samwel, Léo Weissbart, Zhengyu Zhao, Dirk Lauret, Lejla Batina, and Martha A. Larson. "Screen Gleaning: A Screen Reading TEMPEST Attack on Mobile Devices Exploiting an Electromagnetic Side Channel." In: *28th Annual Network and Distributed System Security Symposium, NDSS 2021, virtually, February 21-25, 2021*.

[Wei] Léo Weissbart. "Performance Analysis of Multilayer Perceptron in Profiling Side-Channel Analysis." In: *Applied Cryptography and Network Security Workshops - ACNS, Rome, Italy, October 19-22, 2020, Proceedings*. Vol. 12418, pp. 198–216.

[Wei+20] Léo Weissbart, Lukasz Chmielewski, Stjepan Picek, and Lejla Batina. "Systematic Side-Channel Analysis of Curve25519 with Machine Learning." In: *J. Hardw. Syst. Secur.* 4.4 (2020), pp. 314–328.

[WP23] Léo Weissbart and Stjepan Picek. *Lightweight but Not Easy: Side-channel Analysis of the Ascon Authenticated Cipher on a 32-bit Microcontroller*. Cryptology ePrint Archive, Paper 2023/1598. https://eprint.iacr.org/2023/1598. 2023.

[WPB] Léo Weissbart, Stjepan Picek, and Lejla Batina. "One Trace Is All It Takes: Machine Learning-Based Side-Channel Attack on EdDSA." In: *Security, Privacy, and Applied Cryptography Engineering - 9th International Conference, SPACE 2019, Gandhinagar, India, December 3-7, 2019, Proceedings*. Vol. 11947, pp. 86–105.

Other publications:

[Ami+] Parisa Amiri-Eliasi, Silvia Mella, Léo Weissbart, Lejla Batina, and Stjepan Picek. "Xoodyak Under SCA Siege." In: *27th International Symposium on Design & Diagnostics of Electronic Circuits & Systems, DDECS 2024, Kielce, Poland, April 3-5, 2024*, pp. 61–66.

[EBW24] Trevor Yap Hong Eng, Shivam Bhasin, and Léo Weissbart. "Train Wisely: Multifidelity Bayesian Optimization Hyperparameter Tuning in Side-Channel Analysis." In: *IACR Cryptol. ePrint Arch.* (2024), p. 170.

[Hor+] Péter Horváth, Lukasz Chmielewski, Léo Weissbart, Lejla Batina, and Yuval Yarom. "CNN Architecture Extraction on Edge GPU." In: *Applied Cryptography and Network Security Workshops - ACNS 2024, Abu Dhabi, United Arab Emirates, March 5-8, 2024, Proceedings, Part I*. Vol. 14586, pp. 158–175.

[Rez+] Azade Rezaeezade, Abraham Basurto-Becerra, Léo Weissbart, and Guilherme Perin. "One for All, All for Ascon: Ensemble-Based Deep Learning Side-Channel Analysis." In: *Applied Cryptography and Network Security Workshops - ACNS 2024, Abu Dhabi, United Arab Emirates, March 5-8, 2024, Proceedings, Part I*. Vol. 14586, pp. 139–157.

[SWB] Gabriele Serafini, Léo Weissbart, and Lejla Batina. "Everything All at Once: Deep Learning Side-Channel Analysis Optimization Framework." In: *Applied Cryptography and Network Security Workshops - ACNS 2024, Abu Dhabi, United Arab Emirates, March 5-8, 2024, Proceedings, Part I*. Vol. 14586, pp. 195–212.

[Uet+a] Yoshinori Uetake, Akihiro Sanada, Takuya Kusaka, Yasuyuki Nogami, Léo Weissbart, and Sylvain Duquesne. "Side-Channel Attack using Order 4 Element against Curve25519 on ATmega328P." In: *International Symposium on Information Theory and Its Applications, ISITA 2018, Singapore, October 28-31, 2018*, pp. 618–622.

[Uet+b] Yoshinori Uetake, Keiji Yoshimoto, Yuta Kodera, Léo Weissbart, Takuya Kusaka, and Yasuyuki Nogami. "A Side-Channel Attack Using Order 8 Rational Points against Curve25519 on an 8-Bit Microcontroller." In: *International Symposium on Computing and Net-*

working, CANDAR 2019, Nagasaki, Japan, November 25-28, 2019, pp. 225–231.

[Wu+23] Lichao Wu, Léo Weissbart, Marina Krcek, Huimin Li, Guilherme Perin, Lejla Batina, and Stjepan Picek. "Label Correlation in Deep Learning-Based Side-Channel Analysis." In: *IEEE Trans. Inf. Forensics Secur.* 18 (2023), pp. 3849–3861.

ABOUT THE AUTHOR

Léo Weissbart was born in Colmar, France on June 15 1994. He obtained his Electrical Engineering degree from Grenoble INP - Esisar, Valence, France in 2018.

In 2018, he started a Ph.D. candidate jointly in the Digital Security group supervised by Prof.dr. L. Batina and in the cybersecurity research group at the Delft University of Technology, supervised by Prof.Dr.ir R.L. Lagendijk and Dr. S. Picek. His main research interests are cryptographic implementation attacks and deep learning. During his Ph.D., he worked on improving implementation attacks with artificial intelligence. He presented his works in multiple international conferences. In the meantime, he reviewed several papers from flagship security journals and conferences, given talks, and also supervised multiple master students.