SORBONNE UNIVERSITÉ

ÉCOLE DOCTORALE N° 130

INFORMATIQUE, TÉLÉCOMMUNICATIONS, ÉLECTRONIQUE DE PARIS

CEA GRENOBLE - LETI

LABORATOIRE D'INFORMATIQUE DE PARIS 6

TOWARDS A BETTER COMPREHENSION OF DEEP LEARNING FOR SIDE-CHANNEL ANALYSIS

par Loïc MASURE

Thèse de doctorat d' Informatique

Dirigée par Emmanuel PROUFF et Cécile DUMAS

Présentée et soutenue publiquement le 18 décembre 2020

Devant un jury composé de:

Pr. Hichem SAHBI,	CNRS / Sorbonne Université	Président du jury
Pr. François-Xavier STANDAERT,	Université Catholique de Louvain	Rapporteur
Pr. Lilian BOSSUET,	Université Jean-Monnet, Saint-Étienne	Rapporteur
Dr. Vanessa VITSE,	Université Grenoble Alpes	Examinatrice
Dr. Annelie HEUSER,	CNRS	Examinatrice
Dr. Benoît GÉRARD,	DGA	Membre invité
Dr. Emmanuel PROUFF,	ANSSI	Directeur de thèse
Dr. Cécile DUMAS,	CEA - LETI	Encadrante, membre invitée

Remerciements

Je voudrais en premier lieu remercier François-Xavier STANDAERT et Lilian BOSSUET pour avoir accepté d'être les rapporteurs de ce manuscrit, et pour leurs retours très instructifs. Je remercie également Vanessa VITSE et Hichem SAHBI, qui en plus d'avoir accepté de faire partie de mon jury, ont participé à mon comité de suivi de thèse ces deux dernières années. Leurs retours en cours de route m'ont aidé dans l'orientation et la planification de mes travaux. Je suis reconnaissant à Annelie HEUSER et Benoît GÉRARD d'avoir également accepté de faire partie de mon jury. En outre, j'aimerais les remercier tous les deux pour leur implication indirecte dans cette thèse : Annelie en qualité de *sheperd* pour le papier à CHES 2020; et Benoît pour nos échanges à la conférence C&ESAR 2018, ainsi que pour son invitation au séminaire SEMSECUELEC à Rennes en Janvier 2020, du temps où les présentations orales ne se tenaient pas encore sur Zoom.

Cette thèse n'aurait pas pu voir le jour sans Cécile DUMAS, mon encadrante. Cécile m'a fait confiance dès le début, alors que j'étais presque profane en cryptographie. C'était un pari risqué, mais sa bienveillance et son encadrement au quotidien m'ont très vite rassuré. Alors pour toutes nos grandes discussions au cours de cette thèse, pour avoir moult fois relu mes écrits, des fois jusque très tard dans la nuit (ou tôt le matin), et même le jour de ton anniversaire; pour la transmission de tout ton savoir et ton savoir-faire, je t'adresse un immense merci et te témoigne ma reconnaissance éternelle.

De même, cette thèse n'aurait pas eu la même envergure sans Emmanuel PROUFF, mon directeur de thèse. Diriger une thèse à distance me semblait un grand défi *a priori*. Néanmoins, ces trois années ont démontré que cet obstacle pouvait être très bien surmonté et n'enlevait rien à la qualité de l'encadrement, notamment grâce à ta disponibilité, ta pédagogie, et ta vaste connaissance de la littérature scientifique dans la discipline. À cela, j'ajouterais une immense gratitude pour la finesse de ta relecture de tous mes travaux. Tes retours et conseils sur la rédaction/construction d'un papier m'ont donné le plaisir d'écrire ; paradoxe pour le non-littéraire que je suis.

Outre mes deux encadrants, je suis reconnaissant envers deux personnes qui m'ont également servi de mentor au cours de cette thèse. Tout d'abord Eleonora, qui m'a prodigué ses conseils et avis à chaque étape de la thèse au CESTI. Cette thèse étant le prolongement de la sienne, la lecture de ses travaux et de son manuscrit ont été des références incontournables pour moi. Ensuite, Pierre-Alain, pour nos conversations ponctuées de conseils techniques. Tes travaux et ton expertise dans le domaine de l'apprentissage m'ont souvent apporté une nouvelle perspective lorsque j'étais confronté à des problèmes techniques.

Pour faire une bonne thèse, un bon encadrement est une condition nécessaire, mais pas suffisante. Il faut aussi un bon environnement de travail. C'est à mettre au crédit de l'ensemble du CESTI, le laboratoire dans lequel j'ai passé ces trois dernières années, et en premier lieu de Anne, qui en a pris les rênes au moment-même où j'intégrais l'équipe. En bonne cheffe d'orchestre, elle a su tout mettre en oeuvre pour s'assurer du bien-être des thésards. Je lui suis très reconnaissant de m'avoir accordé sa confiance à en juger par les nombreuses fois où j'ai été "désigné volontaire" pour représenter le laboratoire lors de présentations orales.

Un grand merci aux doctorants du labo, mes compagnons d'infortune qui m'ont rejoint progressivement au fil des années, j'ai nommé Vincent (a.k.a. le crypto-trader), Gabriel (a.k.a. la machine de Boltzmann), et plus récemment Raphaël. J'espère que vous vous remet-

trez de mon déménagement, même si je ne vais qu'au bout du couloir.

Merci à Vincent, Laurent et Thomas du CCRC[1], pour toutes nos sorties vélos sur les hauteurs de Chartreuse ou à l'assaut des plus hauts cols des Alpes ; ponctuées de débats dont le niveau n'avait souvent d'égal que le dénivelé parcouru.

Merci à Marie, ange gardien du labo, toujours présente quand on a besoin d'elle, que ce soit pour réaligner des traces, trouver un banc de mesures disponible, ou payer sa tournée à la Nat'. Merci à Claire et Benoît, pour les soirées arrosées dans votre repaire du Vercors, et pour m'avoir sorti d'une bonne fringale l'été dernier : je vous dois encore quelques biscuits ...Merci à tous les autres membres du CESTI, actuels ou passés : Jean-Christophe, Damien, David, Julien, Guillaume, Elisabeth, Frédéric, Antton, Olivier ($\times 3$), Philippe ($\times 3$), Roland, Marielle, Nicolas, Véronique, Jessy, Stéphanie, Emrick, Yann, Aurélien, Charles, Dorian (félicitations pour ton bébé). Je n'oublie pas le LSOSP (parce qu'on les aime quand-même) : je remercie en particulier Antoine, Maxime, Valence ainsi qu'Alexis pour nos échanges sur nos travaux respectifs. De même, j'adresse mes remerciements à Bruno CHARRAT et Assia TRIA qui, même de loin, ont su suivre l'évolution de mes travaux avec bienveillance; aux assistantes du DSYS, Aurélie, Majda, Sandrine et Virginie, pour leur aide et leur réactivité; au service RH du département avec Franck pour m'avoir grandement aidé à surmonter quelques difficultés administratives lors de l'inscription en thèse, et plus récemment Sophie.

Grâce à certains cités précédemment, j'ai pu élargir mon horizon de travail au delà simplement du CESTI, et je tiens à exprimer ma gratitude à ceux qui ont collaboré avec moi sur différents projets. Tout d'abord, Damien et Nicolas, du LIALP, avec qui nous avons travaillé – et continuons – pour le projet CLAPS, qui m'a permis de rythmer mon confinement. Ensuite, Rémi STRULLU de l'ANSSI, pour sa collaboration sur le projet ASCAD-v2, matérialisée par nos nombreux échanges de mails et appels téléphoniques. J'espère que cela ouvrira de nouveaux horizons de recherche, à nous et aux autres membres de la communauté ML / SCA.

Je tiens à remercier toutes les personnes avec qui j'ai pu échanger lors des divers séminaires / conférences auxquels (du temps où c'était physiquement possible). Je pense en particulier à Mathieu CARBONE, Élie BURZSTEIN, Rémi AUDEBERT, Yanis LINGE, Houssem MAGHREBI, et tous les autres que j'ai rencontrés à Amsterdam, Darmstadt ou Gardanne. Leurs retours m'ont été précieux pour orienter mes premières contributions au domaine.

Merci à ceux que j'ai côtoyés à GEM : Angèle pour m'avoir incité à donner des cours en école de commerce, Mustapha et Barthelemy pour la confiance qu'ils m'ont accordée alors que l'enseignement était une toute nouvelle expérience pour moi, et qu'ils m'ont renouvelée en compagnie d'Alain plus récemment.

Viennent alors des remerciements plus personnels, mais non moins importants dans la réalisation de cette thèse. Mes amis de prépa à Lyon, en particulier Clémentine et Aurélien qui ont commencé leur thèse au CEA en même temps que moi. Cela m'a énormément rassuré de pouvoir discuter de nos tracas respectifs sur la thèse. Courage à Théo, Gabriel, Romain, et plus récemment Bastien qui ont choisi cette voie.

Il est sans doute vexé de ne pas avoir été cité précédemment parmi, mais c'est parce qu'il un mérite remerciement à lui tout seul pour son soutien psychologique et sa finesse qui le caractérisent tant : j'ai nommé Michel. Mon moral ces dernières années aurait été beaucoup plus erratique sans nos innombrables appels Skype transatlantiques. Merci à Hugo pour nos escapades à Montréal, Grenoble, et Barcelone, qui m'ont permis de souffler un peu dans la frénésie de la thèse. Merci à tous les autres membres de *la secte*: Mathilde pour m'avoir fait visiter Darmstadt à l'occasion de COSADE, Tiphaine, Oksana et Alex, Pierre, Anthéa, Léa. Vous avez tous, à un moment ou à un autre, contribué à mon épanouissement lors de ces deux années à Lyon et celles qui ont suivies, et je vous en suis très reconnaissant.

Merci à mes amis Grenoblois de toujours qui ont contribué à me faire penser à autre chose que le travail : Thibaut, Chloé, Nina, Fabien, Alexis, Hugo, Fanny, Julie, Anaïs, David,

[1]Club Cycliste des Retraités du CESTI.

Constance, JB, Lætitia. J'espère qu'on pourra fêter ça très bientôt, comme au bon vieux temps ! Merci à Edo et Miguel, pour les super sorties en montagne, en ski de rando ou à VTT, et qui, malgré les multiples plans foireux dans lesquels je les ai emmenés, me sont restés fidèles.

Enfin, je remercie l'ensemble de ma famille pour son soutien indéfectible de toujours. Ma grand-mère, pour l'immense soutien logistique tout au long de cet été si particulier à Salon-de-Provence lorsque j'écrivais ce manuscrit, en dépit des entraînements de la patrouille de France, de jour comme de nuit. Merci à mes grands-parents de Sully-sur-Loire, qui m'ont aussi hébergé plusieurs fois durant cette thèse lorsque j'étais de passage sur Paris. Je réserve mes tout derniers remerciements – parce que ce sont les plus importants – à mes parents Pierre & Solen pour l'immense soutien tout au long de mes études et plus particulièrement ces derniers mois qui n'étaient pas de tout repos ; ainsi que mes trois petites soeurs, Pauline, Claire & Anne-Lise pour toutes ces années à me supporter, surtout durant ce confinement.

À mon grand-père.

Contents

Chapter 1

Context, Objectives and Contributions

Contents

1.1 Frame of the Thesis

1.1.1 A World Full of Embedded Cryptography

Research in cryptology has always benefited the competition between its two sides, namely cryptography and cryptanalysis. Depending on the epochs, one of the two sides may have taken the advantage on the other. Some important events in History, such as wars and conflicts, have been indirectly impacted by this latent rivalry [Sin99]. Yet, the past forty years have seen the emergence of a period in which cryptography has taken the advantage. Along with other external factors, this has implied a pervasiveness of cryptography, shaping our mordern lives. This trend may be explained by three factors that we detail hereafter:

1. The emergence of *trust* in cryptography.
2. The digitization of our modern world.
3. The advent of standardization in cryptographic protocols.

The Emergence of Trusted Primitives. Crypto-systems may be seen as the set of cryptographic primitives which, combined to each other, provide the security functionalities required by the user, such as confidentiality, integrity, authenticity, non-repudiation. As such, the security and reliability of the whole crypto-system is grounded on those of the primitives. The quest of perfect primitives is so far a mission that motivates most of the research work in cryptology.

Hopefully, over the last decades, the cryptology community has seen the emergence of *secure* primitives, which currently allows to bring enough trust in the crypto-systems. The notion of security of such primitives may be defined in two ways, that we recall hereafter.

- **Provably secure**: A primitive is said to be *provably secure* if and only if (i.f.f.) defeating this primitive is strictly equivalent to solving a known mathematical problem, so that any attacker willing to defeat the primitive would have to solve this mathematical problem. If the latter is known to be intractable enough, then this brings strong guarantees in the security of the primitive. As examples, the Diffie-Hellman key exchange protocol relies on the *discrete logarithm* problem [DH76], while the Rivest-Shamir-Adleman (RSA) protocol used in asymmetric cryptography relies either on the factorization of large prime numbers [RSA78] or on the so-called RSA problem [BV98]. Both problems are known to be intractable on classical computers.

- **Reputed secure**: A primitive is said to be *reputed secure* i.f.f. no efficient attack has been emphasized over a long period. Although this does not rigorously prove the security of the primitive, the fact that many people from the research community in cryptology have attempted to find a vulnerability without success gives strong evidences of the soundness of the given primitive. This is particularly the case of the Advanced Encryption Standard (AES), that we will further describe in Subsection 1.1.2: the best known attack requires a complexity that is roughly of the same order of magnitude as a brute-force attack enumerating the 2^{128} possible keys, which makes a practical attack intractable [BKR11].

Although the security property of the mentioned primitives may vary in the next years, in the case where new attack paths might be found,[1] it is noticeable that the mentioned primitives have earned their reputation over a quite long period of time, thereby reinforcing trust in them.

[1] Actually quantum computers would be able to efficiently resolve prime factorization and discrete logarithm [Sho94]. Hopefully, quantum computers are not up to date yet, which would let enough time to find *post-quantum* algorithms resilient to such threat. The National Institute of Standard and Technology (NIST) is currently running a competition to select new post-quantum designs. The interested reader may refer to https://csrc.nist.gov/projects/post-quantum-cryptography.

From (Electro)-Mechanical to Electronic Systems. Crypto-systems propose solutions to secure communications asking some secret keys for computations on which is ground their security. Keys are represented as long strings from an alphabet specified by the underlying cryptographic primitives. A sound crypto-system must let the parties communicating with each other to be able to securely store and manipulate those keys, without delivering them in clear over insecure channels. However, from a practical perspective, using trusted cryptographic primitives is not enough to spread their use: they must be efficiently usable in any context. Thanks to the progressive replacement during the 20-th century of crypto-systems implemented on mechanical or electro-mechanical devices, by fully electronic devices realizing such operations is nowadays drastically faster [Sin99].

As an example, *smart cards* were historically conceived as a practical solution to such a key storage issue: they consist in small devices that a user can easily carry around with, which not only store secret keys as long strings of bits, but also are able to internally perform cryptographic operations, in such a way that they can be involved in secure communication protocols, that do not require the delivering of secret keys. Smart cards are pocket-sized plastic-made cards equipped with a secure component, which is typically an integrated circuit containing some computational units and some memories. They have been patented by Moreno in 1974 [Mor74] – as a memory device – and Ugon in 1977 [Ugo77] – a a computing device.

Today, about 40 years after its invention, smart cards still have a huge diffusion, both in terms of applicative domains and in terms of quantity of copies. Indeed, they serve as credit or ATM cards, healthy cards, ID cards, public transportation payment cards, fuel cards, identification and access badges, authorization cards for pay television, *etc.* Slightly changing the card support, we find other applications of the same kind of integrated circuits, for example the mobile phone Subscriber Identity Modules (SIMs) and the electronic passports. In terms of quantity, a marketing research found out that in 2014, 8.8 billion smart cards have been sold [ABI], *i.e.*, the same order of magnitude of the global population.

In addition to smart cards, the recent growing and variation of security needs lead to the development and specification of other kinds of secure solutions, for example the Trusted Platform Module (TPM), which is a secure element providing cryptographic functionalities to a motherboard, or completely different solutions based on software layers, which are today in great expansions. An example is provided by the Trusted Execution Environment (TEE), a software environment of the main processor of a smartphone or tablet, designed to assure resistance to software and even hardware threats.

The Standardization of Cryptographic Primitives. The last ingredient contributing to the spread of applications relying on cryptography is the standardization of the protocols and the primitives. Standardization enables to make different crypto-systems compatible, so that any couple of instances (people, device) equipped with a same cryptographic primitive can then securely communicate, without necessarily having to first agree on a communication protocol, since the standardized one is implicitly chosen. Standardization is usually done by a third part, able to endorse the security of the proposed primitive after comprehensively verifying its security. Typically, this is done by institutions or government agencies such as the NIST. In this thesis, we will mainly focus on a standard edited by the latter institution, called AES, presented in Subsection 1.1.2. The economic impact of the adoption of this standard, along with the two other factors we previously described, has been evaluated 2018 to 250 billion dollars at a global scale, according to a NIST report [LFS18].

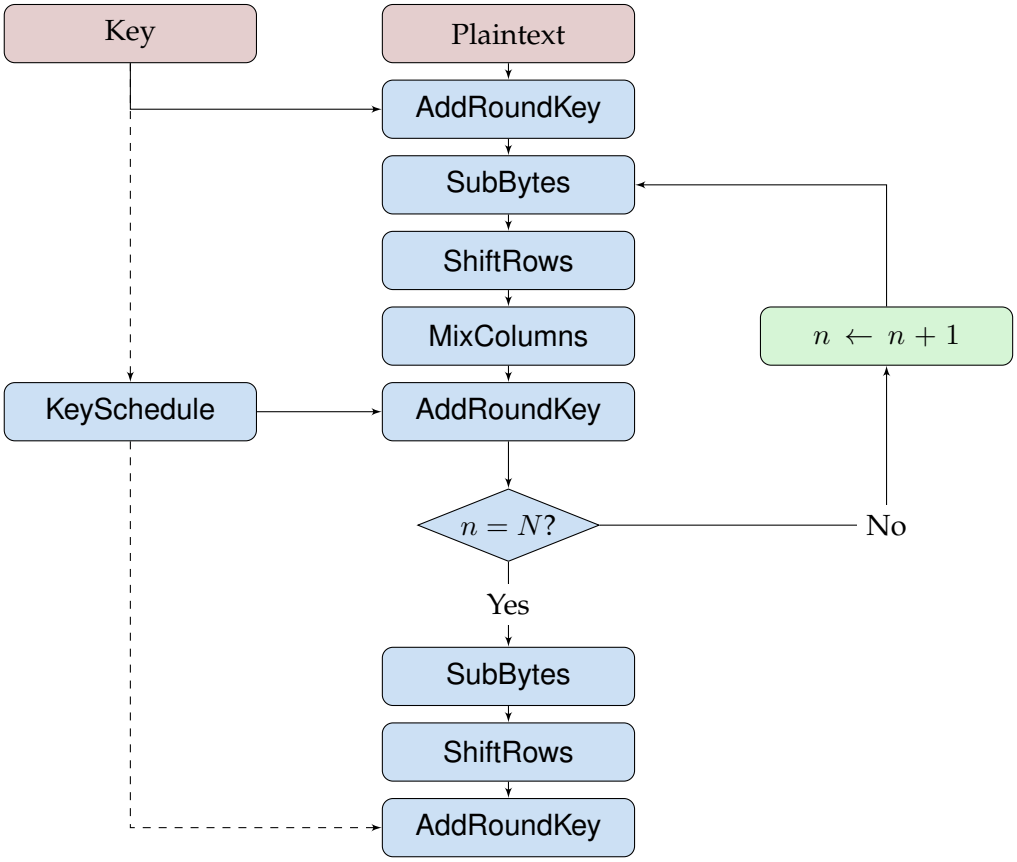


Figure 1.1: The rounds of AES.

1.1.2 The Advanced Encryption Standard (AES)

In this thesis, we will mainly focus on the AES algorithm,[2] though many aspect may be generalized to other algorithms. The AES is a block cipher used for encryption or authentication in embedded devices. Its specifications are established by the Federal Information Processing Standard (FIPS) publication 197 released by the NIST in 2001 [Nat01]. It proceeds blocks of 128 bits by running several *rounds*, as depicted in Figure 1.1, composed of elementary operations, called AddRoundKey, SubBytes, ShiftRows, MixColumns, thanks to a secret key of 128 bits as well.[3] The diffusion is ensured by the ShiftRows and the MixColumns operations, while the confusion is ensured by the AddRoundKey and the SubBytes operations. The two latter operations are involved in the different aspects on which this thesis is dedicated. That is why we detail them in Section 2.4.

The AES cryptographic primitive has been carefully designed to be optimally robust against some classes of attacks such as linear cryptanalysis and differential cryptanalysis. Therefore, it has the advantage to be reputed practically secure against classical cryptanalysis. In the latter threat model, a.k.a. *black-box* attack scenario, an attacker is assumed to know the algorithm (according to the Kerckhoff's principle) and of some inputs and/or outputs. Starting from these data, his goal is to retrieve the secret key. In other words, no internal variable can be observed during the execution. The best known black-box attacks on AES assume to reduce the number of rounds or have a marginal gain compared to a brute force enumeration of the 2^{128} keys, thereby leading to intractable secret key recoveries [BKR11].

Another key argument for its choice as a standard is that any elementary operation can

[2] This algorithm is also known under the name of *Rijndael* after the name of its two creators Vincent Rijmen and Joan Daemen.

[3] There exists AES versions using keys of respectively 192 and 256 bits. In this thesis, we will mainly focus on the 128 bit version of the algorithm, without loss of generality.

be done at the scale of one byte, *i.e.*, 8 bits. This is particularly true for the AddRoundKey and the SubBytes, which are byte-wise operations. This makes it compatible with most of software and hardware architectures, based on at least 8 bits and most of the time 32 for Micro-Controller Unit (MCU) or 64 bits for Central Processing Unit (CPU) nowadays. This compatibility with most of electronic device architectures is one of the cornerstone of its success in modern crypto-systems. However, this success comes with a major drawback in terms of security, despite its robustness against classical cryptanalysis. This drawback does not come directly from the algorithm itself, but rather from its physical implementation, as we will see in the next sections.

1.1.3 The Physical Attacks

The black-box threat model fits well with an adversary having access to the communication channel between two parties. However, this scenario does not fit anymore with modern crypto-systems embedded in electronic devices for one simple reason: the secret key is physically stored in the device, and the adversary may have a physical access to the latter one storing the secret key. As an example, a credit card containing sensitive information useful to proceed banking transactions may be stolen. A physical access to this device is likely to be exploited by a malicious person to proceed fraudulent transactions. Likewise, a payment-TV service may send a smart card every month to its customer on which is implemented a cryptographic primitive using a key. This key generates dedicated information necessary to decrypt a signal sent on a public channel by the service. Provided that the customer guesses the key stored inside the smart card, he may be able to clone it and to sell those clones at the expense of the payment-TV service. The black-box model stipulates that the customer does not know this key. Yet, we can see here that it may have a physical access to the device storing it, which is likely to break the black-box model assumption.

Of course, having a physical access to the device is not sufficient to have a perfect knowledge of of the secret key which is stored inside. Since this is not a public information, it is expected that the design of the implementation considers the key as a private variable, *i.e.*, a variable for which the access (in reading or over-writing) by an external program is denied. This is particularly the case for smart cards, as mentioned in Subsection 1.1.1. Therefore, the attacker must circumvent this constraint by proceeding a so-called *physical* attack, which aims at exploiting the weaknesses of the implementation of a cryptographic primitive, rather than the algorithm itself, by *observing* and/or *interacting* with its physical environment.

The word "observing" refers to *passive* attacks, in which the device runs as expected by its specifications. The attacker only observes its behavior through the acquisition of physical measurements, without provoking any alteration. On the contrary, the word "interacting" refers to *active* attacks, during which a special manipulation is performed, either on the target device or on its physical environment, in order to corrupt the expected behavior of the device.

Physical attacks englobe a wider scope than just attempting to recover a secret key used by a cryptographic primitive: in particular, active attacks are often dedicated to bypass some security measures implemented in a program – not necessarily a cryptographic primitive – embedded on the target device. To this end, an attacker may perturb the implementation with fluctuations of its physical environment, *e.g.* with glitches occurred by power consumption or Electro-Magnetic (EM) emanations, laser pulses on the device, or physical reconfiguration of electronic circuits thanks to a Focused Ion Beamer (F.I.B.). Active attacks are beyond the scope of this thesis: we will only focus on passive attacks, and more precisely attacks where an adversary observes the so-called *side-channels* that we describe in the following section.

1.1.4 The Side-Channel Attacks

It turns out that depending on the nature of its implementation, a cryptographic primitive may also spread key-dependent signals on non-purposed *side channels*, besides the main communication channel on which the ciphertext message is supposed to be broadcast. Those side channels are depicted in Figure 1.2. Inside the electronic device, all the data related to the cryptographic primitive – *i.e.* the plaintexts, the secret key, the ciphertexts and all the intermediate computations – are stored in the memory, loaded through the bus, and manipulated in the CPU registers, under the physical form of electric signals. All those elements are made of gates whose power consumption depends on the binary data they are supposed to store or to drive. Therefore, depending on the processed data, an oscilloscope can notice slight changes in the measured power consumption, so an attacker can monitor this physical measurement to recover some information about the key, and thereby breaking the target device, as shown by Kocher [KJJ99]. A more detailed discussion about this dependency is proposed in Section 3.5.

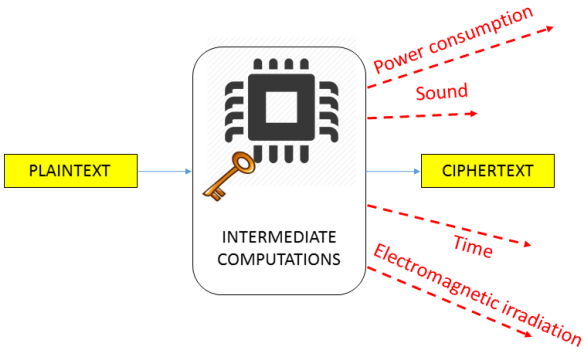


Figure 1.2: The different side channels encountered by an electronic device. Courtesy of Eleonora Cagli.

Likewise, any change of value in the data passed through a given gate would result in a change of current in the gate, resulting in the emission of Electro-Magnetic (EM) radiations, which can be monitored thanks to an EM probe [GMO01, QS01].

Other non-desired channels can be exploited by a malicious person: the intermediate computations processed by the electronic device can emit specific sounds allowing to distinguish secret values [GST14]. Moreover, if until a few years ago it was thought that only small devices, equipped with slow micro-processors and with small-sized architecture, such as smart cards, were vulnerable to this kind of side-channel attacks, the last cited recent work about acoustic emanations, together with other works exploiting electro-magnetic fluctuations, pointed out that much faster and bigger devices, *i.e.* laptops and desktop computers, are vulnerable as well [GPT15, GPPT15, GPPT16]. Finally, the runtime of the implementation of a cryptographic primitive can also carry some sensitive information – *i.e.* depending on secret values, as emphasized by the seminal work of Kocher in 1996 [Koc96]. This is particularly true when the implementation of the cryptographic primitive contains branches whose evaluation depends on sensitive values: if the different branches do not have the same runtime, it is therefore possible to guess which branch has been selected, which in turn provides information on the secret-related variable tested to branch. Recently, in 2018, Kocher *et al.* proposed a timing attack based on modern CPU's architectures, involving low-level optimizations such as *branch prediction* and *speculative execution* [KHF+19]. Those optimization tricks concern nowadays most of the CPUs such as INTEL, AMD, or ARM processors, therefore making the vulnerability pervasive.

1.2 From the Attacks towards the Evaluation

1.2.1 Goals and Stakeholders of the Evaluation and the Certification

The emergence of side-channel attacks as pervasive and credible threats on modern cryptosystems has contributed to the trend from the industrial and institutional stakeholders of assessing and mitigating them, in order to still ensure the reliability on the security of such

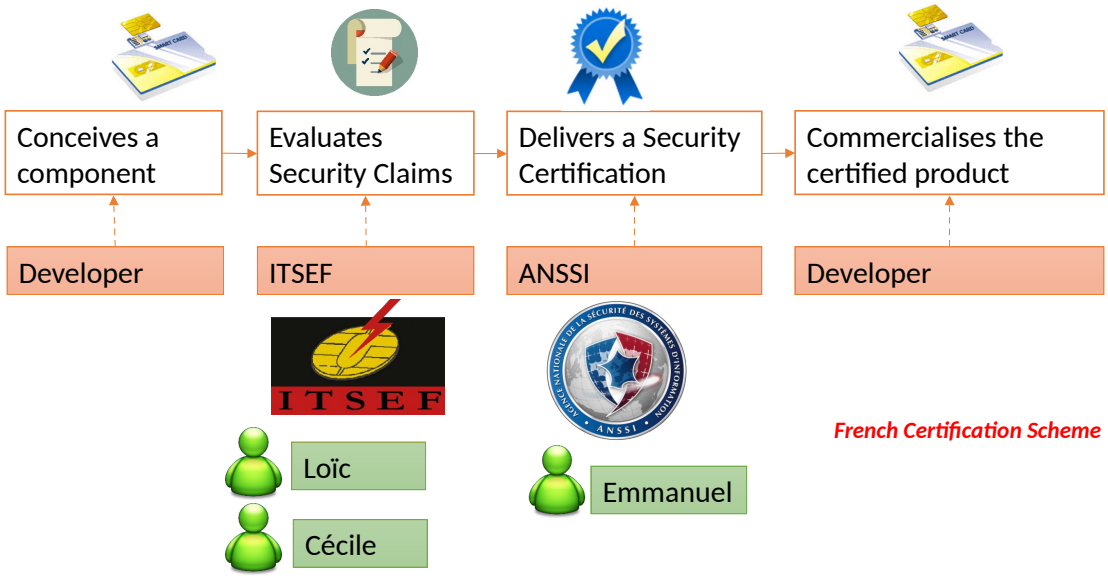


Figure 1.3: The French certification scheme. Inspired from Cagli [Cag18].

crypto-systems. This is concretely materialized by the emergence of *certification* schemes. This is a – sequence of – process(es) that aim at ensuring that the security claims of a given Target of Evaluation (T.O.E.) are indeed verified, up to the point that a *certification body* can endorse those claims by delivering a security certification. The certification scope depends on the levels of the security, and on the scope of the T.O.E.: the latter one can either englobe a whole product – *e.g.* a smart card – or only a specific part of it – *e.g.* its Integrated Circuit (IC). Likewise, the certification is delivered for a given period (*e.g.* 3 years), during which it can still be revised depending on the emergence of new threats compromising the security claims.

The most famous scheme is the Common Criteria for Information Technology Security Evaluation (CC) created in 1999, gathering several national certification bodies around the world. The fact that those different certification bodies use the same scheme contributes to the standardization of the certification of security assessment. Hereafter, we briefly present the different stakeholders of a certification scheme, depicted on Figure 1.3.

- **The Developers** conceive the product for which they ask a security certification. Asking a security certification is not mandatory, but often represents key stakes for the final product, *e.g.* commercial advantage over a similar product. In the certification scheme, the product is referred as the T.O.E.

- **The Evaluators**: When the certification query is claimed by the developer, the latter one ask an evaluation laboratory to assess the security of the T.O.E. An evaluation laboratory is often referred under the name of Information Technology Security Evaluation Facility (ITSEF) – Centre d'Évaluation de la Sécurité des Technologies de l'Information (CESTI) in French. To evaluate the security of the T.O.E. claimed by the developers, the ITSEF verifies the expected functionalities, by inspecting not only the specifications of the T.O.E., the T.O.E. as itself and – depending on the required level of security – the other components of the products which interact with it. Eventually, depending on the whole lifecycle of the T.O.E., some inspections on site – *e.g.* foundries – may be done. If vulnerabilities are identified, the evaluator imagines threat scenarios by building attack paths, in order to assess the required means to succeed the attack, in terms of human, material and financial resources, or technical expertise. If need be, the evaluator realizes himself the attack – or at least a part of it – in order to verify the reliability of its assessment. Based on his investigations, the evaluator produces

an Evaluation Technical Report (ETR), sent to the developers and to the certification body.

- **The Certification Body** delivers the security certification, based on the ETR produced by the ITSEF. Usually, the certification body is a governmental organization, such as the Agence Nationale de la Sécurité des Systèmes d'Information (ANSSI) in France or the Bundesamt für Sicherheit in der Informationstechnik (BSI) in Germany, but it can also be a private organization such as the Europay-Mastercard-Visa Consortium (EMVCo). Sometimes if needed, the certification body can ask the ITSEF for further investigation. Likewise, it can even verify that the evaluation has been correctly conducted by the ITSEF, either by reproducing some parts of the evaluation on its own or by ordering audits into the evaluation laboratory. The certificate is often made public to ensure not only the developers but also the final users about the claimed security of the T.O.E.[4]

1.2.2 The Need of Constant Improvement of the State of the Art

As a starting point of this section, it is interesting to more precisely define what the term "*security*" means, and more particularly to point out the difference with the meaning of the term "*safety*", despite their closeness.

> "The safety describes a machine designed to prevent inadvertent or hazardous operation" [MW20b], *i.e.* "depending on the effect of *unpredictable* and *unanalyzable* forces in determining events" [MW20a].

In other words, this definition assumes that an undesirable event mainly involves randomness. Assessing the safety of a device consists in verifying that the final user, is not likely to make – unintentionnaly – the device having a behavior not expected by its functional specifications.

Security is defined as

> "to relieve from exposure to danger : act to make safe against *adverse* contingencies" [MW20c].

Thus, assessing the security suggests to consider an *adversarial* model threat: if there is a vulnerability in the device, one must assume that such an adversary – a.k.a. the *attacker* – will do all its possible to exploit it, provided it fits with the required means of this attack. On the contrary, what is considered as a vulnerability in the T.O.E. from a security point of view is not necessarily harmful from a safety point of view, as long as the probability to randomly encounter this vulnerability is sufficiently low.

Since the electronic devices embedding security functionalities are usually widely spread and are often used for critical tasks, such as telecommunications, banking transactions, *etc.* it is preferrable for the developer to consider an adversarial threat model, even if it requires to protect a device against potential attacks that are not likely to happen in real life. Hence, one of the goals of developers and evaluators is to succeed in proving that a crypto-system is sound against *any* attacker instantiating a given threat model.

At first sight, the latter task seems untractable, especially if there are infinite ways to instantiate an attacker from a threat model: it becomes impossible to test them all. One way to circumvent this issue is to find a way to sort the different instances of a model threat

[4] In France, security certifications are available on the ANSSI's website: https://www.ssi.gouv.fr/entreprise/produits-certifies/. An equivalent list of certification stamped by the BSI is available at https://www.bsi.bund.de/DE/Themen/ZertifizierungundAnerkennung/Produktzertifizierung/ZertifizierungnachTR/ZertifizierteProdukte/zertifiz_produkte.html?nn=6618104

according to their *efficiency*,[5] in the sense that any attack being successful would necessarily imply that any more efficient attack would succeed. Therefore, if the most efficient instance from a model threat does not succeed in breaking a target, one is guaranteed that any other attack within this threat model would fail too.

As a consequence, it is necessary for an evaluator to always reach the optimal attack, which consists in assessing the worst-case security, in order to provide strong guarantees about the security of the T.O.E. This requires an ITSEF to always know the state-of-the-art attacks, and to be willing to always investigate how to push the limits of these attacks, in order to assess to what extent the security guarantees of a T.O.E. may decrease through the time, as the SCA literature improves at the same time, making attacks of a given threat model more powerful.

1.3 Deep Learning based Attacks

1.3.1 A Recent Emergence in Side-Channel Analysis

DL is a special type of Machine Learning (ML). Historically, they have been created in the 1950's as models for simulating the behavior of simple neurons connected to each other in a brain. A complete description of DL is proposed in Section 4.2. Deep Neural Networks (DNNs) have recently shown impressive performances at some image recognition tasks known to be hard to efficiently solve until the beginning of the 2010's. In particular, the success of the model proposed by Krizhevsky [KSH12] at the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) 2012, definitely paved the way towards spreading the use of DL in many application fields. This also holds for SCA, where DL has also started to be used since 2013, with the seminal works of Martinasek *et al.* [MZ13]. A few years later, Maghrebi *et al.* [MPP16] have shown that DNNs were particularly efficient to break implementations of AES protected with a Boolean secret-sharing – see Subsection 3.7.1 – whereas other types of SCA failed. Likewise, Cagli *et al.* [CDP17] have succeeded in breaking some software and hardware implementations protected with de-synchronization counter-measures. Those two milestones have convinced an important part of people inside the SCA community that this line of work is worth being more deeply investigated, as depicted by Figure 1.4: the number of dedicated papers follows an increasing trend over the past few years, according to several scientific literature databases. The CHES[6] workshop, which is the flagship conference in embedded cryptography, had only one paper dealing with DL-based SCA in 2017, but respectively 5 and 6 papers for the 2019 and 2020 editions, with now a dedicated session on the topic. Likewise the CARDIS conference now includes the term "Deep Learning Analysis" in its topics of interest. Finally, this interest for DL-based SCA has particularly been consecrated by the fact that it is now considered as part of the state-of-the-art attacks by the Joint Interpretation Library (JIL).[7]

1.3.2 The Drawbacks of Deep Learning in Side-Channel Analysis

The successful breakthrough of the DL approach in many application fields gave the opportunity to their respective specialists to draw comparisons with former state-of-the-art ML algorithms. Most of the time, the same criticism emerged from those comparisons: DL appeared as alchemy. Indeed, some intriguing results emphasized that DNNs are more prone to *over-fitting*, a phenomenon where the learning algorithm starts to learn *by heart* in order to improve its performances, although this strategy generalizes poorly for most of the investigated learning problems. Likewise, research has shown that machine learning algorithms

[5] A formal definition of the term "efficiency" will be given in Subsection 3.2.1.

[6] ches.iacr.org

[7] The interested reader may find useful information on the CC website: https://commoncriteriaportal.org/.

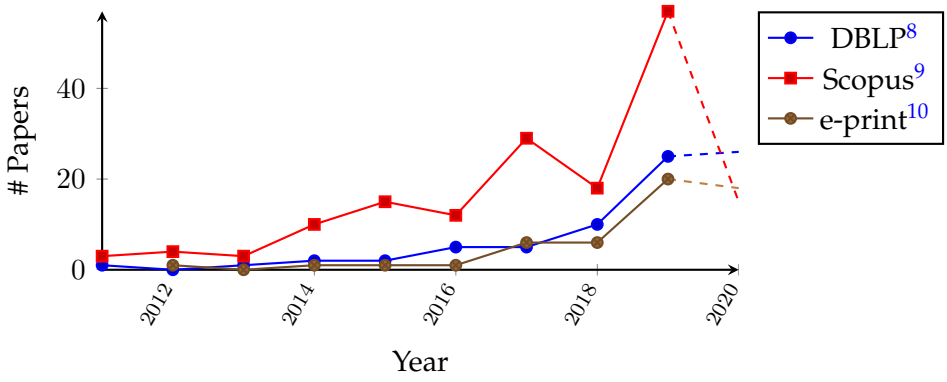


Figure 1.4: Queries to scientific databases, by August 31, 2020.

could be fooled by a malicious person, thus questioning the reliability of such algorithms in a security context. This line of works, entitled Generative Adversarial Networks (GANs) has recently skyrocketed in the DL literature with impressive results [GPAM+14].

Those drawbacks have legitimately found some echoes in the SCA community, where only realizing and assessing an attack is not sufficient to draw exhaustive conclusions for a security evaluation. Therefore, one may question the interest of such an approach in a security evaluation. In particular, the different attack methodologies proposed so far in the literature could be easily interpretable by simple statistical tools, which is not the case of DL-based algorithms which are often seen as black-boxes, so it is hard for the SCA practitioner to interpret and to rely on such results. Moreover, the fact that DL algorithms can be fooled does help to bring trust in a community whose work is especially grounded on trust. It is noticeable yet that the scenarios where the DL algorithms are fooled assume that the latter ones are the target themselves and that the malicious entity have access to some inputs/outputs of the algorithm. Actually, this is the convert situation of ours, in which this is the attacker who is equipped with DL methods, and not the target. Still, this confusion may lead the layman attacker to have misconceptions about the strengths and the weaknesses of DL for SCA.

1.4 Contributions of this Thesis

The observations reported in Subsection 1.3.2 are specifically at the origin of this thesis, whose common thread is to bring trust in DL algorithms by better understanding their behavior and their potential in an SCA context.

I, for one, truly believe that such tools, without necessarily triggering a Copernican revolution of the whole field of embedded cryptography, can still represent a milestone by drastically improving some attacks against targets whose robustness was so far taken for granted, an to incite developers to not rely on some potential weaknesses of such attackers: paradoxically, this might therefore improve the security of embedded electronic devices. For these reasons, it seemed to me that it was worth further investigating this line of works. The contributions presented in this thesis aim at grounding the use of DL in an SCA context, at different steps of an evaluation workflow.

Theoretical Study of Deep Learning in Side-Channel Analysis. In Chapter 5, we revisit the DL-based SCA approach under the theoretical framework of statistical learning. The

[8] https://dblp.uni-trier.de/search?q=side%20channel%20learning

[9] Query "TITLE-ABS-KEY(("learning") AND ("SCA" OR "side-channel") AND ("attacks") AND ("cryptographic" OR "cryptography" OR "crypto")) " on www.scopus.com

[10] Query "learning side channel" at eprint.iacr.org, excluding the Learning With Errors (LWE)-related keywords

latter framework is by the way formally presented in Chapter 4. We analyze how the goal of the evaluator, namely finding the optimal attack in view of assessing the worst-case scenario, is translated into a machine learning problem and to what extent the approach is sound. This leads to confirm that the choice of the loss function has a meaning from an SCA point of view, since the values returned by the loss function can be linked to the efficiency of an attack. With this finding in mind, it is therefore possible to precisely assess the soundness of a software or hardware protection brought to the target device. We have indeed verified that DNNs could efficiently address the key recovery in presence of different counter-measures, such as (high order) secret-sharing or hiding.

The Use of Convolutional Neural Networks in Practice. As an example, we propose in Chapter 6 a case study of a software device protected by a code polymorphism counter-measure, consisting in changing the machine translated from a source file programming a cryptographic primitive. Such a protection implies technical challenges in view of the DL-based attack, as the acquired leakage measurements are of very high – *i.e.* around 10^5 time samples – dimensionality. The DL-based literature being mostly inspired from computer vision, the different algorithms are not adapted anymore to those kind of data. We propose slights modifications to the design of DNNs to circumvent this issue.

Gradient Visualization for General Characterization in Profiling Attacks. Finally, we tackle in Chapter 7 the problem of the interpretability of DNNs to show that such algorithms may not be only seen as black-box models in an SCA context. By analyzing the specific properties of our problem, we are able to propose a simple method, so far known to be sub-optimal in other fields such as computer vision, but efficient in SCA to emphasize the time samples that carry the informative leakage in the measured data, the so-called Points of Interest (P.o.Is). The advantage of this method is that its efficiency to emphasize those points works as long as the DNNs on which it is applied is able to succeed an attack. Since we would have emphasized in Chapter 5 that DNNs-based attacks are sound against mostly all the protected implementations, this method could potentially be applied on any evaluation of implementation. Compared to other methods, the characterization can be done on each acquisition separately. The diagnosis that an evaluator can build based on this method may enable to identify the vulnerabilities in the source code, in order to mitigate their effect on potential attacks.

All together, those contributions propose some improvements of DL-based attacks at several steps of an evaluation.

Chapter 2

Preliminaries

Contents

2.1 Notations and Conventions

We use calligraphic letters as $\mathcal{X}$ to denote sets. If $\mathcal{X}$ is finite, the number of elements in $\mathcal{X}$ a.k.a. its cardinality is denoted by $|\mathcal{X}|$. We use bold notations $\mathbf{x}$ to denote vectors of elements from a set $\mathcal{X}$.

Throughout this thesis, the finite set $\mathcal{Z} = \{s_1, \dots, s_N\}$ will be often considered: it will always denote the possible values for a *sensitive* variable Z. We will denote by s a generic element of $\mathcal{Z}$, in contexts in which specifying its index is unnecessary.

When the vectors' orientation minds, they are understood as column vectors. The i-th entry of a vector $\mathbf{x}$ is denoted by $\mathbf{x}[i]$, while the transposed of a vector $\mathbf{x}$ is denoted as $\mathbf{x}^\intercal$. We will use the transposed mark to refer to row vectors $\mathbf{x}^\intercal$.

In this thesis, $\mathbb{Z}_p$ denotes the set of relative integers modulo p, and $\mathbb{R}_+$ denotes the set of non-negative integers.

The symbol $\triangleq$ denotes an equality by definition. The range of integers from a to b included is denoted by $[\![a, b]\!]$. If $\mathcal{D}$ is a logical statement, we define the *characteristic* function as:

$$\mathbf{1}_{\mathcal{D}} = \begin{cases} 1 \text{ i.f.f. } \mathcal{D} \text{ is true} \\ 0 \text{ otherwise} \end{cases} . \tag{2.1}$$

Finally, terms in blue are defined in the glossary at the end of this thesis.

2.2 Recalls in Probability and Statistics

2.2.1 Probability

We consider a probabilistic structure $(\Omega, \mathcal{A}, \Pr)$, where $\mathcal{A}$ denotes the σ-algebra of the set of all possible events Ω. Formally, the *probability measure* $\Pr$ is a mapping $\mathcal{A} \to [0, 1]$ such that:

1. the probability of all the possible events is 1, *i.e.*, $\Pr(\Omega) = 1$;
2. the probability of the countable union of several mutually exclusive events $(A_n)_{n\in\mathbb{N}}$ equals the sum of their probabilities:

$$\forall i, j \in \mathbb{N}, \text{if } A_i \cap A_j = \emptyset \text{ then } \Pr(A_i \cup A_j) = \Pr(A_i) + \Pr(A_j) . \tag{2.2}$$

Random Variables. We call *random variable* (resp. random *vector*), denoted by upper-case letters X (resp. bold letters **X**), any measurable map from $(\Omega, \mathcal{A})$ to a σ-algebra $\mathcal{X} \subset \mathbb{R}$ (resp. $\mathbb{R}^d, d \in \mathbb{N}^*$). The probability of a random variable X taking value in a subset $\mathcal{U} \subset \mathcal{X}$ is denoted by $\Pr(\mathrm{X} \in \mathcal{U})$. When $\mathcal{U}$ is reduced to a singleton $\mathcal{U} = \{x\}$ the same probability is denoted by $\Pr(\mathrm{X} = x)$. If $\mathcal{X}$ is a finite or countable subset of $\mathbb{R}$, X is called *discrete* and the mapping $\mathcal{U} \mapsto \Pr(\mathrm{X} \in \mathcal{U})$, called Probability Mass Function (p.m.f.), verifies $\Pr(\mathrm{X} \in \mathcal{U}) = \sum_{x\in\mathcal{U}} \Pr(\mathrm{X} = x)$. Therefore the p.m.f. can be fully defined by a $|\mathcal{X}|$-dimensional vector whose entries are non-negative reals that sum to one. The set of every p.m.f. is denoted by $\mathcal{P}(\mathcal{X})$. If $\mathcal{X}$ is not finite nor countable, X is said *continuous* and the mapping $\mathcal{U} \mapsto \Pr(\mathrm{X} \in \mathcal{U})$ is fully defined by the Probability Density Function (p.d.f.) $x \in \mathcal{X} \mapsto f_{\mathrm{X}}(x) \in \mathbb{R}_+$ that verifies $\Pr(\mathrm{X} \in \mathcal{U}) = \int_{x\in\mathcal{U}} f_{\mathrm{X}}(x)\, dx$ [Kle13, Thm. 1.104].

Couple of Random Variables. When two variables X and Y are considered, their *joint* probability is denoted by $\Pr(\mathrm{X} = x, \mathrm{Y} = y)$. We call *marginal* probability the following quantity: $\Pr(\mathrm{X} = x) = \int_{y\in\mathcal{Y}} \Pr(\mathrm{X} = x, \mathrm{Y} = y)\, dy$. The *conditional* probability of X assuming the value x given an outcome y for Y is denoted by $\Pr(\mathrm{X} = x \mid \mathrm{Y} = y)$. By definition, we have $\Pr\left(\mathrm{X} \triangleq x, \mathrm{Y} = y\right) = \Pr(\mathrm{X} = x \mid \mathrm{Y} = y) \Pr(\mathrm{Y} = y)$. In particular, i.f.f. $\Pr(\mathrm{X} = x \mid \mathrm{Y} = y) = \Pr(\mathrm{X} = x)$ we say that X and Y are *independent* and the joint probability is then the product of the two *marginal* probabilities. The mapping $y \mapsto \Pr(\mathrm{X} = x \mid \mathrm{Y} = y)$ is denoted by $\Pr(\mathrm{X} = x \mid \mathrm{Y})$.

Moments of a Random Variable. The symbol $\mathbb{E}\left[\phi(\mathrm{X})\right]$, or equivalently $\mathop{\mathbb{E}}_{\mathrm{X}}\left[\phi(\mathrm{X})\right]$, denotes the expected value of a function ϕ of the random variable X, under the distribution of X. We recall that in the continuous case $\mathop{\mathbb{E}}_{\mathrm{X}}\left[\phi(\mathrm{X})\right] \triangleq \int_{x\in\mathcal{X}} \phi(x) f_{\mathrm{X}}(x)\, dx$. Likewise, symbols $\mathrm{Var}\left(\mathrm{X}\right)$ and $\mathop{\mathrm{Var}}_{\mathrm{X}}\left(\mathrm{X}\right)$ denote the variance of X. We recall that $\mathrm{Var}\left(\mathrm{X}\right) \triangleq \mathbb{E}\left[\left(\mathrm{X}-\mathbb{E}\left[\mathrm{X}\right]\right)^2\right]$. We note $\mathrm{Cov}\left(\mathrm{X},\mathrm{Y}\right) = \mathbb{E}\left[\left(\mathrm{X}-\mathbb{E}\left[\mathrm{X}\right]\right)\left(\mathrm{Y}-\mathbb{E}\left[\mathrm{Y}\right]\right)\right]$ the *covariance* of two random variables X, Y. It is worth emphasizing that $\mathrm{Var}\left(\mathrm{X}\right) = \mathrm{Cov}\left(\mathrm{X},\mathrm{X}\right)$. The mapping $y \mapsto \mathop{\mathbb{E}}_{\mathrm{X}\mid\mathrm{Y}=y}\left[\mathrm{X}\right]$ is called *conditional expected value* and is denoted by $\mathbb{E}\left[\mathrm{X} \mid \mathrm{Y}\right]$.

We now recall some useful probability results.

Total Probabilities. A consequence of Equation 2.2 and the definition of a conditional probability is the *total probabilities* formula. Given two random variables $\mathrm{X} \in \mathcal{X}$, $\mathrm{Y} \in \mathcal{Y}$, and a subset $\mathcal{U} \subset \mathcal{X}$ we have:

$$\begin{aligned} \Pr\left(\mathrm{X}\in\mathcal{U}\right) &= \sum_{y\in\mathcal{Y}} \Pr\left(\mathrm{X}\in\mathcal{U},\mathrm{Y}=y\right) && (2.3)\\ &= \sum_{y\in\mathcal{Y}} \Pr\left(\mathrm{X}\in\mathcal{U} \mid \mathrm{Y}=y\right)\Pr\left(\mathrm{Y}=y\right) \ . && (2.4)\end{aligned}$$

In case Y is a continuous random variable, the latter formula involves integrals instead of sums.

Bayes Theorem. Since the random variables X and Y have symmetric roles in the definition of the joint probability, it is possible to easily *invert* the conditional probability according to the Bayes' theorem:

$$\Pr\left(\mathrm{X}=x \mid \mathrm{Y}=y\right) = \frac{\Pr\left(\mathrm{Y}=y \mid \mathrm{X}=x\right)\Pr\left(\mathrm{X}=x\right)}{\Pr\left(\mathrm{Y}=y\right)} \ . \qquad (2.5)$$

In this context, the mapping $x \mapsto \Pr\left(\mathrm{X}=x\right)$ is called the *prior* of X, and describes the p.m.f. (of p.d.f. if continuous) of X without taking into account the information that observing Y may give about X. The mapping $x \mapsto \Pr\left(\mathrm{X}=x \mid \mathrm{Y}=y\right)$ is referred to as *posterior* probability of X, and gives the distribution of X once the outcome y of Y is taken into account. Finally, for a fixed $y \in \mathcal{Y}$, the mapping $x \mapsto \Pr\left(\mathrm{Y}=y \mid \mathrm{X}=x\right)$ is called the *likelihood* of x given the observation y. It is worth mentioning that the likelihood is *not* a probability distribution as is, since it is not normalized. Notions of measure's theory are needed to show that Bayes' theorem is valid and keeps unchanged in case of continuous random variables and in cases in which one of the two involved variables is discrete and the other one is continuous. The interested reader might refer to [Kle13, Sec. 8.1, 8.2].

Remarkable Probability Distributions. This thesis will manipulate several different probability distributions that we detail hereafter.

Discrete Uniform Law. We say that a random variable X follows a *discrete random uniform* law over a finite set $\mathcal{X}$ if for each value $x \in \mathcal{X}$ we have $\Pr\left(\mathrm{X}=x\right) = \frac{1}{|\mathcal{X}|}$, that is, the probability of observing an outcome x does not depend on the value of the outcome itself.

Bernoulli Law. A discrete random variable $\mathrm{X} \in \{0,1\}$ follows a Bernoulli law of parameter p, denoted by $\mathcal{B}(p)$, if $\Pr\left(\mathrm{X}=1\right) = p$ or equivalently $\Pr\left(\mathrm{X}=0\right) = 1-p$. The expected value of a Bernoulli law is p and its variance is $p(1-p)$.

Binomial Law. A discrete random variable $\mathrm{X} \in [\![0, n]\!]$ follows a binomial law of parameters n, p, denoted by $\mathcal{B}(n, p)$, if $\forall k \in [\![0, n]\!]$, $\Pr(\mathrm{X} = k) = \binom{n}{k} p^k (1-p)^{n-k}$. The expected value of a binomial random variable is $\mathbb{E}[\mathrm{X}] = np$ and its variance is $\mathrm{Var}(\mathrm{X}) = np(1-p)$.

Gaussian Law. The Gaussian or *normal* distribution is a widely used model for the distribution of continuous variables. We use the symbol $\mathrm{X} \sim \mathcal{N}(\mu, \sigma^2)$ to denote a random variable X following a Gaussian distribution of parameters $\mu \in \mathbb{R}$ and $\sigma^2 \in \mathbb{R}_+$. For a D-dimensional random vector $\mathbf{X}$, we use the symbol $\mathbf{X} \sim \mathcal{N}(\mathbf{M}, \mathbf{\Sigma})$ to denote a vector that follows a multi-variate Gaussian distribution of parameters $\mathbf{M} \in \mathbb{R}^D$ and $\mathbf{\Sigma} \in \mathbb{R}^{D \times D}$, positive-definite. The p.d.f. of a Gaussian distribution is completely determined by the value of its two parameters. It is given by the following expressions, respectively in uni-variate and multi-variate cases:

$$f_{\mathrm{X}}(x) \triangleq \frac{1}{\sqrt{2\pi\sigma^2}} \exp -\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2 , \tag{2.6}$$

$$f_{\mathbf{X}}(\mathbf{x}) \triangleq \frac{1}{\sqrt{(2\pi)^D \det \mathbf{\Sigma}}} \exp -\frac{1}{2}(\mathbf{x}-\mathbf{M})^\intercal \mathbf{\Sigma}^{-1}(\mathbf{x}-\mathbf{M}) . \tag{2.7}$$

The expected value of a Gaussian coincides with the parameter μ for the uni-variate case and with $\mathbf{M}$ for the multi-variate case. The parameter σ^2 coincides with the variance of the uni-variate distribution, while $\mathbf{\Sigma}$ coincides with the *covariance matrix* of the multi-variate one, *i.e.* such that the coefficient $\mathbf{\Sigma}[i, j]$ is $\mathrm{Cov}(\mathbf{X}[i], \mathbf{X}[j])$.

Chebyshev's Inequality for Confidence Intervals. The ML literature proposes several *concentration* inequalities. They provide bounds on the probabilities followed by a random variable, depending on some information assumed to be known about the probability distribution. Chebyshev's inequality is one of them. Let X be a real-valued random variable, then:

$$\forall a > 0, \quad \Pr\left(\left|\mathrm{X} - \mathbb{E}[\mathrm{X}]\right|\right) \leq \frac{\mathrm{Var}(\mathrm{X})}{a^2} . \tag{2.8}$$

Notion of Convergence. Let $(\mathrm{A}_n)_n$ be a sequence of random variables and let A be another random variable. We say that A_n converges *in probabilities* towards A, denoted as $\mathrm{A}_n \xrightarrow[n\to\infty]{\mathcal{P}} \mathrm{A}$ when the following property holds:

$$\forall \epsilon > 0, \Pr(|\mathrm{A}_n - \mathrm{A}| \geq \epsilon) \underset{n\to\infty}{\longrightarrow} 0. \tag{2.9}$$

Like with the classical definition of convergence, we may define a notion of *convergence rate* as a function m defined hereafter:

$$m : \left| \begin{array}{rcl}]0,1[^2 & \longrightarrow & \mathbb{N} \\ \epsilon, \delta & \longmapsto & \mathrm{argmin}\{n \in \mathbb{N} \mid \Pr(|\mathrm{A}_n - \mathrm{A}| \geq \epsilon) \leq \delta\} \end{array} \right. . \tag{2.10}$$

The existence of such function is ensured by the convergence in probabilities, *i.e.* Equation 2.9.

Likewise, we say that A_n converges *in law* towards A, denoted as $\mathrm{A}_n \xrightarrow[n\to\infty]{\mathcal{L}} \mathrm{A}$ when for all continuous bounded function ϕ of random variable we have:

$$\mathop{\mathbb{E}}_{\mathrm{A}_n}[\phi(\mathrm{A}_n)] \underset{n\to\infty}{\longrightarrow} \mathbb{E}[\phi(\mathrm{A})] . \tag{2.11}$$

2.2.2 Statistics

We use the notation $\mathcal{S} = \{x_1, \ldots, x_N\}$ to denote a *dataset* of N Independent and Identically Distributed (i.i.d.) observations of a random variable X. This means that it can be seen as one observation of the random tuple $(\mathrm{X}_1, \ldots, \mathrm{X}_N)$, where the X_i are i.i.d. variables of same distribution as X. The term *statistics* refers to a branch of mathematics that aims to analyze, describe or interpret observed data. Differently, the word *statistic* refers to any measure obtained applying a function to some observed data $\mathcal{S}$. As a consequence (and unless considering trivial cases), a statistic which depends on random variables is itself a random variable.

Descriptive vs. Inferential Statistics. We might distinguish two sub-branches in statistics: the *descriptive* statistics, and the *inferential* statistics. In descriptive statistics, data are described by means of more or less complex statistics (in the sense of measures) that may capture the relevant information necessary to exhaustively describe the data. The most common of them being the *empirical arithmetic mean*, the *empirical covariance* and the *empirical variance*, respectively:

$$\overline{\mathrm{X}} \triangleq \frac{1}{N}\sum_{i=1}^{N}\mathrm{X}_i \ , \tag{2.12}$$

$$S_{\mathrm{X,Y}} \triangleq \frac{1}{N-1}\sum_{i=1}^{N}(\mathrm{X}_i - \overline{\mathrm{X}})\cdot(\mathrm{Y}_i - \overline{\mathrm{Y}}) \ , \tag{2.13}$$

$$S^2_{\mathrm{X}} \triangleq S_{\mathrm{X,X}} \ , \tag{2.14}$$

where the Y_i are i.i.d. It is noticeable that the statistics defined in Equation 2.12 and Equation 2.14 may be seen as polynomial of the random variables X_i denoting the observations from a dataset. They are qualified as statistical moments of order respectively one and two, since the degree of the underlying polynomial is respectively one and two.

In inferential statistics, data are considered as sample observations of random variables and the data analysis aims at modeling the distribution of such variables. Dealing with random variables, inferential statistics exploit the probability theory framework and theorems. Statistics of data (in the sense of measures) play an important role in inferential statistics as well, usually with two goals. The first one aims at estimating random variable parameters. In this case, the statistics are called *estimators* and will be denoted by a hat: for example $\widehat{\mathbb{E}[\mathrm{X}]}$ denotes an estimator for the expected value of X. Likewise, the realization of an estimator random variable is called *estimate* (or estimation). The second one aims at realizing *statistical hypothesis* tests, in order to statistically validate or refute an hypothesis about the random variable X.

The most classical and intuitive estimator for the expected value is the empirical mean $\overline{\mathrm{X}}$, in the sense that the expected value of the estimator is exactly $\mathbb{E}[\mathrm{X}]$ (we say that it is *unbiased*), and its variance is shown to be minimal for a given number of observations. Therefore, such an estimator is said to be *optimal*.

Maximum Likelihood. There exists a generic method to find optimal estimators, called *maximum likelihood*. The idea is to consider the parameter θ of a probability law as a possible realization of a random variable Θ which is linked to the observations $\mathrm{X}_1, \ldots, \mathrm{X}_N$. In this context, the *likelihood function*, introduced in Bayes' theorem (see Equation 2.5), can be reformulated as $\theta \mapsto \Pr(\mathrm{X}_1 = x_1, \ldots, \mathrm{X}_N = x_N \mid \Theta = \theta)$. The maximum likelihood estimator of θ, denoted by $\hat{\theta}$, is therefore obtained by maximizing the likelihood function. Informally, $\hat{\theta}$ is the value which must be assigned to the parameter θ in order to maximize the probability of observing the dataset $\mathcal{S}$.

By concavity of the log function, and since the observations are assumed to be i.i.d., this is equivalent to minimizing the so-called Negative Log Likelihood (NLL) function:

$$\mathcal{L}_{\mathcal{S}}(\theta) \quad = \quad -\sum_{i=1}^{N} \log \Pr(\mathsf{X}_i = x_i \mid \Theta = \theta) \tag{2.15}$$

Therefore, $\hat{\theta} = \operatorname{argmin}_\theta \mathcal{L}_{\mathcal{S}}(\theta)$.[1]

2.2.3 Information Theory

We now define some Information Theoretic quantities. An interested reader may refer to the book of Cover and Thomas [CT06]. Let $\mathsf{Z} \in \mathcal{Z}$ be a discrete random variable. The *entropy* of Z, denoted by $\mathsf{H}(\mathsf{Z})$, describes the uncertainty to guess the value of a realization of a discrete random variable Z. It is formally defined by:

$$\mathsf{H}(\mathsf{Z}) \triangleq -\sum_{s \in \mathcal{Z}} \Pr(\mathsf{Z} = s) \log_2 \Pr(\mathsf{Z} = s). \tag{2.16}$$

The latter definition can straightforwardly extend to the entropy of conditional random variables. Let $\mathsf{X} \in \mathcal{X}$ be a random variable and let $x \in \mathcal{X}$, then:

$$\mathsf{H}(\mathsf{Z} \mid \mathsf{X} = x) \triangleq -\sum_{s \in \mathcal{Z}} \Pr(\mathsf{Z} = s \mid \mathsf{X} = x) \log_2 \Pr(\mathsf{Z} = s \mid \mathsf{X} = x). \tag{2.17}$$

This value depends on the observation x, so we may generalize by defining the *conditional entropy* of a discrete random variable Z given another random variable X. It is formally defined as:

$$\mathsf{H}(\mathsf{Z} \mid \mathsf{X}) \triangleq \mathop{\mathbb{E}}_{\mathsf{X}} [\mathsf{H}(\mathsf{Z} \mid \mathsf{X} = x)]. \tag{2.18}$$

Informally, the conditional entropy quantifies the remaining uncertainty on the guess of Z once X is known. In the latter definitions, it is worth emphasizing that the random variables are implicitly assumed to be discrete. The extension to continuous variables would require a thorough discussion. Nevertheless, despite some random variables observed in this thesis are continuous, their measure remains always discrete, so such a discussion can be still avoided here.

If P and Q are two probability distributions on $\mathcal{Z}$, we define the Kullback - Leibler (KL) divergence as:

$$\mathsf{D}(\mathrm{P} \parallel \mathrm{Q}) \triangleq \sum_{s \in \mathcal{Z}} \mathrm{P}(s) \log_2 \frac{\mathrm{P}(s)}{\mathrm{Q}(s)}. \tag{2.19}$$

This quantity is typically used to measure the difference between two discrete probability distributions, since it is always non-negative and equals zero i.f.f. $\mathrm{P} = \mathrm{Q}$. Thanks to the previous definitions, we can introduce the Mutual Information (MI) between two variables Z and X as:

$$\mathsf{MI}(\mathsf{Z}; \mathsf{X}) \triangleq \mathsf{H}(\mathsf{Z}) - \mathsf{H}(\mathsf{Z} \mid \mathsf{X}) = \mathsf{D}(\Pr(\mathsf{X}, \mathsf{Z}) \parallel \Pr(\mathsf{X}) \Pr(\mathsf{Z})). \tag{2.20}$$

This characterizes how much information can be obtained about Z by observing X.

2.2.4 Monte-Carlo Methods

In Section 5.4, we will be interested in computing the MI between a discrete random variable Z denoting a random secret byte, and a continuous random vector $\mathbf{X}$, denoting the time series of a physical measurement. In this context, we assume to know the generative

[1] The maximum likelihood estimation principle will be extended in Section 4.3 and Equation 9.

p.d.f. $\Pr(\mathbf{X} \mid \mathsf{Z})$. According to Equation 2.20, computing the MI is equivalent to computing $\mathsf{H}(\mathsf{Z})$ and $\mathsf{H}(\mathsf{Z} \mid \mathbf{X})$. The former term is straightforward to compute, since in this thesis Z will always be assumed to follow a uniform discrete law over 2^n values, hence $\mathsf{H}(\mathsf{Z}) = n$. However, Equation 2.18 tells us that computing the conditional entropy term $\mathsf{H}(\mathsf{Z} \mid \mathbf{X})$ involves a D-dimensional integral, where D is the dimensionality of the random vector $\mathbf{X}$. Therefore, it is likely to be intractable. Hopefully, the conditional entropy term may still be efficiently estimated by the so-called *Monte-Carlo* stochastic method. The idea is to replace the expected value in Equation 2.18 by an empirical mean based on the random draw of a dataset $\mathcal{S} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$:

$$\mathsf{H}(\mathsf{Z} \mid \mathbf{X}) \approx \overline{\mathrm{H}}_N \triangleq \frac{1}{N} \sum_{i=1}^{N} \mathsf{H}(\mathsf{Z} \mid \mathbf{X} = \mathbf{x}_i)\,. \tag{2.21}$$

We have said that the empirical mean was an optimal estimator of an expected value, since it is unbiased, and its variance is minimal among every possible estimator of the expected value based on a dataset of N observations. But more interestingly, the *Central Limit Theorem* [Kle13, Thm. 15.37] states that this estimation is consistent, that is, $\overline{\mathrm{H}}_N \xrightarrow[N\to\infty]{\mathcal{L}} \mathsf{H}(\mathsf{Z} \mid \mathsf{X})$, with a convergence speed of $\mathcal{O}\left(\frac{1}{\sqrt{N}}\right)$. This leads to Algorithm 1, describing the way the mutual information can be estimated.[2]

Algorithm 1 Conditional entropy estimation with Monte-Carlo method

Require: $N \in \mathbb{N}^*$, \$: Random Number Generator (RNG)
Ensure: $\overline{\mathrm{H}}_N \xrightarrow[\to\infty]{\mathcal{L}} \mathsf{H}(\mathsf{Z} \mid \mathbf{X})$
  for $i \leftarrow 1$ to N **do**
    $z \leftarrow \$(\mathsf{Z})$
    $x \leftarrow \$(\mathsf{X} \mid \mathsf{Z} = z)$ ▷ Draws a random observation x
    for $s \in \mathcal{Z}$ **do**
      `tab_P`$[s] \leftarrow \Pr(\mathsf{X} = x \mid \mathsf{Z} = s)$ ▷ Compute the likelihood given x
    end for
    `tab_P` $\leftarrow$ `normDist(tab_P)` ▷ Normalizes by computing (2.5)
    `tab_H` $\leftarrow$ `computeH(tab_P)` ▷ Computes the entropy with (2.17)
    $\overline{\mathrm{H}}_N \leftarrow$ `RunningMean(tab_H)` ▷ Averages to estimate (2.18)
  end for

2.3 Recalls in Discrete Mathematics

In this thesis, $\mathbb{F}_{p^q}$ denotes the *finite field* of p^q elements. For the representation of this field, the specifications of the AES consider the set of polynomials with coefficients in $\mathbb{Z}_p$, whose addition (denoted by $\oplus$) and multiplication (denoted by $\times$) are done modulo an irreducible polynomial of degree q. The parameter p, necessarily prime, is called the characteristic of the field. In this thesis, we will only be interested in the *Rijndael field* $\mathbb{F}_{2^8} = \mathbb{Z}_2[X]/P(X)$, where $P(X) = X^8 + X^4 + X^3 + X + 1$, on which all the AES operations are defined. Its characteristic being 2, it has two consequences. First, the polynomial coefficients are binary. Since any polynomial is fully represented by its coefficients, any element in $\mathbb{F}_{2^8}$ can be seen as a byte value. Second, the addition between two polynomials being nothing but the element-wise addition of their coefficients in $\mathbb{Z}_2$, the field addition $\oplus$ coincides with the bit-wise `xor` operation between two bytes, and thereby the addition coincides with the subtraction.

[2] Algorithm 1 will be used in Section 5.4.

2.4 Recalls on AES

As recalled in the introduction, the AES is a round-based block-cipher encrypting blocks of 128-bit plaintexts chunks. Such a chunk is called a *state*. Concretely, it is represented by a 4×4 array of bytes, denoted by a. The byte lying at the i-th row, j-th column of a will be denoted by $a[i, j]$ for $i, j \in \{0, 1, 2, 3\}$. The 16 bytes of the state are indexed column-wise. Each element $a[i, j]$ of the state is mathematically seen as an element of the Rijndael field $\mathbb{F}_{2^8}$ defined in Section 2.3. The AES-128 on which we focus through this thesis loops over 10 similar rounds – see Figure 1.1 – during which it will progressively transform the state from the plaintext to the ciphertext through the rounds, with the help of one subkey for each round. The subkeys are derived from a master key according to a routine called KeySchedule, and are also represented by a 4×4 array of bytes, like the current state. In the AES-128 bit version on which this thesis focuses, the KeySchedule operation is invertible. In other words, perfectly knowing one complete subkey is equivalent to knowing the whole master key, and in particular, the subkey derived at the first round equals the master key. In this thesis, we will particularly focus on the first steps of the cryptographic primitive, as it is the most prone to SCA. That is why it is not necessary to describe the KeySchedule operation here.[3]

The very first step consists in the application of the AddRoundKey operation. Each byte $a[i, j]$ of the state is xor-ed with the corresponding byte of the round key $k[i, j]$.

The next operation is the byte-wise application of a non-linear invertible mapping called SubBytes devoted to introduce confusion in the state. It is composed of the following two functions:

1. The inversion in $\mathbb{F}_{2^8}$, where the null element 0 of the field is mapped to itself. An interesting property of the fact that the group $(\mathbb{F}_{2^8} \backslash \{0\}, \times)$ is cyclic is that computing s^{-1} for $s \neq 0$ is equivalent to computing s^{2^8-2} [Ter18, Lem.5.3.4]. That is why this step is also known under the name of the *power* function.

2. An affine transformation.

Concretely, the SubBytes operation may be implemented thanks to a Look-Up Table (l.u.t.) called Sbox, given in the FIPS-197 [Nat01].

Then, the ShiftRows operation is applied during which the bytes in the second, third and fourth rows of the state are cyclically shifted of 1, 2, and 3 byte(s) respectively.

Finally, the last operation of the round is called MixColumns and is devoted to introduce diffusion in the encryption algorithm by mixing the bytes between them. In this thesis, we will only consider the different intermediate computations of the cryptographic primitive occurring at the output of the AddRoundKey, the SubBytes, and the ShiftRows.

2.5 Recalls on Vectorial Calculus

2.5.1 Gradient and Jacobian Matrix

In the following, $\mathbb{R}^n$ denotes the n-dimensional vector space, provided with the scalar product $\langle \cdot, \cdot \rangle$. Let $f : \mathbb{R}^n \to \mathbb{R}$ be a function of several real-valued variables. We denote its partial derivative with respect to the i-th entry of the input vector $\mathbf{x}$ by $\frac{\partial}{\partial \mathbf{x}[i]} f$. The vector $\nabla f(\mathbf{x}) \triangleq \left(\frac{\partial}{\partial \mathbf{x}[1]} f(\mathbf{x}), \ldots, \frac{\partial}{\partial \mathbf{x}[n]} f(\mathbf{x}) \right)^\intercal$ denotes the gradient of the function f. If there is an ambiguity, the gradient will be denoted by $\nabla_{\mathbf{x}} f(\mathbf{x})$ to emphasize that it is computed with respect to $\mathbf{x}$ only.

We recall that $\mathbf{x}$ is said to be a critical point of f if $\nabla_{\mathbf{x}} f(\mathbf{x}) = 0$, a *local* minimizer if it minimizes f over a neighbourhood of $\mathbf{x}$, and a *global* minimizer if it minimizes f over the

[3] A complete description may be found in the FIPS-197 [Nat01].

whole domain of f. If f is defined over an open set of $\mathbb{R}^n$, a (local or global) minimizer is necessarily a critical point, but the converse is not always true. In that case, such points are called *saddle points*.

If f is a function from $\mathbb{R}^n$ to $\mathbb{R}^m$, then $\mathrm{J}_f(\mathbf{x}) \in \mathbb{R}^{m,n}$ denotes the Jacobian matrix of size (m, n), whose rows are the transposed gradient of each elementary function $\mathbf{x} \mapsto f(\mathbf{x})[i] \in \mathbb{R}$, $i \in [\![1, m]\!]$.

When computing the derivatives of composed functions, it is useful to know the *chaining rule*. The following lemma recalls this calculus rule.

Lemma 1 (Chaining Rule [GBC16, p. 199]). *Let $f : \mathbb{R}^m \to \mathbb{R}^p$ be a real-valued function and $g : \mathbb{R}^n \to \mathbb{R}^m$ be a vectorial-valued function. Let $\varphi : \mathbf{x} \in \mathbb{R}^n \mapsto f \circ g(\mathbf{x}) \in \mathbb{R}^p$. The* chaining rule *states that*

$$\mathrm{J}_\varphi(\mathbf{x}) = \mathrm{J}_f(g(\mathbf{x})) \cdot \mathrm{J}_g(\mathbf{x}) \quad . \tag{2.22}$$

In particular, if $p = 1$, the Jacobian matrices of f and φ are their transposed gradients, so:

$$\nabla_{\mathbf{x}} \varphi(\mathbf{x})^\intercal = \nabla f(g(\mathbf{x}))^\intercal \cdot \mathrm{J}_g(\mathbf{x}) \quad . \tag{2.23}$$

As an example, if one takes $g(\mathbf{x}) = M \cdot \mathbf{x}$ where $M \in \mathbb{R}^{m \times n}$ and $f(\mathbf{y}) = \frac{1}{2}\langle \mathbf{y}, \mathbf{y} \rangle$, one gets $\varphi(\mathbf{x}) = \frac{1}{2}\langle M\mathbf{x}, M\mathbf{x} \rangle = \frac{1}{2}\langle \mathbf{x}, M^\intercal M\mathbf{x} \rangle$ so the gradient is $\nabla_{\mathbf{x}}\varphi(\mathbf{x}) = M^\intercal M\mathbf{x}$. It can then be verified that it corresponds to the product in Equation 2.22, where $\mathrm{J}_g(\mathbf{x}) = M$ and $\nabla_{\mathbf{y}} f(\mathbf{y}) = \mathbf{y}$.

2.5.2 The Gradient Descent Optimization Algorithms

We will see in Subsection 4.1.3 that machine learning (almost) always consists in solving a functional optimization problem which can often be rephrased itself as a numerical optimization problem. This is why we briefly present here the optimization algorithms used in this thesis. Numerical and functional optimization are wide topics in machine learning, hence naturally beyond the scope of this thesis. The interested reader may refer to the books of Boyd *et al.* [BV14], Goodfellow *et al.* [GBC16, Chap. 8] or Shalev-Shwartz and Ben-David [SSBD14, Chap. 14].

The Stochastic Gradient Descent (SGD) Algorithm. Let $f : \mathbb{R}^d \to \mathbb{R}$. We are given a random initial point $\mathbf{x}_0 \in \mathbb{R}^d$, and a parameter $\eta > 0$ called *learning rate*. The SGD step consists in updating the current point $\mathbf{x}_t$ as follows:

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \eta \nabla_{\mathbf{x}} f(\mathbf{x}_t) \,. \tag{2.24}$$

It can be shown that when f is convex and *smooth* (*i.e.*, the norm of the gradient is bounded), SGD converges towards the unique point $\mathbf{x}^\star$ minimizing f with speed $\mathcal{O}\left(\frac{1}{\sqrt{T}}\right)$ where T is the number of steps [SSBD14, Thm. 14.8].[4] Actually, the SGD step given in Equation 2.24 denotes the regular gradient descent on functions differentiable everywhere. The convergence of the SGD can be extended without loss of generality to functions that are differentiable almost everywhere, provided that where it is defined, the gradient has a bounded norm. Likewise, the exact gradient of f can be replaced by an unbiased statistical estimator of $\nabla f(\mathbf{x}_t)$, without changing the convergence properties of SGD [SSBD14, Thm. 14.8]. Unfortunately, SGD is not guaranteed to converge towards the minimum of f if the latter one is not convex, which will be the case in our context as we will see in Subsection 4.3.3. More precisely, provided that the learning rate is small enough, the SGD can still converge *almost surely* towards a local minimizer [LSJR16], which means that the convergence towards a saddle point has probability zero if the initial point $\mathbf{x}_0$ is randomly chosen. Nevertheless,

[4] We recall that unicity of a minimizer of f is ensured by convexity [BV14, Sec. 4.2.2].

this minimizer is not necessarily the global minimizer. Even worse, the learning rate is a sensitive parameter. If the learning rate is too high, the SGD can diverge [BV14, Sec. 9.3]. On the opposite, if the learning rate is too low, the convergence may be prohibitively long. For this reason, the SGD is not widely used in practice.

Instead of using the SGD as is, we will rather use a slight variant called Adaptive Moment Estimation (Adam), proposed by Kingma *et al.* at ICLR'15 [KB15]. It is based on adaptive estimates of lower-order (*i.e.* 1 and 2) moments of the gradients computed at each iteration of the descent. Those moments are then used to slightly modify the descent direction – originally set to $-\nabla_{\mathbf{x}} f(\mathbf{x})$. It is nowadays one of the most used gradient descent based optimization algorithms for machine learning. A complete description and study of this algorithm is beyond the scope of this thesis. Nevertheless, the interested reader may refer to the *Deep Learning* book by Goodfellow *et al.* for more information [GBC16, Sec. 8.5.3], or directly to the Kingma *et al.*'s paper.

Chapter 3

Side-Channel Attacks

" All models are wrong, but some models are useful."

George Box

Contents

3.1 Definition of a Side-Channel Attack

3.1.1 The Attack Scenario

Let $\mathcal{T}$ be the instance of the *target* device under an attack conducted by an adversary, a.k.a. *attacker*, denoted by $\mathcal{A}$. We assume that $\mathcal{T}$ runs a cryptographic primitive $\mathbf{E}()$ each time that a query, represented by a plaintext $\mathbf{p}$, is sent by the attacker. The primitive being set by a secret encryption key $\mathbf{k}^\star$, the target returns the ciphertext corresponding to the encryption of the sent plaintext, that is, $\mathbf{c} = \mathbf{E}(\mathbf{p}, \mathbf{k}^\star)$. The goal of the attacker is then to guess the secret key, assumed to belong to a known key space $\mathcal{K}$.

In this thesis, we modelize a SCA by the following scenario, that is illustrated in Figure 3.1. First, the target randomly draws a secret key $\mathbf{k}^\star$ that is used for the encryption. Then, $\mathcal{A}$ sends a given number N_a of queries to the target $\mathcal{T}$. Those queries are materialized by the input plaintexts $\mathbf{p}_1, \ldots, \mathbf{p}_{N_a}$. For each plaintext $\mathbf{p}_i \in \mathcal{P}$, the target $\mathcal{T}$ returns the corresponding ciphertext $\mathbf{c}_i$ but also a measurement $\mathbf{x}_i \in \mathcal{X}$, a.k.a. SCA *trace*, corresponding to the physical leakage occuring during the computation $\mathbf{c}_i = \mathbf{E}(\mathbf{p}_i, \mathbf{k}^\star)$. In the remaining of this thesis, we denote by $\mathcal{S}_a \triangleq \{(\mathbf{x}_1, \mathbf{p}_1), \ldots, (\mathbf{x}_{N_a}, \mathbf{p}_{N_a})\}$ the *attack set* acquired by the attacker $\mathcal{A}$ during the SCA. This attack scenario is often called a *gray-box* attack, in opposition to a black-box scenario corresponding to classical cryptanalysis, where the attacker does not have access to the SCA traces in the attack set, but rather the ciphertexts instead.

From a probabilistic point of view, $\mathbf{p}, \mathbf{k}^\star, \mathbf{x}$ can be respectively seen as the realizations of the corresponding random variables $\mathbf{P}, \mathbf{K}, \mathbf{X}$, according to the probabilistic graph presented in Figure 3.2. More precisely, we make the assumption that $\mathbf{X}$ only depends on a random variable $\mathbf{Z}$ resulting in an intermediate computation denoted by $\mathbf{C}()$ involving chunks of $\mathbf{P}$ and $\mathbf{K}$,[1] *e.g.*, $\mathbf{C}(\mathbf{P}, \mathbf{K}) = \mathbf{P} \oplus \mathbf{K}$. This random variable is called *sensitive* since it depends on the secret key, and *intermediate* since it corresponds to an intermediate state between the plaintext and the final ciphertext. In other words, some knowledge about the values $\mathbf{z}_i = \mathbf{C}(\mathbf{p}_i, \mathbf{k}^\star), i \in [\![1, N_a]\!]$ of the sensitive intermediate variable, induces some knowledge about the underlying secret key $\mathbf{k}^\star$ used for the encryption.

Moreover, we assume that the couple $(\mathbf{P}, \mathbf{X})$ is not independent from the secret key $\mathbf{K}$. Otherwise, considering the gray-box scenario has no further interest compared to the black-box one. Finally, the random variable $\mathbf{X}$ is usually assumed to be drawn from a continuous p.d.f., since it measures a physical phenomenon. However in practice, the observations of this random variable are discretized during the acquisition phase by the oscilloscope. That is why the leakage space is often of the type $\mathcal{X} = [\![0, 2^\omega - 1]\!]^D$, where D is the dimensionality of the observations, and where ω denotes the resolution of the oscilloscope – typically $\omega = 8$.

In the pursuit of his ultimate goal, the attacker can process the attack set in order to

[1] Subsection 3.1.2 will discuss how to reduce the problem to chunks of plaintexts and keys.

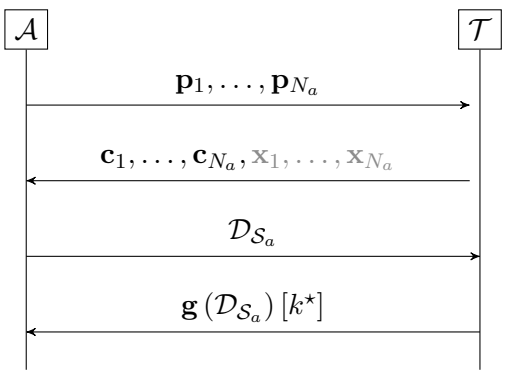


Figure 3.1: Black: scenario of the black-box model. Grey: additional information added in the gray-box scenario.

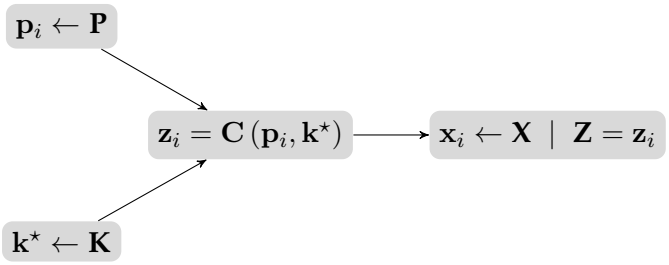


Figure 3.2: Probabilistic graph denoting the links between the data.

extract *information* on (a chunk of) the secret key. Depending on how the attacker wants to exploit this information, the latter one can take different forms.

- Either the attacker aims at directly recovering the secret key from $\mathcal{S}_a$, without additional investigation. Then, he returns a value $\widehat{\mathbf{k}}$ that he believes to correspond to the right key $\mathbf{k}^\star$, according to the extracted information from $\mathcal{S}_a$.
- Or the attacker $\mathcal{A}$ may want to combine a SCA with other attack techniques such as:
  - *Algebraic* attacks, *e.g.*, by using a Propositional Satisfiability Problem (SAT) solver to find the right key among a restricted list of o key candidates $\widehat{\mathbf{k}}_1, \ldots, \widehat{\mathbf{k}}_o$ returned after the SCA – where $o \ll |\mathcal{K}|$. Such techniques have been initially introduced by Renauld *et al.* [RS09, RSV09].[2]
  - *Brute-force* attacks, *e.g.*, by using a key enumeration algorithm [VGRS12, MOOS15, BKM+15, Pou18] fed with the set of key hypotheses $\mathcal{K}$ sorted in a decreasing order of preference, returned by the SCA until it reaches the right key $\mathbf{k}^\star$.

More generally, those different approaches may be encompassed by the following one: the SCA attacker $\mathcal{A}$ returns a vector assigning a score to each hypothetical value of the key. This vector is computed thanks to a so-called *distinguisher* that we define hereafter.

Definition 1 (Distinguisher). *Let $\mathcal{S}_a$ be an attack set. A* distinguisher *is a mapping from $\mathcal{S}_a$ to a* score vector *in $\mathbb{R}^{|\mathcal{K}|}$:*

$$\mathcal{D} : \mathcal{S}_a \mapsto \begin{pmatrix} \vdots \\ \mathcal{D}_{\mathcal{S}_a}[\mathbf{k}] \\ \vdots \end{pmatrix}. \tag{3.1}$$

Remark 1. *The definition of the distinguisher may be refined by constraining the scores to belong to the interval $[0, 1]$, 0 denoting the least confidence in the corresponding key hypothesis while 1 denoting the greatest confidence. This constraint can still be obtained by applying a normalization of the scores.*

This definition encompasses the different forms of information exploitation from the SCA presented so far. For the first way, the attacker $\mathcal{A}$ takes $\widehat{\mathbf{k}} = \operatorname{argmax}_{\mathbf{k} \in \mathcal{K}} \mathcal{D}_{\mathcal{S}_a}[\mathbf{k}]$. The attack is then said *successful* i.f.f. $\widehat{\mathbf{k}} = \mathbf{k}^\star$. For the second way, *i.e.* the algebraic attack, the attacker $\mathcal{A}$ returns a list of key candidates corresponding to the o first scores from $\mathcal{D}_{\mathcal{S}_a}$. The attack is said successful i.f.f. $\mathbf{k}^\star \in \{\widehat{\mathbf{k}}_1, \ldots, \widehat{\mathbf{k}}_o\}$. Finally, for the last way, the attacker enumerates the key candidates by decreasing order of their scores in $\mathcal{D}_{\mathcal{S}_a}$. The rank of the right key therefore quantifies the amount of enumeration necessary to succeed the key recovery. In Subsection 3.2.2, we will further discuss this rank through the notion of Guessing Entropy (GE). Although unknown in advance by a pure attacker, this quantity is known by a developer/evaluator in an evaluation context.

[2] This approach has been rewarded at the CHES 2018 Capture the Flag (C.t.F.) [GJS19, HZF+19, GJS20].

Remark 2 (Vertical Attacks). *The SCA literature sometimes makes a discrepancy between* vertical *attacks, namely "technique analyzing the same sample time regions of several [...] traces" in opposition to* horizontal *attacks that "analyze many portions of a single trace" [Cag18]. The attack scenario presented in this thesis particularly fits the case of vertical attacks, which are typically used against block ciphers, whereas horizontal attacks are rather used on asymmetric cryptography, e.g. based on RSA.*

3.1.2 Reducing the Problem.

At this stage of the description, it is noticeable that $\mathcal{A}$ has two main degrees of freedom, namely the choice of the distinguisher and the strategy[3] to choose the plaintexts $(\mathbf{p}_i)_{i\in[\![1,N_a]\!]}$ for the queries, materialized by the p.m.f. of $\mathbf{P}$ that is set by the attacker. We discuss both degrees of freedom hereafter.

For the AES, it is usual to take the input or the output of the first SubBytes operation. Indeed, at this step of the algorithm, no diffusion operation has been applied yet in the encryption so $\mathbf{Z}$ is the byte-wise output of the composition $p, k \mapsto \mathrm{Sbox}[p \oplus k]$. Since the p.m.f. of the secret key is assumed to be uniform over the AES field $\mathbb{F}_{2^8}$,[4] we have that for all value $\mathbf{p}$ chosen by $\mathcal{A}$, the sensitive random variable $\mathbf{Z}$ is also uniform and independent from $\mathbf{P}$, and, likewise, $\mathbf{X}$ is independent from $\mathbf{P}$. In other words, the attacker has no reason to prefer the sending of a plaintext value from another. Since we assume in our scenario that the plaintexts are all sent at the same time, this concern all the plaintexts. That is why in the following, we will assume that the plaintexts are i.i.d. and randomly chosen according to the uniform law.

More interestingly, this also means that all the bytes of the sensitive variable are independent from each other, and so are the bytes of the key and those of the plaintext. Therefore the j-th byte $\mathbf{Z}[j]$ of the sensitive variable only depends on the j-th byte of plaintext $\mathbf{p}[j]$ and on the j-th byte of the secret key $\mathbf{k}^\star[j]$. This allows us to recover the secret key $\mathbf{k}^\star$ as a byte-wise manner, in a so-called *divide-and-conquer* strategy. The recovery of one key byte at the time enables the attacker $\mathcal{A}$ to drastically reduce the key chunk search space from 2^{128} to 2^8, thereby breaking the high complexity usually required to run an attack in the black-box threat model. The whole secret key can then be recovered by replicating the reduced attack on the 16 key bytes independently. This reduction makes the SCA particularly efficient regarding cryptanalytic attacks.

In the remaining of this thesis, unless not precised, we will only consider the recovery of one byte of the secret key – hence implicitly assuming that it applies similarly to all key bytes. This means that we substitute the plaintext random vector $\mathbf{P} \in (\mathbb{F}_{2^8})^{16}$ by a random variable $\mathrm{P} \in \mathbb{F}_{2^8}$. Likewise, we substitute the secret key $\mathbf{K}$ by K and the sensitive vector $\mathbf{Z}$ by Z. The reader's attention is drawn however on the fact that the leakage $\mathbf{X}$ is still considered as a vector.

3.1.3 Beyond our Attack Scenario.

The gray-box scenario considered here presented the powers and degrees of freedom of an attacker aiming at recovering the secret key. Despite being beyond the scope of this thesis, we also provide hereafter a (non-exhaustive) list of ways to build an augmented attack compared to what is assumed here for the attacker.

Adaptive Chosen Plaintexts. Rather than sending the N_a queries to the target $\mathcal{T}$ and then waiting for the acquisitions of the N_a corresponding traces, a more realistic scenario would

[3] The actual term used in reinforcement learning is *policy* [SB98].

[4] Otherwise, the uncertainty on the key does not ensure anymore the security, and a simple brute-force attack may become affordable for the attacker.

be to send a first query $\mathbf{p}_1$, then to acquire the first trace $\mathbf{x}_1$ along with the cipher text $\mathbf{c}_1$ before sending the next query, and so on. In that case, the attacker may already collect some information about the secret key after each acquisition, or equivalently before each query. This way, he may eventually use an adaptive chosen-plaintext strategy that may help making a discrepancy between particular key hypotheses faster, *i.e.* requiring less queries to the target $\mathcal{T}$. Such strategies have been investigated by Köpf *et al.* [KB07, KB11] and Veyrat-Charvillon *et al.* [VS10], with promising results on simulations. An extension of those works involving the *reinforcement learning* [SB98] framework would be a promising track in the coming years. Yet, this remains beyond the scope of this thesis.

Key Rank Estimation. Whereas an attacker would be interested in recovering the whole key enumeration, an evaluator would just be interested in knowing how many keys should be enumerated according to the guessing vector in order to reach the right key, rather than actually enumerating them. Several works propose some ranking estimation methods [VGS13, YEM14, GGP+15, MMOS16, MMO18, DW19, APSV20] allowing to save some time compared to a naive key enumeration.

Other Ways to Partition. Finally it is worth emphasizing that although they will not be investigated in this thesis, other divide-and-conquer strategies may be used, *e.g.*, at a bit-wise level. More generally speaking, the choice of such a strategy depends on the nature of both the cryptographic primitive and the physical leakage occurred by the target. This may typically lead to the reduction on different key chunks. As an example, the AddRoundKey of the last round of AES can be targeted instead of the first one, with the same complexity by just swapping the ciphertexts with the plaintexts. In this case, the recovered key bytes form the last derived subkey from the KeySchedule operation, rather than the ones of the master key directly. Yet, since the KeySchedule is invertible for the AES-128, this attack path equivalently leads to the master key recovery. An example of such an attack path is provided in Subsection 3.8.4.

3.2 Assessing an Attack

Whereas an attacker is ultimately interested in the value of the secret key embedded in the target device, a developer or an evaluator is much more interested in the effort required by the attacker to succeed, from which ultimately depends the security level of the target implementation. This different perspective leads the SCA community to design and adopt conventions on the performance metrics used for the quotation of the vulnerabilities of the target implementation. We present in this section the different aspects to take into account in the efficiency evaluation of an attack, before introducing the related performance metrics.

3.2.1 The Different Factors of Attack Complexity

Depending on who actually instantiates the attack $\mathcal{A}$, *i.e.*, whether this is an actual adversary or a developer/evaluator, one will differently define the notion of complexity. We particularly distinguish the *effectiveness* of an attack – *i.e.* can $\mathcal{A}$ succeed – from its *efficiency* – *i.e.* to what extent can $\mathcal{A}$ succeed. This distinction is necessary because nowadays crypto-systems are designed to be *resilient* from a potential informative leakage about some secret data, *e.g.*, the key. This concretely means that those crypto-systems often refresh the secret keys used in their communication protocols by the cryptographic primitives, after a given number of uses for encryption and/or decryption. On the one hand, if an attack requires a number of queries N_a beyond the refreshing period of a key, then it will be harmless for the crypto-system. On the other hand, refreshing the secret key in a communication protocol might limit the runtime performance of the upper communication layers of the target device, and

thereby its global performance desired by the developer. Thanks to this mechanism, the developer can control the trade-off between performance and security, depending on the efficiency of potential attacks. In this context, an attacker is interested in the effectiveness of its particular attack, whereas a developer is more interested in the best efficiency of a wider class of attacks against which he wants to protect its device.

Therefore, the notion of efficiency can be more precisely translated into the required number N_a of queries to succeed the attack $\mathcal{A}$ whereas the effectiveness can be defined by the existence of such an N_a ensuring the success of the attack.

3.2.2 The Performance Metrics

To assess the effectiveness and the efficiency of an attack, it has initially been suggested to measure or estimate the minimum number of traces required to get a successful key recovery [MOP07]. This can be done by computing the *guessing vector* $\mathbf{g}\left(\mathcal{D}_{\mathcal{S}_a}\right)$ of a score vector $\mathcal{D}_{\mathcal{S}_a}$. The coordinates of the guessing vector are defined as follows:

$$\mathbf{g}\left(\mathcal{D}_{\mathcal{S}_a}\right)[k] \triangleq \sum_{k' \in \mathcal{K}} 1_{\mathcal{D}_{\mathcal{S}_a}[k'] \geq \mathcal{D}_{\mathcal{S}_a}[k]} \quad , \tag{3.2}$$

where 1 is the characteristic function defined in Section 2.1. In particular, $\mathbf{g}\left(\mathcal{D}_{\mathcal{S}_a}\right)[k^\star]$ denotes the *rank* of the right key, which determines the success of an attack depending on the form of the exploitation of the scores by the attacker, as discussed in Subsection 3.1.1. Although unknown by a pure attacker, this quantity is known by a developer/evaluator in an evaluation context.

Yet, many random factors may be involved during the attack: we have seen that the traces and the plaintexts may be seen as the realizations of N_a couples of i.i.d. random variables $(\mathbf{X}, \mathbf{P})$, so the attack set $\mathcal{S}_a$ may be seen itself as the realization of a random vector. In other words, one cannot consistently compare two attackers $\mathcal{A}_1$ and $\mathcal{A}_2$ from one attack set, since the comparison could lead to different conclusions on another attack set $\mathcal{S}'_a$ acquired on the same target $\mathcal{T}$. So any measure of success must be refined to remove any dependency on random factors.

To circumvent this issue, the SCA community has agreed on a metric called the Success Rate (SR) *at order* o:[5]

$$\mathrm{SR}(N_a, \mathcal{D}, o) \triangleq \Pr\left(\mathbf{g}\left(\mathcal{D}_{\mathcal{S}_a}\right)[k^\star] \leq o \;\middle|\; |\mathcal{S}_a| = N_a\right) \quad , \tag{3.3}$$

where o is set according to the desired definitions of "success" among those proposed in Subsection 3.1.1. The SR quantifies the probability that the attacker $\mathcal{A}$ succeeds in finding the secret key stored in the target $\mathcal{T}$ within a given number N_a of queries done during the attack phase. If the attack is effective, the SR is expected to increase with N_a and to converge towards 1. Following the discussion in Subsection 3.2.1, the efficiency of the attack $\mathcal{A}$, *at probability* β, is likewise materialized by:

$$N_a(\mathcal{D}, o, \beta) \triangleq \min\left\{N_a \in \mathbb{N} \;\middle|\; \mathrm{SR}(N_a, \mathcal{D}, o) \geq \beta\right\} \quad , \tag{3.4}$$

where $\beta \in [0, 1]$ is a threshold set by the evaluator, typically $\beta = 90\%$. Figure 3.3 illustrates the relationship between the different quantities introduced so far in this section.

One can accordingly compare two attackers $\mathcal{A}_1$ and $\mathcal{A}_2$ by comparing the efficiency of their respective distinguishers at a given threshold and a given success order. In the remaining of this thesis we will lighten the notations, by removing the reference to the success order o when the latter one is implicitly fixed to one, and by removing the reference to β when the latter one is implicitly fixed to 90%. Likewise, since so far an attacker is fully defined by its distinguisher, we may equivalently substitute $\mathcal{D}$ with $\mathcal{A}$ in the notations.

[5] The notion of order of the success rate shall not be confused with the notion of order of secret-sharing defined in Subsection 3.7.1.

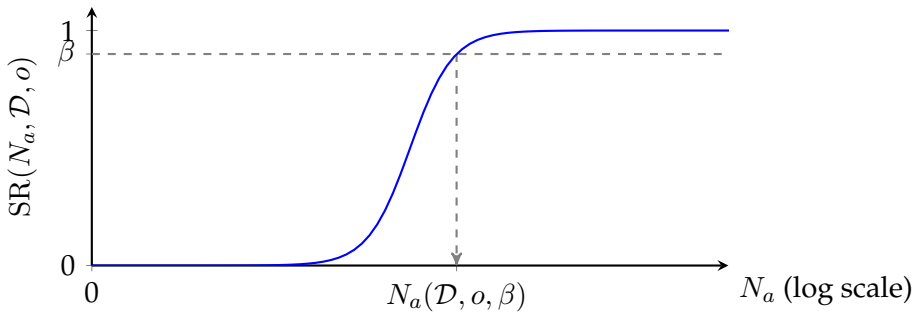


Figure 3.3: Typical shape of a Success Rate (SR) plot, illustrating how $N_a(\mathcal{D}, o, \beta)$ is defined.

Within this framework, it is then common to formulate the evaluator's task as assessing the worst-case scenario from the developer's point-of-view. The pursuit of such scenario is the cornerstone of the evaluation, as stated by the following problem.

Problem 1 (SCA Optimization). *Given a target $\mathcal{T}$, a threshold $\beta \in [0, 1]$ and a success order o, find the most efficient attacker $\mathcal{A}$, i.e., the one instantiating the distinguisher $\mathcal{D}$ minimizing $N_a(\mathcal{D}, o, \beta)$. We denote by*

$$N_a^\star(o, \beta) \triangleq \min_{\mathcal{D}} \{N_a(\mathcal{D}, o, \beta)\} \tag{3.5}$$

the efficiency of the optimal attack.

Remark 3. *Rather than the success rate, one can equivalently consider the average ranking of the correct guess, a.k.a. the Guessing Entropy (GE) [SMY09], defined as:*

$$\mathrm{GE}(N_a, \mathcal{D}) \triangleq \mathop{\mathbb{E}}_{\mathcal{S}_a} \left[\mathbf{g}\left(\mathcal{D}_{\mathcal{S}_a}\right)[k^\star] \;\middle|\; |\mathcal{S}_a| = N_a \right]. \tag{3.6}$$

In that case, the efficiency is defined by :

$$N_a(\mathcal{D}, \tau) \triangleq \min \{N_a \;|\; \mathrm{GE}(N_a, \mathcal{D}) \leq \tau\} \quad , \tag{3.7}$$

where $\tau \geq 1$ is a threshold set by the evaluator. An illustration of the metrics related to the GE is proposed in Figure 3.4. The GE quantifies the average amount of enumeration which is yet to be done after the key recovery phase if the right key is not ranked in the first place in the guessing vector.

Since an acceptable amount of enumeration for the whole key – i.e. made of the 16 bytes for AES – is generally set to 2^{32}, it is usual to set the threshold to recover only one byte to $\tau = 2$. This way, it ensures the average amount of enumeration for the whole key to lie below $\tau^{16} \leq 2^{32}$. In the remaining of this thesis, we will let the parameter τ implicitly set to 2, in order to lighten the notations.

The GE is of great interest for attack scenarios in which the attacker $\mathcal{A}$ is allowed to proceed a key enumeration after the attack phase, and we will provide later in this thesis an illustration of a GE plot in Figure 7.9. Moreover, one can draw a parallel with the eponymous notion of GE defined by the NIST [BDP06], which "measures [...] the difficulty that an attacker has to guess the average password used in a system". Nevertheless, we will favor the SR in this thesis.

3.2.3 Estimating the Metrics in Practice

In practice, to estimate $\mathrm{SR}(N_a, \mathcal{D}, o)$, sampling many attack sets may be very prohibitive in an evaluation context, especially if we need to reproduce the estimations for many values of N_a until we find $N_a(\mathcal{D}, o, \beta)$. One solution to circumvent this problem is, given a validation set of N_v traces, to sample some attack sets by permuting the order of the traces into the validation set (*e.g.* 50 times). $\mathcal{D}_{\mathcal{S}_a}$ can then be computed with a cumulative sum to get a score for each $N_a \in [\![1, N_v]\!]$. For each value of N_a, the success rate is estimated by the occurrence frequency of the event "$\mathrm{argmax}_{k \in \mathcal{K}} \mathcal{D}_{\mathcal{S}_a}[k] = k^\star$".[6] While this trick gives good

[6] The GE is likewise estimated by computing the average value of $\mathbf{g}\left(\mathcal{D}_{\mathcal{S}_a}\right)[k^\star]$.

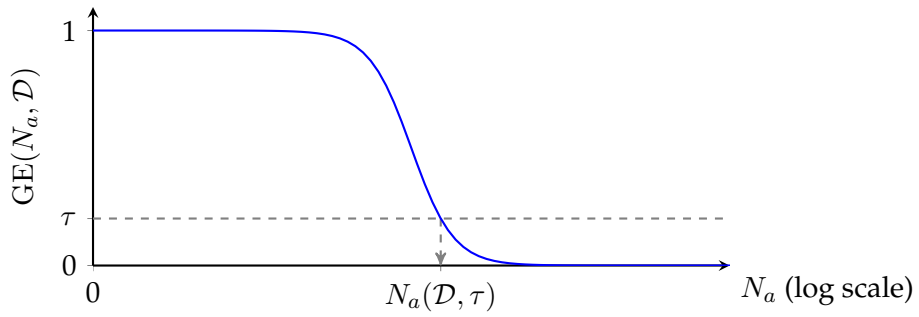


Figure 3.4: Typical shape of a Guessing Entropy (GE) plot, illustrating how $N_a(\mathcal{D}, \tau)$ is defined.

estimations for $N_a \ll N_v$, one has to keep in mind that the estimates become biased when $N_a \to N_v$. Retrospectively, we must verify in each experiment that the result $N_a(\mathcal{D}, o, \beta)$ is indeed much lower than N_v.

3.3 Conditions of an Optimal Attack

When addressing Problem 1, it is relevant to first recall a key result from Heuser *et al.* presented at CHES'14: an analytical optimal solution to Problem 1 is given by the following theorem.

Theorem 1 (Optimal Distinguisher [HRG14, Thm. 1])**.** *The most efficient attacker $\mathcal{A}$ for the device $\mathcal{T}$ is the one using the maximum likelihood – defined in Section 2.2.2 – as a distinguisher, i.e.,*

$$\mathcal{D}^{\mathrm{ML}}_{\mathcal{S}_a}[k] = \sum_{i=1}^{N_a} \log \Pr\left(\mathbf{X}_i = \mathbf{x}_i \;\middle|\; \mathrm{Z}_i = \mathbf{C}\left(p_i, k\right)\right) \quad , \tag{3.8}$$

i.e., $N_a\left(\mathcal{D}^{\mathrm{ML}}_{\mathcal{S}_a}\right) = N^{\star}_a$.

At this stage, it is relevant to comment the different elements of Equation 3.8:

- **The Leakage model** denotes here the p.d.f. $\Pr\left(\mathbf{X} \mid \mathrm{Z}\right)$. More generally, it is the way to describe the physical dependency between one leakage trace $\mathbf{X}$ and the sensitive target variable Z.
- **The Distinguisher** properly said is the way how the information extracted on each trace through the leakage model is combined to compute the scores. Here in particular, the distinguisher is the sum of the log probabilities of the likelihood function.

We may discuss its impact on our attack gray-box scenario. On the one hand, it implies that the optimal attacker $\mathcal{A}$ is fully determined by the choice of the maximum likelihood distinguisher, thereby addressing the last remaining degree of freedom. This is useful in order to build provably secure implementations against any type of attacker: it suffices to prove that the given implementation is secure against an attacker using the maximum likelihood distinguisher.

On the other hand, the major drawback of such a distinguisher is that it implicitly requires the full knowledge of the leakage model. The latter one typically depends on the target implementation $\mathcal{T}$, both at software and hardware levels, as on the acquisition environment of the physical measurements. Therefore, perfectly knowing the leakage model turns out to be practically impossible as is. In other words, the analytical solution of Problem 1 is not informative for the SCA evaluator. To circumvent this issue two approaches have been proposed in the literature.

The first one – historically speaking – consists in making assumptions on the leakage model depending on the knowledge of the attacker or the evaluator on the device. These assumptions may be strong and even non-realistic although representing reasonable approximation errors. The counter-part to this approach is that other distinguishers, possibly less sensitive to approximation errors, may lead to more efficient attacks compared to the maximum likelihood distinguisher. This approach, presented in Section 3.5 is nowadays called *unprofiled* attacks, in opposition to the second approach, hence called *profiled* attacks.

The second scenario allows to still assume the attacker to have access to the exact leakage model – or at least a good approximation of it for some metric that must be previously defined, in order to address the worst-case scenario.

To this end, a preliminary phase of the attack requires to characterize the leakage behavior of the device. This will be detailed in Section 3.4. This approach enables to reformulate Problem 1 in a slightly modified version, namely Problem 2, that can be more practically useful.

3.4 Profiled Attacks

Profiled attacks provide a way to help the attacker to accurately approximate the leakage model, in order to allow the use of the maximum likelihood distinguisher – see Equation 3.8 – to be practically used in an SCA context. From an evaluator's point of view, it is also relevant, since it allows him to implement an attack close to the optimal one, rather than just considering it as theoretical. This is useful when assessing the performance of a *real*, possibly non perfect attacker. It relies on the existence of a clone device $\mathcal{T}'$ of the actual target $\mathcal{T}$. The clone device is assumed to behave as an *open sample*, *i.e.* it is fully controlled by the attacker; especially the knowledge (and eventually the choice) of all the parameters and intermediate computations processed during the execution of the primitive, including the random values used to secure the processing – see Subsection 3.7.1.

A profiled attack, depicted in Figure 3.5, is divided into two distinct phases. The first one, called *profiling phase*, as depicted on the left of Figure 3.5, exploits so-called *profiling traces*. Profiling traces are acquisitions taken under known values for the sensitive variable Z, so the attacker collects a *profiling set* $\mathcal{S}_p \triangleq \{(\mathbf{x}_1, z_1), \ldots, (\mathbf{x}_{N_p}, z_{N_p})\}$, for which the correct association trace/sensitive variable is known. The profiling phase is typically done on the clone device $\mathcal{T}'$, assumed to have the same physical and algorithmic behavior as the target $\mathcal{T}$. Intuitively, the less similar the behavior of the clone device $\mathcal{T}'$ with respect to the target device $\mathcal{T}$, the more loss in the attack performance. That is why in practice, an evaluator aiming at finding the worst-case scenario often considers the target device $\mathcal{T}$ to be the exact clone $\mathcal{T}'$ on which he is working.[7]

The second phase of a profiling attack is the *attack phase* strictly speaking, during which the attacker $\mathcal{A}$ proceeds exactly as in the gray-box scenario depicted in Figure 3.1. Therefore, $\mathcal{A}$ may take the advantage of the previous profiling phase to infer over it.

Assessing a Profiled Attack. Considering a profiled attack scenario allows an evaluator to conduct a worst-case scenario analysis of the target security. Such a scenario typically covers very powerful attackers, potentially without restriction in terms of financial, material and human resources. Regarding this analysis, it is common to assume the attacker to have an *unbounded* profiling power, in order to fully exploit the behavior of the open sample. This means concretely that the resources used by an attacker in a profiling scenario, in terms of human expertise, time and technical means, are not critical here and therefore, are considered as negligible. In particular, no bound on the number N_p of acquired traces in the profiling set $\mathcal{S}_p$ is assumed in this thesis, contrary to the number N_a of traces in the attack

[7] This assumption is discussed in Section 4.4 when we review the literature working on the *portability* issue.

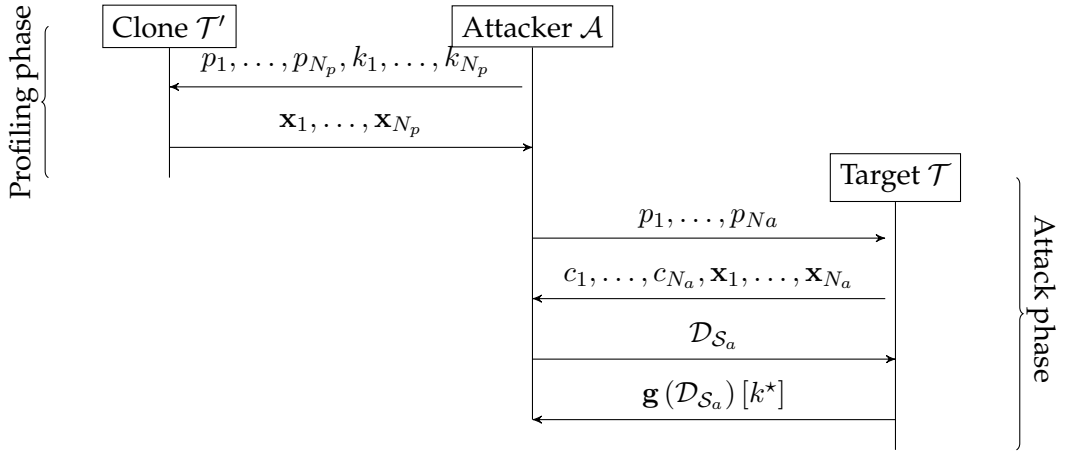


Figure 3.5: Profiling attack scenario: a gray-box attack scenario with a preliminary profiling phase.

set $\mathcal{S}_a$ that we assess during the evaluation of a target. However, from an evaluator's point-of-view, this assumption is questionable. If the target device is not provably secure against profiled SCA, the security guarantees come from a practical evaluation, which cannot be mounted with infinite resources. Hence, the worst-case scenario analysis is not always affordable in a profiled attack context. That is why some works also consider the case of restricted profiling power for the evaluator too [PHG19]. Chapters 4 and 5 will discuss the impact of the number N_p of profiling traces on the quality of the profiling phase.

Template Attacks (TA). The most known estimation method of the leakage behavior is the use of Gaussian Templates (GTs), as initially proposed by Chari *et al.* at CHES'02 [CRR02]. Precising the term "Gaussian" here means that one assumes the likelihood to follow a (eventually multivariate) Gaussian law:

$$\mathbf{X} \mid \mathrm{Z} = s \sim \mathcal{N}(\mathbf{M}_s, \mathbf{\Sigma}_s), \tag{3.9}$$

whose parameters $\mathbf{M}_s, \mathbf{\Sigma}_s$ (possibly) depend on the sensitive value s processed by the intermediate computation Z. During the profiling phase, those parameters are estimated respectively thanks to the empirical mean and the empirical covariance matrix for each cluster of traces sharing the same value s.

The critical task here comes from the estimation of the $\frac{D\times(D-1)}{2}$ different coefficients of the covariance matrix $\mathbf{\Sigma}_s$ for each value $s \in \mathcal{Z}$. Choudary *et al.* recall that the latter one must be invertible – see Equation 2.7, and explain that a necessary condition is that $N_p \geq D$. The latter condition is usually not sufficient because of the noise in the leakage. Indeed, making estimations of the parameters accurate enough in order to make a strong discrepancy between each template would require much more profiling traces N_p. Although not critical at first sight since we do not assume any limitation on the number of profiling traces, it may become a practical issue for the evaluator if the input dimensionality becomes too high.

To circumvent this problem, two solutions are proposed in the literature. First, a dimensionality reduction pre-processing can be done on the acquired traces from the profiling set. This solution aims at decreasing D. Some of those techniques will be discussed in Section 3.6. Second, one may consider other assumptions on the covariance matrix, in order to decrease the required number of data for the estimation. The literature in statistics proposes a wide spectrum of such techniques. Yet, we mention hereafter the ones used in the specific case of profiled SCA:

- When no additional assumption is done, one remains with all covariance matrices, one for each cluster tagged with the sensitive value s. In that case, the covariance matrices

are said *heteroscedastic*. The combination of a Gaussian template with heteroscedastic covariance matrices and the maximum likelihood distinguisher is also known as a Quadratic Discriminant Analysis (QDA) in the machine learning terminology [HTF09, Chap. 4.3].

- In opposition to the heteroscedastic assumption, the covariance matrices may eventually be assumed to be all equal to each other. In that case, the covariance matrices are said *homoscedastic*. This is an interesting assumption when one is guaranteed that the discriminative information is contained in the mean vector $\mathbf{M}_s$, since this enables to estimate only one covariance matrix, which we explained to be the critical task. The use of a Gaussian template with homoscedastic covariance matrices and the maximum likelihood distinguisher is also known as a Linear Discriminant Analysis (LDA) in the machine learning terminology [HTF09, Chap. 4.3]. This approach has been proposed by Choudary *et al.* at CARDIS'13 [CK13] under the name of *pooled* template.[8]

- In addition to the homoscedastic assumption, the covariance matrices may even be assumed to be diagonal. In other words, this means that all the samples $\mathbf{X}[t], t \in D$ are assumed to leak independently from each other. The use of a Gaussian template with a single diagonal covariance matrix and the maximum likelihood distinguisher is also known as a *naive Bayes* classifier in the machine learning terminology [HTF09, Chap. 6.6.3]. The soundness of this approach has been discussed by Picek *et al.* [PHG17].

It is worth emphasizing that although the SCA community almost always assumes that the leakage $\mathbf{X}$ follows a multivariate Gaussian law, templates may be obviously extended beyond this case. The interested reader may refer to the works of Heuser *et al.* discussing the latter assumption at CHES'14 [HRG14].

Generative vs. Discriminative. GTs are an example of a so-called *generative* model. This means that the leakage model, *i.e.*, the likelihood function $\Pr(\mathbf{X} \mid Z)$ may be used to generate synthetic traces. To this end, one may take a RNG to draw a random value $z \in \mathcal{Z}$ according to a uniform distribution; before a vector $\mathbf{x}$ according to the likelihood distribution $\Pr(\mathbf{X} \mid Z = z)$, estimated thanks to the profiling phase.

In other words, a generative model is able to make a discrepancy between the values $s \in \mathcal{Z}$ of the sensitive variable Z, by completely modelizing how such values would affect the input trace $\mathbf{X}$, even if some parts of the modelization do not enable to make any discrepancy between the underlying values of the sensitive variable. This is a more general problem than just guessing which is the most likely value of Z that is leaking from a given trace $\mathbf{x}$. The latter approach is called the *discriminative* model. Such models are typically estimated with machine learning algorithms, that we will present in Chapter 4.

Intuitively, estimating a discriminative model is a simpler task than estimating a generative one, since the latter one requires to build a complete model of $\mathbf{X}$, whereas the former one only focuses on the discriminative features of $\mathbf{X}$ in order to guess Z. This was phrased as follows by Vladimir Vapnik in his principle for solving problems using a restricted amount of information:

> "When solving a given problem, try to avoid a more general problem as an intermediate step." [Vap00, Chap. 1.9].

However, contrary to generative models, discriminative ones are more seen as black-box, since they do not always reveal the latent mechanism linking the variable to explain Z to the explaining variable $\mathbf{X}$.

[8] In SCA, the term LDA may either refer to the pooled templates, or to a dimensionality reduction technique, a.k.a. the *Fisher's LDA* [SA08].

Replacing a generative model by a discriminative one turns out to be possible in profiled attacks, according to the following lemma.

Lemma 2 (Discriminative distinguisher)**.** *Suppose that the attacker $\mathcal{A}$ knows the posterior probability distribution $\Pr(\mathrm{Z} \mid \mathbf{X})$ instead of the likelihood distribution $\Pr(\mathbf{X} \mid \mathrm{Z})$. Then the maximum likelihood distinguisher can equivalently be defined as:*

$$\mathcal{D}^{\mathrm{ML}}_{\mathcal{S}_a}[k] = \sum_{i=1}^{N_a} \log \Pr(\mathrm{Z} = \mathbf{C}(p_i, k) \mid \mathbf{X} = \mathbf{x}_i) + u \ , \tag{3.10}$$

where $u \in \mathbb{R}$ is a constant independent of k.

Proof. According to the Bayes' Theorem (see Equation 2.5), one have:

$$\Pr(\mathrm{Z} = \mathbf{C}(p_i, k) \mid \mathbf{X} = \mathbf{x}_i) \quad = \quad \Pr(\mathbf{X} = \mathbf{x}_i \mid \mathrm{Z} = \mathbf{C}(p_i, k)) \times \frac{\Pr(\mathrm{Z} = \mathbf{C}(p_i, k))}{\Pr(\mathbf{X} = \mathbf{x}_i)} \tag{3.11}$$

Since Z is uniform, $\Pr(\mathrm{Z} = \mathbf{C}(p_i, k))$ does not depend on k. That is why the logarithm of the likelihood and the posterior probability distributions are equal, up to an additive constant (*i.e.*, independent of k). Therefore, the distinguisher, as defined in both Equation 3.10 and Equation 3.8, will always imply the same key hypotheses ranking. □

The equivalent definitions of the maximum likelihood distinguisher enables the attacker to choose the distribution which fits the most its constraints, and in particular, opens the way towards the use of ML algorithms as we will see in Chapter 4.

3.5 Unprofiled Attacks

When profiled attacks are impossible – *e.g.* when one lacks a clone device behaving as an open sample – the attacker is reduced to make (strong) hypotheses on the leakage model, instead of accurately estimating it. The error induced on the guessing of the secret key may then be more or less sensitive to those hypotheses. That is why such a weaker attacker should also adapt its strategy in the key recovery.

Two main strategies can be used when facing a non-profiled attack context. The first one consists in using leakage models commonly adopted in the SCA literature. The latter one typically proposes some simple generative models although often relevant. Combined with those simple models, another distinguisher, namely the *correlation* distinguisher, is widely used. This gives an approach known under the name of Correlation Power Analysis (CPA), which will be addressed in Subsection 3.5.1.

The second one aims at addressing the case where no sound leakage model can be assumed concerning the acquired traces. In that case, the MI can be somehow used as a distinguisher, leading to the so-called Mutual Information Analysis (MIA). This approach being beyond the scope of this thesis, the interested reader may refer to the study conducted by Batina *et al.* [BGP+11].

3.5.1 Correlation Power Analysis

The aim of CPA is as follows. The attacker is given a uni-variate *additive noise* leakage model, typically under the form

$$(\mathbf{X}[t] \mid \mathrm{Z} = s) \propto \varphi(s) + \mathrm{B} \ , \tag{3.12}$$

where $\varphi : \mathcal{Z} \to \mathbb{R}$ is a deterministic function of the observation s of the sensitive target variable Z leaking at a time coordinate t, and B is a Gaussian zero-mean random variable independent from Z denoting the ambient noise. The attacker wants to test for which hypothetical value of the secret key the acquired traces from the attack set $\mathcal{S}_a$ fit *the most* the

assumed leakage model. Since Equation 3.12 assumes a linear relation between the random variables $(\mathbf{X}[t] \mid \mathrm{Z})$ and $\varphi(\mathrm{Z})$, one materializes this test by computing the correlation coefficient between $\mathbf{X}[t]$ and $\varphi\left(\mathbf{C}\left(\mathrm{P}, k\right)\right)$, for each time sample t and for each hypothetical value k, as stated by the following definition.

Definition 2 (Correlation Distinguisher)**.** *Given an attack set $\mathcal{S}_a$ and a leakage model φ, the* Correlation Distinguisher *is defined as:*

$$\mathcal{D}^{\mathrm{CPA}}_{\mathcal{S}_a}[k] \triangleq \max_{t \in [\![1, D]\!]} \left| \rho\left(\mathbf{X}[t], \varphi\left(\mathbf{C}\left(\mathrm{P}, k\right)\right)\right)\right| \ , \tag{3.13}$$

where

$$\rho\left(\mathrm{X}, \mathrm{Y}\right) \triangleq \frac{S_{\mathrm{X},\mathrm{Y}}}{\sqrt{S^2_{\mathrm{X}} \cdot S^2_{\mathrm{Y}}}} \tag{3.14}$$

denotes the empirical – a.k.a. Pearson *– correlation coefficient between* X *and* Y*.*

Indeed, given a key hypothesis k, the computation of the correlation coefficient for each time coordinate of the traces give a vector ρ of size D.[9] Provided that the leakage model is sound, when the right key $k^\star$ is tested, the computed correlation coefficient is expected to reach a significantly higher value where the leakage happens, *i.e.*, at Points of Interest (P.o.Is), than elsewhere in the resulting vector. Instead, when another wrong key candidate $\tilde{k}$ is tested, it is equivalent to test the fitness of another leakage model, namely $\varphi\left(\mathbf{C}\left(\mathrm{P}, \tilde{k}\right)\right)$. If this leakage model is highly non-linear with respect to the true leakage model $\varphi\left(\mathbf{C}\left(\mathrm{P}, k^\star\right)\right)$, then the resulting correlation coefficient computed at the same P.o.I is not expected to be distinguishable from the non-informative time coordinates.

Since the computed correlation coefficients are empirical estimations, the more traces in the attack set, the more likely an attacker is able to make a discrepancy between the score of the right key hypothesis and the wrong ones. Hence the interest of the correlation distinguisher to mount a SCA.

CPA has been first introduced, as is, by Brier *et al.* at CHES 2004 [BCO04].[10] But Le *et al.* [LCC+06] and Doget *et al.* [DPRS11] have shown that several attacks proposed in the early years of SCA since the seminal work of Kocher *et al.* [KJJ99], may be retrospectively reformulated as a CPA. The only difference with the Brier *et al.*'s work relies on the underlying leakage model, which will be discussed hereafter.

Heuser *et al.* have shown that if the true leakage model is uni-variate and is perfectly known by the attacker, then the linear correlation distinguisher is equivalent to the maximum likelihood distinguisher regarding Problem 1 [HRG14]. However, in a non-profiled context, one cannot guarantee that the assumed leakage model φ perfectly fits the true one, and here is where the correlation distinguisher takes advantage on the maximum likelihood: it is often less sensitive to approximation errors in the leakage model. Hence its wide use in non-profiled attacks.

Models to Approximate the Leakage. We review hereafter the different leakage models which can be proposed to approximate the true one. The choice depends on the knowledge of the attacker on the software and/or hardware architecture of the target device $\mathcal{T}$.

Classical leakage models come from the fact that, in Complementary Metal Oxide Semiconductor (CMOS) technology – which is used to realize the majority of existing integrated circuits, peaks of power consumption are observable when the output of the gates transition from either a '0' to '1' or a '1' to '0' logic state. At the scope of an 8, 16 or 32-bit register storing a targeted intermediate computation Z, the leakage model can then be described by

[9] D is recall to be the dimensionality of the traces, *i.e.*, the number of time samples in them.

[10] The term Correlation Electro-Magnetic Analysis (CEMA) may also be found when the traces denote acquisitions from an EM probe. Yet, we will not make any discrepancy between both terms.

the values of its bits and those of the previous intermediate computation Z' stored in the same register, that is:

$$\mathbf{X}[t] \;\mid\; \mathrm{Z} \propto \varphi\left(\mathrm{Z}, \mathrm{Z}'\right) + \mathrm{B} \;\;, \tag{3.15}$$

where φ is a deterministic function of two elements from $\mathcal{Z}$, and B is a random variable denoting the noise coming from the environment, *i.e.*, the neighbor registers and the measurement noise. The knowledge of Z and Z' may be guessed from the source code or the hardware architecture depending on the context. In that case, φ can even be simplified to only depend on the target variable Z: the dependency on the previous state is implicitly included in φ and in the noise term.

Since φ is a deterministic function of a discrete random variable taking finite values, it may be reformulated as a polynomial of the bits of Z, denoted hereafter as $\mathrm{Z}[0], \ldots, \mathrm{Z}[n-1]$, where n is the number of bits:

$$\varphi\left(\mathrm{Z}\right) = \sum_{\mathcal{J} \subset [\![0, n-1]\!]} \alpha_{\mathcal{J}} \prod_{j \in \mathcal{J}} \mathrm{Z}[j] \;\;, \tag{3.16}$$

where $\alpha_{\mathcal{J}} \in \mathbb{R}$. Doget *et al.* argue that for most targets $\mathcal{T}$, the attacker $\mathcal{A}$ may assume that the degree of φ, seen as a polynomial, is lower or equal to one [DPRS11]. In other words, the bits contribute to the power consumption – or the EM emanation – independently from each other, therefore ignoring the possible coupling effects between the logic gates storing those bits. Thus, the leakage model may be simplified to:

$$\varphi\left(\mathrm{Z}\right) = \sum_{i=0}^{n-1} \alpha_i \mathrm{Z}[i] \;\;, \tag{3.17}$$

where $\alpha_i \in \mathbb{R}$, which has the advantage to be linear with respect to the bits of Z. At this stage, the attacker must only guess the coefficients α_i. Depending on its knowledge and on the approximation error margin he may accept on its leakage modelization, the attacker may choose between several assumptions suggested hereafter, although not exhaustive.

- The coefficients may be assumed to be equal to each other. This corresponds to the so-called *Hamming weights* leakage model, denoted as $\mathrm{hw}(\mathrm{Z})$. This model is the one proposed by Brier *et al.* for their CPA [BCO04], and is, so far, the most widely used leakage model.

- The coefficients may otherwise be ignored – *i.e.* they are set to 0 – except one of them. Such a leakage corresponds to a *monobit* leakage model. In particular, when the non-null coefficient is α_0 (resp. α_{n-1}), the model is also known as a Least Significant Bit (l.s.b.) (resp. Most Significant Bit (m.s.b.)) model. Although not realistic, this model has the advantage to be very robust against approximation errors, since it involves few approximation assumptions on the leakage, thereby making the bridge with legacy Differential Power Analysis (DPA) [KJJ99].

- Eventually, the coefficients may be adjusted thanks to a linear regression with the traces from the attack set, for each key hypothesis. This approach is known as a *stochastic* attack or Linear Regression Analysis (l.r.a.) [SLP05].

It is also noticeable that Brier *et al.* suggest to choose the target sensitive variable Z as the output of the SubBytes instead of the output of the AddRoundKey. By including the Sbox inside φ, one ensures that the leakage induced by z wrong key hypothesis will be highly non-linear with respect to the one induced by the right key hypothesis. Therefore, this decreases the required number N_a of queries to distinguish the right key $k^\star$ with the correlation distinguisher – see [MOP07, Sec. 6.3.1] for an explanation.

3.6 Leakage Characterization and Pre-Processing

So far, we have seen that to emulate a sound attacker $\mathcal{A}$ in order to evaluate a target, one must get an estimation of the conditional p.m.f. $\Pr(\mathrm{Z} \mid \mathbf{X})$. This can be done either manually by the attacker thanks to the preliminary knowledge of the target behavior, or thanks to measured leakages during a preliminary characterization phase. The problem of estimating the latter conditional p.m.f. $\Pr(\mathrm{Z} \mid \mathbf{X} = \mathbf{x})$ based on measured data $\mathbf{x}$ is singular for two reasons.

On the one hand, SCA traces depict physical quantities varying through the time. By definition, these can be seen as continuous functions of the time, so they cannot be perfectly measured through the acquisition. A high-sampling rate oscilloscope can nevertheless discretize this signal with high fidelity, but at the cost of high dimensionality in the traces, ranging from several hundreds to several millions of samples, depending on the target implementation. This feature yields technical challenges concerning the use of some estimation algorithms. As an example, naively applying Gaussian Template Attacks (TA) on the raw traces of length D would require to estimate the $\mathcal{O}(D^2)$ coefficients of the covariance matrices. Hence, such a leakage model poorly scales when increasing the dimensionality of the traces.

On the other hand, in our cryptographic contexts, the relevant informative leakage is empirically known to be only localized in few time samples, the so-called P.o.Is. By relevant, we mean that those P.o.Is statistically depend – independently or jointly – on a sensitive variable, as formally stated by the following assumption.

Assumption 1 (Sparsity). *There only exists a small set of coordinates* $\mathcal{I}_{\mathrm{Z}} \triangleq \{t_1, \ldots, t_C \mid C \ll D\}$ *such that* $\Pr(\mathrm{Z}|\mathbf{X}) = \Pr(\mathrm{Z} \mid \mathbf{X}[t_1], \ldots, \mathbf{X}[t_C])$.

That is why it is worth considering methods able to target those P.o.Is. Not only, from an attacker's point of view, it enables to reduce the attack complexity by decreasing the dimensionality of the input traces, while keeping enough exploitable information to enable an attack to succeed. But also, by identifying the precise time samples where the informative leakage occurs, it allows an evaluator or a developer to guess the origin of the vulnerability, whether it comes from an element in a hardware circuit, or from a particular instruction in the assembly code for software implementations. In a sense, it helps to build a full diagnosis of the target device weaknesses.

There are two ways to proceed a dimensionality reduction. Either one may try to directly localize the P.o.Is, thanks to some tools exploiting the statistical dependencies of the trace at the informative time samples – detailed in the next section. Or one may use standard data compression techniques, such as Principal Component Analysis (PCA) [CDP15, SA08, EPW10, CK13, CK14], Kernel Discriminant Analysis (KDA) [CDP16], Discrete Fourier Transform (D.F.T.) or wavelet-based signal decomposition [DDFP20]. Contrary to the former way, the latter one does not enable to localize the P.o.Is. Hereafter, we propose an overview of the former way, especially since it will be used as a benchmark in Subsection 7.3.3.

3.6.1 Research of Points of Interest

In this section we present some basic tools, able to emphasize P.o.Is in the traces. We recalled in Subsection 3.5.1 that some non-profiled leakage models rely on a deterministic function φ of the sensitive variable, which coincides with statistical moments of the traces on the specific time coordinate where the leakage occurs. This is somehow a first method emphasizing P.o.Is. That is why assessing similarly statistical hypothesis tests on those samples can make a discrepancy between uninformative time samples and informative ones. We detail hereafter two ways to implement this idea.

T-test. The T-test characterization [MOBW13] is based on the eponymous statistical test. Its idea is to gather a profiling set into two classes: one, denoted by $\mathcal{S}_A$ of size n_A, with a same fixed sensitive value z_A, and another, denoted by $\mathcal{S}_B$ of size n_B with random sensitive values. The T-test assesses whether the two datasets share the same expected value or not. To this end, a *T-statistic* is computed according to Equation 3.18, for each time sample of the traces:

$$T[t] = \frac{\overline{\mathrm{X}_A}[t] - \overline{\mathrm{X}_B}[t]}{\sqrt{\frac{S_A^2[t]}{n_A} + \frac{S_B^2[t]}{n_B}}} \quad , \tag{3.18}$$

where $\overline{\mathrm{X}_A}[t]$ (resp. $\overline{\mathrm{X}_B}[t]$) denotes the empirical mean and $S_A^2[t]$ (resp. $S_B^2[t]$) denotes the empirical variance of the traces from $\mathcal{S}_A$ (resp. $\mathcal{S}_B$) at a given time coordinate t – see Subsection 2.2.1. Therefore, it outputs a characterization vector of the same dimensionality of that of the input traces, namely D.

If a time coordinate t does carry informative leakage, then there is at least one value $s \in \mathcal{Z}$ of the sensitive variable Z such that the probability distribution $\Pr\left(\mathbf{X}[t] \mid \mathrm{Z} = s\right)$ would differ from $\Pr\left(\mathbf{X}[t]\right)$. Therefore, it is likely that the expected values of the corresponding p.d.f.s, namely $\mathbb{E}\left[\mathbf{X}[t] \mid \mathrm{Z} = s\right]$ and $\mathbb{E}\left[\mathbf{X}[t]\right]$ are significantly different, hence a T-statistic significantly different from 0.[11] That is why such a test should reveal a vector emphasizing P.o.Is at the time samples where some informative leakage is carried. Moreover, according to Assumption 1, this vector should be sparse. An example of leakage characterization with a T-test is given in Figure 3.9a.

However, depending on the true leakage model, not all values $s \in \mathcal{Z}$ may yield a significant T-statistic. To circumvent this issue, the T-statistic may be averaged over the set $\mathcal{Z}$ of all possible values that Z may take. Likewise, the p.d.f.s $\Pr\left(\mathbf{X}[t] \mid \mathrm{Z} = s\right)$ and $\Pr\left(\mathbf{X}[t]\right)$ being different does not necessarily mean that their corresponding expected values also differ. This particularly happens when facing protected implementations, where the discriminative part of the leakage occurs in higher-order statistical moments.[12] To circumvent this issue, Standaert suggested at CARDIS'18 to replace $\mathrm{X}_A[t]$ $\left(\text{resp. } \mathrm{X}_B[t]\right)$ with $\left(\mathrm{X}_A[t] - \overline{\mathrm{X}_A}[t]\right)^d$ $\left(\text{resp. } \left(\mathrm{X}_B[t] - \overline{\mathrm{X}_B}[t]\right)^d\right)$, for a given $d > 1$.

Signal-to-Noise Ratio. The Signal-to-Noise Ratio (SNR) follows roughly the same idea. For each time sample t, the following statistic is estimated:

$$\mathrm{SNR}[t] \triangleq \frac{\underset{\mathrm{Z}}{\mathrm{Var}}\left(\mathbb{E}\left[\mathbf{X}[t] | \mathrm{Z} = s\right]\right)}{\underset{\mathrm{Z}}{\mathbb{E}}\left[\mathrm{Var}\left(\mathbf{X}[t] | \mathrm{Z} = s\right)\right]} \quad , \tag{3.19}$$

where the numerator denotes the signal magnitude and the denominator denotes the noise magnitude estimate. Like the T-test, provided that a time coordinate does carry some informative leakage, the deterministic part $\mathbb{E}\left[\mathbf{X}[t] | \mathrm{Z} = s\right]$ should not be constant with s, hence a non-zero variance, and thereby a non-zero SNR when the latter one is computed at an informative time coordinate. It is noticeable here that the denominator does not bring more information concerning the relevance of a time coordinate, but only scales the leakage characterization vector with the noise amplitude of the traces. This is then particularly useful when one wants to compare the relevance between several informative time coordinates.[13] Nevertheless, although requiring more traces to draw significant conclusions from the SNR than from a T-test, the former one does not depend on the choice of the the sets of traces to statistically compare [Sta18]. The interested reader may refer to see [MOP07, Sec. 4.3.2] for more details on its application in the SCA context.

[11] The T-statistic should be typically higher than 4.5.
[12] This will be discussed in Section 3.7.
[13] Provided that those time coordinates carry the same redundant information about the sensitive variable.

Characterization of Multi-Variate Leakages The SNR characterization techniques we have presented so far work in the case of uni-variate leakage, that is, when the marginal p.d.f. of $\mathrm{X}[t]$, for a given time coordinate $t \in [\![1, D]\!]$, depends on the sensitive variable Z. Therefore, both characterization techniques are able to emphasize all the P.o.Is verifying the preceding property. However, for some more complex leakage models, some time coordinates may still carry some information about the sensitive variable, *i.e.*, $\Pr(\mathbf{X}[t_1], \mathbf{X}[t_2] \mid \mathrm{Z} = s)$ is non-constant with respect to $s \in \mathcal{Z}$; although their respective marginal p.d.f.s, namely $\Pr(\mathbf{X}[t_1] \mid \mathrm{Z} = s)$ and $\Pr(\mathbf{X}[t_2] \mid \mathrm{Z} = s)$ remain constant with respect to s. Such leakage models are called *multi-variate* to emphasize the fact that several time coordinates must be jointly considered to find a dependency with the sensitive variable Z. Those leakage models are widely met with protected implementations – see Section 3.7.

To deal with the statistically based characterization methods, it is possible to use a pre-processing step of the traces. It consists in building a new pre-processed trace $\mathbf{X}'$, computed thanks to a so-called *re-combination* function ε applied on each possible tuple of a given size q of time coordinates from the trace, *i.e.*,

$$\mathbf{X}'[t_1, t_2, \ldots, t_q] = \varepsilon(\mathbf{X}[t_1], \mathbf{X}[t_2], \ldots \mathbf{X}[t_q]) \quad , t_1, t_2, \ldots, t_q \in [\![1, D]\!] \ . \tag{3.20}$$

The hope is that, for a value of q high enough, the uni-variate statistical moments of the new trace $\mathbf{X}'$ may contain new P.o.Is which would not be constant with respect to the sensitive variable Z. Prouff *et al.* have proposed sound recombination functions in the case of second order leakage models, *i.e.*, when $q = 2$ [PRB09]. The drawback of this method is that it mechanically increases the dimensionality of the new trace $\mathbf{X}'$ to D^q where D is the dimensionality of the original trace $\mathbf{X}$. This limitation is critical, even for small values of q, and is especially a cornerstone of the design of counter-measures by the developers.

3.7 Counter-Measures

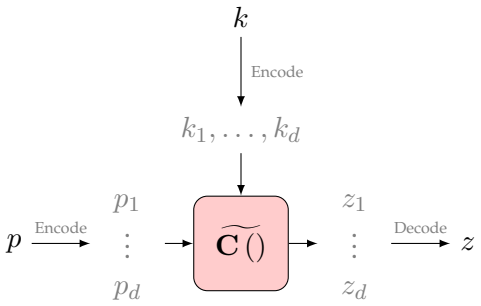
So far, we have seen that side-channel attacks may be easily exploitable by a potential attacker, as is. The ultimate goal of the developers is then to design secure implementations of a cryptographic primitive, while keeping the latter one running efficiently. The SCA terminology often uses the term *counter-measure*, defined hereafter, to refer to protections brought to an implementation.

Definition 3 (Counter-measure [MOP07, p. 167])**.** *A counter-measure is a set of modifications brought to an original implementation of a cryptographic primitive aiming at avoiding or at least reducing the dependency between the leakage and the sensitive intermediate values processed by the primitive.*

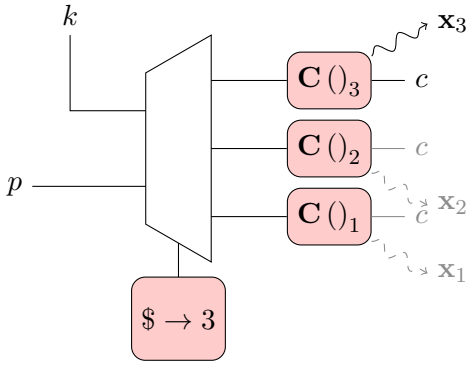
Behind Definition 3, there is the idea that a developer may find a way to control the quantity of informative leakage through its implementation, despite the fact that he may not have the control on the behavior of some hardware leaky components of the target device $\mathcal{T}$. In that sense, this draws a link with the notion of *leakage resilient* cryptography [KR19].

When dealing with the implementation of counter-measures, the general idea is to incorporate randomness, *e.g.*, by using values drawn from an RNG, unknown by the attacker during the attack phase – although those random values may also leak themselves through the acquired traces. This randomness added to the implementation will harden the key recovery in two ways. First, it acts as a source of entropy which is added to the p.m.f. of the true leakage model. This higher entropy mechanically decreases the information an attacker might optimally gather about the secret key. Second, it artificially increases the leakage model "complexity", which may protect against attackers with bounded power to guess the true leakage model.

There are mainly two approaches in the SCA literature about the design of counter-measures:



(a) Random data encoding.



(b) Random primitive code.

Figure 3.6: The two families of counter-measures in SCA.

- **Random data encoding**, depicted in Figure 3.6a: the developer uses a source of randomness to change the way any sensitive variable are encoded in the algorithm at each new execution.
- **Random primitive code**, depicted in Figure 3.6b: the developer uses a source of randomness – depicted as $ – to change at each execution the way the elementary operations of the algorithm are processed while keeping their semantic constant. The random behavior in the elementary operations induces many different random patterns – denoted by $\mathbf{x}_i$ in Figure 3.6b – at each new execution, even if the plaintexts and the encryption key remain constant.

The former approach is detailed in Subsection 3.7.1, while the latter approach is detailed in Subsection 3.7.2.

3.7.1 Random Data Encoding

Principle of Secret-Sharing. The first idea introduced so far to protect devices against SCA was to substitute the direct use of sensitive intermediate computations with a *secret-sharing* of those computations. It gathers lots of techniques, investigated under several names over the past 20 years, such as *duplication*, *masking*, *blinding*[14] or *random encoding*. We will briefly review those techniques.

Definition 4 (Encoding). *An n-encoding of a variable z representing the intermediate state of a computation, is a tuple $(z_0, \ldots, z_{n-1})$ of size n such that there exists a* decoding *function verifying:*

$$\mathsf{Dec}\left(z_0, \ldots, z_{n-1}\right) = z. \tag{3.21}$$

Definition 5 (Secret-Sharing [Bei11]). *Let $d \leq n+1$. An (n, d)-*secret-sharing *of the random variable $\mathrm{Z} \in \mathcal{Z}$ is an n-encoding verifying the two following properties:*

1. *For any subset $\mathcal{I}_{d+1} = (i_1, \ldots, i_{d+1}) \subset [\![0, n-1]\!]$ of size $d+1$, there exists a decoding function verifying*
$$\mathsf{Dec}_{\mathcal{I}_{d+1}}\left(\mathrm{Z}_{i_1}, \ldots, \mathrm{Z}_{i_{d+1}}\right) = \mathrm{Z} \ . \tag{3.22}$$
2. *For any subset $\mathcal{I}_d = (i_1, \ldots, i_d) \subset [\![0, n-1]\!]$ of size d, the secret variable Z and the d-tuple of random variables $\mathrm{Z}_{i_1}, \ldots, \mathrm{Z}_{i_d}$ are independent,*[15] *that is:*
$$\Pr\left(\mathrm{Z} \mid \mathrm{Z}_{i_1}, \ldots, \mathrm{Z}_{i_d}\right) = \Pr\left(\mathrm{Z}\right) \ . \tag{3.23}$$

[14] This term is specifically used for implementation of asymmetric cryptographic primitives.

[15] This condition may be relaxed for the so-called *non-perfect* secret-sharing. However, this is not used in this thesis and the interested reader may refer to the survey of Beimel [Bei11].

Remark 4. *It is straightforward to show that Equation 3.22 holding for any set of size d implies that it also hodls for any set of size strictly greater than $d+1$. Likewise, Equation 3.23 holding for any subset of size d implies that it also holds for any subset of size strictly lower than d.*

The parameter d is called the *order* of security induced by the secret-sharing. We call *scheme* a set of design rules enabling to:

1. compute a tuple $(\mathsf{Z}_0, \ldots, \mathsf{Z}_{n-1})$ which is a (n, d)-secret-sharing of a sensitive random variable Z,
2. propagate the encoding through the different elementary operations of a cryptographic primitive.

The goal of a developer is to find a scheme ensuring the given security requirements while minimizing the runtime and memory overhead due to the scheme. We will discuss these aspects while presenting the main schemes used so far in the literature.

Group Based Encodings. At first sight, finding an encoding function may look non trivial. Hopefully, there exists a generic way to implement secret-sharing schemes: if there is an inner operator $\cdot : \mathcal{Z}^2 \to \mathcal{Z}$ such that $(\mathcal{Z}, \cdot)$ is a group, then one may consider the scheme given in Algorithm 2.

Algorithm 2 Secret-sharing scheme based on a group operator

Require: $\mathsf{Z} \in \mathcal{Z}$, \$: RNG
Ensure: $\mathsf{Z}_0, \ldots, \mathsf{Z}_{n-1}$ is an $(n, n-1)$-secret-sharing
  $\mathsf{Z}_0 \leftarrow \mathsf{Z}$
  for $i = 1$ to $n-1$ **do**
    $\mathsf{Z}_i \leftarrow \$$
    $\mathsf{Z}_0 \leftarrow \mathsf{Z}_0 \cdot \mathsf{Z}_i$
  end for

The first scheme introduced so far is the *Boolean* scheme, which has been proposed by Goubin *et al.* at CHES'99 [GP99] and Chari *et al.* at CRYPTO'99 [CJRR99]. It considers the addition $\oplus$ in $\mathbb{F}_{2^8}$ as a group operator, which is nothing but the bit-wise `xor` between two bytes,[16] hence the name "Boolean". Other group laws have also been proposed: the *arithmetical* secret-sharing considers the modular addition $+$ over $\mathbb{Z}_n$, and has been introduced by Messerges *et al.* [Mes00], the *multiplicative* secret-sharing uses the field multiplication $\times$ between non-null elements in $\mathbb{F}_{2^8}$ and has been first used by Golic *et al.* [GT02]. The reason to prefer one group law from another is that depending on the cryptographic primitive, the sharing may be more or less easy to propagate through the elementary operations. In particular, it is trivial to apply an elementary operation which is commutative with the considered group law: it suffices to apply the operation on each share separately. As a consequence, any cryptographic primitive made of elementary operations commuting with a group law can be protected by a d-th order secret-sharing with a linear complexity with d. As an example for AES, the operations AddRoundKey, ShiftRows, MixColumns commute with the group law $\oplus$ of the AES field $\mathbb{F}_{2^8}$ but the SubBytes operation does not. Similarly, the power function in the latter operation commutes with the field multiplication $\times$ in $\mathbb{F}_{2^8}$ provided that the shares are non-null. We see here that all the operations of the AES do not commute with the same group law, which implies an important negative result: there is no trivial way to implement a secret-sharing scheme for this cryptographic primitive – at least based on group laws. We see hereafter how to cope with this difficulty. Since except for SubBytes, every other elementary operation of AES commutes with the addition $\oplus$ in $\mathbb{F}_{2^8}$, the idea is to

[16] See Section 2.3.

keep a Boolean scheme anyway, to propagate the sharing through the commutative operations, and to find a way to propagate the encoding through the SubBytes operation. Several methods have been proposed in the literature that we briefly review hereafter.

The first idea simply consists in proposing ways to switch between different schemes during the execution of the algorithm, so that at any time during the operation, the sensitive target variables are always shared according to a scheme commuting with the next elementary operation. This line of works has been initially considered by Genelle *et al.* [GPQ10], and further improved by Bettale *et al.* [BCZ18]. By switching from a Boolean secret-sharing to a multiplicative one just before the SubBytes, and switching back to Boolean secret-sharing after the power function, we circumvent the difficulty of the encoding propagation through the SubBytes. Unfortunately, the switching operations have a runtime complexity of $\mathcal{O}(d^2)$, which mitigates the advantages of having a commutative secret-sharing with the power function.

The second idea is known as *table re-computation* [AG01, PR07]. It is widely used for operations relying on l.u.ts.[17] We briefly describe its principle in the case of a $(2,1)$-secret-sharing based on a generic group law $\cdot$, although it can be extended without loss of generality to higher-orders [CRZ18]. It consists in initially drawing a pair of random elements $\mathsf{r}_{\mathsf{in}}, \mathsf{r}_{\mathsf{out}} \in \mathcal{Z}$, and based on the initial l.u.t. T, a modified table $\widetilde{T}$ is generated, such that:

$$\widetilde{T}[\mathsf{r}_{\mathsf{in}} \cdot \mathrm{Z}] = \mathsf{r}_{\mathsf{out}} \cdot T[\mathrm{Z}]. \tag{3.24}$$

Later in the algorithm, when one needs to apply T to a $(2,1)$-secret-sharing of a secret variable Z, it suffices to apply Algorithm 3. Coron investigated the extension of this scheme to higher-order [Cor14].

Algorithm 3 Propagation of a secret-sharing through a l.u.t.

Require: $A_0 = \mathrm{Z} \cdot \mathrm{M}, \mathrm{M} \in \mathcal{Z}$
Ensure: A_3, M is a $(2,1)$-secret-sharing of $T[\mathrm{Z}]$, A_1, A_2, A_3 independent from Z
$A_1 \leftarrow \mathsf{r}_{\mathsf{in}} \cdot A_0 \cdot \mathrm{M}^{-1}$
$A_2 \leftarrow \widetilde{T}[A_1]$
$A_3 \leftarrow {\mathsf{r}_{\mathsf{out}}}^{-1} \cdot A_2 \cdot \mathrm{M}$

The third idea is to exploit the algebraic properties of the Sbox, as proposed by Rivain and Prouff [RP10]. They remark that the non-linear part of the Sbox – namely the raising to the power 254 as recalled in Section 2.4 – can be decomposed into a sequence of few field multiplications and (linear) raisings to powers of the form s^{2^p} [Ter18, Lem. 5.3.4]. It turns out that the latter operations also commute with the field addition in $\mathbb{F}_{2^8}$. The problem can then be reduced to finding an implementation that propagates the secret-sharing through the remaining field multiplications – which are not of the form s^{2^p}. This can be done by extending to the Rijndael field $\mathbb{F}_{2^8}$ the so-called Ishai-Sahai-Wagner (I.S.W.) scheme [ISW03], originally computing field multiplications over the smaller field $\mathbb{F}_2$ for a Boolean secret-sharing at any order. It results in a global Boolean scheme with complexity $\mathcal{O}(d^2)$.

Beyond Group Law Based Encodings. We have seen a generic principle allowing to derive simple secret-sharing schemes based on group laws. The literature in cryptography however proposes many more ways to change the encoding of a sensitive information in a secured way.

Von Willich has first proposed the combined use of Boolean and multiplicative group laws into a so-called *affine* scheme [vW01]. This has been further investigated by Fumaroli *et al.* [FMPR10]. More precisely, the bytes of the AES state carrying the sensitive variable Z

[17] In the particular case where the Sbox may be randomly set, table re-computation might be the only working method.

are shared into α, β, and $\alpha \times \mathrm{Z} \oplus \beta$, where $\alpha \in \mathcal{Z}\backslash\{0\}$, and $\beta \in \mathcal{Z}$ are randomly drawn. The interest of the scheme relies on an improved security compared to a second order Boolean scheme at the cost of a runtime overhead close to that obtained for a first-order Boolean scheme. However, the affine scheme has only been proposed for a specific order, contrary to group based schemes, potentially usable for any order d.

The so-called *Shamir's* secret-sharing scheme, initially introduced in 1979 [Sha79], may also be used to share a sensitive intermediate computation. It has first been investigated simultaneously by Prouff *et al.* and Goubin *et al.* at CHES 2011 [PR11, GM11]. The scheme enables to generate a $(d+1, d)$-sharing of a sensitive intermediate variable Z for any order d. The principle is as follows. One first defines a polynomial P of degree at most $d+1$, whose coefficients are randomly drawn from $\mathbb{F}_{2^n}$ and such that the constant coefficient verifies $P(0) = \mathrm{Z}$. One then draws $d+1$ random public points $\alpha_i \in \mathbb{F}_{2^n}$. The $d+1$ shares are finally given by the evaluation $P(\alpha_i)$ of the random polynomial over the public points. According to the Lagrange interpolation, one is ensured that a necessary and sufficient condition to recover Z is to know the whole polynomial, *i.e.* the $d+1$ shares $(P(\alpha_i))_{i \in [\![1, d+1]\!]}$.

The *inner-product* scheme, introduced by Dziembowski *et al.* [DF12, GR15], then improved [BFG+17] and even generalized to code-based schemes [WMCS20, CGC+21], follows a similar idea. It consists in defining two vectors of random variables $\mathbf{L}$ and $\mathbf{R}$, each made of d elements from $\mathbb{F}_{2^n}$, such that the inner product $\langle \mathbf{L}, \mathbf{R} \rangle \triangleq \bigoplus_{i=1}^{d} \mathbf{L}[i] \times \mathbf{R}[i] = \mathrm{Z}$, where $\mathrm{Z} \in \mathbb{F}_{2^n}$ is the sensitive variable to protect. The vector $\mathbf{L}$, although randomly drawn, is typically let publicly known whereas the shares from $\mathbf{R}$ are supposed to be secret.

Soundness of Random Data Encoding A (n, d) secret-sharing is provably secure against any attack involving less than d shares, since according to Definition 5, no subset of less than d shares carry information about the sensitive shared variable Z. This result implies for example that uni-variate attacks such as CPAs cannot succeed against an implementation protected with a d-th order scheme – for $d \geq 1$ – as is, and that a pre-processing step such as the one described in Section 3.6 is necessary. Nevertheless, this does not fit the threat model presented in Section 3.1 where the attacker $\mathcal{A}$ has access to noisy observations of all the shares, since they are supposed to leak through the acquired traces from the attack set $\mathcal{S}_a$. In other words, this does not guarantee that any attacker is prevented from succeeding an SCA against the protected target. Hopefully, it has been recently shown in a series of papers, extending the seminal work of Chari *et al.* [CJRR99], that secret-sharing schemes remain theoretically *sound* against SCA. Informally, this means that for the cost of a polynomial growth in the performance overhead, the number of queries $N_a^\star$ required for an optimal attacker to recover the secret key would be lower bounded by $\mathcal{O}\left(\sigma^d\right)$, where σ is a parameter denoting the level of noise present in the traces for each of the shares, *e.g.*, the standard deviation [PR13, DDF19, DFS19, DFS16, PGMP19]. In a nutshell, the proofs use a common ingredient at the cornerstone of the secret-sharing soundness: the *noise amplification* effect. Indeed, if a scheme relies on a group operation, then the p.m.f. of the sensitive variable, recombined from the p.m.f. of each share separately, may be seen as a discrete convolution operation. Intuitively, convolutions are known to *smoothen* any function or distribution, provided that the initial shares' p.m.f.s are noisy *enough*. This smoothen effect is the source of noise amplification.[18]

3.7.2 Randomizing the Primitive Code

In Subsection 3.7.1 we have presented a way to protect an implementation by randomizing the encoding of the sensitive data processed through the execution. Though the latter approach may be theoretically sound, it is practically limited due to the performance overhead

[18] The interested reader may find more thorough discussion about the noise amplification effect in Appendix A.

incurred by the counter-measure.

An alternative approach consists in designing counter-measures especially sound against a specific type of attacker, not necessarily optimal but most likely to happen in reality. Therefore, the designed counter-measures, although not theoretically sound, can represent a particularly efficient alternative from a runtime and memory performance point-of-view. This is the idea behind the development of randomizing the operations processing the intermediate computations.

In a nutshell, without randomized code all the traces follow the same pattern and the sensitive intermediate computations are leaking at the same time coordinates, which makes the use of simple statistical tools particularly relevant for an attacker. Instead, some randomness in the execution of the elementary operations prevents the attacker from perfectly knowing the expected behavior of the traces, unless adopting specific strategies to mitigate the effects of randomness. In the following, we review how to implement randomization of the primitive code.

The implementation of a cryptographic primitive can be described at different levels, from the source code – if the target is a software device – to the hardware architecture. This implies that a developer is provided with a large spectrum of scopes on which randomization may be applied in the operations.

At the software or hardware levels, round-based cryptographic primitives – such as the AES – may be modified in order to incorporate *dummy* rounds. Inside those rounds, elementary operations may be augmented themselves with dummy operations which are not necessary to proceed the encryption/decryption. Likewise, for any function looping over independent elementary operations, the latter ones may be executed in a random order. At a thinner scope, the transcription of the source code into machine code may be randomized thanks to a code polymorphism approach.

Additional scopes of randomization are available at a hardware level. The use of asynchronous architecture [Ren00] makes the device prone to a *jitter* effect, causing misalignment in the traces. Likewise, the use of *dual-rail* logic [SS06] enables the target device to smoothen the power consumption or the EM emanations through the execution time, so that it removes the influence of the sensitive intermediate computation.

In the remaining of this section, we further describe some of those approaches which will be investigated in this thesis. The interested reader can also refer to Chapters 7 and 8 of the DPA book by Mangard *et al.* [MOP07].

Shuffling. This approach has first been introduced by Herbst *et al.* [HOM06], and extended by Rivain *et al.* [RPD09] and Veyrat-Charvillon *et al.* [VMKS12]. The idea of shuffling is to benefit from the many elementary functions in the AES made of independent (sequences of) operations. As an example in the AES, the AddRoundKey and SubBytes process each byte of the state independently from each other. Likewise, the ShiftRows and the MixColumns process respectively each row and each column independently from each other. This means that they can be processed in an arbitrary order. Whereas a naive implementation would process those operations in a trivial order, a protected implementation would leverage the independence in order to process them in a *random* order for each execution of the shuffled algorithm. This randomness prevents the attacker from perfectly knowing which sensitive variable may be targeted at a given time sample of the acquired trace. Indeed, the intermediate computation effectively leaking at a given time sample depends on the shuffled indices ordering the independent operations, that have been randomly drawn for this execution. Those indices cannot be assumed to be known by the attacker, though they can be guessed since the SCA traces also leak information about them. Intuitively, the more informative leakage about those indices, the less sound the shuffling counter-measure is.

The effect of shuffling on some attackers can even be quantified. Indeed, it has been shown that a shuffling over t different operations had the effect to divide the amplitude

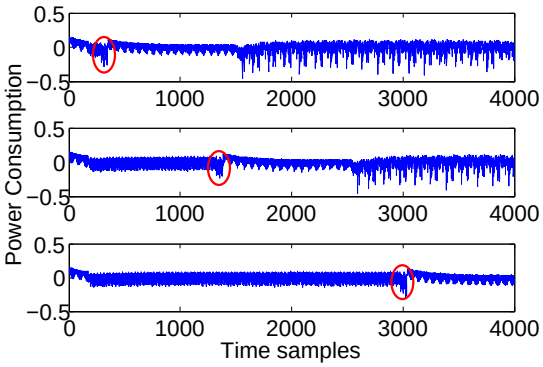


Figure 3.7: An example of dummy operation (nop) randomly inserted before an informative leakage (in red). Courtesy of Cagli *et al.* [CDP17].

of the peaks of SNR and CPA by a factor t [CCD00, Man04], which therefore requires t times more traces to succeed the attack compared to the same unprotected implementation. Moreover, Veyrat-Charvillon *et al.* showed an analogous effect of the shuffling on the MI between the leakage and the target sensitive variable, when considering an attacker with uni-variate leakage models [VMKS12].

Insertion of Dummy Operations. Although practically sound against attackers focusing only on a few time samples of the trace, the shuffling counter-measure suffers from an intrinsic limitation, *e.g.*, when applied to AES: there is not enough independent elementary operations to shuffle to make this approach practically sound against an attacker. That is why developers also consider adding dummy operations to the execution of the targeted implementation, such as suggested by Coron *et al.* at CHES'09 [CK09]. By definition, a dummy operation does not have any effect on the final computation, so it allows to somehow artificially increase the number of independent operations to shuffle. More precisely, rather than a perfect augmented shuffling, the insertion of dummy operations particularly provokes misalignment (a.k.a. *de-synchronization*) which are then propagated along the remaining of the trace, whereas shuffling does not induce this effect. Therefore, the developer can choose the desired effect on the statistical tools used by some potential attackers – see Section 3.5 and Subsection 3.6.1, to guarantee a security level against them.

Nevertheless, if the dummy operations do not leak in the same way as the sensitive operations, the former ones may be easily distinguished from the latter ones by the attacker, and a re-alignment operation may be proceeded in order to mitigate the effect of the counter-measure.

An example is given in Figure 3.7, depicting a trace chunk containing a leakage from the access of a l.u.t. carrying a sensitive information, preceded by the insertion of a random number of nop instructions. The location of the informative leakage is circled in red. Since the access to the l.u.t. and the nop do not span the same pattern in the leakage trace, it is not hard for the attacker to intuitively localize the informative leakage. Hence, provided that the attacker selects the right method, the attack is not likely to be hardened by this dummy operation insertion.

Code Polymorphism. Due to the skyrocketing production of Internet of Things (IoT), there is a need for the automated application of protections to improve products' resistance against SCA while keeping the performance overhead sufficiently low. In this context, some recent works proposed compiler toolchains to automatically apply counter-measures such as bit-slice masking [BDM+20] or a software hiding counter-measure called code polymorphism [BCHC18]. The working principle of the latter counter-measure relies on the execu-

tion of many variants of the machine code of the software device to protect, produced by a runtime code generator. The successive execution of many variants aims at producing variable side-channel traces in order to increase the difficulty to realize SCA. One must keep in mind that if code polymorphism is the only counter-measure applied to the target component, information leakage is still present in the side-channel traces. Yet, several works have shown the ability of code polymorphism and similar software mechanisms to be effective in practice against vertical SCA [ABP12, CBR+16], *i.e.*, up to the point that the leakage characterization techniques presented in Subsection 3.6.1, would not be able to detect information leakage in the traces, and that a CPA would require several millions of queries whereas the same attack on the unprotected version of the targeted implementation succeeded within a few hundreds traces [ABPS15, BCHC18].

We briefly describe the code polymorphism counter-measure applied by the toolchain used by Belleville *et al.* [BCHC18]. The compiler applies the counter-measure to selected critical parts of an unprotected source code: it inserts, in the target program, machine code generators, called Specialized Generators of Polymorphic Code (SGPCs), which can produce so-called polymorphic instances, *i.e.*, many different but functionally-equivalent implementations of the protected components. At runtime, SGPCs are regularly executed to produce new machine code instances of the polymorphic components. Thus, the device will behave differently after each code generation but the results of the computations are not altered. The toolchain supports several polymorphic code transformations, which can be selected separately in the toolchain, and most of them offer a set of configuration parameters. A developer can then set the level and the nature of polymorphic transformations, hence the amount of behavioral variability.

Hereafter, we detail some code transformations used in this thesis:

- **Register shuffling**: the index of the general purpose callee saved registers are randomly permuted.
- **Instruction shuffling**: the independent instructions are randomly permuted.
- **Semantic variants**: some instructions are randomly replaced by another semantically equivalent (sequence of) instruction(s). For example, variants of arithmetic instructions (e.g. `eor`, `sub`), remain arithmetically equivalent to the original instruction.
- **Noise instructions**: a random number of dummy instructions is added between the useful instructions in order to break the alignment of the leakage in the traces. Noise instructions are interleaved with the useful ones by the instruction shuffling transformation.

We emphasize on the fact that the sensitive variables are only manipulated by the polymorphic instances (*i.e.*, the generated machine code), and not by the SGPCs themselves. SGPCs are specialized code generators, and their only input is a source of random data (a RNG internal to the code generation runtime) driving the code generation. Hence, SGPCs only manipulate instruction and register encodings, and never manipulate secret data. Thus, performing an SCA on side-channel traces of executions of SGPCs cannot reveal a secret nor an information leakage. However, SGPCs manipulate data that are related to the contents of the buffer instances, *i.e.*, the structure of the generated code, the nature of the generated machine instructions (useful and noise instructions), *etc.* SCA performed on SGPC traces could possibly be helpful to reveal sensitive information about the code used by the polymorphic instances, but to the best of our knowledge, there is no such work in the literature. As such, this research question is out of the scope of this thesis.

3.8 Overview of the Used Datasets

We present in this section the different datasets of SCA traces which will be used for the experimental validation of our work in this thesis. Those datasets cover a large spectrum of use cases. Moreover, most of them are publicly available, which is of great interest for – fair – comparison with the state of the art. Some of those datasets are used in several parts and contributions in this thesis. That is why, for conciseness, we gather their description in a devoted section.

3.8.1 Chip Whisperer Dataset (CW)

This dataset has been used in the work we presented at CHES 2020 [MDP19b]. Although it does not depict the execution of a whole cryptographic primitive, it emulates the behavior of leakages of any secret-sharing scheme that may occur during the execution of assembly instructions in a software implementation.

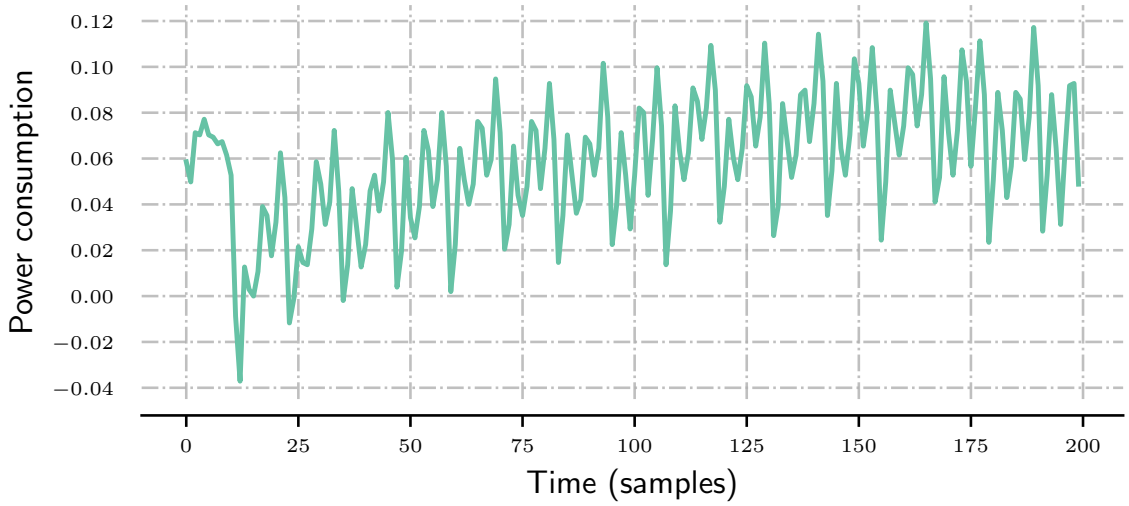
The Target. The leakage traces represent the power consumption of a XMEGA128D4 chip supported on a Chip Whisperer Lite board [OC14]. The firmware is directly written in assembly code. A pseudo-code is provided in Algorithm 4. It consists in iteratively loading a byte of a 16-byte plaintext array to a register of the MCU in order to provoke a physical leakage, then setting the value of the byte to zero and then storing it back in Random Access Memory (RAM). The operations are then repeated for each byte of the array.

Algorithm 4 loadData

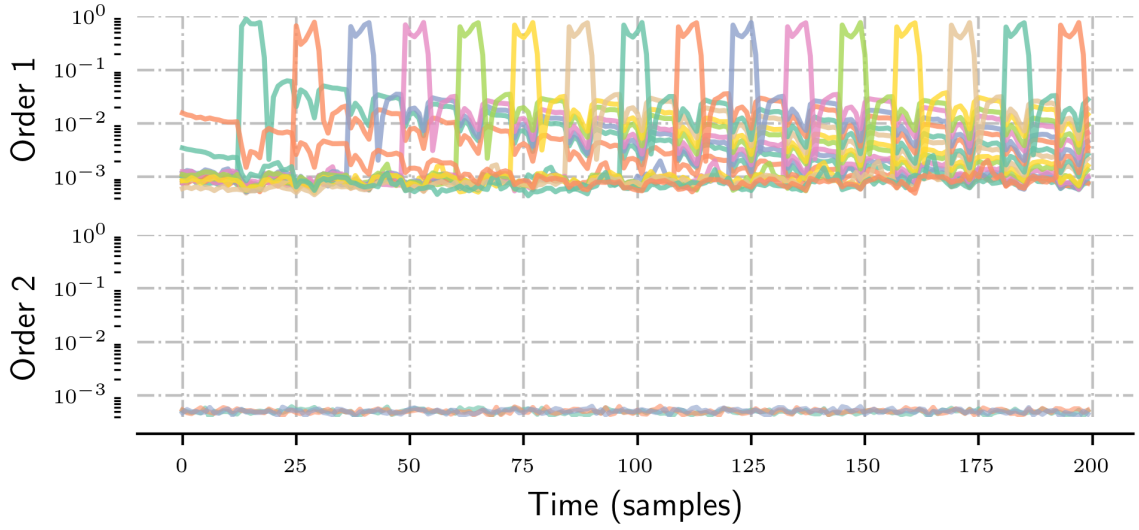
```
1: LD r0, X          ▷ Loads the first byte in r0
2: CLR r0            ▷ Clears the register
3: ST X, r0          ▷ Stores 0 in the plaintext array
4: LD r0, X          ▷ Do it again to clear the bus
5: CLR r0
6: ST X, r0
7: LD r0, X          ▷ One more time to be sure
8: CLR r0
9: ST X+, r0
```

$500,000$ traces of $2,500$ time samples each have been acquired, along with the corresponding bytes array denoted by `plain[i]`, $i \in [\![0, 15]\!]$. The complete acquisition has been done within 15 hours.

Quick Analysis of the Traces. Since this dataset will mainly be used to investigate DL-SCA in presence of secret-sharing, we would like to prevent any leakage jointly involving two bytes of the array. Figure 3.8a shows an example of one trace acquired through the platform. The 16 patterns denoting the execution of Algorithm 4 on each byte of the array are clearly distinguishable. We provide the corresponding SNR in Figure 3.8 (top) in different colors for each byte. In addition, we have also computed a SNR of order 2, that is, targeting the `xor` between two bytes, for any couple of bytes. The absence of peaks tends to confirm that there is no undesirable leakage, at least involving such a `xor`. However, a leakage between two bytes involving another relationship than a `xor` might still be informative, this does not allow to draw a sharp conclusion. Emphasizing such a leakage would require many more traces to reach a significant conclusion. That is why we let open this eventuality, although we remain confident that it is not likely to happen.



(a) Illustration of one trace



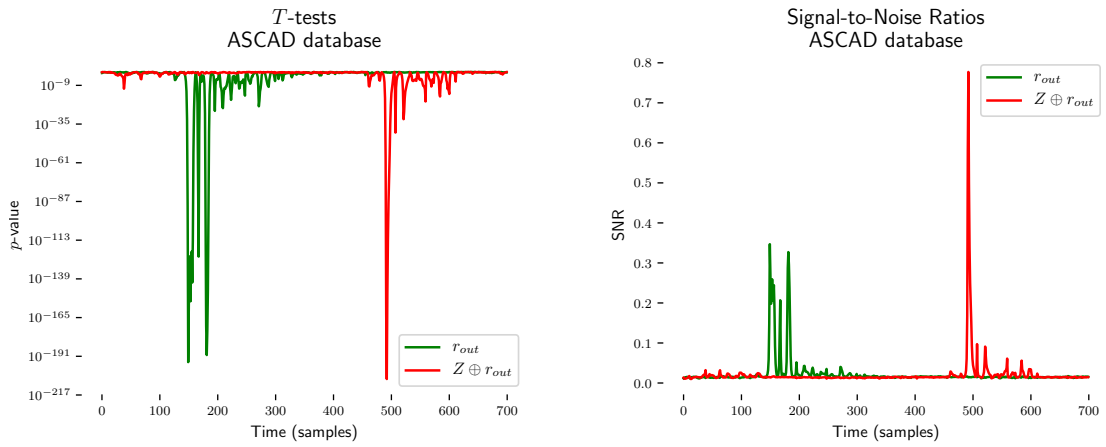
(b) The SNRs of order 1 and 2 (in log scale).

Figure 3.8: The CW dataset.

3.8.2 The ASCAD Dataset

The ANSSI's SCA Databases (ASCAD) dataset has been introduced in 2018 by Benadjila *et al.* [BPS+19] to provide the SCA community a benchmark to investigate and compare DL-based attacks. In particular, the aim is to assess to what extent deep learning is relevant to mount attacks against protected implementations.

The Target. The target is a protected software AES-128 implementation running over an ATMEGA-8515, which has an 8-bit AVR architecture. The software aims at protecting against first-order SCA, by using a Boolean secret-sharing scheme based on the table re-computation method (*cf* Algorithm 3), although the first two bytes of the AES state are not protected, for the sake of comparison. Typically, the targeted variable on this dataset is the third byte of the state, at the output of the Sbox in the first round, *i.e.*, $Z = \mathrm{Sbox}\,(p[3] \oplus k[3])$. The dataset provides $60,000$ traces, where $N_p = 50,000$ traces are used for profiling and $N_v = 10,000$ for validation, *i.e.*, for emulating attacks phases – see Subsection 3.2.3. For both datasets, the same fixed key has been used, while the plaintexts and the shares have been



(a) T-test (b) SNR

Figure 3.9: Leakage characterization with statistical tools over the ASCAD dataset, without artificial shift.

randomly drawn.[19] Whereas the whole traces, focused on the first AES round are made of $100,000$ time samples, this thesis will focus on the chunk corresponding to the interval $[\![45,400;46,100]\!]$, *i.e.*, $D = 700$. This window corresponds to the joint leakage of $Z \oplus r_{out}$ and r_{out}. Three versions of this dataset are available: the first one provides the traces as is, whereas the second and third ones provide the same traces on which an artificial shift of maximum amplitude of respectively 50 and 100 points has been applied. The traces are publicly available at https://github.com/ANSSI-FR/ASCAD.

Quick Analysis of the Traces. A characterization with the statistical methods presented in Subsection 3.6.1 is provided in Figure 3.9. Figure 3.9a depicts the characterization thanks to a T-test, while Figure 3.9b depicts the characterization done with a SNR. On those two plots, the green peaks emphasize the leakage of the random share r_{out}, while the red peaks denote the leakage of the output of the re-computed Sbox, namely $Z \oplus r_{out}$. The recombination of the two leakages would give access to information about the sensitive variable Z, which might be a privileged target for an attacker.

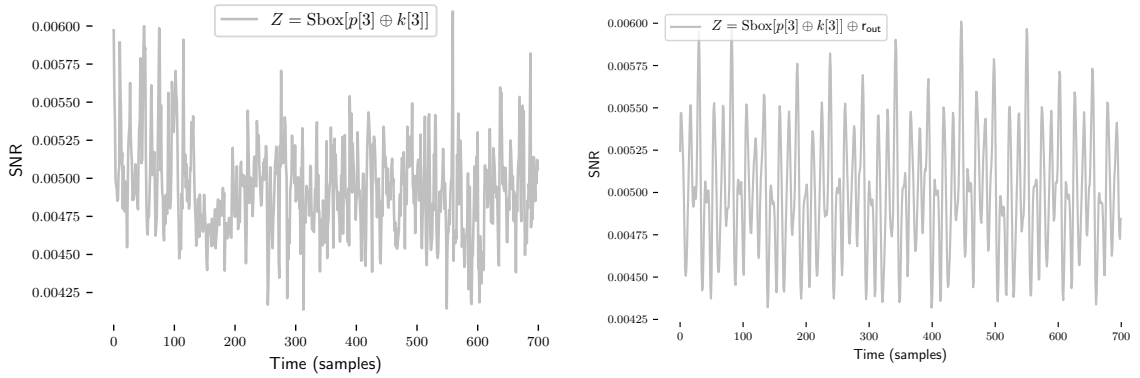
Nevertheless the latter characterization is successful because we have assumed the attacker (or the evaluator) to have access to the values of the random share r_{out}. This assumption turns out to be critical, as pointed in Figure 3.10a. This figure denotes the SNR directly computed over the sensitive variable Z, *i.e.*, without knowledge of the random share r_{out}. One can see on the plot that no clear peak appears, and that the level of SNR is much lower than in Figure 3.9b. This shows that without knowledge of the random share r_{out}, the SNR becomes unable to localize any P.o.I.

Likewise, the random shift artificially applied to the traces occurs a similar effect on the SNR computation, as shown in Figure 3.10b. Here again, no clear peak is emphasized on the plot. Hence, no P.o.I selection can be done on the traces thanks to statistical tools usually used for characterization. A way to circumvent this difficulty will be presented in Chapter 7.

3.8.3 Random Delay Dataset (AES-RD)

The AES - Random Delay (AES-RD) dataset has been released, following the works of Coron *et al.* on the insertion of random delays in the implementation of a software AES, as a way to implement the dummy operation insertion [CK09, CK10]. In a nutshell, it consists in drawing – thanks to the RNG – a random number of cycles during which a loop will iterate.

[19] Since the first release, a new version of the traces has been published, using a random key in the profiling traces, which are besides twice larger. This version is not investigated in this thesis.



(a) Characterization without knowledge of the secret shares.

(b) Characterization on randomly shifted traces ($T = 50$).

Figure 3.10: Effect of the counter-measures to the characterization on the ASCAD dataset.

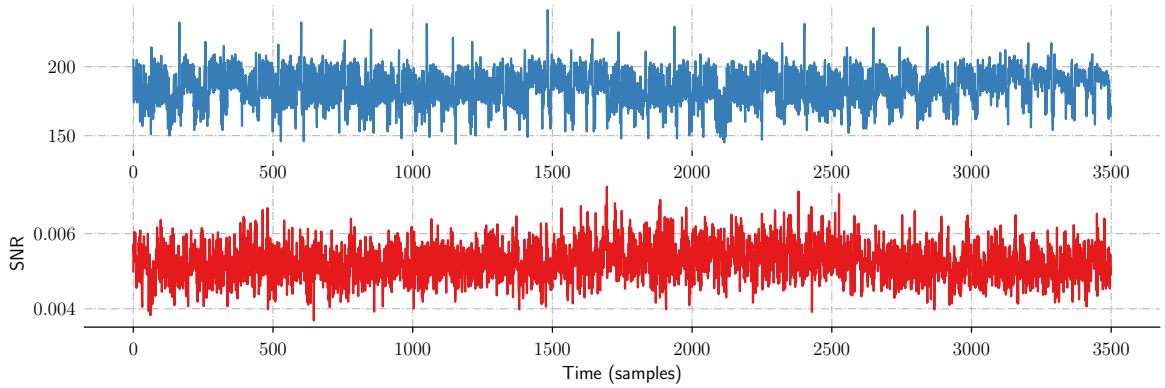


Figure 3.11: Top: An example of a trace from the AES-RD dataset. Bottom: the SNR computed over the whole dataset.

The Target. The target smart-card is an 8-bit Atmel AVR micro-controller, protected by a random delay counter-measure, which has an effect on the misalignment of the traces, making some attacks such as with GTs much harder. The targeted variable is the output of the first Sbox. The dataset is publicly available at https://github.com/ikizhvatov/randomdelays-traces. $50,000$ traces of $D = 3,500$ time samples each are provided, denoting the power consumption of the target. These power traces have been previously compressed by selecting 1 sample (peak) from each CPU clock cycle. At least the first (non-dummy) AES round is covered. In this thesis, we split the dataset into $N_p = 40,000$ profiling traces and $N_v = 10,000$ validation traces.

Quick Analysis of the Traces. We provide in Figure 3.11 an example of trace from the dataset (top), along with a characterization with SNR over the whole dataset (bottom). As expected, due to the misalignment effect of the random delay counter-measure, no peak denoting a leakage is emphasized. This confirms that either a pre-processing phase including re-alignment or the use of a misalignment-resilient methods is necessary to deal with those traces.

3.8.4 AES on FPGA (AES-HD)

The AES - Hamming Distance (AES-HD) dataset has been released by Picek *et al.* at CHES 2019 [PHJ+18], in order to introduce a dataset of SCA traces targeting a hardware implementation whereas the majority of the public datasets are focused on software implementations.

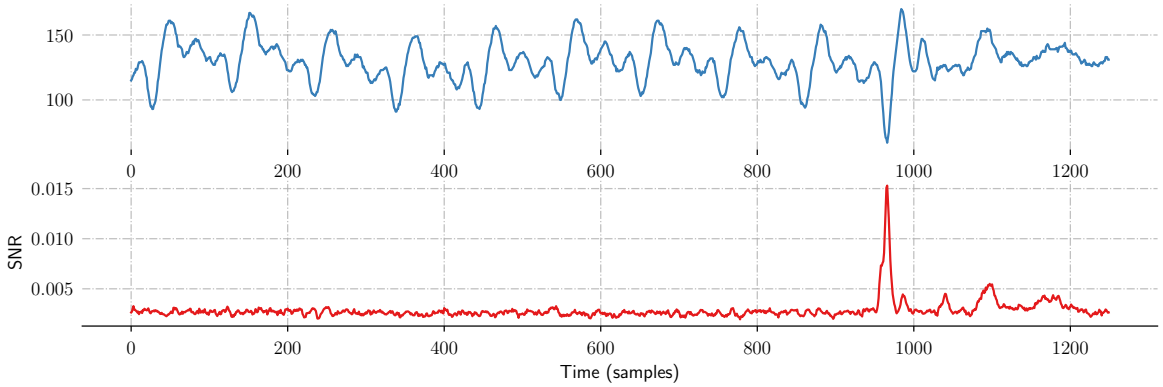


Figure 3.12: Top: one trace of the AES-HD dataset. Bottom: The SNR computed over the whole dataset.

Moreover, this dataset is an example of use-case where the targeted sensitive variable comes from the last round of the AES encryption.

The Target. We recall hereafter the description provided by Picek *et al.* in their paper. This is an "unprotected implementation of AES, written in VHSIC Hardware Description Language (VHDL) in a round based architecture taking 11 clock cycles for each encryption. The design was implemented on a Xilinx Virtex-5 Field-Programmable Gate Array (FPGA) of a SASEBO GII evaluation board. Side-channel traces were measured using a high sensitivity near-field EM probe, placed over a decoupling capacitor on the power line." The dataset is publicly available at `https://github.com/AESHD/AES_HD_Dataset`.

The authors recommend to use an intermediate leakage model φ corresponding to the Hamming distance between the targeted byte of the state before applying the Sbox of the last round, and the final ciphertext byte. In the following of this thesis, we will rather target the input of the last AddRoundKey, without considering any prior leakage model.

Quick Analysis of the Traces. Figure 3.12 provides a brief insight of the traces. One can guess 10 similar patterns corresponding to the 10 rounds of AES. The peak of SNR appears on the last pattern, confirming that the targeted sensitive variable is an intermediate computation of the last round. Although the peak is clearly distinguishable, one may remark that the peak is at approximately 0.15, which is lower than the preceding SNRs observed in Figure 3.8 and Figure 3.9b. This is expected since hardware implementations are known to usually leak less information.

3.8.5 Polymorphism Dataset

In Chapter 6, we will present the investigations conducted on the security of the code polymorphism counter-measure proposed by Belleville *et al.* [BCHC18]. To this end, we conducted an acquisition campaign of SCA traces over two out of the 15 implementations used in their benchmarks, namely the AES 8-bit and the mbedTLS that we briefly describe hereafter, along with the details of the experimental setup and a preliminary analysis.

The mbedTLS Implementation. This 32-bit implementation of AES from the ARM library [ARM19] follows the so-called *T-table* technique [DR02]: the 16-byte state of AES is encoded into four `uint32_t` variables, each representing a column of the state. Each round of the AES is done by applying four different constant l.u.ts stored in flash memory.

The AES 8-bit Implementation. This is a simple software unprotected implementation of AES written in C, and manipulating only variables of type `uint8_t`, similar to [Sma20]. The SubBytes operation is computed byte-wise thanks to a l.u.t., stored in RAM. This reduces information leakage on memory accesses, compared to the use of the same l.u.t. stored in flash memory.

Target Device. We ran the different AES implementations on an STM32 NUCLEO F303 board, embedding an ARM Cortex-M4 32-bit core [STM]. This device does not provide any hardware security mechanisms against side-channel attacks. This core originally operates at 72 MHz, but the core frequency was reduced to 8 MHz for the purpose of side-channel measurements. The target is similar to the one used by Belleville *et al.* [BCHC18], who considered a Cortex-M3 core running at 24 MHz. These two micro-controllers have an in-order pipeline architecture, but with a different pipeline organization. Thus, we cannot expect those two platforms to exhibit the same side-channel characteristics. However, our experience indicates that these two experimental setups would lead to similar conclusions regarding the attacker models considered in our study. Similar findings on similar targets have also been reported by Heuser *et al.* [HGMG20]. Therefore, we assume that the differences of side-channel characteristics between our targets and the Belleville *et al.*'s ones should not induce major differences in the results of such side-channel analysis.

Configuration of the Code Polymorphism Counter-Measure. For each evaluated implementation, the code polymorphism counter-measure is applied with a level corresponding to the configuration "high" described by Belleville *et al.* [BCHC18]: all the polymorphic code transformations are activated, the number of inserted noise instructions follows a probability distribution based on a truncated geometric law. The dynamic noise is activated and SGPCs produce a new polymorphic instance of the protected code for *each* execution (*i.e.*, the regeneration period is set to 1).

Acquisition Setup. We measured SCA traces corresponding to EM emanations with an EM probe RF-B 0.3-3 from Langer, equipped with a pre-amplifier, and a Rohde & Schwarz RTO 2044 oscilloscope with a 4 GHz bandwidth and a vertical resolution of 8 bits. We set the sampling rate to 200 MS/sec., with the acquisition mode "*peak-detect*" which collects the minimum and the maximum voltage values over each sampling period. We first verify that our acquisition setup is properly set. This is done by acquiring several traces where the code polymorphism is de-activated. Thus, we can verify that those traces are synchronized. Then, computing the T-test (*cf* Subsection 3.6.1) enables to quickly[20] assess whether the probe is correctly positioned and the sampling rate is high enough. Then, after re-activating the code polymorphism, $100,000$ profiling traces are acquired for each target implementation. Each acquisition campaign lasts about 12 hours.

Preliminary Analysis of the Traces. We detail hereafter a preliminary analysis of the acquired traces. The aim is to restrict as much as possible the target region acquired to a window covering the entire first AES round. Therefore there would not be any loss of informative leakage about the sensitive intermediate variable targeted in those experiments. In addition to that, a uni-variate leakage assessment, by computing the SNR, is provided hereafter in order to verify that there is no trivial leakage.

mbedTLS. We ran some preliminary acquisitions on 10^7 samples, in order to visualize all the execution. We could clearly distinguish the AES execution with sparse EM peaks,

[20] *i.e.*, faster than with a SNR computation.

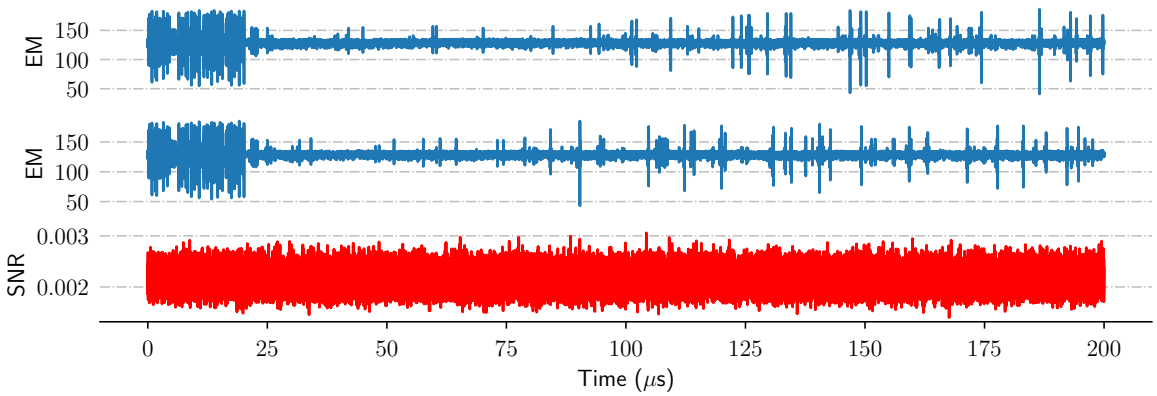


Figure 3.13: Acquisitions on the mbedTLS implementation. Top: two traces containing the first AES round. Bottom: SNR computed on the $100,000$ profiling traces.

from the call to the SGPC with more frequent EM peaks. This enabled to focus on the first 10^6 samples of the traces corresponding to the AES execution.

Actually, the traces restricted to the AES execution seem to remain globally the same between each other, up to local elastic deformations along the time axis. This is in line with the expected effect of code polymorphism, since it involves transformations at the machine code level. Likewise, 10 patterns could be distinguished on each trace, which were clues to expect that it would correspond to the 10 rounds of AES. That is how we could restrict again our target window to the first round, up to comfortable margins because of the misalignment effect of code polymorphism. This represents $80,000$ samples. An illustration of two traces restricted to the first round is given by Figure 3.13 (top).[21]

The SNR denoting here the potential uni-variate leakage of the first output byte of the SubBytes operation is computed based on the $100,000$ acquired profiling traces, and is plotted in Figure 3.13 (bottom). No distinguishing peak can be observed, which confirms the soundness of code polymorphism against attacks requiring the P.o.Is of the raw traces to be aligned with each other, *e.g.*, the CPA or the GT.

However, we observe in each trace approximately 16 EM peaks corresponding to the number of memory accesses to the l.u.ts per encryption round. These memory accesses are known to carry sensitive information. This suggests that a trace re-alignment on EM peaks might be relevant to successfully achieve such attacks.[22]

AES 8-bit. We proceed in the same way for the evaluation of AES 8-bit implementation. As with the mbedTLS traces, we could identify 10 successive patterns likely to correspond to the 10 AES rounds. Therefore we reduce the target window at the oscilloscope to the first AES round, which represents $160,000$-dimensional traces plotted in Figure 3.14 (top). This growth in the size of the traces is expected, since the naive AES 8-bit implementation is not optimized to be fast, contrary to the mbedTLS one.

Yet, here the peaks are hardly distinguishable from the level of noise. This is expected, due to the l.u.ts being moved from the flash memory to the RAM. As a consequence, the memory accesses are less remarkable. That is why a trace re-alignment does not seem relevant here.

Finally, Figure 3.14 (bottom) shows the SNR on the raw traces, ensuring once again that no trivial leakage can be exploited to recover the secret key.

[21] Here the first 20 μs of both traces correspond to a non-critical part of the implementation, probably due to the Operating System (OS) of the chip. Hence the apparent synchronization here.

[22] A re-alignment process is proposed in Section 6.2.

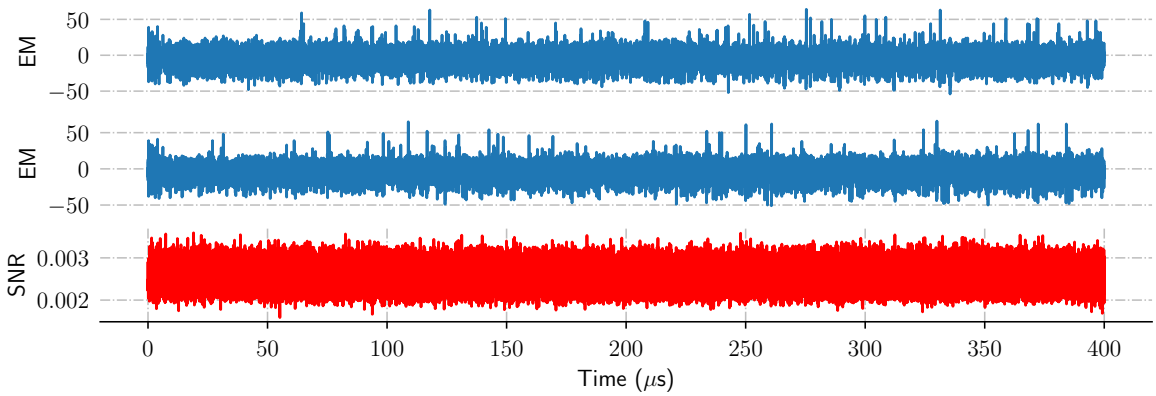


Figure 3.14: Acquisitions on the AES 8-bit implementation. Top: two traces containing the first AES round. Bottom: SNR computed on the $100,000$ profiling traces.

3.9 Conclusion

In this chapter, we have presented the attack scenario considered in this thesis. Starting from the so-called "gray-box" scenario (Section 3.1), the latter one may be augmented with a preliminary profiling phase with the help of an open-sample clone device of the target. This leads to the profiling attack scenario presented here (Figure 3.5). We have stated that in this scenario, the attack is jointly defined by two elements: the choice of the distinguisher (Definition 1), and the design of the leakage model. In particular, the profiling phase aims at estimating the true leakage model as precisely as possible, in order to reach a satisfying solution of the fundamental goal of the SCA evaluator, as stated by Problem 1 and Theorem 1.

From a practical perspective, the design of the leakage model often requires a preliminary dimensionality reduction, *e.g.*, with the help of a P.o.Is selection method. We have briefly presented two statistical tools in Subsection 3.6.1 enabling such a selection, illustrated in the presentation of the datasets investigated through the experiments of this thesis (Section 3.8).

Regarding the way that the vulnerabilities of a leaky device could be exploited by potential attackers, a developer is not unarmed. Hopefully, he can indeed control the quantity of informative leakage resulting from the execution of the targeted implementation through several means, namely the randomization of the sensitive data encoding (a.k.a. secret-sharing, Subsection 3.7.1) or the randomization of the operations (a.k.a. hiding, Subsection 3.7.2). Provided with those counter-measures, the developer has the power to trade off runtime and memory performance against security.

Yet, several questions remain unanswered at the end of this chapter. Indeed, although we have presented the attack techniques against unprotected implementations, we did not thoroughly discussed how the attacker can adapt its techniques – or even adopt new ones – to the counter-measures presented so far. This issue will be investigated in Chapter 5, when we will address the efficiency of neural networks as a way to modelize the posterior conditional p.m.f. of the leakage, for a protected implementation.

Besides, we have quickly mentioned to what extent the machine learning techniques can be integrated to the profiling attack framework presented in this chapter. In Chapter 4, we will dive in more details about this fact, and we will see that the profiling attack framework and the machine learning framework are somehow intertwined.

Chapter 4

Deep Learning for Side-Channel Analysis

" If someone can explain every phenomenon, his explanations are worthless."

Shalev-Shwartz and Ben-David [SSBD14]

Contents

We have presented the SCA framework in Chapter 3, and the profiling attacks in particular. Indeed, the profiling phase in our considered attack scenario aims at leveraging the access to the traces measured on the clone device in order to improve the modelization of the leakage behavior of the target device. We will see in this chapter that the latter process may be encompassed into the field of machine learning. This chapter is devoted to introduce the necessary notions of this field to discuss the use of DL-based SCA later in this thesis.

In Section 4.1 we present the theoretical notions of ML. Then, Section 4.2 and Section 4.3 briefly recall the main principles of DL, before we propose a review of its use in SCA in Section 4.4. This will serve as a way to legitimate the outcomes of our research through this thesis in the next chapters.

4.1 The Statistical Learning Theory

4.1.1 Position of the Problem

We stated in Section 3.3 that from an evaluator's point of view, it would be optimal to use the maximum likelihood distinguisher defined by Equation 3.8. Unfortunately, it requires to know the true conditional p.m.f. $\Pr(Z \mid \mathbf{X})$, which is unknown in practice. Instead, the evaluator or the attacker can substitute the latter one with a model $F : \mathcal{X} \to \mathcal{P}(\mathcal{Z})$, giving the following surrogate distinguisher:

$$\mathcal{D}^F_{\mathcal{S}_a}[k] = \sum_{i=1}^{N_a} \log F(\mathbf{x}_i)\left[\mathbf{C}\left(p_i, k\right)\right] \quad , \tag{4.1}$$

where $\mathcal{S}_a \triangleq \{(\mathbf{x}_1, p_1), \ldots, (\mathbf{x}_{N_a}, p_{N_a})\}$ is the attack set acquired on the actual target $\mathcal{T}$ – see Subsection 3.1.1.

We consider hereafter the framework of profiled attacks presented in Section 3.4: the attacker $\mathcal{A}$ has a clone device $\mathcal{T}'$ of the actual T.O.E. $\mathcal{T}$, on which he acquires the profiling dataset $\mathcal{S}_p \triangleq \{(\mathbf{x}_1, z_1), \ldots, (\mathbf{x}_{N_p}, z_{N_p})\}$. The clone is behaving as an open sample, so the values $z_1, \ldots, z_{N_p}$ of the sensitive intermediate variable targeted by the attacker during the profiling phase are known, contrary to the same values processed throughout the attack phase. Based on $\mathcal{S}_p$, the role of the profiling phase is, to build a *sound* surrogate model $F : \mathcal{X} \to \mathcal{P}(\mathcal{Z})$. Here, "sound" refers to the efficiency of an attack defined by Equation 3.4. In the remaining of this thesis, we will denote by $N_a\left(F, o, \beta\right)$ the efficiency of the attack using the distinguisher $\mathcal{D}^F_{\mathcal{S}_a}$, following the definition given by Equation 4.1, namely $N_a\left(\mathcal{D}^F_{\mathcal{S}_a}, o, \beta\right)$. Likewise, as in Subsection 3.2.2, we may omit the mention to o and β, implicitly set respectively to 1 and 90 %, in order to lighten the notations.

As a consequence, we may also refine the main goal of the evaluator emulating an attacker $\mathcal{A}$ in view of assessing the worst-case attack scenario, as stated in Problem 1:

Problem 2 (Profiled SCA Optimization)**.** *Given a profiling set $\mathcal{S}_p$, find the model $\mathcal{A}(\mathcal{S}_p)$ minimizing the SCA efficiency metric $F \mapsto N_a\left(F\right)$, as defined in Equation 3.4.*

We will see in the next section that the latter problem can be encompassed into the more general framework of Machine Learning (ML). This point of view enables to better understand how to efficiently address Problem 2. To this end, we first provide in the next section a clear definition of the term "learning".

4.1.2 Definition of a Learning Algorithm

The more cited definition of *learning* has been proposed by Mitchell in 1997 [Mit97]:

> "A computer program is said to learn from experience E with respect to some task T and performance measure P, if its performance on T, as measured by P, improves with experience E."

Programming a machine to achieve a task by learning is particularly useful when the given task is too complex to be programmed by hand. We detail hereafter the different elements of the definition of learning in our profiled SCA context.

The Task. In the context recalled in Subsection 4.1.1, the task of the attacker is to build a mapping $\mathcal{X} \to \mathcal{P}(\mathcal{Z})$. In machine learning, it is usual to precise the *hypothesis class*, denoted by $\mathcal{H} \subset \{F : \mathcal{X} \to \mathcal{P}(\mathcal{Z})\}$, from which the model is selected. A learning algorithm is not only defined by the hypothesis class $\mathcal{H}$ from which it selects the *best* model – according to the defined performance measure – but also by the method it uses to select the model – *i.e.* the algorithm as itself.

The Performance Measure. Likewise, Problem 2 directly provides the relevant performance measure in our context, namely the SCA efficiency metric $F \mapsto N_a(F)$. Contrary to the common sense, the performance is said to improve whenever the measure performance decreases. This convention is more generally adopted by the machine learning community, where the performance metric is usually called the *loss*, in order to remove the ambiguity. Formally speaking, a loss function is a mapping:

$$\ell : \left| \begin{array}{ccc} \mathcal{P}(\mathcal{Z}) \times \mathcal{Z} & \longrightarrow & \mathbb{R}_+ \\ \mathbf{y}, z & \longmapsto & \ell(\mathbf{y}, z) \end{array} \right. , \tag{4.2}$$

where $\mathbf{y} = F(\mathbf{x})$ would denote the output – *i.e.* a vector describing a p.m.f. here – of the model F returned by the learning algorithm for an input data $\mathbf{x}$, and z would denote the value that one expects the learned model to predict given $\mathbf{x}$.

The Experience. The experience describes the way data and information are accessed by the learning algorithm during learning. Two types of experience may be distinguished, a.k.a. *active* vs. *passive*.[1] In a passive experience, the learning algorithm is given some data collected by a third part which it has no way to interact with, *i.e.* the process of data collection is independent from the learning algorithm. On the opposite, an active experience allows the learning algorithm to influence the data collection process. The latter type of experience covers the *reinforcement* learning framework [SB98]. Although beyond the scope of this thesis, this approach may be interesting in order to find relevant strategies[2] of adaptive chosen-plaintext attacks, as mentioned in Section 3.1. Nevertheless, the definition of our attack scenario implies that we consider only learning algorithms with passive experience.

In the context of the profiling attack scenario described in Section 3.4, the experience is fully defined by the profiling set $\mathcal{S}_p$ of size N_p. That is why we can say that the experience increases whenever N_p increases. Since the profiling set $\mathcal{S}_p$ contains the values $(z_i)_i$ of the targeted intermediate computations corresponding to the acquired profiling traces – referred as *labels* in the ML terminology, the learning is said to be *supervised*. In a more restricted case, beyond the scope of this thesis, the learner is not assumed to know those labels, hence denoted as an *unsupervised* learning.

Although intuitive and simple, the definition given by Mitchell is not sufficient, since the notion of "improvement" in the definition is not precise enough. That is why, we complete it with the definition of learnability given by Shalev-Shwartz and Ben-David hereafter.

Definition 6 (Learnability [SSBD14, Def. 3.4]). *A hypothesis class $\mathcal{H}$ is* learnable *with respect to an input data space $\mathcal{X}$, an output data space $\mathcal{Z}$, and a loss function ℓ, if there exists a learning*

[1] This terminology must not be confounded with a similar one introduced in p. 5. In the latter one, the term "active" is often used for a scenario of physical attacks, *e.g.* fault attacks, where the attacker attempts to perturb the behavior of the target device, in opposition to passive attacks such as SCA where the attacker only observes the target device.

[2] The exact term used in reinforcement learning is *policy*.

algorithm[3] $\mathcal{A}$ *such that for every probability distribution over* $\mathcal{X} \times \mathcal{Z}$*, when running the learning algorithm on a profiling set* $\mathcal{S}_p$ *of* $N_p = |\mathcal{S}_p|$ *i.i.d. samples, the algorithm returns* $\mathcal{A}(\mathcal{S}_p) \in \mathcal{H}$ *such that:*

$$\mathcal{L}_{\mathbf{X},\mathrm{Z}}\left(\mathcal{A}(\mathcal{S}_p)\right) \xrightarrow[|\mathcal{S}_p|\to\infty]{\mathcal{P}} \min_{F\in\mathcal{H}} \mathcal{L}_{\mathbf{X},\mathrm{Z}}\left(F\right) \quad , \tag{4.3}$$

where $\mathcal{L}_{\mathbf{X},\mathrm{Z}}\left(F\right) \triangleq \underset{\mathbf{x},z}{\mathbb{E}}\left[\ell\left(F(\mathbf{x}), z\right)\right]$.

In other words, the definition of "learning" refers to a convergence in probabilities to the best possible model, according to the loss function. Like every notion of convergence, one may define a notion of speed of convergence. In machine learning, this notion is also known under the name *sample complexity*, that we define hereafter.

Definition 7 (Sample Complexity [SSBD14, Sec. 3.2])**.** *The* sample complexity *of a hypothesis class* $\mathcal{H}$ *is the maximum convergence rate – as defined by Equation 2.10 – of the sequence* $\left(\mathcal{L}_{\mathbf{X},\mathrm{Z}}\left(\mathcal{A}\left(\mathcal{S}_p\right)\right)\right)_{N_p}$ *over the set of every probability distribution over* $\mathcal{X} \times \mathcal{Z}$*.*

In a nutshell, the sample complexity gives some clues about the required number of profiling traces so that the learning algorithm $\mathcal{A}$ returns a model that is likely to provide a satisfying performance for the task it is assigned.

4.1.3 The Empirical Risk Minimization Paradigm

The definition of learnability states whether it is possible or not to find a learning algorithm. However, it does not give any clue about how to find such a learning algorithm. An intuitive and generic approach is to rephrase the problem by finding a model which, rather than minimizing the loss over the whole unknown joint distribution of $(\mathbf{X}, \mathrm{Z})$, would minimize the loss $\ell\left(\right)$ over the samples from the profiling set only, a.k.a. the *training loss* denoted by $\mathcal{L}_{\mathbf{X},\mathrm{Z}}\left(\right)$. That is:

$$\mathcal{L}_{\mathcal{S}_p}\left(F\right) \triangleq \frac{1}{|\mathcal{S}_p|} \sum_{i=1}^{|\mathcal{S}_p|} \ell\left(F(\mathbf{x}_i), z_i\right) \quad . \tag{4.4}$$

This principle is known under the name of Empirical Risk Minimization (ERM), and covers many situations such as linear regression, or maximum likelihood estimation. It translates the learning problem into a functional optimization problem – *i.e.* finding the model F from $\mathcal{H}$ minimizing the training loss – that the attacker may directly address.

Soundness of the Empirical Risk Minimization (ERM) Principle. The question arising when substituting Problem 2 with ERM, is whether the latter one is actually a learning algorithm, according to Definition 6. In other words, does one have the guarantee that the more profiling traces, the higher the performance metric of the obtained solution $\mathcal{A}(\mathcal{S}_p)$? And if so, what is the required size of the profiling set in order to get a satisfying performance, according to Definition 7?

The Fundamental Theorem of Learning, that we present hereafter, aims at addressing this issue. It gives a necessary and sufficient condition on the hypothesis class $\mathcal{H}$, for the ERM to be a learning algorithm. This condition relies on the so-called Vapnik-Chervonenkis (VC)-dimension of the considered hypothesis class $\mathcal{H}$, a way to characterize its *size*. The formal definition of the VC-dimension is beyond the scope of this thesis, but will be briefly discussed after we introduce the following fundamental theorem.[4]

[3] We deliberately overlap the notation $\mathcal{A}$ referring to the learning algorithm with the same notation addressing the attacker, since they represent the same entity in a profiling SCA scenario.

[4] The interested reader may refer to the book of Shalev-Shwartz and Ben-David [SSBD14] or to the book of Vapnik [Vap95].

Theorem 2 (Foundamental Theorem of Learning [Vap99, SSBD14]). *Let $\mathcal{A}$ be a learning algorithm and let $\mathcal{S}_p$ be a profiling set of size $|\mathcal{S}_p|$. Assume that $\mathcal{H}$ is a hypothesis class of finite VC-dimension. Then:*

$$\sup_{F \in \mathcal{H}} \left\{ \mathcal{L}_{\mathbf{X},\mathrm{Z}}(F) - \mathcal{L}_{\mathcal{S}_p}(F) \right\} \xrightarrow[|\mathcal{S}_p| \to \infty]{\mathcal{P}} 0 \tag{4.5}$$

In particular, it follows that:

$$\mathcal{L}_{\mathcal{S}_p}(\mathcal{A}(\mathcal{S}_p)) \xrightarrow[|\mathcal{S}_p| \to \infty]{\mathcal{P}} \min_{F \in \mathcal{H}} \mathcal{L}_{\mathbf{X},\mathrm{Z}}(F) \ , \tag{4.6}$$

$$\mathcal{L}_{\mathbf{X},\mathrm{Z}}(\mathcal{A}(\mathcal{S}_p)) \xrightarrow[|\mathcal{S}_p| \to \infty]{\mathcal{P}} \min_{F \in \mathcal{H}} \mathcal{L}_{\mathbf{X},\mathrm{Z}}(F) \ . \tag{4.7}$$

This result will be of great interest in Chapter 5.

The VC-dimension implicitly impacts the sample complexity of the ERM: roughly speaking, the higher the VC-dimension, the slower the convergence in Equation 4.6 and Equation 4.7 [Vap99]. Hence a set $\mathcal{H}$ with finite VC-dimension is necessary for the ERM to be sound. A brief discussion about the characterization of the VC-dimension in our context will be proposed in Subsection 4.2.3

A Utopian Approach. So far we have said that the ERM approach allows the attacker to transform the SCA optimization problem into a fully defined functional optimization problem. However, it now remains to get an algorithm able to solve this functional optimization problem. This is the major drawback of this approach: depending on the considered hypothesis class $\mathcal{H}$, the optimization problem yield by the ERM approach may be *hard* to solve. Instead, most of the time, one uses heuristics which are not always guaranteed to return the model that the ERM algorithm would return, as we will discuss in Subsection 4.3.1 and Subsection 4.3.3. That is why one must make a discrepancy between a theoretical attacker $\mathcal{A}(\mathcal{S}_p)$ using the true ERM approach, and a practical attacker $\tilde{\mathcal{A}}(\mathcal{S}_p)$, that would use some heuristics.[5]

Decomposition of Error Terms. In view of all the elements of the ML theory introduced so far in this chapter, it becomes of natural interest to study the final loss returned by our learning algorithm $\tilde{\mathcal{A}}(\mathcal{S}_p)$. This term can be decomposed into four parts, as follows:

$$\begin{aligned}
\mathcal{L}_{\mathcal{S}_p}\left(\tilde{\mathcal{A}}(\mathcal{S}_p)\right) = \quad & \mathcal{L}_{\mathcal{S}_p}\left(\tilde{\mathcal{A}}(\mathcal{S}_p)\right) - \mathcal{L}_{\mathcal{S}_p}(\mathcal{A}(\mathcal{S}_p)) && \geq 0 && (4.8)\\
+ \quad & \mathcal{L}_{\mathcal{S}_p}(\mathcal{A}(\mathcal{S}_p)) - \min_{F \in \mathcal{H}} \mathcal{L}_{\mathbf{X},\mathrm{Z}}(F) && \leq 0 && (4.9)\\
+ \quad & \min_{F \in \mathcal{H}} \mathcal{L}_{\mathbf{X},\mathrm{Z}}(F) - \mathcal{L}_{\mathbf{X},\mathrm{Z}}(F^\star) && \geq 0 && (4.10)\\
+ \quad & \mathcal{L}_{\mathbf{X},\mathrm{Z}}(F^\star) && \geq 0 \ , && (4.11)
\end{aligned}$$

where $\mathcal{S}_p$ is the profiling set of traces introduced in Section 3.4, F denotes an abstract model from the hypothesis class $\mathcal{H}$ considered by the attacker, and $\mathcal{L}_{\mathbf{X},\mathrm{Z}}(F)$ denotes the expected value of the loss function over the joint distribution of $\mathbf{X}, \mathrm{Z}$.

The term (4.11) denotes the so-called *Bayes'* error, *i.e.*, the minimal value of a loss achieved by the optimal solution to Problem 2. This minimum value only depends on the nature of the loss function and on the unknown joint distribution, but does not depend on any choice from the learner/attacker. As the expected value of a non-negative random variable, the Bayes' error is itself non-negative.

The term (4.10) corresponds to the *approximation* error: this error is due to the choice of a restricted hypothesis class $\mathcal{H}$ from which we select our model F – *e.g.* $F^\star$ may not belong to the hypothesis class $\mathcal{H}$ considered by the attacker. Since $\mathcal{L}_{\mathbf{X},\mathrm{Z}}(F^\star)$ is itself a minimum

[5] Examples of heuristics will be given in the description of DNNs.

over a wider set of functions than $\mathcal{H}$, it is always lower than $\min_{F \in \mathcal{H}} \mathcal{L}_{\mathbf{X},Z}(F)$. Hence, the approximation error is always non-negative.

The term (4.9) corresponds to the *estimation* error. It is the error due to the fact that we do not maximize the expected value of the loss – as the true p.m.f. is unknown – but rather its empirical estimation, *i.e.*, the training loss computed over a finite set $\mathcal{S}_p$ of profiling traces. This error term is always non-negative.[6] Moreover, according to the property of a learning algorithm given in Definition 6, this error term is supposed to decrease with the number of profiling traces. On the contrary, this error term increases with the VC-dimension of the hypothesis class $\mathcal{H}$ [Vap99].

The term (4.8), a.k.a. the *optimization* error, appears when considering an attacker $\mathcal{S}_p \mapsto \widetilde{\mathcal{A}}(\mathcal{S}_p)$ using a heuristic algorithm rather than the exact ERM approach. Since by definition, the theoretical attacker $\mathcal{A}(\mathcal{S}_p)$ minimizes the training loss, the optimization error is always non-negative.

We remark that each error term refers to a restriction in the capacity of an evaluator (optimal attacker, restricted attacker with finite hypothesis class, finite profiling set, heuristic for the ERM). That is why, in order to practically assess the quality of the model returned by the learning algorithm, it is interesting to emulate cases where such restrictions can be ignored, so that each error term can be evaluated separately. The work presented in Chapter 5 will be devoted to thoroughly discuss each error term.

4.2 The Neural Networks Class Hypothesis

So far we have presented the ML framework for a generic hypothesis class $\mathcal{H}$. This framework is of particular interest for our profiled attacks scenario, since it encompasses the majority of the attacks presented so far. As an example, we can remark that GTs may be considered as a particular hypothesis class. Indeed, it may be shown that the way the mean vectors and the covariance matrices are estimated from the profiling set actually follows the ERM principle [GBC16].

The main interest in formalizing profiling attacks more generally as a machine learning problem, is that one is not necessarily restricted anymore to the limited number of leakage models presented so far in Section 3.4. That is why the SCA community started considering different hypothesis classes over the last few years, among the wide zoology of ML algorithms, such as Support Vector Machine (SVM) [CV95] or random forest [Bre01, PSK+18]. This section – and more generally this thesis – will be exclusively devoted to the particular hypothesis class of Deep Neural Networks (DNNs). Nevertheless, the interested reader may refer to the comprehensive survey of Hettwer *et al.* about the use of every ML approach to SCA [HGG20].

We briefly describe the DNNs in Subsection 4.2.1, and the different *architectures* in Subsection 4.2.3. We also present some of their useful properties, especially the *Universal Approximation* theorem, in Section 4.2.3, before detailing in Section 4.3 how to implement them in practice. Finally, Section 4.4 is devoted to review the use of DNNs in SCA, over the recent literature.

4.2.1 General Description

Deep Learning (DL) aims at constructing a function $F\colon \mathcal{X} \to \mathcal{P}(\mathcal{Z})$ that takes a datum $\mathbf{x}$ and outputs a p.m.f. over a finite domain, represented as a vector $\mathbf{y}$. The result can then be used for different tasks. For example, in a classification task the goal is to predict among a given number of mutually exclusive classes, the one which has been assigned to the input data $\mathbf{x}$.

[6] Unless the attacker includes some inductive bias into the ERM, *e.g.* with *regularization* techniques, if he thinks that some solutions to the ERM would be likely to better generalize than some others. The inclusion of dropout layers – see Subsection 4.2.2 – is an example of inductive bias.

The vector returned by the DNN may then expresses scores depicting the preference to each class that the input data $\mathbf{x}$ might stand for. The final prediction is made by returning the class for which the highest score has been assigned. The output $\mathbf{y}$ can also be used for soft decision contexts, which correspond more to SCA as the DNN outputs on attack traces may be used as scores to feed a distinguisher.

In a very general way, a DNN may be seen as a Directed Acyclic Graph (DAG) of computation, where different functions may be applied at each node. Each function may be fixed by the operator, or may belong to a class of functions $f_i(\cdot, \theta_i)$, each being typically fully described by real vectors θ_i, a.k.a. *parameters*. The shape of the DAG and the nature of the classes of functions is called the *architecture* of the DNN.

4.2.2 The Elementary Layers

Most of the time and in the remaining of this thesis,[7] the architecture is organized so that the DAG is a simple chain of nodes. In other words, the model is a sequence of compositions between several simpler functions called *layers*. More precisely, this sequence generally alternates layers denoting linear operations with respect to each of their inputs – hence called *linear* layers, and non-linear functions, often referred as *activation* layers. This section is devoted to introducing the different layers that we will use in this thesis, before presenting the general architectures in Subsection 4.2.3.

Dense Layers λ_C. They consist in applying to a vectorial input $\mathbf{x} \in \mathbb{R}^D$ a matrix multiplication:

$$\lambda_C(\mathbf{x}) = \mathbf{M} \cdot \mathbf{x} \ , \tag{4.12}$$

where $\mathbf{M} \in \mathbb{R}^{D \times C}$ denotes the *weight* matrix, and C denotes the *size* of the dense layer. These weights are the trainable parameters of this layer. The term "dense" denotes the fact that when representing separately each entry of the output as a single node in the DAG, those nodes are all connected to all the nodes representing the entries of the input of the layer.

Convolutional Layers $\gamma_{W,K}$. A convolution layer consists in computing a series of discrete convolutions between an input and one or several *filters* – a.k.a. *kernels*. We detail hereafter the meaning of the layer.

Let $\mathbf{x}$ be a 2D-array of size (D, V) denoting the input of a convolution layer. V denotes the number of *channels* in $\mathbf{x}$.[8] In particular, for the first layer, $\mathbf{x}$ coincides with the input trace, so $V = 1$. Let also $\mathbf{w}$ be a 3D-array of size (W, V, K) denoting the set of K convolution filters of size W to apply to the input signal.[9] The output of the convolution is a 2D-array $\gamma_{W,K}$ of size $(D - W + 1, K)$ such that:

$$\gamma_{W,K}(\mathbf{x})[i, j] = \sum_{d=1}^{V} \sum_{m=0}^{W-1} \mathbf{x}[i + m, d] \cdot \mathbf{w}[m, d, j] \ , \tag{4.13}$$

for $0 \leq i \leq D - W, \ \ 1 \leq j \leq K$.

The parameters which can be adjusted with an ERM are the coefficients of the K filters. Therefore, there are $W \times V \times K$ real parameters to learn, and W and K are the hyper-parameters defining the layer. Hence, we use the notation $\gamma_{W,K}$ to describe such a convolution layer.

A convolution being a linear operation, it can be rephrased as a dense layer where the weights matrix has constraints decreasing the number of coefficients to learn compared to

[7] Exceptions to this restriction are discussed in Section 4.2.3, *e.g.* with Resnet architectures.

[8] This terminology encompasses the fact that a black-and-white picture has one channel, a stereo sound has two channels (left and right) and a colored picture has three channels (RGB).

[9] We distinguish K denoting the number of filters in a convolutional layer from K, the random discrete variable denoting the secret key chunk to ultimately recover.

a regular dense layer [GBC16, Sec. 9.1]. One must remark that the convolution layer commutes with shifts of maximum size W, which is useful when one wants to encode the possible invariants of the input trace as an inductive bias of the hypothesis class. This is particularly of great interest against misalignment-based counter-measures, as shown by Cagli *et al.* [CDP17].

Management of the Side Effects. As one may remark in Equation 4.13, the output size along the time dimension of a convolution layer decreases from D to $D - W + 1$. It is often useful to maintain the time dimensionality constant through the convolution layer. To tackle this problem, the input $\mathbf{x}$ may be *padded* by one or several ranges of zeros, around the two edges of the input, so that the time dimensionality is artificially increased to $D' = D+W-1$. The usual convention imposes to pad the input with the same number of zeros in both sides, which then constrains the filter size W to be odd.

Pooling Layers δ_p. An *average* pooling layer is a mapping made of two steps. First, one applies a particular case of convolution layer, yet without any learning parameter. It considers constant filters of size p with value $\frac{1}{p}$. In other words, this computes the average over a *pool* of p entries. Second, a *sub-sampling* operation is applied, consisting in keeping only one entry in each pool. The hyper-parameter p is called the pooling *stride*, and fully defines the pooling layer, hence the notation δ_p to denote such a pooling layer. A pooling layer of stride p has the effect of dividing the time dimensionality of the output by p. As a consequence, it is less sensitive to shifts of maximum size p compared to the input of the pooling layer, which is here again useful to encode inductive bias on the input traces.

It is worth emphasizing that there exist other types of pooling layers, such as the *max* pooling layer, consisting in keeping the maximum value of $\mathbf{x}$ for each window of size p. This makes the pooling layer not linear anymore, yet throughout this thesis, we will only consider average pooling layers.

Batch Normalization Layers μ. This type of layer has been introduced by Ioffe *et al.* at ICML 2015 [IS15], and is generally inserted after each linear layer. According to the authors, the first intuition behind this layer is to avoid the *internal covariate shift*, namely "the distribution of each layer's inputs changes during training, as the parameters of the previous layers change. This slows down the training[...]"

To circumvent this problem, they propose to insert after each linear layer the following operation:

$$\mathbf{x}' = \frac{\mathbf{x} - \mathbf{M}}{\sqrt{\mathbf{\Sigma}}} \cdot \mathbf{b} + \mathbf{a} \ , \tag{4.14}$$

where $\mathbf{x}$ denotes the input, $\mathbf{x}'$ denotes the output of same size, and $\mathbf{M}, \mathbf{\Sigma}, \mathbf{a}, \mathbf{b}$ have also the same dimensionality – $\sqrt{\mathbf{\Sigma}}$ denoting the square root applied element-wise on $\mathbf{\Sigma}$. $\mathbf{M}$ and $\mathbf{\Sigma}$ are respectively estimated according to the (element-wise) empirical mean and variance directly during the training, while $\mathbf{a}$ and $\mathbf{b}$ are learning parameters, *i.e.*, they are both included in the parameter vector θ to fit during the training loss minimization.

Many empirical evidences of its efficiency have been emphasized through the past few years, hence the batch normalization layer has been successfully used in many DNN architectures. Yet, the theoretical reasons behind this efficiency are still debated, and other arguments emphasize the effect of batch normalization on the training loss smoothness, rather than the internal covariate shift [STIM18].

Dropout Layer ω_q. Dropout is a layer introduced by Srivastava [SHK+14], aiming at decreasing the estimation error of a hypothesis class.

Given an input layer $\mathbf{x}$ of size m, the dropout layer samples a random vector $\mathbf{u}$ of same dimensionality, each entry independently following a Bernoulli law of parameter $q \in [0, 1]$.[10] The dropout layer outputs the element-wise product:

$$\omega_q(\mathbf{x})[i] \triangleq \mathbf{x}[i] \times \mathbf{u}[i] \quad 1 \leq i \leq m \ . \tag{4.15}$$

Therefore, q is the hyper-parameter defining the layer. Dropout is known to be a way to control the estimation error by trading-off a bit of the approximation error [GBC16, Sec. 7.12].[11]

Activation Layers. The role of activation layers is to insert non-linear –more precisely non-polynomial – functions in the architecture. The underlying reason will be quickly explained afterwards in Subsection 4.2.3 when we will introduce the *universal approximation* theorem. Historically, activation layers were used to modelize the response of a neuron cell by the stimuli of several neighbor neuron cells. Throughout this thesis, we will use two types of activation layers.

- **Rectified Linear Unit (ReLU):** It consists in the element-wise application of the $\max$ real function
$$\sigma(x) = \max(0, x) \ . \tag{4.16}$$
- **Softmax:** This function aims at normalizing a vector to make it fit a discrete p.m.f.
$$\mathrm{s}(\mathbf{x})[i] \triangleq \frac{e^{\mathbf{x}[i]}}{\sum_j e^{\mathbf{x}[j]}} \ . \tag{4.17}$$
The composition of a linear layer and a softmax is often referred as a softmax classifier in the ML literature [GBC16, Sec. 6.2.2.3]. Note that contrary to the ReLU, the softmax layer is not applied element-wise, since an output entry depends on all the input entries.

There exist many other activation layers used so far in the DL literature, especially those based on *sigmoids* – *i.e.* with the shape of an 'S', although beyond the scope of this thesis.

4.2.3 The Architectures

Once we have introduced the building blocks of the DNNs, we can now present all the architectures used in this thesis.

Multi-Layer Perceptron. The simplest architecture, called Multi-Layer Perceptron (MLP), consists in alternatively composing dense layers, batch norm layers, ReLUs and a final softmax layer, in order to output a p.m.f.:

$$F(\mathbf{x}) = \mathrm{s} \circ \lambda_C \circ [\sigma \circ \mu \circ \lambda_C]^L \circ \mu(\mathbf{x}) \ , \tag{4.18}$$

where $L \geq 1$ denotes the *depth* of the MLP.

The Universal Approximation Theorem. A remarkable result specific to MLPs, known as the *Universal Approximation* Theorem, states that when considering a L^2 error as a loss function, the approximation error term (4.10) converges towards 0 when the number C of neurons in the layers increases [Pet98]. In other words, an MLP, even with only one intermediate layer, can approximate a wide range of functions. The theorem only requires the activation function of the MLP to be non-polynomial, which is the case for the ReLU.

[10] See Subsection 2.2.1 for a description of the Bernoulli law.
[11] A dropout layer will be used in Subsection 5.5.1.

Actually there exist many versions of the universal approximation theorem, relying on different notions of convergence, or on particular properties of the activation function of the neural network. The drawback of this result is that without any additional assumption on the function to approximate, the required number of neurons exponentially increases with the inverse of the approximation error. Hopefully, this negative result may be mitigated by increasing the depth of the MLP, rather than the number of neurons on each layer [Tel16]. The interested reader may refer to the survey of Pinkus [Pin99].

The VC-dimension of MLPs. As presented in Subsection 4.1.3, the VC-dimension of an hypothesis class has an impact on the estimation error, and thereby the sample compexity. When considering the class of MLPs, the VC-dimension can be upper-bounded by a polynomial depending on the number of learning parameters, *i.e.*, the weights [SSBD14, Sec. 20.4]. In other words, with DNNs, the more parameters to learn, the higher the estimation error.

Convolutional Neural Networks. In pattern recognition tasks involving signals such as time series, images or videos, some elementary deformations, such as random shifts, do not usually affect the information carried through the data. Instead of letting the ERM learn implicitly those invariants at the cost of a higher sample complexity, they can be explicitly encoded through the architecture. This is the main idea behind the introduction during the 1990's of Convolutional Neural Networks (CNNs) by LeCun and Bengio [BLH93, LB94]. By remarking that some typical patterns appeared in the weights of the dense layers, they suggested to replace those layers by convolutional and pooling layers. Indeed, thanks to the properties of convolution and pooling layers on small shifts, stacking those layers enable to better encode the semantic invariants of the input data, while decreasing the number of parameters to learn – and so intuitively the sample complexity.

VGG-like Architecture. The Visual Geometry Group (VGG)-like architecture has first been introduced by Simonyan *et al.* [SZ15], after winning the ILSVRC in 2014. Its architecture is as follows:

$$\mathrm{s} \circ \lambda_{|\mathcal{Z}|} \circ [\sigma \circ \lambda_C]^{n_1} \circ [\delta_p \circ \sigma \circ \mu \circ \gamma_{W,K}]^{n_2} \circ \mu \ , \tag{4.19}$$

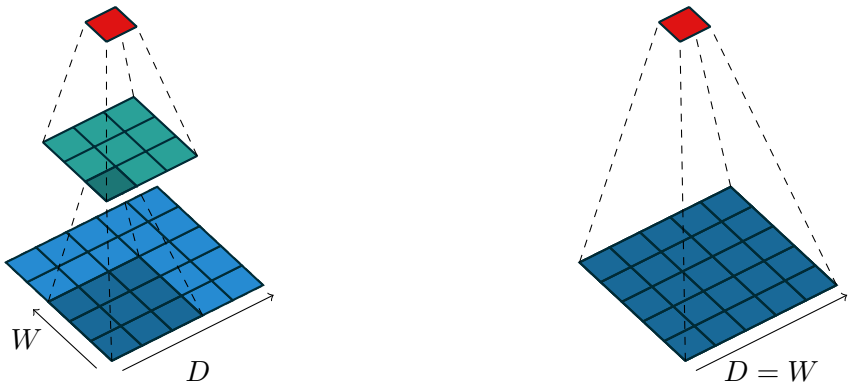
where $\gamma_{W,K}$ denotes a convolution layer made of K filters of size W, and δ_p denotes a pooling layer of stride p. Furthermore, the composition $[\delta_p \circ \sigma \circ \mu \circ \gamma_{W,K}]$ is denoted as a convolutional *block*. Likewise, $[\sigma \circ \lambda_C]$ denotes a dense block. We note n_1 (resp. n_2) the number of dense (resp. convolutional) blocks.

Remark 5. *More precisely, the VGG architecture of a convolutional block is slightly different in the original paper: the convolution layer $\mu \circ \gamma_{W,K}$ inside the convolution block is replaced by a stack $[\mu \circ \gamma_{W,K}]^{n_3}$ of $n_3 > 1$ convolution layers. The authors indeed suggest a recipe, behind the success of the VGG architecture at the ILSVRC. This recipe may be summarized as follows:* stack more convolution layers, with small filters.

The reason is that when dealing with images, to be able to capture patterns from a $D \times D$ square, the stack of convolution layers must verify the following condition resulting from the side effects of successive convolution layers, as explained in Subsection 4.2.2:

$$D = n_3(W - 1) \ . \tag{4.20}$$

When this condition holds, the number of learning parameters is $n_3 W^2 = \frac{DW^2}{W-1} \approx DW$, which may be minimized by setting W to small values and by increasing n_3 accordingly.



(a) $n_3 = 2, W = 3$. (b) $n_3 = 1, W = 5$.

Figure 4.1: A 2D receptive field of size $D \times D$, captured by two different settings. Inspired from Dumoulin *et al.* [DV16].

An illustration is proposed in Figure 4.1: on Figure 4.1a two convolutional layers are applied to a 5×5 patch, with filters of size 3×3 for each layer, hence $2 \times 3 \times 3$ filters weights to learn. On the contrary, as depicted in Figure 4.1b, by only using one layer, one must use 5×5 filters to cover the same area. This represents 25 filter weights to learn, i.e., more than by using two stacked layers. Therefore, the convolution layers are believed to keep their capacity of expression approximately constant by covering the same area, while decreasing the number of learning parameters and so the sample complexity. Hence the global trend consisting in increasing more and more the depth of CNNs in a computer vision context over the past few years.

For 1D-data, such as SCA traces, the argument discussed in Remark 5 does not hold anymore since the required number of learning parameters becomes $n_3 W = D \frac{W}{W-1} \approx D$ rather than DW, as depicted on the two examples on Figure 4.2. In other words, no matter the filter size chosen, the number of filter weights to learn remains globally constant, so stacking more convolutional layers does not seem necessary anymore; at least not for the reason developed in Remark 5. Benadjila *et al.* empirically confirmed that it was not necessary to

(a) $n_3 = 2, W = 3$. (b) $n_3 = 1, W = 5$.

Figure 4.2: A 1D receptive field of size $D = 5$, captured either by one or two convolution layers. Inspired from Dumoulin *et al.* [DV16].

stack more than $n_3 = 1$ layer inside a convolution block [BPS+19] for an SCA context. That is why in the remaining of this thesis, we will keep the VGG architecture such as described in Equation 4.19. However, the discussion concerning the ideal filter size remains open, depending on the context. In particular, we will discuss this setting in Chapter 6.

Other CNN-Based Architectures. Beside the VGG architecture, the DL community has seen the emergence of many other types of architectures, such as GoogleNet [SLJ+15], Inception [SVI+15], Resnet [HZRS16], DenseNet [HLW16], *etc.* The study of those architectures are yet beyond the scope of this thesis, although some of them start to be used in DL-based SCA [ZS19, GJS20] and some of them will be briefly discussed in Chapter 6.

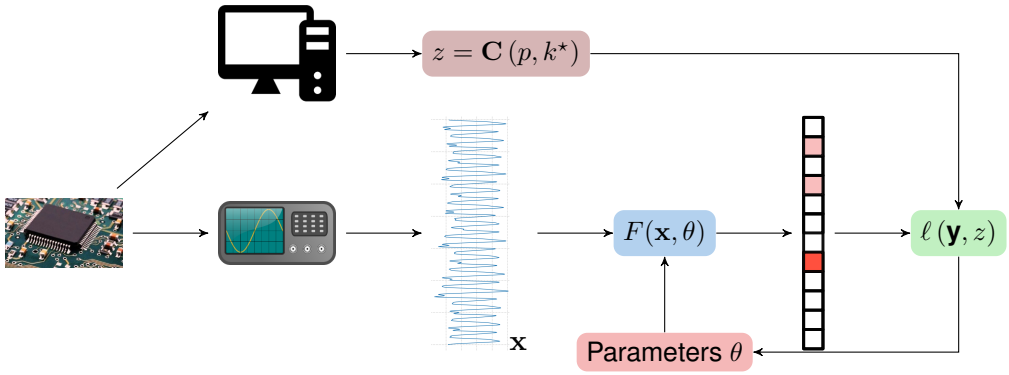


Figure 4.3: Illustration of the workflow of the training of a DNN in a profiled SCA context.

4.3 Implementing the ERM with Neural Networks

As we have seen, the hypothesis class of DNNs is fully defined by the hyper-parameters describing the architecture, whereas a specific instance F of neural network among $\mathcal{H}$ is also defined by the vector θ of all the learning parameters. In other words, one can re-write a model from $\mathcal{H}$ under the form $F(\cdot, \theta)$. As a consequence, implementing the ERM principle can be translated from a functional optimization perspective – *i.e.* finding the model $F \in \mathcal{H}$ minimizing the training loss – into a numerical optimization problem – *i.e.* finding the vector θ such that the function $F(\cdot, \theta)$ minimizes the training loss. Since closed-form solutions do not exist, this numerical optimization problem is usually solved thanks to an iterative optimization algorithm such as SGD[12] by iteratively updating the values of the learning parameter θ, as illustrated in Figure 4.3.

Interestingly, the ERM principle introduced by the ML framework remains a purely theoretical tool in most of the applications, and particularly in deep learning. Indeed, it often cannot be implemented in practice, for several reasons that we will describe in the following subsections. Finally, we explain in Subsection 4.3.4 how the derivatives of the loss function, a crucial element required by the optimization algorithm, is efficiently computed. This description will serve as a discussion in Chapter 7.

4.3.1 SCA Metrics are Hard to Optimize

So far in this chapter, we have presented the ML framework for a generic learning problem. In particular, we have considered so far an abstract loss function $\ell(\,)$ to minimize, whose generic definition has been given in Equation 4.2. It is indeed tempting at first sight to use the final performance metric as a loss function, in order to guarantee the convergence of the trained model towards the best possible one, according to Theorem 2. Unfortunately, one cannot optimize with respect to any loss function, regardless the underlying hypothesis class $\mathcal{H}$.

For example, in supervised classification the ultimate performance metric is the *accuracy*, namely the rate of accurate predictions among the possible labels that may be recognized by the model. For any function $F : \mathcal{X} \rightarrow \mathbb{R}^{|\mathcal{Z}|}$, the accuracy is denoted by $\mathrm{Acc}_{\mathbf{X},\mathrm{Z}}(F)$ and is defined as:

$$\underset{\mathbf{X},\mathrm{Z}}{\mathrm{Acc}}(F) \triangleq \Pr\left(\underset{s\in\mathcal{Z}}{\operatorname{argmax}}\, F(\mathbf{X})[s] = \mathrm{Z}\right) = \underset{\mathbf{X},\mathrm{Z}}{\mathbb{E}}\left[\mathbf{1}_{\operatorname{argmax}_{s\in\mathcal{Z}} F(\mathbf{X})[s]=\mathrm{Z}}\right] . \tag{4.21}$$

That is, this metric gives the rate of right predictions, or more precisely the probability that the highest score returned by a learning algorithm based on a single input data $\mathbf{X}$ corre-

[12] We recall that Subsection 2.5.2 introduced the SGD algorithm.

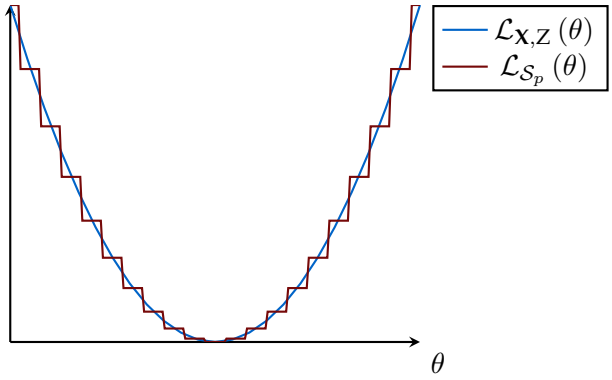


Figure 4.4: Toy example of a training loss made of characteristic functions with respect to a real valued learning parameter θ. Paradoxically, although the training loss $\mathcal{L}_{\mathcal{S}_p}(\theta)$ has zero derivatives almost everywhere, the generalization loss $\mathcal{L}_{\mathbf{X},Z}(\theta)$ may have non-null derivatives.

sponds to the class Z it is assigned. Therefore, maximizing the accuracy is suitable to address the *classification* task.

Unfortunately, solving the ERM for DNNs with the accuracy as a loss function turns out to be NP-hard [SSBD14, Sec. 20.5]. In a nutshell, the reason comes from the fact that the resulting training loss can be formulated as a sum of characteristic functions:[13]

$$\underset{\mathcal{S}_p}{\mathrm{Acc}}(F) \triangleq \frac{1}{|\mathcal{S}_p|} \sum_{\mathbf{x},z \in \mathcal{S}_p} \mathbf{1}_{\mathrm{argmax}_{s \in \mathcal{Z}} F(\mathbf{x})[s]=z} \quad . \tag{4.22}$$

Each characteristic function in Equation 4.22 has null derivatives almost everywhere, and so has the training loss of the accuracy, as depicted in Figure 4.4 with the staged red curve depicting the training loss, as a sum of characteristic functions. So gradient-descent-based optimization algorithms are useless, and no other efficient alternative could circumvent this issue.

Interestingly, this drawback also concerns the efficiency $F \mapsto N_a(F)$, defined in Subsection 3.2.2 as the minimal number of attack traces to succeed the attack beyond a probability threshold β fixed by the evaluator, and that we chose as an ultimate performance metric according to Problem 2. The corresponding training loss to minimize in the ERM would depend on the guessing vector defined in Equation 3.2. But the latter quantity is a sum of characteristic functions, hence meeting the same issues as the training loss corresponding to accuracy. That is why the SCA metrics are hard to directly optimize.

More generally, this also holds for Random Forest and SVMs, two other learning algorithms used in the SCA literature [PHJ+18]: the former one is based on heuristics working reasonably well in practice [SSBD14, Chap. 18.2] whereas the latter one must minimize another loss function called *Hinge* loss [SSBD14, Chap. 15.2.3].

4.3.2 The Need for a Surrogate Loss

The previous section raises the need for a suitable *surrogate* loss function, either found among the usual functions considered in the DL literature, or designed specifically for our problem. In the literature, mostly two surrogate loss functions have been used in the SCA context: the Negative Log Likelihood (NLL) [CDP17, BPS+19, KPH+19] and the Mean Square Error (MSE)[14] [MPP16, Tim19, WMM19]. Until a few years ago, nobody particularly raised the issue into the SCA community since the empirical results obtained for any loss function on several use cases got promising results from an attacker's point of view.

[13] See Section 2.1 for a definition of a characteristic function.

[14] From a purely optimization point of view, the MSE might suffer from problems [Nie18]. From a SCA evaluation point of view, the relevance of MSE is an open question [vdVP19], beyond the scope of this thesis.

Nevertheless, from an evaluator's point-of-view, it remains necessary to assess whether the problem of minimizing the chosen training loss is actually equivalent to the profiled SCA optimization problem. More precisely, whether:

1. both problems share the same analytical optimal solution $F^\star$;
2. improving a sub-optimal solution for one problem directly leads to get an improved sub-optimal solution for the other one.

Tackling this issue is of great interest in SCA. Indeed nowadays there might still be a gap between the recent practical successes of this class of attacks, and the theoretical soundness of DL-based SCA: what is the sense of training a DNN by minimizing a surrogate loss function from an SCA point of view? This issue will be at the core of Chapter 5.

4.3.3 The Challenge of Optimization

We have seen that the interest of the surrogate loss function is to be differentiable with respect to all the learning parameters,[15] in order to use a gradient based optimization algorithm such as SGD. Nevertheless, in DL, this still raises an important issue, even when using a surrogate loss. Indeed, when considering the specific hypothesis class of DNNs, the resulting *objective*[16] function to optimize is shown to be highly non-convex [CHM+15], contrary to SVMs whose surrogate loss, *i.e.* Hinge loss, spans a convex optimization problem. Moreover, the solution to the problem is not unique: considering one DNN model minimizing the training loss, the same model whose entries of the intermediate layers are permuted – and so are the learning parameters accordingly – would return the same result. Hence, the number of equivalent solutions is *combinatorial* with respect to the output size of the intermediate layers of the model.

As a consequence, the usual numerical optimization algorithms are not theoretically guaranteed to converge towards an optimal solution, which prevents the SGD algorithm and its variants to perfectly instantiate the ERM principle. In other words, the optimization error cannot be assumed to be negligible, as already discussed in Subsection 4.1.3. Nevertheless, experience has surprisingly shown that those algorithms represent a satisfying heuristic [LBOM12], and the recent literature started to provide theoretical insights about this fact [DLL+19, DZPS19].

4.3.4 Computing the Gradient

So far in this section, we have explained that the ERM principle could be implemented by addressing a non-convex numerical optimization problem. We explained in the previous sections to what extent perfectly solving this problem is hard in practice, although sound approximate solutions could be returned by an optimization algorithm such as the SGD. The latter algorithm – and its variants – rely on the computation of *descent* directions based on the gradient of the loss function defined in Equation 4.4 and computed with respect to the parameter vector, namely $\nabla_\theta \mathcal{L}_{\mathcal{S}_p}\left(F(\cdot,\theta)\right)$. It is therefore of great interest to study to what extent providing the gradient of the loss function to the optimization algorithm is affordable when considering models from the hypothesis class of DNNs. This subsection is devoted to this discussion.

Remark 6. *The details provided in this subsection concern more generally any DL-based problem, and not only SCA. Nevertheless, it will be useful for future discussions in Chapter 7.*

[15] We recall that the parameters describing the architecture for which the loss is not differentiable are called *hyper-parameters*.

[16] The term "objective" function is the terminology used by the numerical optimization research community. It refers to the training loss function in the specific case of machine learning. In the following, we will rather use the term "loss" to design the function to minimize.

The training loss to minimize being a sum of elementary losses over the profiling set, so is the gradient:

$$\nabla_\theta \mathcal{L}_{\mathcal{S}_p}\left(F(\cdot,\theta)\right) = \frac{1}{N_p}\sum_{i=1}^{N_p} \nabla_\theta \ell\left(F(\mathbf{x}_i,\theta), z_i\right) \quad . \tag{4.23}$$

It turns out that an algorithm called *backward propagation* (a.k.a. backprop) can exactly compute the gradient of $\ell\left(F(\mathbf{x}_i,\theta), z_i\right)$ with respect to θ for roughly the same complexity of computing $\ell\left(F(\mathbf{x}_i,\theta), z_i\right)$ itself. It relies on the use of the chaining rule recalled in Lemma 1. Indeed, due to the layer-wise nature of $F(\cdot,\theta)$, the loss function, seen as a function of the parameter vector θ, can also be seen as a sequence of compositions of elementary functions whose derivatives can be computed in a closed-form solution. Therefore, by using recursively the chaining rule on the given sequence of functions, one is able to exhibit an efficient procedure to exactly compute the gradient.

It is noticeable that for a composition of $n > 2$ elementary functions, the chaining rule can be recursively applied in two manners, respectively denoted as *forward* and *reverse*. We explain hereafter the stakes behind those two automatic differentiation modes on an example of compositions of n elementary functions $(f_i)_{1\leq i\leq n}$ such that:

$$f_i : \mathbb{R}^{m_{i-1}} \to \mathbb{R}^{m_i} \quad , \tag{4.24}$$

where $m_i \geq 1$ for $0 \leq i \leq n-1$, $m_{n-1} = |\mathcal{Z}|$, and $m_n = 1$. The resulting function to differentiate $\ell : \mathbb{R}^{m_n} \to \mathbb{R}$ can then be written as:

$$\ell\left(\theta\right) = \overbrace{f_n \circ \ldots \circ \underbrace{f_i \circ \ldots \circ f_1}_{\psi_i \text{(Forward)}}}^{\varphi_{n-i}\text{(Reverse)}} \tag{4.25}$$

We recall furthermore that the elementary functions $(f_i)_{1\leq i\leq n}$ are assumed to be simple enough to derive closed-form expressions of their Jacobian matrices.

In a *forward* mode, the chaining rule is applied from right to left[17] in Equation 4.25. That is, by considering the sequence of mappings defined by $\psi_0 = I_d$ the identity mapping and $\psi_{i+1} = f_{i+1} \circ \psi_i, \;\; 0 \leq i \leq n-1$, and by applying Lemma 1, it follows that:

$$\underbrace{\mathrm{J}_{\psi_{i+1}}\left(\theta\right)}_{(m_{i+1},m_0)} = \underbrace{\mathrm{J}_{f_i}\left(\psi_i(\theta)\right)}_{(m_{i+1},m_i)} \cdot \underbrace{\mathrm{J}_{\psi_i}\left(\theta\right)}_{(m_i,m_0)} \tag{4.26}$$

Since one remarks that $\psi_n(\theta) = \ell\left(\theta\right)$, it is then possible to compute its gradient by iteratively applying Equation 4.26. But this method requires to store the Jacobian matrices of the mappings ψ_i whose sizes are respectively $m_i \times m_0$, which might be prohibitive if the intermediate outputs are of high dimensionality.

In a *reverse* mode, the chaining rule is applied from left to right in Equation 4.25. That is, by considering the sequence of mappings defined by $\varphi_1 = f_n$ and $\varphi_{i+1} = \varphi_i \circ f_{n-i}, \;\; 1 \leq i \leq n-1$, and by applying Lemma 1, it follows that for every $\mathbf{x} \in \mathbb{R}^{m_{n-i}}$:

$$\mathrm{J}_{\varphi_{i+1}}\left(\mathbf{x}\right) = \mathrm{J}_{\varphi_i}\left(f_{n-i}(\mathbf{x})\right) \cdot \mathrm{J}_{f_{n-i}}\left(\mathbf{x}\right) \tag{4.27}$$

By taking $\mathbf{x} = \psi_{n-(i+1)}(\theta)$, it follows that:

$$\underbrace{\mathrm{J}_{\varphi_{i+1}}\left(\psi_{n-(i+1)}(\theta)\right)}_{(1,m_{n-i})} = \underbrace{\mathrm{J}_{\varphi_i}\left(\psi_{n-i}(\theta)\right)}_{(1,m_{n-i-1})} \cdot \underbrace{\mathrm{J}_{f_{n-i}}\left(\psi_{n-(i+1)}(\theta)\right)}_{(m_{n-i-1},m_{n-i})} \tag{4.28}$$

Since one remarks that $\varphi_n = \ell$ it is here again possible to compute its gradient by iteratively applying Equation 4.28. Yet, compared to the forward mode, two main differences should

[17] We draw the attention of the reader on the fact that when composing several functions, the notation $f_2 \circ f_1$ must be read backward, hence "forward" means here "right-to-left".

be noticed. First, it is necessary to store all the intermediate computations $(\psi_i(\theta))_{1 \leq i \leq n}$ when previously computing the loss function $\ell(\theta)$, whereas in a forward mode the gradient can be directly computed without necessarily computing $\ell(\theta)$. Second, rather than storing a Jacobian matrix after each iteration, the reverse mode enables to only store a gradient of size m_{n-i}, which is much lighter than in the forward mode. That is why all the APIs devoted to DNNs only use the reverse mode in their implementations, and are optimized in order to avoid the storage of any Jacobian matrix, hence the name of "backprop" which explicitly refers to the "reverse" mode. This point will be later discussed in Chapter 7.

The backprop algorithm has been independently discovered many times during the 70's and the 80's, in particular by Hinton *et al.* in 1986 [RHW86b, RHW86a]. More generally, the backprop algorithm has paved the way towards *automatic differentiation* which studies how to efficiently differentiate functions *numerically*,[18] which is often crucial in machine learning. The interested reader may refer to the survey of Baydin *et al.* [BPRS17].

4.3.5 Some APIs

Training DNNs with optimization algorithms is nothing but linear algebra computations – *e.g.* scalar and matrix product – in very high dimensionality, typically $10^5 - 10^6$. To scale with this range, the implementation must leverage massively parallel programming, either in CPU clusters of on (General Purpose) Graphic Processing Units (GPUs). That is why until few years ago, the ML practitioners used to need strong skills in a broad spectrum of programming fields. Those skills were hard to gather into a research team, and most of the time of researchers was devoted to implementing models which were not easily reproducible by the scientific community.

That is why, with the emergence of deep learning in the beginning of the 2010's, several APIs have been released by the machine learning community. Although the very first public library Theano [The16] is not maintained anymore by its authors, several other APIs have been publicly developed over the last few years, such as Pytorch [PGM+19], supported by Facebook; Tensorflow [AAB+15], supported by Google; or Keras [C+15], an extension of Tensorflow initiated by Chollet. In particular, the source code developed for this thesis has been written in Python with the help of the Pytorch API,[19] and is run on a workstation with a Nvidia Quadro M4000 GPU with 8 GB memory and 1664 cores, and 32 GB of RAM.

4.3.6 On the Accuracy as a Monitoring Metric

We have seen in Subsection 4.3.1 that in supervised classification, it is not possible to directly maximize the accuracy of DNNs, since it is a NP-hard problem. Despite its impossibility to directly minimize, most of the works considering DL-based SCA made use, explicitly or implicitly, of the accuracy as a monitoring metric, assuming that such a performance measure may still be informative of the quality of a trained model.

Cagli *et al.* [CDP17] and Picek *et al.* [PHJ+18] were the first to raise this issue, namely the relevance of the *accuracy*, a widely used ML performance metric, in the context of SCA. Unfortunately, we explain hereafter that it is not the case.

It turns out that the optimal model for SCA, which is $F^\star = \Pr(\mathbf{Z} \mid \mathbf{X})$, is also the optimal classifier for the supervised classification task.[20] This means that for any function $F : \mathcal{X} \to \mathcal{P}(\mathcal{Z})$ we have:

$$\mathrm{Acc}(F) \leq \mathrm{Acc}(F^\star) \ . \tag{4.29}$$

In other words, both supervised classification problem and Problem 2 share the same analytical optimal solution $F^\star$, which is the first of the two conditions stated in Subsection 4.3.2

[18] *i.e.* in opposition to *symbolic* differentiation considering only algebraic formulations of a function.

[19] The library is available at `pytorch.org`.

[20] The ML literature often uses the term *Bayes' classifier* to denote the optimal classifier for the classification task.

to get a loss function equivalent to our SCA efficiency metric. Thus, the accuracy might be a good metric candidate. Yet, both problems sharing the same optimal solution $F^\star$ does not necessarily mean that they are equivalent, since it remains to verify that improving sub-optimal solutions for one problem should lead to improved sub-optimal solutions for the other.

Cagli *et al.* recalled that the accuracy could be translated into another SCA metric, namely the success rate when recovering the key with one trace $\mathrm{SR}(1)$. In a sense, $\mathrm{Acc}(F)$ is the *dual* metric of $N_a(F, 1, \beta)$: the former one fixes $N_a = 1$ and estimates the corresponding threshold β, whereas the latter one fixes the threshold and estimates the minimum value N_a such that $\mathrm{SR}(N_a, \mathcal{D}^F_{\mathcal{S}_a}, 1) \geq \beta$ – see Equation 3.3. Cagli *et al.* have emphasized that although this could have a sense in some specific scenarios, *e.g.*, when evaluating implementations of asymmetric cryptographic primitives, this does not necessarily have a sense to evaluate the robustness of a target against an attack involving only one trace. That is why at first sight, accuracy does not seem appropriate in our context.

Picek *et al.* empirically confirmed a similar observation at CHES'19 [PHJ+18]. For several learning algorithms, such as SVMs and Random Forests, they empirically compared the obtained GE with the accuracy, and they found out that there is no clear link between them. More precisely, they argue that a high accuracy – *i.e.*, with respect to the one obtained with a model providing completely random predictions – is a clue for effective attacks, though the inverse does not empirically hold: a low accuracy does not necessarily imply a failed key recovery. However, the latter case may often happen, especially with protected implementations where the noise artificially induced by the counter-measures reduces the performances of the optimal model, and so the accuracy.

As a conclusion to the two latter sections, the accuracy is not only useless for the implementation of the ERM, as already concluded in the end of Subsection 4.3.1, but is also a poor metric to monitor in an SCA context. Recently, new ways to monitor the quality of a DNN have been proposed, in particular by Robissout *et al.* [RZC+20] and Perin *et al.* [PBP20]. Although this brings new insights, those monitoring metrics partially circumvent the problems raised by the accuracy, since the metrics proposed there are not optimizable. Finding a proper loss function which is useful not only as a quantity to optimize but also as a metric to monitor is the core problem which will be addressed in Chapter 5.

4.4 An Overview of the Literature

The past few years have seen the emergence of contributions on SCA using more and more DL techniques. The community committed to investigate several models leading to practical attacks against several implementations. The very first works came from Martinasek *et al.* [MZ13, MDM16], Gilmore *et al.* [GHO15], whereas other ML techniques have been investigated by Heuser *et al.* [HZ12] and Lerman *et al.* [LBM14, LBM15]. Hereafter, we focus on the specific use of DL rather than on other ML algorithms. The reader interested in a complete review of the use of every learning algorithm in SCA may refer to the comprehensive survey of Hettwer *et al.* [HGG20].

The asymmetric cryptography has been by now investigated by Carbone *et al.* concerning the RSA primitive [CCC+19], and by Weissbart *et al.* [WPB19] concerning elliptic curves. In both works, results are as promising as for the symmetric context.

Auto-Encoders appeared as a valid solution to perform dimensionality reduction and pre-processing of side-channel signals [MPP16]. The temporal aspect of side-channel traces leads the community to explore as well some recurrent neural network structures, in particular the Long Short-Term Memory (LSTM) one [Mag19]. CNNs appeared more suitable in presence of signal desynchronization, and thus in presence of counter-measures injecting desynchronization in signals [CDP17, BPS+19, KPH+19].

4.4.1 Unsupervised Learning for SCA

Moreover one may remark that the great majority of works apply deep learning to perform profiling attacks, and logically exploits for the learning algorithm a supervised experience. A very few works proposed a non-profiling deep-learning-based attack. First, Timon proposed at CHES'19 [Tim19] an extension of the l.r.a. attack [SLP05]. However, this means that the learning algorithm still exploits a supervised experience, in which labels are assigned according to different key hypotheses.

Second, Ramezanpour *et al.* [RAD20] extended the works of Timon in several ways, by using LSTMs as an unsupervised feature extractor, and by using some analysis developed by Wang *et al.* [WYS+18] as a leakage modeling method.

A full-non-supervised track, based for example onto deep clustering techniques recently proposed in the computer vision domain, is still unexplored in the side-channel context.

4.4.2 Exploring the DL Strategies for SCA

This vast panorama of experimentally investigated tools have subsequently emphasized the need for a deep understanding of their architectural properties. A well-established methodology – beyond those already proposed [BPS+19, ZBHV19] – to tune the (very) high number of hyper-parameters in these models would be very useful. Furthermore, since in ML the learning algorithms are driven by data, the data management is a crucial point and related issues and good practices have been investigated in this sense. Cagli *et al.* [CDP17] and Kim *et al.* [KPH+19] proposed data augmentation techniques to control and decrease the estimation error (4.9). But many questions about the utility and /or necessity of performing some pre-processing like dimensionality reduction [Mag19], realignment [CDP17, ZS19], de-noising [WP19], under/over-sampling to deal with class imbalance [PHJ+18], or the conversion of data into the frequency domain by means of Fourier or Wavelets transforms [YLMZ18, DDFP20] have been raised.

4.4.3 Support for Understanding

Although DNNs show encouraging results in an evaluation context of SCA, following the recent hype of deep learning in pattern recognition, many people in the community remain skeptical and reluctant to this approach. This is mostly due to the *black-box*[21] aspect of those algorithms, *i.e.* the fact that they do not provide any insight about how the informative leakage occurs in the SCA traces. Although not of great interest for the attackers whose ultimate goal is only to recover the secret key, this represents a huge stake for evaluators and developers.

To provide understanding in DL models behavior, a track of recent works – including ours – proposed the use of some visualization techniques, with the threefold intent of characterizing the sensitive leaking part of the side-channel traces, understanding the nature of the signal information that a given neural network is able or not to exploit [MDP19a, HGG19, Tim19, vdVPB19] and validating the hyper-parameters choices [ZBHV19]. Furthermore, visualization techniques aiming at focusing on DL in order to help tuning their hyper-parameters or understanding their prediction is a long running challenge in the visualization and machine learning community [HKPC19, GTdS+18]. This issue will be the core discussion of Chapter 7.

[21] This terminology shall not be confound with the same term used for black-box attack scenarios discussed in Chapter 3.

4.4.4 DL-based SCA and Counter-Measures

The community already wondered about the efficiency of existing side-channel counter-measures against DL-based SCA. Many works – including ours – recently investigated the robustness of classic counter-measures, in particular the high-order secret-sharing [MPP16, BPS+19, KPH+19, Tim19, Mag19, MDP19b, ZS19, BS20]. The DL-based SCA showed very fast outperforming results with respect to the previous state-of-the-art attacks. The main advantage of DL, compared to regular SCA attacks is that DL is not technically limited by the minimal number of points which must be jointly processed, which was originally one of the strong practical arguments to use high-order secret-sharing, as we argued in Subsection 3.7.1. Nevertheless, Bronchain *et al.* recently emphasized a use case where automated attacks with DL did not succeed against a software target protected with affine secret-sharing whereas classical template attacks involving a subtle dissection of the open source code [BS20]. This lets one think that DL-based SCA could not always represent a better approach than classical Gaussian templates. We further discuss this case in the perspectives presented in the global conclusion of this thesis.

This also raises the challenge of knowing exactly the necessary number of traces for the training phase of a DL model – *i.e.* the sample complexity, and how secret-sharing could have an impact on this constraint. Until now, only Picek *et al.* started tackling the question [PHG19].

In addition, to the best of our knowledge, no sound counter-measure has been exhibited so far in the literature to specifically counteract deep learning techniques in side-channel analysis. Nevertheless, it seems that perturbing the inputs or adding dummy operations to fool a network could help developers in the protection of their implementation against deep learning attackers [BBCS20].

4.4.5 Multi-Task Learning

Some works make the hypothesis that several sensitive variables, processed in a similar way by the device during the cryptographic algorithm execution, may be targeted while keeping unchanged the neural network architecture (*i.e.* the hyper-parameters are tuned only once) [GBO19]. A similar approach has been proposed by Wang *et al.* [WD20], who combined the predictions of three different models targeting three different sensitive intermediate computations in an FPGA implementation of AES. A recent work from Maghrebi [Mag20] leverages this finding by proposing to solve the SCA problem with a single multi-labeling classifier. However, his solution, as is, is limited to learning only two labels at the same time. In the general conclusion of this thesis, we will further discuss this new line of work.

The multiple classifier concept is analogously proposed in [DDFP20], where several overlapping modelizations of a sensitive variable are independently recognized by several classifiers whose outputs are jointly exploited in the attack phase. In addition to these multiple outputs, it has been observed that training a deep neural network in a multi-task fashion results in having the performances of each single classifier increased.

4.4.6 Multi-Sources

The multi-source idea to enrich signal databases, meaning harvesting at the same time several side-channel signals (for example power consumption and EM irradiation captured with multiple probes placed nearby different areas of the device) and exploiting them synergistically, has been explored by Genevey-Metat *et al.* at C&ESAR 2019 [GMGH19]. Furthermore, learning with multiple and even heterogeneous sources remains an open topic in the machine learning community.

4.4.7 Portability

As a related topic, Carbone *et al.* [CCC+19] and Bhasin *et al.* [BCH+20b] tackled the portability issue: these works aim to understand the performance effects observed on DL models while conducting a profiling phase on a device which may not be a perfect clone of the target, in opposition to what is assumed throughout this thesis.

4.5 Conclusion

The machine learning framework enables to extend the leakage modelization in a profiling attack scenario to much more powerful hypothesis classes, which has been at the origin of new recent milestones in SCA. Nevertheless, in view of the current state of the art, we are today in an uncomfortable situation. Indeed, the replacement of the Profiled SCA Optimization Problem – *i.e.* Problem 2, so far tackled classical profiling attacks such as GTs, by the Supervised Classification Problem – *i.e.* finding a model maximizing the accuracy defined by Equation 4.22, thanks to DNNs, shows promising efficiency gains. Nevertheless, several recent papers question the theoretical soundness of the latter problem substitution [CDP17, PHJ+18]. This situation prevents the SCA community to get a clear picture of the potential impact of ML and DL, especially from the developers' perspective. Indeed, though an attacker only needs to know an efficient practical approach to train a DNN, a developer needs a theoretically grounded approach to be able to give the best security bounds on the complexity of mounting a profiling attack, especially when the implementation is protected by counter-measures.

More specifically, through the review of the deep learning approach and the literature review, we have emphasized several caveats which should be addressed by the SCA community:

- How can one prove that the underlying optimization problem materializing the training phase is a theoretically sound approach for Problem 2, beyond the recent empirical success? This requires to address the issue of choice of the loss function, and the study of the optimization error yield by the SGD or its variants.
- How far can deep learning go to approximate the true leakage model, even in presence of counter-measures? Said equivalently, is there any sound DL-oriented counter-measure which could protect the implementations of cryptographic primitives against this threat?
- How can an evaluator get a clear understanding of what happens during the profiling phase with DNNs, in order to draw a fair diagnosis of the device under evaluation?

Our work in the remaining of this thesis aims at grounding the use of DNNs in the SCA context, by addressing those issues in the next three chapters.

Chapter 5

Theoretical Aspects of Deep Learning Based Side-Channel Analysis

" In Statistical Inference, nothing is more practical than a good theory."

Vladimir Vapnik [Vap00]

This chapter is inspired from the results published in TCHES'20 in collaboration with Cécile Dumas and Emmanuel Prouff [MDP19b].

Contents

5.1 Introduction

The work presented in this chapter aims at grounding the use of DNNs in the SCA context, especially when classical counter-measures like secret-sharing and hiding are involved. In this chapter, we investigate to what extent the profiled SCA optimization problem – *i.e.* Problem 2 – can be solved through the machine learning framework by carefully choosing the underlying loss function. This is the main problem addressed in this chapter.

We propose a theoretical study of the Negative Log Likelihood (NLL) loss in different steps depicted in Figure 5.1, by enlightening the fact that such a function is strongly linked to a side-channel information theoretic quantity called PI, formally introduced by Renauld *et al.* at EUROCRYPT 2011 [RSV+11], and recently studied by Bronchain *et al.* at CRYPTO 2019 [BHM+19]. As a direct consequence, the PI can be straightforwardly computed from the NLL loss. More interestingly, this implies that the training phase of a deep learning model, through the minimization of the NLL loss, is actually equivalent to giving the PI estimation which is the closest to the MI between the leakage and the target sensitive variable.

In parallel, we benefit from the recent works of Chésirey *et al.* at CHES 2019 [dCGRP19]. The latter ones provide a lower bound of the optimal efficiency metric $N_a^\star(1,\beta)$ – defined in Equation 3.4 – depending on the MI. By combining those two results and by translating them into the ML terminology, we show that the SCA efficiency metric can be accurately estimated without even mounting a key recovery, which justifies the *soundness* of the DL approach when the latter one is addressed by training DNNs through the minimization of the NLL loss.

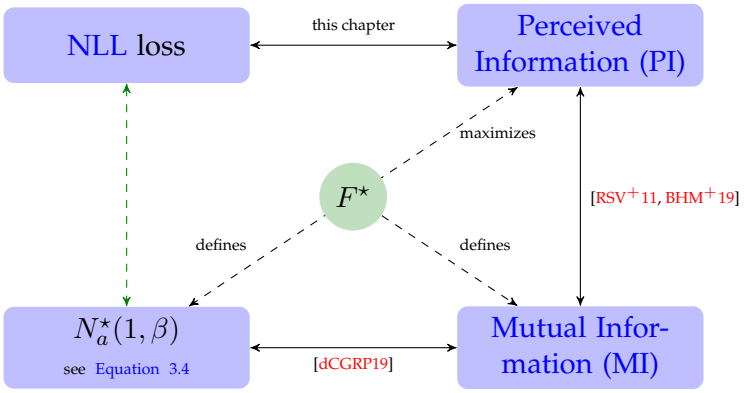


Figure 5.1: Link between the NLL loss and the efficiency metric in SCA.

As we shall show, the latter result has several direct impacts. First, the training of DNNs with the NLL loss can be considered as an efficient and effective estimation of the PI, and thereby of the MI – known to be complex to accurately estimate in the context of secure implementations [PR10, BGP+11]. Secondly, it implies that in an SCA context, choosing the NLL loss function to drive the training is sound when it comes to address the profiled SCA optimization problem. Thirdly, it enables to quantitatively study the impact of classical SCA counter-measures on the efficiency of deep learning based SCA and to formally verify that they stay sound.

Outline. The remaining of the chapter is organized as follows. Section 5.2 proposes another way to tackle the evaluation by introducing the *Leakage Assessment* problem. Section 5.3 states the soundness of minimizing the NLL loss since it is nothing but maximizing the PI. A discussion about the tightness of such a lower bound can be found in Subsection 5.3.3. The second part of the chapter is dedicated to the validation of our theoretical results through several simulations in Section 5.4 and experiments presented in Section 5.5, in the context of implementations secured by secret-sharing, shuffling and desynchronization.

5.2 Model Training for Leakage Assessment

The problem substitution that we present in this section comes as a direct consequence of a series of works stating a close link between $N_a^\star(1,\beta)$, namely the efficiency corresponding to the optimal solution to Problem 2, and the MI between the target sensitive variable and the leakage. We briefly recall this link hereafter in Subsection 5.2.1, before stating its interest in our context in Subsection 5.2.2.

5.2.1 The Link between MI and SCA Efficiency

Mangard *et al.* [Man04, MOP07] have first stated a link between the SCA efficiency and ρ, the correlation coefficient between the true leakage and its model, in the case of first-order leakage. Following the notations introduced in Equation 3.12 and Equation 3.13, the authors show that:

$$N_a\left(\mathcal{D}_{\mathcal{S}_a}^{\mathrm{CPA}},1,\beta\right) = 3 + 8\frac{p_{1-\beta}^2}{\log^2\frac{1-\rho}{1+\rho}} \approx \frac{28}{\rho^2} \quad , \tag{5.1}$$

where $\beta \in [0,1]$ is the threshold defined in Equation 3.4, $p_{1-\beta}$ is a tabulated constant given by a statistical test. Later, Mangard *et al.* have emphasized a link between the correlation coefficient and the MI of a first-order additive Gaussian noise leakage model [MOS11]:

$$\mathsf{MI}\left(\mathbf{X}[t];\mathrm{Z}\right) = -\frac{1}{2}\log_2\left(1-\rho^2(\mathbf{X}[t],\varphi(\mathrm{Z}))\right) \approx \frac{\rho^2(\mathbf{X}[t],\varphi(\mathrm{Z}))}{2} \quad . \tag{5.2}$$

By combining Equation 5.1 and Equation 5.2, we get a first link between the MI and the CPA efficiency:

$$N_a\left(\mathcal{D}_{\mathcal{S}_a}^{\mathrm{CPA}},1,\beta\right) \propto \frac{1}{\mathsf{MI}\left(\mathbf{X}[t];\mathrm{Z}\right)} \quad . \tag{5.3}$$

Unfortunately, the link emphasized in Equation 5.3 implicitly involves the value of the SNR of the univariate leakage, which implies that this statement does not necessarily holds for any arbitrary leakage model. Even when considering univariate leakages, this only covers the efficiency of a CPA, and not necessarily the optimal attack.

The later issues have drawn a great interest into the SCA community, especially in view of the counter-measures used in SCA. Following the results of Prouff *et al.* [PR13] and Duc *et al.* [DDF19] in proving the soundness of higher-order secret-sharing schemes against SCA, the latter authors have extended Equation 5.3 to an optimal attacker in presence of such a scheme [DFS19, Eq. 18].[1] If $\mathsf{MI}\left(\mathbf{X};\mathrm{Z}_i\right)$ denotes the MI between a leakage $\mathbf{X}$ and one share Z_i of the sensitive target variable Z,[2] then one can bound $N_a^\star(1,\beta)$, namely the efficiency corresponding to the optimal solution to Problem 2, as follows:

$$\frac{cst\cdot\beta}{\mathsf{MI}\left(\mathbf{X};\mathrm{Z}_i\right)^{d/2}} \leq N_a^\star(1,\beta) \quad , \tag{5.4}$$

where cst is a constant, and d is the order of the secret-sharing scheme.

The works of Chésirey *et al.* at CHES 2019 [dCGRP19] extend the previous results to any arbitrary leakage model:

$$\frac{f(\beta)}{\mathsf{MI}\left(\mathrm{Z};\mathbf{X}\right)} \leq N_a^\star(1,\beta) \quad , \tag{5.5}$$

where f is a known, invertible, strictly increasing function defined in the authors' paper.

[1] The threat model of Duc *et al.*'s works [DFS19, DFS19] also covers attackers with adaptive message strategies, which is not necessarily the case of other similar results presented in this subsection. Yet, as stated in Subsection 3.1.3, adaptive strategies are beyond the scope of this thesis.

[2] And assuming to simplify that the MIs between $\mathbf{X}$ and every share Z_i are of the same order of magnitude.

In other words, one is ensured that no attack can succeed with a success rate higher than β within $\left\lceil \frac{f(\beta)}{\mathsf{MI}(\mathsf{Z};\mathbf{X})} \right\rceil$ queries to the target device $\mathcal{T}$. Chésirey *et al.* argue that the lower $N_a^\star$, the tighter Inequality (5.5). Nevertheless, from the point of view of conservative security evaluations, it remains interesting to compute the value of the left-hand side in Inequality (5.5), no matter the value of $N_a^\star$.

5.2.2 The Link between MI and PI

Unfortunately, computing the MI in the denominator also requires to perfectly know the true p.m.f. $\Pr(\mathsf{Z} \mid \mathbf{X})$. Like with the profiled SCA optimization problem – *i.e.* Problem 2 – and the supervised classification problem – *i.e.* maximizing the accuracy – this cannot be assumed in practice. To circumvent this issue, we can fortunately use the notion of PI extending the MI to accept p.m.f.s estimations [RSV+11].

Definition 8 (Perceived Information [BHM+19])**.** *Let $F : \mathcal{X} \to \mathcal{P}(\mathcal{Z})$. The* Perceived Information *between Z and $\mathbf{X}$ for the model F is denoted by $\mathsf{PI}(\mathsf{Z};\mathbf{X};F)$ and defined as:*

$$\mathsf{PI}(\mathsf{Z};\mathbf{X};F) \triangleq \mathsf{H}(\mathsf{Z}) + \sum_{s\in\mathcal{Z}} \Pr(\mathsf{Z}=s) \mathop{\mathbb{E}}_{\mathbf{X}\mid \mathsf{Z}=s} \left[\log_2 F(\mathbf{X})[s]\right] \quad . \tag{5.6}$$

Intuitively, when the p.m.f. $\Pr(\mathsf{Z} \mid \mathbf{X})$ is perfectly learned, the PI equals the MI, otherwise the first one is always lower than the latter one [BHM+19]. This is of great interest here since it enables to derive an upper bound of the left-hand side in Inequality (5.5), namely

$$\frac{f(\beta)}{\mathsf{MI}(\mathsf{Z};\mathbf{X})} \leq \frac{f(\beta)}{\mathsf{PI}(\mathsf{Z};\mathbf{X};F)} \triangleq \widetilde{N_a}(F,\beta) \quad . \tag{5.7}$$

We may shorten the previous notation to $\widetilde{N_a}(F)$ when β is implicitly set to the value 90%. Moreover, we can then compare different models in terms of their PI: the higher the PI, the lower the distance to MI and thereby the better the approximation of $\frac{f(\beta)}{\mathsf{MI}(\mathsf{Z};\mathbf{X})}$ by $\widetilde{N_a}(F)$. This leads to introduce a new intermediate problem, named *Leakage Assessment*.

Problem 3 (Leakage Assessment)**.** *Given a profiling set $\mathcal{S}_p \triangleq \{(\mathbf{x}_1, z_1), \ldots, (\mathbf{x}_{N_p}, z_{N_p})\}$, find the model $\mathcal{A}(\mathcal{S}_p)$ maximizing the PI over $\mathcal{S}_p$, i.e. such that:*

$$\forall F \in \mathcal{H}, \;\; \mathsf{PI}(\mathsf{Z};\mathbf{X};\mathcal{A}(\mathcal{S}_p)) \geq \mathsf{PI}(\mathsf{Z};\mathbf{X};F) \quad . \tag{5.8}$$

At this point, we have argued that addressing the Leakage Assessment Problem is *sound* for the profiled SCA optimization problem – *i.e.* Problem 2 – in the sense that it will enable to estimate a lower-bound of the optimal solution $N_a^\star$ of the latter problem. The following section aims at deeply studying Problem 3. We will show that training deep learning models with the NLL loss is asymptotically equivalent to this problem which implies that conducting profiled SCA with deep learning can be argued to be relevant within this framework.

5.3 NLL Minimization is PI Maximization

This section is devoted to show that a deep learning model trained by minimizing the NLL loss fits with Problem 3. Subsection 5.3.1 studies the link between the NLL loss and an information theoretic quantity called *cross entropy*, that we will define hereafter. Then, Subsection 5.3.2 will make a link between cross entropy and PI. Finally, Subsection 5.3.3 discusses the gap between the MI and a PI estimated by training deep learning based models. Eventually, it will be concluded that the MI can be accurately estimated thanks to this approach.

5.3.1 Recall on the Consistency of the NLL Loss with Cross Entropy

This subsection is devoted to recall to the unfamiliar reader some of the important machine learning notions which will be used afterwards in Subsection 5.3.2. In particular, the NLL loss minimization is asymptotically equivalent to the minimization of the cross entropy. We start by recalling those notions hereafter.

Definition 9 (Negative Log Likelihood). *Given* $\mathcal{S}_p = \{(\mathbf{x}_1, z_1), \ldots, (\mathbf{x}_{N_p}, z_{N_p})\}$*, and a model* $F : \mathcal{X} \to \mathcal{P}(\mathcal{Z})$*, the Negative Log Likelihood (NLL)*[3] *is defined as:*

$$\mathcal{L}_{\mathcal{S}_p}(F) \triangleq \frac{1}{|\mathcal{S}_p|} \sum_{(\mathbf{x},z)\in\mathcal{S}_p} -\log_2 F(\mathbf{x})[z] \ . \tag{5.9}$$

Furthermore, we define the maximum likelihood estimator[4] $\mathcal{A}(\mathcal{S}_p)$ *as the model from* $\mathcal{H}$ *that minimizes the NLL loss computed over the profiling set* $\mathcal{S}_p$*:* $\mathcal{A}(\mathcal{S}_p) \triangleq \operatorname{argmin}_{F\in\mathcal{H}} \mathcal{L}_{\mathcal{S}_p}(F)$ *.*

Definition 10 (Cross Entropy). *Given a joint probability distribution of a target sensitive variable* Z *and its leakage* $\mathbf{X}$ *denoted as* $\Pr(\mathbf{X}, \mathrm{Z})$*, we define the* cross entropy *as the expected value of each term in Equation 5.9:*

$$\mathcal{L}_{\mathbf{X},\mathrm{Z}}(F) \triangleq \mathop{\mathbb{E}}_{\mathbf{X},\mathrm{Z}} [-\log_2 F(\mathbf{X})[\mathrm{Z}]] \ . \tag{5.10}$$

The cross entropy is actually nothing but the expected value of the NLL loss computed over the profiling set of traces. Besides, according to the law of large numbers, for any fixed F the NLL loss converges in probabilities towards the cross entropy [SSBD14]. However, since the true joint distribution of Z and $\mathbf{X}$ is actually unknown, one cannot exactly compute the cross entropy. The hope behind the NLL minimization is that for a number N_p of profiling traces high enough, the obtained maximum likelihood estimator $\mathcal{A}(\mathcal{S}_p)$ will be a good candidate to minimize the cross entropy.

It is not trivial though that $\mathcal{L}_{\mathcal{S}_p}(\mathcal{A}(\mathcal{S}_p))$ converges in probabilities towards the minimal cross entropy, as $\mathcal{A}(\mathcal{S}_p)$ is varying for each value of N_p. Actually, the Cramer-Rao bound, a well known result in Statistics [Cra99], guarantees the latter convergence, but relies on assumptions that cannot be taken for granted, in particular the assumption that $\Pr(\mathrm{Z}|\mathbf{X}) \in \mathcal{H}$.

Thankfully, as a consequence of Theorem 2 introduced in Chapter 4, we are indeed ensured that $\mathcal{L}_{\mathcal{S}_p}(\mathcal{A}(\mathcal{S}_p)) \xrightarrow[|\mathcal{S}_p|\to\infty]{\mathcal{P}} \min_{F\in\mathcal{H}} \mathcal{L}_{\mathbf{X},\mathrm{Z}}(F)$. Therefore when the number of profiling traces converges towards infinity, we can substitute the analysis of the NLL loss with that of the cross entropy. It remains now to draw the link between cross entropy and PI, in order to address the Leakage Assessment Problem – *i.e.* Problem 3.

5.3.2 The Link between Cross Entropy and Perceived Information

This section aims at explaining to what extent the PI and the cross entropy introduced in the previous section are linked. It is argued here that the PI actually equals the cross entropy up to constant factors. Such a link and the reduction argued in Subsection 5.3.1 will allow us to guarantee that minimizing the NLL loss is a consistent approach for solving the Leakage Assessment Problem. It is recalled that the PI has been formally defined in Section 5.2. We also introduce hereafter the *Empirical Perceived Information*, as given in [BHM+19].

[3] The NLL has already been introduced in Subsection 2.2.1.

[4] Minimizing the NLL loss is equivalent to maximizing the Log Likelihood.

Definition 11 (Empirical Perceived Information [BHM+19]). *Let $F : \mathcal{X} \to \mathcal{P}(\mathcal{Z})$. The* Empirical Perceived Information, *denoted as $\widehat{\mathsf{PI}}_{\mathcal{S}_p}(\mathrm{Z}; \mathbf{X}; F)$, is defined from a profiling set $\mathcal{S}_p$ as follows:*

$$\widehat{\mathsf{PI}}_{\mathcal{S}_p}(\mathrm{Z}; \mathbf{X}; F) \triangleq \mathsf{H}(\mathrm{Z}) + \sum_{s \in \mathcal{Z}} \Pr(\mathrm{Z} = s) \frac{1}{|\mathcal{S}_p|} \sum_{\substack{(\mathbf{x}, z) \in \mathcal{S}_p \\ z = s}} \log_2 F(\mathbf{x})[z] \ . \tag{5.11}$$

Informally, the PI is defined the same way as the MI, but by substituting the *uncertainty* of the true p.m.f., namely $\log_2 \Pr(\mathrm{Z} \mid \mathbf{X} = \mathbf{x})$, with the uncertainty of the approximating p.m.f., namely $\log_2 F(\mathbf{X})$. Surprisingly, this substitution is exactly what defines the cross entropy.

Proposition 1. *Let Z be a random variable with uniform distribution over $\mathcal{Z}$ of cardinal $|\mathcal{Z}|$. Then, the cross entropy and the NLL loss of a model $F \in \mathcal{H}$ are respectively linked to the PI and its empirical estimation as follows:*

$$\log_2 |\mathcal{Z}| - \mathcal{L}_{\mathbf{X},\mathrm{Z}}(F) = \mathsf{PI}(\mathrm{Z}; \mathbf{X}; F) \ , \tag{5.12}$$

$$\log_2 |\mathcal{Z}| - \mathcal{L}_{\mathcal{S}_p}(F) = \widehat{\mathsf{PI}}_{\mathcal{S}_p}(\mathrm{Z}; \mathbf{X}; F) \ . \tag{5.13}$$

Proof. The assumption about Z implies that $\mathsf{H}(\mathrm{Z}) = \log_2 |\mathcal{Z}|$. Injecting the latter result into the definition of the PI, and by using the formula of total probabilities for the expected value we have:

$$\begin{aligned}
\mathsf{PI}(\mathrm{Z}; \mathbf{X}; F) &\triangleq \mathsf{H}(\mathrm{Z}) + \sum_{s \in \mathcal{Z}} \Pr(\mathrm{Z} = s) \mathop{\mathbb{E}}_{\mathbf{X} \mid \mathrm{Z} = s} [\log_2 F(\mathbf{X})[s]] \ , \\
&= \log_2 |\mathcal{Z}| + \mathop{\mathbb{E}}_{\mathrm{Z}} \left[\mathop{\mathbb{E}}_{\mathbf{X} \mid \mathrm{Z}} [\log_2 F(\mathbf{X})[\mathrm{Z}]] \right] \ , \\
&= \log_2 |\mathcal{Z}| - \mathop{\mathbb{E}}_{\mathbf{X}, \mathrm{Z}} [-\log_2 F(\mathbf{X})[\mathrm{Z}]] \ , \\
&= \log_2 |\mathcal{Z}| - \mathcal{L}_{\mathbf{X},\mathrm{Z}}(F) \ .
\end{aligned}$$

The proof for the empirical PI follows exactly the same reasoning substituting expected values with averages.[5] □

Proposition 1, which is illustrated on Figure 5.2, tells us that the PI can be expressed as a cross entropy, provided that the entropy of the sensitive variable Z is known. As already pointed out in [BHM+19, Thm. 6], for all $F \in \mathcal{H}$ we have $\mathsf{PI}(\mathrm{Z}; \mathbf{X}; F) \leq \mathsf{MI}(\mathrm{Z}; \mathbf{X})$.[6] In other words, computing the cross entropy of any deep learning model enables to get a lower bound of the MI. This tells nothing about the tightness of such a bound yet. Hopefully, based on the previous results stated in this section, we now know how to tighten this inequality, as stated by the following proposition.

Proposition 2. *Let $\mathcal{S}_p$ be a profiling set and let $\mathcal{A}(\mathcal{S}_p)$ be the maximum likelihood estimator, namely such that $\mathcal{A}(\mathcal{S}_p) \triangleq \operatorname{argmin}_{F \in \mathcal{H}} \mathcal{L}_{\mathcal{S}_p}(F)$. Then:*

1. *$\mathcal{A}(\mathcal{S}_p)$ maximizes the empirical PI,*
2. *the information perceived by $\mathcal{A}(\mathcal{S}_p)$ converges in probabilities towards the maximum of PI over $\mathcal{H}$.*

[5] Actually, Equation 5.13 only holds if $\mathcal{S}_p$ is *balanced*, *i.e.* if the number of traces is the same for each class in $\mathcal{S}_p$. This can be assumed without loss of generality, since Z is drawn uniformly.

[6] This comes from the fact that the gap between the MI and the PI can be rephrased as a KL-divergence term, which is always non-negative.

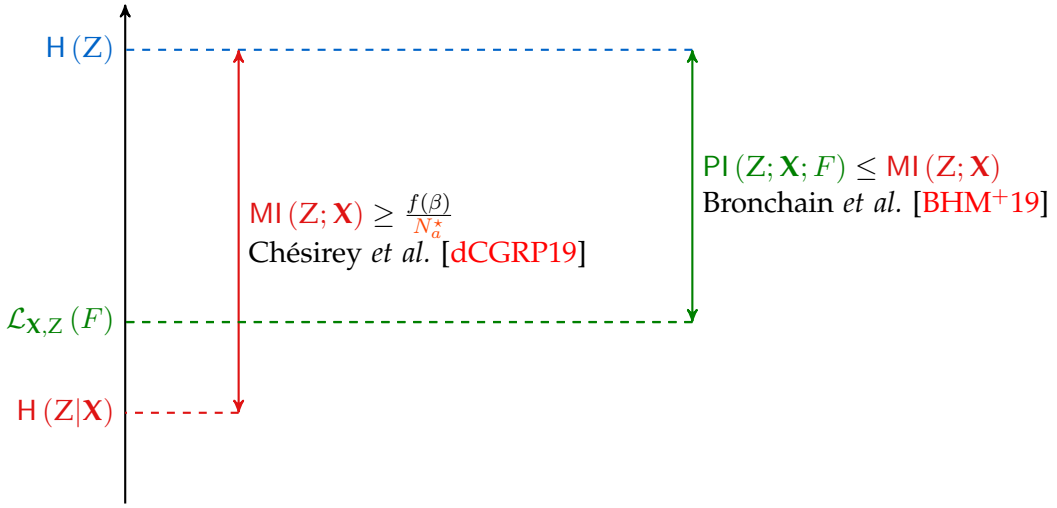


Figure 5.2: Illustration of Proposition 1.

$$\widehat{\mathsf{PI}}_{\mathcal{S}_p}\left(\mathsf{Z};\mathbf{X};\mathcal{A}\left(\mathcal{S}_p\right)\right) \xrightarrow[|\mathcal{S}_p|\to\infty]{\mathcal{P}} \max_{F\in\mathcal{H}} \mathsf{PI}\left(\mathsf{Z};\mathbf{X};F\right) \leq \mathsf{MI}\left(\mathsf{Z};\mathbf{X}\right) \tag{5.14}$$

$$\mathsf{PI}\left(\mathsf{Z};\mathbf{X};\mathcal{A}\left(\mathcal{S}_p\right)\right) \xrightarrow[|\mathcal{S}_p|\to\infty]{\mathcal{P}} \max_{F\in\mathcal{H}} \mathsf{PI}\left(\mathsf{Z};\mathbf{X};F\right) \leq \mathsf{MI}\left(\mathsf{Z};\mathbf{X}\right) \tag{5.15}$$

Roughly speaking, Proposition 2 states that the NLL loss minimization is asymptotically equivalent to the PI maximization mentioned in the Leakage Assessment Problem (*i.e.* Problem 3).

Proof. Starting from Equation 5.12, and applying Equation 4.6 from Theorem 2 to get

$$\begin{aligned}
\log_2|\mathcal{Z}| - \mathsf{PI}\left(\mathsf{Z};\mathbf{X};\mathcal{A}\left(\mathcal{S}_p\right)\right) \overset{(5.12)}{=} \mathcal{L}_{\mathbf{X},\mathsf{Z}}\left(\mathcal{A}\left(\mathcal{S}_p\right)\right) &\xrightarrow[|\mathcal{S}_p|\to\infty]{\overset{(4.6)}{\mathcal{P}}} \min_{F\in\mathcal{H}} \mathcal{L}_{\mathbf{X},\mathsf{Z}}\left(F\right) \\
&= \min_{F\in\mathcal{H}} \left(\log_2|\mathcal{Z}| - \mathsf{PI}\left(\mathsf{Z};\mathbf{X};F\right)\right) \\
&= \log_2|\mathcal{Z}| - \max_{F\in\mathcal{H}} \mathsf{PI}\left(\mathsf{Z};\mathbf{X};F\right)
\end{aligned}$$

Hence the result given in Equation 5.15. The proof for Equation 5.14 follows exactly the same reasoning, replacing $\mathsf{PI}\left(\mathsf{Z};\mathbf{X};\mathcal{A}\left(\mathcal{S}_p\right)\right)$ by $\widehat{\mathsf{PI}}_{\mathcal{S}_p}\left(\mathsf{Z};\mathbf{X};\mathcal{A}\left(\mathcal{S}_p\right)\right)$, and applying Equation 4.7 instead of Equation 4.6. □

Therefore, on the one hand, we have a theoretically grounded method to address the Leakage Assessment Problem (*i.e.* Problem 3) thanks to Proposition 2, namely by minimizing the NLL loss. On the other hand, since it has been argued in Section 5.2 that solving the Leakage Assessment was sound in order to address the SCA Optimization Problem, it follows from Proposition 2 the main result of this chapter, given hereafter.

Corollary 1 (Main Result). *Let $\mathcal{A}\left(\mathcal{S}_p\right)$ be the maximum likelihood estimator with respect to the profiling set $\mathcal{S}_p$, namely such that $\mathcal{A}\left(\mathcal{S}_p\right) \triangleq \operatorname{argmin}_{F\in\mathcal{H}} \mathcal{L}_{\mathcal{S}_p}\left(F\right)$. Then $\mathcal{A}\left(\mathcal{S}_p\right)$ asymptotically minimizes the quantity $\widetilde{N}_a\left(\mathcal{A}\left(\mathcal{S}_p\right),\beta\right) \triangleq \frac{f(\beta)}{\mathsf{PI}(\mathsf{Z};\mathbf{X};F)}$ when the size of $\mathcal{S}_p$ tends towards infinity,[7] which is an upper-bound of the left-hand side in Inequality (5.5).*

Proof. By applying Proposition 2, we get

$$\widetilde{N}_a\left(\mathcal{A}\left(\mathcal{S}_p\right),\beta\right) \xrightarrow[|\mathcal{S}_p|\to\infty]{\mathcal{P}} \frac{f(\beta)}{\max_{F\in\mathcal{H}} \mathsf{PI}\left(\mathsf{Z};\mathbf{X};\mathcal{A}\left(\mathcal{S}_p\right)\right)} = \min_{F\in\mathcal{H}} \frac{f(\beta)}{\mathsf{PI}\left(\mathsf{Z};\mathbf{X};F\right)} \geq \frac{f(\beta)}{\mathsf{MI}\left(\mathsf{Z};\mathbf{X}\right)}$$

[7] We recall that f is a known, invertible, strictly increasing function defined by Chésirey *et al.* [dCGRP19], and $\beta \in [0,1]$ is the threshold defined in Equation 3.4.

□

In other words, Corollary 1 tells us that minimizing the NLL loss is sound for the SCA Optimization Problem (*i.e.* Problem 2), in the sense that has been argued in Section 5.2, and that the term $\widetilde{N}_a\left(\mathcal{A}\left(\mathcal{S}_p\right), \beta\right)$ might be a good approximation of the lower bound of Inequality (5.5). However, this also emphasizes that in the pursuit of estimating $N_a^\star(1, \beta)$ through the NLL minimization, some weaknesses must be discussed.

First, as recalled in Section 5.2, the higher $N_a^\star(1, \beta)$, the looser Inequality (5.5). It is therefore of natural interest to verify to what extent the tightness of the latter inequality holds, in view of estimating $N_a^\star(1, \beta)$ by $\frac{f(\beta)}{\mathsf{MI}(\mathsf{Z};\mathbf{X})}$. This must be at least empirically verified. Second, the tightness of Inequality (5.15) is another possible source of imprecision when one wants to substitute the MI with the PI. This will be discussed in the next section, and will eventually be verified through simulations and experiments in Section 5.4 and Section 5.5.

5.3.3 Tightness of the Obtained Bound

So far we have argued that minimizing the NLL loss is a sound approach to tackle Problem 3: it is indeed equivalent to minimizing the cross entropy – *cf* Equation 5.10, thereby equivalent to maximizing the PI – *cf* Equation 5.6. In the particular case where the hypothesis class $\mathcal{H}$ is a set of neural networks, it becomes now of natural interest to study the gap between the MI and the NLL loss we are minimizing (or equivalently the empirical PI we are maximizing) to assess the quality of the built solution: the tighter the gap, the more accurate our estimation $\widetilde{N}_a\left(F\right)$ of the efficiency of the optimal attack $N_a^\star$ in view of assessing the worst-case scenario from an evaluator's point of view.

To this end, we may start from the discussion proposed in Subsection 4.1.3 about the decomposition of the error terms, by updating them after instantiating the loss function with the NLL:

$$
\begin{aligned}
\widehat{\mathsf{PI}_{N_p}}\left(\mathsf{Z};\mathbf{X};\widetilde{\mathcal{A}}\left(\mathcal{S}_p\right)\right) = & \left(\widehat{\mathsf{PI}_{N_p}}\left(\mathsf{Z};\mathbf{X};\widetilde{\mathcal{A}}\left(\mathcal{S}_p\right)\right) - \widehat{\mathsf{PI}_{N_p}}\left(\mathsf{Z};\mathbf{X};\mathcal{A}\left(\mathcal{S}_p\right)\right)\right) & \leq 0 & \quad (5.16)\\
+ & \left(\widehat{\mathsf{PI}_{N_p}}\left(\mathsf{Z};\mathbf{X};\mathcal{A}\left(\mathcal{S}_p\right)\right) - \sup_{F\in\mathcal{H}} \mathsf{PI}\left(\mathsf{Z};\mathbf{X};F\right)\right) & \geq 0 & \quad (5.17)\\
+ & \left(\sup_{F\in\mathcal{H}} \mathsf{PI}\left(\mathsf{Z};\mathbf{X};F\right) - \mathsf{MI}\left(\mathsf{Z};\mathbf{X}\right)\right) & \leq 0 & \quad (5.18)\\
+ & \mathsf{MI}\left(\mathsf{Z};\mathbf{X}\right) \quad , & \geq 0 & \quad (5.19)
\end{aligned}
$$

where $\widetilde{\mathcal{A}}\left(\mathcal{S}_p\right)$ denotes the model returned by the heuristic optimization algorithm – *e.g.*, SGD or its variants such as Adam, see Subsection 2.5.2 – rather than the true maximum likelihood estimator $\mathcal{A}\left(\mathcal{S}_p\right)$.

The Bayes' error term (4.11) becomes here the informational security bound on the leakage (5.19). The approximation error term (4.10) quantifies now the gap (5.18) to the *computational* bound, namely the best profiled attacker based on the given hypothesis class $\mathcal{H}$. The estimation error (4.9) is updated as (5.17), and the optimization error (4.8) is now quantified by the term (5.16). It is worth mentioning we argued that both terms (5.16) and (5.18) are negative, as they respectively result from the opposite of the error terms (4.8) and (4.10) discussed in Subsection 4.1.3.

Therefore, this instantiation of the error terms enables to draw an insightful parallel between the ML metrics and the SCA ones. Eventually, the whole discussion conducted in this section can be synthesized in Table 5.1.

Table 5.1: Machine learning metrics and their meaning in Side-Channel Analysis

ML meaning	ML metric		SCA metric	SCA meaning
Perfect model	$\mathsf{H}(\mathsf{Z} \mid \mathbf{X})$	=	$\log_2 \lvert\mathcal{Z}\rvert - \mathsf{MI}(\mathsf{Z}; \mathbf{X})$	Informational security bound on $\mathsf{Z} \mid \mathbf{X}$
+ Approximation error	$\inf_{F \in \mathcal{H}} \mathcal{L}_{\mathbf{X},\mathsf{Z}}(F)$	$\iff$	$\sup_{F \in \mathcal{H}} \mathsf{PI}(\mathbf{X}; \mathsf{Z}; F)$	Computational bound
Cross Entropy	$\mathcal{L}_{\mathbf{X},\mathsf{Z}}(F)$	=	$\log_2 \lvert\mathcal{Z}\rvert - \mathsf{PI}(\mathsf{Z}; \mathbf{X}; F)$	Perceived Information
NLL loss	$\mathcal{L}_{\mathcal{S}_p}(F)$	=	$\log_2 \lvert\mathcal{Z}\rvert - \widehat{\mathsf{PI}}_{N_p}(\mathsf{Z}; \mathbf{X}; F)$	Estimated PI

5.3.4 Partial Conclusions

The results we have stated so far are threefold.

First, it has been argued that addressing the profiled SCA optimization problem may be done by considering the Leakage Assessment – *i.e.* Problem 3 – aiming at finding a model extracting the most perceived information, rather than choosing, for example, the model maximizing the accuracy.

Second, the loss function we are usually minimizing, namely the NLL loss can be interpreted as a perceived information to maximize. That is why in Section 5.4 and Section 5.5, we will plot the PI, as computed with Equation 5.13, since it will enable to replace the accuracy in order to compare and evaluate the efficiency of a trained model.

Third, to discuss the tightness of Inequality (5.14), we can decompose the gap into three terms, namely the approximation error, the estimation error and the optimization error. Each error term refers to a restriction in the capacity of an evaluator. The experiments conducted in Sections 5.4 and 5.5 study the practical impact of each term.

5.4 Study on Simulated Data

This section confronts the different propositions made so far with simulated experiments. The aim of these experiments are:

- to show experimentally that the PI, as computed in Equation 5.12, is indeed a lower bound of the MI;
- to show, in some cases where we can compute the exact MI between a sensitive target variable and a leakage, that the latter lower bound is tight, so that the PI gives an accurate estimation of the MI;
- to see to what extent the commonly used counter-measures adapted for SCA have a practical impact on the training of DNNs.

To this end, we first present the settings of our simulations in Subsection 5.4.1, and we afterwards analyze them in Subsection 5.4.2.

5.4.1 Settings of the Experiments

To verify the tightness of the bounds, we simulate simple D-dimensional leakages from an n-bit sensitive variable Z. The traces are defined such that for every $t \in [\![1, D]\!]$:

$$\mathbf{x}_i[t] = \begin{cases} u_i + b_i, \text{ if } t \notin \{t_1, \ldots, t_{d+1}\} \\ \mathrm{hw}(z_{t,i}) + b_i \text{ otherwise} \end{cases}, \tag{5.20}$$

where $(u_i)_i, (b_i)_i$ and all $(z_{t,i})_i$ are i.i.d. draws from the following independent random variables. Respectively, $\mathsf{U} \sim \mathcal{B}(n, 0.5)$, $\mathsf{B} \sim \mathcal{N}(0, \sigma^2)$,[8] where and where $(z_{1,i}, \ldots, z_{d+1,i})$ is a

[8] It is recalled that hw denotes the Hamming weight function, see Subsection 3.5.1.

$(d+1)$-sharing of z_i for the bit-wise addition law. This example corresponds to a situation where the leakages of the shares are hidden among values having no relation with the target, but have the same marginal p.m.f. Since the $z_{t,i}$ are drawn uniformly, $\mathrm{hw}(z_{t,i})$ follows a binomial marginal p.m.f. so they are indistinguishable without prior knowledge. Hence the choice of a binomial law for U when emulating non-informative components. In order to have an *exhaustive* dataset, every possible combination of the $(d+1)$-sharing has been generated and replicated q times before adding the noise, where q is given afterwards in the experiments. Therefore, it contains $|\mathcal{S}_p| = q \times 2^{(d+1)n}$ simulated traces. Once the data were generated, we trained a DNN from the hypothesis class $\mathcal{H}$ of the MLPs with $L = 1$ hidden layer made of $C = 1,000$ neurons. The training loss is naturally the NLL loss.[9] The training lasts, after $T = 200$ epochs,[10] by applying the SGD algorithm with a learning rate of 10^{-3}. The trained model is denoted $\hat{\mathcal{A}}(\mathcal{S}_p)$

Our simulations comprise three main campaigns:

Experiment 1 (secret-sharing only): in this experiment, we set $D = d + 1$ in order to avoid to consider irrelevant input features. The simulations are done over $n = 4$ bits, $d \in \{0, 1, 2, 3\}$ and $\sigma \in \{0.01, 0.1, 0.2, 0.4, 0.8, 1.6, 3.2\}$. We also generate enough data so that the training set is exhaustive, *i.e.*, the number of replicas is $q = 2,000$. With such a generated dataset, we expect to make the estimation error (4.9) negligible. The gap between the MI and the PI should therefore only be composed of the optimization error (4.8) and the approximation error (4.10).

Experiment 2 (secret-sharing, with uninformative components): in a second experiment, we have $D = 40$, including the uninformative components. Since all components share the same margin law, we recall that they cannot be distinguished without knowing Z. Compared to Experiment 1, we might expect the optimization error to be more important because of the potential difficulty induced by the presence of uninformative components.

Experiment 3 (shuffling, no secret-sharing): in a third experiment, we set $d = 0$, $D \in \{2, 4, 16\}$; in other words, $D - 1$ uninformative components are added like in Experiment 2, but this time they are randomly shuffled with the only informative component. Note that the shuffling is different for each simulated trace so that one cannot guess in which position the informative leakage lies. Therefore, we expect the information perceived by the model to be lower than without shuffling [VMKS12]. Besides, $\sigma \in \{0.04, 0.2, 0.4, 0.8, 1.6, 3.2\}$ here.

PI and MI Estimation. From those experiments, the PI is estimated thanks to a *hold-out* dataset of $1/5$-th of the size of the training set size, *i.e.* those data are not used by the optimization algorithm. For the sake of comparison, we estimate the MI between the target sensitive 4-bit variable and its simulated leakage model with a Monte Carlo sampling of the leakage p.m.f. $\Pr(\mathbf{X} \mid Z)$. The methodology is described in Subsection 2.2.4. The only difficulty is to efficiently compute the precise p.m.f. $h : s \mapsto h(s) = \Pr(\mathbf{X} = \mathbf{x} \mid Z = s)$ given a simulated trace $\mathbf{x}$ in the presence of the counter-measures, *i.e.* secret-sharing or shuffling. We describe hereafter of such computations can be done.

Secret-sharing. We benefit from the technique suggested by Lomné *et al.* at CHES 2014 [LPR+14] that we recall hereafter. Suppose that one knows the p.m.f. of each share

[9] Beware that in Pytorch and Tensorflow, the NLL loss is computed with natural logarithms, whereas here one ought to consider the logarithm in base 2.

[10] One epoch refers to the number of iterations needed to process the whole dataset through the SGD algorithm.

for a given leakage trace $\mathbf{x}$, denoted as $h_i(s) \triangleq \Pr(\mathbf{X} = \mathbf{x} \mid \mathrm{Z}_i = s)$, applying the total probability formula – see Subsection 2.2.1 – leads to:

$$h(s) = \sum_{s_1} \cdots \sum_{s_d} h_0(s \oplus s_1 \oplus \cdots \oplus s_d)\, h_1(s_1) \cdots h_d(s_d) \quad . \tag{5.21}$$

As described above in this section, we assume for our simulations that the leakage of each share $z_{t,i}$ is following a Gaussian law $\mathcal{N}(\mathrm{hw}(z_{t,i}), \sigma^2)$ so we explicitly know the functions h_i. It turns out that the right hand-side can be seen as the following convolution product $(h_0 * h_1 * \cdots * h_d)(s)$ in the vector space $\mathbb{F}_2^n$.[11] Discrete convolutions in $\mathbb{F}_2^n$ can be efficiently computed by using a variant of the D.F.T. called Walsh-Hadamard (WH) transform. Like with a regular fast Fourier transform, this trick allows to compute $\Pr(\mathbf{X} = \mathbf{x} \mid \mathrm{Z})$ with a runtime complexity of $\mathcal{O}(d \cdot |\mathcal{Z}| \cdot \log |\mathcal{Z}|)$ instead of $\mathcal{O}\left(|\mathcal{Z}|^d\right)$ with a naive computation of Equation 5.21.

The outcomes of those MI estimations are depicted by the orange, blue and pink lines in Figure 5.3a and Figure 5.3b.

Shuffling. We benefit from the analysis conducted by Veyrat-Charvillon *et al.* at ASIACRYPT 2012 [VMKS12]. The shuffling counter-measure simulated here assumes that no leakage from the permutation indices occurs, so the conditional p.d.f. to compute can be written as follows:

$$\Pr(\mathbf{X} = \mathbf{x} \mid \mathrm{Z} = s) = \frac{1}{D} \sum_{j=1}^{D} \Pr(\mathbf{X} = \mathbf{x} \mid \mathrm{Z}_j = s) \quad , \tag{5.22}$$

where Z_j denotes the random variable modelizing all the shares $z_{j,i}$ for $i \in [\![1, |\mathcal{S}_p|]\!]$. Since in all the terms of the sum in Equation 5.22 except one, $\mathbf{X}$ and Z_j are independent, "it boils down to consider $D - 1$ leakages out of the D as algorithmic noise" [VMKS12].

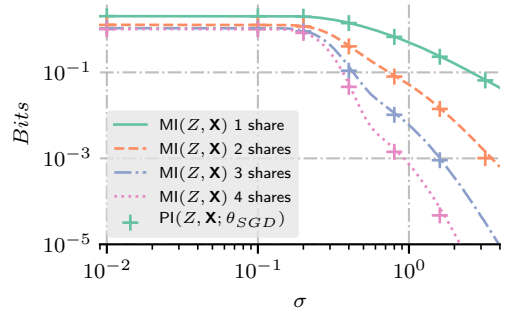
5.4.2 Analysis of the Results

In this section we analyze the results obtained by running Experiments 1 (Figure 5.3a), 2 (Figure 5.3b) and 3 (Figure 5.3c). On each figure, the lines correspond to the estimated MI and the crosses correspond to the information perceived by the trained MLP, as computed from the NLL loss with Equation 5.12. Based on these results, several observations can be done.

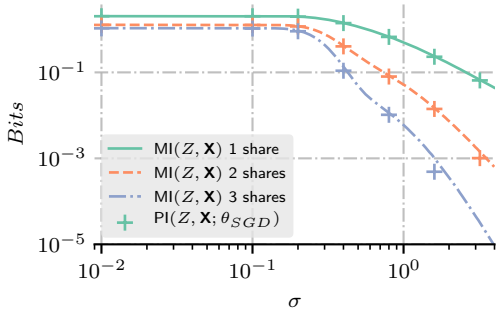
First, on each result the crosses are always below the lines, which is in line with the results given in the literature: the estimated PI is a lower bound of the MI. But more interestingly, the crosses are always close to the line no matter the MI magnitude. In the case of Experiment 1, we argued that the error was composed of the approximation and optimization errors, so their sum is negligible. Since we recalled in Subsection 5.3.3 that those terms are both negative, we conclude that even for a simple MLP with one layer and $1,000$ neurons, both errors can be ignored. This is of particular interest concerning the approximation error, as it always decreases with the number of layers and the number of neurons inside each layer of the MLP. Therefore, in the case of a Hamming weight leakage model with additive Gaussian noise, any *more sophisticated* MLP (*i.e.* with more layers or more neurons by layer) will also have a negligible approximation error.

Secondly, the PI plotted in Figure 5.3b shows that the presence of uninformative components in Experiment 2 does not annihilate the capacity of the MLP to optimally extract information about the target variable, provided that these components are not shuffled with

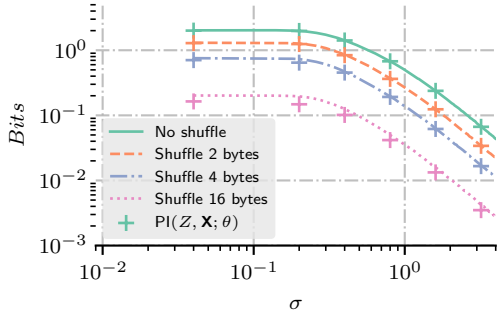
[11] We inform the unfamiliar reader that $\mathbb{F}_{2^n}$ denotes the finite field with 2^n elements, whereas $\mathbb{F}_2^n$ denotes the same set, seen as a vector space of dimensionality n, with respect to the field $\mathbb{F}_2$.



(a) Experiment 1



(b) Experiment 2



(c) Experiment 3

Figure 5.3: Information perceived by the MLP.

informative ones. This shows that the optimization error, which was thought to be increased compared to Experiment 1, remains stable.

Finally, the preceding observations hold regardless the considered counter-measure, namely secret-sharing (Figure 5.3a and Figure 5.3b) or shuffling (Figure 5.3c). This can be interpreted as the fact that the MLP trained through the NLL loss minimization is able to give a model optimally extracting the remaining informative leakage, while being "agnostic" concerning the presence or not of such counter-measures. Nevertheless, since both counter-measures have been shown to decrease the MI – exponentially with the level of noise for secret-sharing as explained in Subsection 3.7.1 or linearly for shuffling as explained in Subsection 3.7.2 – they remain sound against Deep Learning.

At this stage, we have argued thanks to our simulations that the approximation error is negligible, no matter the considered counter-measure, nor the architecture of a MLP, while the optimization error is likely to remain negligible as well. Therefore, our MI estimation obtained by PI maximization seems accurate. This provides an empirical validation of Proposition 2. As another consequence, we are fairly confident that in the case of such simple leakage models, which often happen on real use cases, replacing an optimal architecture by another should not degrade too much the MI estimation.[12] These observations must be challenged by tests on experimental traces, where one cannot have an exhaustive dataset. This will naturally lead to discussions regarding the estimation error which has not been investigated here.

5.5 Application on Experimental Data

So far, we have seen that DNNs could reach the informational security bounds of a leakage in simulated experiments, thereby giving useful estimations for the developer. This success did not rely on any prior knowledge on the leakage, but was achieved thanks to a simple

[12] However, one cannot get such a conclusion if one considers another leakage model.

MLP with one hidden layer. To confirm these observations, we propose to complete the investigations by considering experimental leakage traces from the Chip Whisperer dataset presented in Subsection 3.8.1. Subsection 5.5.1 presents the methodology of our experiments, and Subsection 5.5.2 discusses their results.

5.5.1 Methodology

The hypothesis class $\mathcal{H}$. The hypothesis class $\mathcal{H}$ used for the next two experiments, namely Experiments 4 and 5, is defined as the set of MLP with $L = 1$ hidden layer and $C = 500$ hidden neurons. That is,

$$F = s \circ \lambda_2 \circ \mu \circ \omega_q \circ \sigma \circ \lambda_1 \circ \mu \ , \tag{5.23}$$

where ω_q denotes a dropout layer of parameter q, compared to the architecture presented in Subsection 4.2.3. The dropout parameter has been set to $q = 0.1$ *i.e.* each neuron of the hidden layer is randomly set to 0 with probability q each time an output $F(\mathbf{x})$ is computed during the optimization.

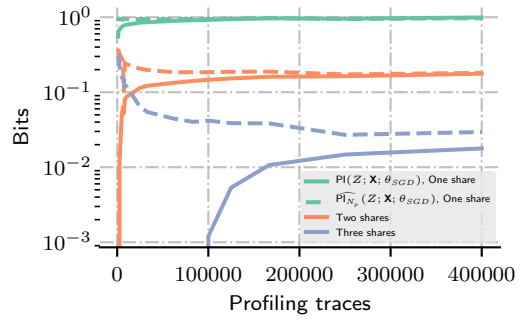
Common settings. The trainings have been done with the Adam algorithm through a number of epochs denoted by T, *i.e.*, each trace has been processed T times by the algorithm. Over the $500,000$ profiling traces, a portion α is used for the training, and the remaining is used as a hold-out set for computing an unbiased estimate of the perceived information. In other words, the profiling set is made of $N_p = \alpha \times 500,000$ traces while the hold-out set is made of $N_v = (1-\alpha) \times 500,000$ traces. We fix the limit $\alpha \leq 4/5$ so that the quality of the estimation over the hold-out set remains satisfying: the error margin will be at most 10^{-2} with a confidence at least 90% in the worst case, according to Chebyshev's inequality – see Subsection 2.2.1.

Experiment 4 on Boolean secret-sharing. When considering secret-sharing, the generated target values are $\mathsf{Z} = \bigoplus_{j \in [\![0,d]\!]} \texttt{plain}[j]$ for $d \in \{0,1,2\}$, where $\oplus$ denotes the `xor` operation between two bytes. This way, it can simulate leakages of order d.

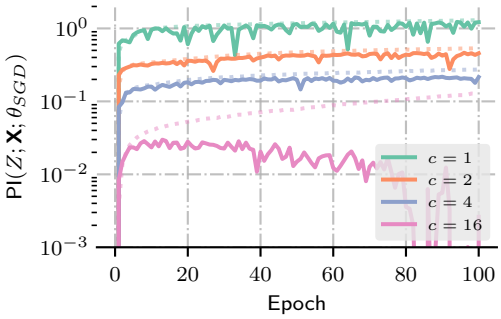
Provided with these target values, we selected P.o.Is based on the magnitude of the SNR: between 4 and 6 P.o.Is are selected in decreasing order of magnitude of SNR from each of the three first bytes of the plaintext array – see Figure 3.8. The time coordinates 13 to 16, 25 to 30 and 37 to 41 correspond respectively to the P.o.Is of the latter bytes manipulation. This gives an input dimension of $D = 15$. This way, we hoped to reduce the quantity of irrelevant components, which would have made the optimization with Adam harder, and therefore hoped to get a good estimate corresponding the most to the approximation error (5.18). Details of the trained MLP can be found hereafter. We set $T = 200$ and let α vary so that $|\mathcal{S}_p| \in [\![1,000; 400,000]\!]$. This way, we will be able to plot the so-called *learning curve*, namely plotting the values of $\mathsf{PI}\left(\mathsf{Z}; \mathbf{X}; \widetilde{\mathcal{A}}\left(\mathcal{S}_p\right)\right)$ and $\widehat{\mathsf{PI}}_{|\mathcal{S}_p|}\left(\mathsf{Z}; \mathbf{X}; \widetilde{\mathcal{A}}\left(\mathcal{S}_p\right)\right)$ depending on $|\mathcal{S}_p|$. This is a classical representation in machine learning. On a learning curve, it is expected that the empirical PI decreases with $|\mathcal{S}_p|$ while the true PI increases, and both converge towards the supremum of the PI [Vap95]. This representation enables to discuss the estimation error (5.17) according to the size of the profiling set.

Experiment 5 on shuffling. When considering shuffling, the generated target values are $\mathsf{Z} = \texttt{plain}[j]$ where j is randomly drawn from a subset of $[\![0,15]\!]$ of size c, c denoting the number of shuffled bytes.

Contrary to the experiments on secret-sharing, we did not selected P.o.Is but only restricted the target window to the $D = 250$ first time samples of the traces, which was sufficient to cover the leakages of every shuffled plaintext byte (see Figure 3.8). Afterwards, a CNN with a VGG-like architecture has been used for those trainings.



(a) Learning curve of Experiment 4 on Boolean secret-sharing.



(b) Results of Experiment 5 on shuffling.

Figure 5.4: Results on experimental data.

We set $\alpha = 4/5$, $T = 100$, and $c \in \{1, 2, 4, 16\}$. The aim of this experiment is to empirically verify the trend observed on the Experiment 3 (Figure 5.3c), namely a linear decrease of PI with the number of shuffled bytes.

5.5.2 Results and Discussions

Figure 5.4a presents the learning curves of Experiment 4, when targeting respectively $1, 2$ or 3 shares among the considered ones. The dotted curves are the estimated PI over the $N_p = |\mathcal{S}_p|$ profiling traces whereas the plain curves denote the PI estimated with the N_v validation traces.

It may first be observed that the amount of information leaking on the sensitive un-split variable seems to decrease at an exponential rate in the number of shares, as expected from both theory – see Subsection 3.7.1 – and our simulations – see Section 5.4. More interestingly, the gap between dotted curves and their corresponding plain ones exactly corresponds to the estimation error term (5.17). It appears then that the latter one becomes negligible relatively to the PI when the profiling set size exceeds respectively a few thousands when targeting one share, or one hundred thousand when targeting two shares. When targeting three shares, the estimation error is not completely negligible, even with $400,000$ profiling traces. It is furthermore particularly noticeable that when profiling the three shares scheme with less than $100,000$ traces, the learning phase completely failed since the PI was null. This indicates that, in addition to the effect on MI predicted by theoretical works, the secret-sharing counter-measure also has an effect on the PI through an increasing estimation error, making the MI estimation poorer.

Figure 5.4b presents the results of Experiment 5 on shuffling. It is recalled that contrary to Experiment 4 where P.o.Is where extracted, here 250-dimensional traces have been processed through a DNN. The gap in Figure 5.3c between each curve remains observable on Figure 5.4b when considering experimental traces. However, the PI obtained when the attack target is shuffled among 16 random values seems decreasing starting the 20-th epoch, while the empirical PI (in dotted curves) keeps increasing. This is a sign of *over-fitting*.

Indeed, if the estimation error is high, the optimization algorithm is expected to return at each iteration a better model with respect to the training loss $\mathcal{L}_{\mathcal{S}_p}()$. Since the latter one is different from the cross entropy $\mathcal{L}_{\mathbf{X},Z}()$, an improvement with respect to the training loss may not be an improvement with respect to the cross entropy, or equivalently, with respect to the PI. That is why there is a moment when the loss computed over the validation traces starts increasing whereas the training loss keep decreasing. In other words, the model starts to learn *by heart* to build its prediction on some uninformative features which would not generalize well during the attack phase on unknown traces. The higher the estimation error, the less similar the NLL loss and the cross entropy so the sooner and the more importantly

over-fitting happens. Therefore, the PI reached in the graph is not necessarily optimal: more profiling traces might be required to decrease the estimation error and thereby mitigating the effect of over-fitting.

Altogether, our experiments show that similarly to the approximation and optimization errors discussed in Section 5.4, the estimation error is also negligible relatively to the MI, when considering unprotected scenarios where the profiling set size is reasonably high (*i.e.* $10,000$ traces or above). This therefore leads to a tight estimation of the MI through the maximization of the PI (*i.e.* the minimization of the NLL loss). When considering protected devices, the investigated counter-measures impact the estimation error, and thereby the tightness of the lower bound computed through PI maximization. Nevertheless this can be controlled by increasing the size of the profiling set. More precisely, the *harder* the counter-measure (*i.e.* the higher the sharing order, or the more shuffled bytes), the higher the profiling set size.

Another way to decrease the estimation error would be to decrease the *capacity* of the hypotheses class, *i.e.* its VC-dimension, by decreasing the number of layers or the number of neurons on each layer. Since we have argued in Subsection 5.4.2 that the approximation error was negligible even for a simple architecture, we are quite confident that this would not strongly affect the quality of the MI estimation.

5.5.3 Application on Public Datasets

So far, we have considered our experimental investigations through the view of the Leakage Assessment Problem (*i.e.* Problem 3). However, we remind that the final task an evaluator is given to achieve is the profiled SCA optimization (*i.e.* Problem 2), namely to find $N_a^\star$.

It is recalled that Corollary 1 argued that by solving the Leakage Assessment Problem, one could get an accurate estimation $\widetilde{N_a}\left(\mathcal{A}\left(\mathcal{S}_p\right)\right)$ of the quantity $\frac{f(\beta)}{\mathsf{MI}(Z;\mathbf{X})}$, known to be a lower-bound of the optimal solution of the profiled SCA optimization problem, namely $N_a^\star$.

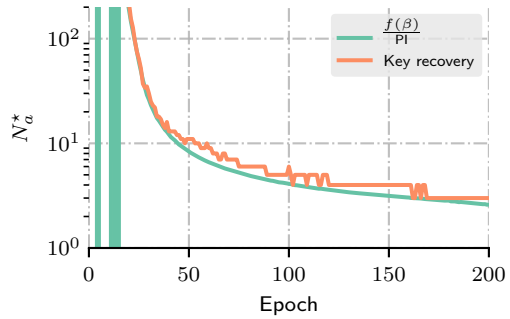
One could wonder whether this inequality still holds for any model, maybe sub-optimal, *i.e.* when estimating the minimal number of queries $N_a(F)$ to the target device for such a model with the quantity $\widetilde{N_a}\left(F\right)$. A formal proof would be a promising further work, though beyond the scope of this thesis.[13] Nevertheless we propose here to empirically verify this hypothesis by training a CNN on all the public datasets presented in Section 3.8 and by implementing the key enumeration in order to evaluate $N_a(F)$.[14] This way, our claim can be tested under several cases covering all the counter-measures considered in this thesis.

Settings. For each training, a VGG-like CNN architecture has been used, as presented in Subsection 4.2.3. More specifically:

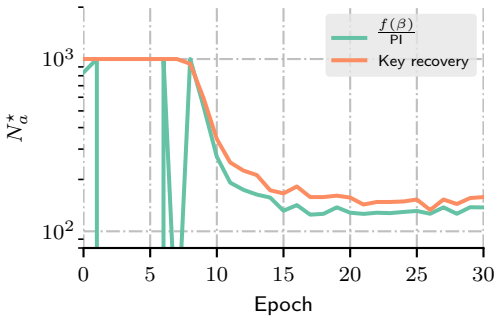
- **ASCAD and AES-HD**: the following parameters have been used: $n_1 = 2$, $n_2 = 7$, the convolutional filters are of length $W = 11$ and the pooling stride is $p = 2$. $K = 10$ filters are in the first layer, and they are doubled at each convolutional layer. The last pooling layer has a stride set so that the output size along the time dimensionality is one. The dense layers contains $1,000$ intermediate neurons. This architecture is based on our works presented at COSADE'19 and is more thoroughly discussed in Subsection 7.5.3

- **AES-RD**: the same VGG-like architecture as the one presented by Kim *et al.* [KPH+19] has been used. More specifically, n_2 is set to 9 so that there is enough pooling layers to get feature maps on the last convolutional layer whose width equals one. Besides, $n_1 = 0$, *i.e.* there is no intermediate dense layer, except softmax.

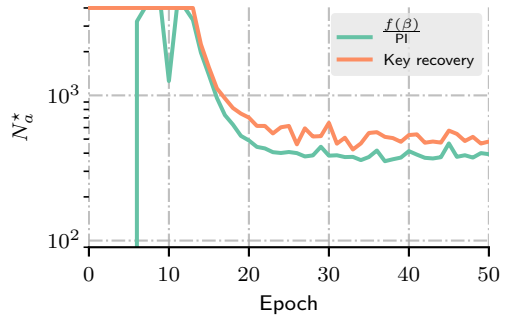
[13] We discuss in Section 5.6 the recent improvements proposed following this work.
[14] See the method described in Subsection 3.2.3 for the practical estimation of $N_a(F)$.



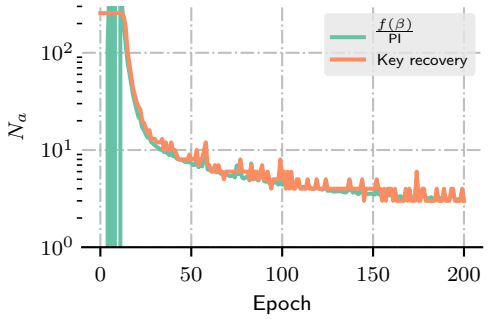
(a) AES-RD dataset.



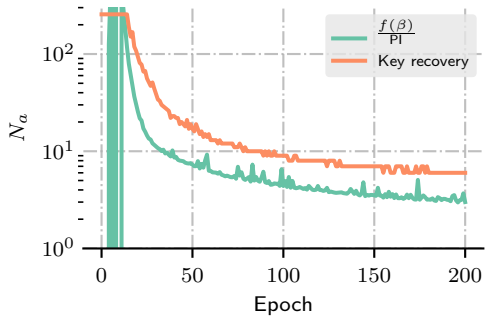
(b) ASCAD dataset



(c) AES-HD dataset



(d) Polymorphism dataset (mbedTLS)



(e) Polymorphism dataset (AES 8-bit)

Figure 5.5: Comparison between the estimation of $N_a(F(\cdot, \theta_t))$ through the lower bound (orange lines) and through a key enumeration (green lines).

- **Polymorphism**: the precise architecture used for the experiments is the one used in our works presented at ESORICS'20 and a whole dedicated discussion is proposed in Subsection 6.2.4.

The training has been run on 200 epochs on the AES-RD and the Polymorphism datasets, 50 epochs on the AES-HD dataset, and stopped after 30 epochs on the ASCAD dataset – see an explanation in Subsection 7.5.3. After each epoch t the optimization algorithm – Adam here – returns an update of the learning parameter vector denoted by θ_t. We can therefore estimate the efficiency of the – partially optimized – model $F(\cdot, \theta_t)$, *i.e.* $N_a\left(F(\cdot, \theta_t)\right)$ thanks to a key enumeration, according to the procedure detailed in Subsection 3.2.3.

Results. The results are given in Figure 5.5. On each graph, $\widetilde{N_a}\left(F(\cdot, \theta_t)\right)$ is denoted in green, whereas the enumeration key estimation $N_a(F(\cdot, \theta_t))$ is denoted by the orange curve. Each curve has been clipped in order to be in $[0, 10^3]$, except for the AES-HD dataset where the high threshold is 4.10^3 since the literature presumed a higher $N_a^\star$ [KPH+19, ZBHV19].

On Figure 5.5a, we can remark that the first epochs of the profiling of the AES-RD dataset

show a chaotic behavior. This is explained by the fact that the NLL loss is initially close to $n = 8$ bits, or in other words, the PI is close to zero, leading to unstable estimations of $\widetilde{N_a}\left(F(\cdot, \theta_t)\right)$. Once the model has started extracting some information, *i.e.* after approximately epochs, the PI starts to be significantly higher than 0 and the unstability vanishes. We can then observe that $\widetilde{N_a}\left(F(\cdot, \theta_t)\right)$ is always lower than the key enumeration estimation $N_a(F(\cdot, \theta_t))$ as expected, while remaining tight through the epochs: the average relative error, computed starting the $t = 20$-th epoch is of 16%. The final model is able to recover the secret key in 3 traces, and has a PI of 2.95 bits/trace.[15]

Likewise, for the ASCAD dataset, the results are presented in Figure 5.5b. We can observe the same unstability at the beginning of the training, though the quantity $\widetilde{N_a}\left(F(\cdot, \theta_t)\right)$ remains lower than the estimation through the enumeration key afterwards, while staying quite tight. The average relative error is also here of 16%, and the final PI is 0.065 bit/trace.

In addition, for the AES-HD, the results are presented in Figure 5.5c. Similarly to the two preceding experiments, a tight estimation is obtained, since the relative error is 18%, while the final PI is 0.020 bit/trace.

Finally, the outcomes on the two Polymorphism datasets are presented. Figure 5.5d deals with the mbedTLS implementation and depicts two superposed curves, hence a tight estimation of $N_a(F(\cdot, \theta_t))$. Similarly, Figure 5.5e deals with the AES 8-bit implementation. Although the estimation seems looser, the relative error remains of 15%.

As a consequence, all those experiments tend to confirm that the quantities $\widetilde{N_a}\left(F(\cdot, \theta_t)\right)$ and $N_a(F(\cdot, \theta_t))$ are effectively related, at least for a threshold $\beta = 90\%$. This is of great interest in the evaluation of the security of a device, since this not only empirically shows the relevance of minimizing the NLL loss, but this also provides a relevant tool to predict the required number of queries to succeed the key recovery, or at least to give a lower-bound to such a number, which is still useful since we look for a worst case scenario in a SCA evaluation.

5.6 Conclusion

In this chapter, we have given some theoretical and experimental reasons why the deep learning paradigm is suitable for evaluating implementations against SCA from a worst-case scenario point of view, regardless the nature of the counter-measures.

Contrary to what was commonly believed until the works of Picek *et al.* [PHJ+18], the supervised classification approach is not theoretically grounded generally speaking as discussed in Chapter 4. Yet, deep learning based attacks still worked. The reason is that in the specific case where the NLL is used as a surrogate loss function, it turns out that the latter one is actually consistent with maximizing the PI, solving the so-called Leakage Assessment Problem. Since the latter problem was argued to be sound with the profiled SCA optimization problem, we conclude that the choice of the NLL as a surrogate loss function is sound from an evaluation point of view, in the sense that it enables to accurately estimate a lower bound of the minimal number of queries required by an attacker provided with an optimal leakage model in order to successfully recover the secret key.

Simulations and experiments verified that the PI maximization via NLL minimization was an efficient method in order to estimate the MI in several configurations, *i.e.* on different architectures and with different types of counter-measures, including higher order secret-sharing, shuffling or de-synchronization through random delays.

This leads to the takeaway messages of this chapter: the minimization of the NLL loss via a neural network model enables to give relevant estimations of the mutual information between a sensitive variable and the corresponding side-channel traces, thereby quantitatively measuring the impact of counter-measures (and their implementations) so that an

[15] Those results coincide with the ones reported by Kim *et al.* [KPH+19].

evaluator can precisely assess whether the latter one stays sound or not.

A possible track of work following the study presented in this chapter could investigate how DL could also be used to estimate the Hypothetical Information (HI), another information theoretic notion extending the MI and the PI. Bronchain *et al.* considered this metric as well in their paper at CRYPTO'19 and showed that it is an upper-bound of the MI whereas the PI is a lower bound. It would be interesting to know whether there is a way to *minimize* the HI of an approximate leakage model, in order to get an insightful confidence interval of the MI, along with the PI. Unfortunately, the computation of HI would rely on generative models, beyond the scope of this thesis. Yet, investigating whether there are sound generative DL models in an SCA context could be promising.

Epilogue. Since the release of our paper at CHES 2020, two recent works have addressed the problem of the choice of the loss function, following our line of works.

Zhang *et al.* have proposed a slight variant of the NLL [ZZN+20]. According to the authors, the latter loss function would have a major drawback when dealing with datasets for which the observed values of the sensitive variable are not uniformly distributed, *e.g.*, if one targets $Z' = \mathrm{hw}(Z)$ instead of Z. If the precise distribution $\Pr(Z')$ is unknown, then so is $\mathsf{H}(Z')$, which means that one cannot compute the PI. Nevertheless, one can still maximize it, since the unknown term does not depend on the considered model F with respect to which the optimization is done. To circumvent this problem, the authors propose the Cross Entropy Ratio (CER), which is the ratio between the NLL computed for the right key hypothesis, and the average of the NLLs computed when assuming any other wrong key hypotheses. They show that an attack is effective i.f.f. this metric is below 1. Moreover, they claim that the lower the metric, the more efficient the attack, which is empirically verified on several public datasets. Yet, a formal proof of this trend still remains to be established.

Zaid *et al.* have recently embraced another approach when tackling the issue of the loss function, leading to proposing the so-called *ranking* loss [ZBD+20]. Contrary to the NLL coming from the supervised classification task, the authors here take inspiration from another learning task, namely *learning to rank*. By translating this task into the profiled SCA optimization framework, they show the soundness of their approach to maximize the SR.

Beyond being useful for estimating the SCA efficiency metric, the computation of the MI could be a goal as itself for the evaluator [BHM+19]. Hence, Cristiani *et al.* recently extended a technique called *Neural Estimation of the Mutual Information*, originally introduced by Belghazi *et al.* [BRB+18], in the aim to derive the best way to to estimate the MI between SCA traces and a sensitive intermediate computation [CLM20].

Chapter 6

DL-based SCA on Large-Scale Traces: A Case Study on a Polymorphic AES

This chapter is inspired from the paper published at ESORICS 2020, in collaboration with Cécile Dumas, Laurent Maingault, Marie-Angela Cornélie and Eleonora Cagli from the LETI - ITSEF, and Damien Couroussé and Nicolas Belleville from the LIST - DSIN [MBC+20], and with the support of Pierre-Alain Moëllic.

Contents

6.1 Introduction

In Section 3.7, when presenting the different counter-measures that a developer can use to protect an implementation against SCA, we essentially focused on two criteria:

1. the resulting protected implementation must still remain acceptable for the final user, in particular in terms of runtime and memory;

2. it must guarantee, formally or empirically, the security level requested by the final user or the developer.

Finding the counter-measure meeting those two constraints, often antagonist, is somehow the holy-grail quest of developers willing to prevent SCA on their devices.

We have seen for example that group-based secret-sharing schemes are able to meet the second condition, to the detriment of the first one. However, we have not discussed yet in this thesis a third condition, namely how a developer can easily turn an unprotected code into a protected one. Indeed, although encryption standards such as AES are designed to be easily implemented, their specifications do not take into account the development cost of versions protected against SCA. In practice, this third constraint turned out to be critical, because it required so far a careful hand-made protection of every possible sensitive intermediate state of the machine running the primitive at a low level – *i.e.* assembly or hardware.

Some recent works propose ways to automatize the secret-sharing of sensitive intermediate computations [BDM+20, BCH+20a], with the hope to decrease the developer's hand-made design. The interest of this approach is thereby to combine this automated generation with a provable security assessment, made possible by the recent works of security proofs on secret-sharing – see Subsection 3.7.1. Nevertheless, the theoretical models proposed by the scientific community on which these tools rely do not always correspond to the physical reality of the devices designed by the industry. As a consequence, a hand-made verification of the automatically generated implementation might not always be excluded, thereby mitigating the interest of automatically generating secret-sharing.

We have presented the code polymorphism counter-measure in Subsection 3.7.2, enabling to automatize the code randomization in a pervasive way at the scope of assembly instructions, in order to implement leakage hiding at a low cost in development. Moreover, we have seen that hiding is a lighter counter-measure in terms of runtime and memory complexity: the performance overhead for hiding is linear with the amount of shuffled operations or the number of dummy operations, while it is quadratic with the sharing order for secret-sharing. Therefore, one may wonder whether code polymorphism could be a good candidate as a way to meet the three constraints of a practically sound counter-measure evocated so far. Belleville *et al.* brought some evidences about the security of code polymorphism in a paper at TACO'19 [BCHC18]. The latter study follows a series of works [ABP12, ABPS15, CBR+16] proposing a way to efficiently (in the sense of the first constraint) implement code polymorphism. They propose a specific configuration of code transformations for which they emphasize empirical evidences of strong security level against the vast majority of the attacks. This particularly concerns the ones requiring the P.o.Is of the raw traces to be aligned with each other, *e.g.* the CPA or the GT: the Test Vector Leakage Assessment (TVLA) method based on T-tests (see Section 3.6) applied on several implementations of cryptographic primitives show that their protection prevents the target device from revealing its leakage, and a CPA mounted against a protected implementation requires about several million traces whereas the same attack against the same unprotected target only required a few hundred traces to recover the secret key.

6.1.1 Problem Addressed in this Chapter

Yet, although very promising, these results cannot draw an exhaustive guarantee concerning the security level against SCA, since other realistic scenarios involving more elaborated attacks have not been investigated.

Indeed, on the one hand the SCA literature proposes other ways to outperform vertical attacks when facing hiding counter-measures. *Re-synchronization* techniques might annihilate the misalignment effect occurred by code polymorphism, since it is successfully applied on hardware devices prone to jitter [NHI+07, vWWB11, DRS+12]. Likewise, CNNs can circumvent some software and hardware de-synchronization counter-measures, in a sense similar to code polymorphism [CDP17, KPH+19]. It is therefore of great interest to use those techniques to assess the security provided by some code polymorphism configurations against more elaborated attackers.

On the other hand, until now the literature has only demonstrated the relevance of CNN attacks on restricted traces whose size did not exceed $5,000$ samples [CDP17, BPS+19, KPH+19, Tim19, ZBHV19], which is small, *e.g.*, regarding the size of the raw traces in the public datasets of software AES implementations used in those papers [NSGD12, BPS+19, CK09]. This requires to focus the trace acquisition to a tight window where the attacker is confident that the relevant leakage occurs. Unfortunately, this is not possible in presence of code polymorphism since it applies hiding in a systematic and pervasive way in the implementation. Likewise, other dimensionality reduction techniques like dedicated variants of PCA [SA08] might be considered prior to the use of CNNs. However, they do not theoretically provide any guarantee that relevant features will be extracted, especially for data prone to misalignment. As a consequence, attacking a polymorphic implementation necessarily requires to deal with large-scale traces. This generally spans serious issues in machine learning tasks known under the name of *curse of dimensionality* [SSBD14]. That is why it currently remains an open question whether CNN attacks can scale on larger traces, or whether it represents a technical issue that some configurations of code polymorphism might benefit against these attacks. Hence, both problems, namely evaluating code polymorphism and addressing large-scale traces SCA, are closely intertwined.

6.1.2 Outline of the Chapter

In the remaining of this chapter, we tackle the two problems presented so far by extending the security evaluation provided by Belleville *et al.* [BCHC18]. The evaluation aims to assess the security of the highest code polymorphism configuration they used, on same implementations, against stronger attackers.

Our evaluation considers a wide spectrum of threat models, ranging from automated attacks affordable by a layman attacker, to state-of-the-art techniques. The whole evaluation setup is detailed in Section 6.2. In particular, we propose to adapt the architectures used in the literature of CNN attacks, in order to handle the technical challenge of large scale traces. This is presented in Subsection 6.2.4.

Finally, the outcomes of our evaluations are presented in Section 6.3, and will serve as a ground for discussions proposed in Section 6.4.

6.2 Evaluation Methodology

6.2.1 Target Device

The evaluation has been conducted on the polymorphism dataset presented in Subsection 3.8.5. We remarked then that the SNRs computed on both mbedTLS and AES 8-bit implementations protected with code polymorphism did not reveal any leakage, although

a re-alignment process was suggested to annihilate the misalignment effect of the countermeasure – see Subsection 3.8.5. In Subsection 6.2.3, we propose a way to re-align the traces on the mbedTLS dataset.

6.2.2 Threat Models

We propose several threat models that we distinguish according to two resources the attacker may access, that we precise hereafter.

First, the attacker may get or not an open sample in order to conduct a profiled attack scenario, as considered in this thesis. We recall that this is a necessary condition in order to evaluate the worst-case scenario from a developer's point of view [HRG14]. Though often seen as strong, this assumption can be considered as realistic in our context: here both the chip and the source codes used for the evaluation are publicly available.

Second, the attacker may eventually incorporate human expertise to improve attacks initially fully automatized. Here, this will concern either the preliminary task of trace re-alignment (see the procedure described in Subsection 6.2.3), or the capacity to properly design a CNN architecture for the deep learning based SCA (see Subsection 6.2.4).

Hence the following attack scenarios:

- $\mathcal{A}_{\textbf{auto}}$**:** considers a fully automatized attack (*i.e.* without any human expertise), without access to any open sample.
- $\mathcal{A}_{\textbf{CPA}}$**:** considers an attack without access to an open sample, but with human expertise to re-align the traces – see Subsection 6.2.3. It results in doing a CPA targeting the output of the SubBytes operation, assuming the Hamming weight leakage model – see Section 3.5.
- $\mathcal{A}_{\textbf{gT}}$**:** considers the same attack as $\mathcal{A}_{\text{CPA}}$, *i.e.* targeting re-aligned traces, with an access to an open sample in addition. The profiling is done thanks to Gaussian Templates (GTs) with pooled covariance matrices – see Section 3.4. No dimensionality reduction technique is used here, beside the implicit reduction done through the re-alignment detailed in Section 3.8.5.
- $\mathcal{A}_{\textbf{CNN}}$**:** considers an attack with access to an open sample and human expertise to build a CNN for the profiled attack. This attack scenario is considered the most effective against de-synchronized traces with first-order leakage [CDP17, KPH+19, BPS+19]. Therefore, we do not assume $\mathcal{A}_{\text{CNN}}$ to need an access to re-aligned traces. In addition, no preliminary dimensionality reduction is done here.

As already mentioned in this section, the SNR of the raw traces, computed on $100,000$ traces, did not emphasize any peak. Thanks to the works of Mangard *et al.* [Man04, MOP07] and Oswald *et al.* [MOBW13] linking the efficiency of a CPA to the amplitude of SNR, we can already draw the following conclusions: $\mathcal{A}_{\text{auto}}$ will not succeed with less than $100,000$ queries.

6.2.3 Re-alignment on the mbedTLS Dataset

To conduct $\mathcal{A}_{\text{CPA}}$ and $\mathcal{A}_{\text{gT}}$, we proceeded a re-alignment on the traces from the mbedTLS dataset that we describe hereafter. For each $80,000$-dimensional trace, the clock cycles corresponding to the region between two EM peaks are identified according to a thresholding on falling edges. Since the EM peak pattern delimiting two identified clock cycles may be spread over a different number of samples from one pattern to another, we only keep the minimum and maximum points. Likewise, since the number of identified clock cycles can also differ from one trace to another, the extracted samples are eventually zero-padded to be of dimension $D = 50$.

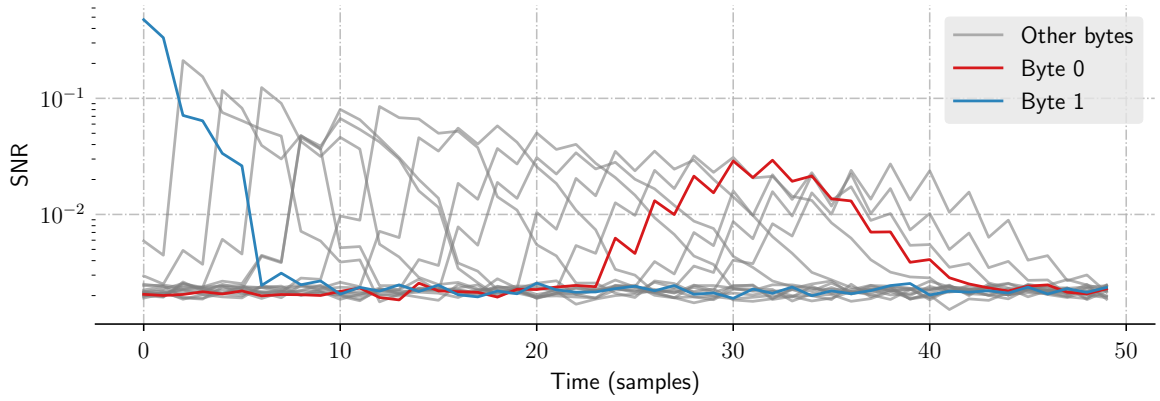


Figure 6.1: The 16 SNR of the acquired traces from the mbedTLS implementation, one for each targeted byte, after re-aligned pattern extraction.

Based on this re-alignment, a new SNR is computed on Figure 6.1. Contrary to the SNR computed on raw traces in Figure 3.13, some leakages are clearly distinguishable though the amplitude of the SNR peaks vary from 5.10^{-2} to 5.10^{-1}, according to the targeted byte. Thanks to this re-alignment, and according to the previous discussion between the CPA efficiency and the amplitude of SNR, we can already bet that $\mathcal{A}_{\mathsf{CPA}}$, $\mathcal{A}_{\mathsf{gT}}$ and $\mathcal{A}_{\mathsf{CNN}}$ are likely to succeed within $100,000$ queries.

Remark 7. *The re-alignment technique used in this work is based on the detection of leakage instants by thresholding. Other re-alignment techniques [NHI+07, vWWB11, DRS+12] may be used. Therefore, the results of $\mathcal{A}_{\mathsf{CPA}}$ and $\mathcal{A}_{\mathsf{gT}}$ might be improved. However, none of the re-alignment techniques in the literature provides strong theoretical guarantees of optimality, especially regarding the use of code polymorphism.*

6.2.4 CNN-Based Profiling Attacks

As mentioned in Subsection 6.2.2, CNN attacks may require some human expertise to properly set the model architecture. This section is devoted to describing the whole settings used to train the CNNs used in the attack scenario $\mathcal{A}_{\mathsf{CNN}}$, in order to tackle the challenge of large-scale traces. We quickly review the guidelines in the SCA literature, and argue why they are not suited to our traces. We then present the used architecture, and we detail the training parameters.

The Literature Guidelines. Although numerous papers have proposed CNN architectures [MPP16, CDP17, BPS+19], the state-of-the-art CNNs are currently given by Kim *et al.* [KPH+19] and Zaid *et al.* [ZBHV19]. Their common point is to rely on the VGG architecture given by Equation 4.19 that we recall hereafter:

$$F = \mathrm{s} \circ \lambda_{|\mathcal{Z}|} \circ [\sigma \circ \lambda_C]^{n_1} \circ [\delta_p \circ \sigma \circ \mu \circ \gamma_{W,K}]^{n_2} \circ \mu \ , \tag{6.1}$$

where $\gamma_{W,K}$ denotes a convolutional layer made of K filters of size W, μ denotes a batch-normalization layer, σ denotes the ReLU activation function, δ_p denotes an average pooling layer of size p, λ denotes a dense layer, and s denotes the softmax layer. Furthermore, the composition $[\delta_p \circ \sigma \circ \mu \circ \gamma_{W,K}]$ is denoted as a convolutional *block*. Likewise, $[\sigma \circ \lambda]$ denotes a dense block. We note n_1 (resp. n_2) the number of dense blocks (resp. convolutional blocks).

An intuitive approach would be to directly set the parameters or our architecture to the ones used by Kim *et al.* or by Zaid *et al.* Unfortunately, we argue in both case that such a transposition is not possible.

Kim *et al.* propose particular guidelines to set the architecture [KPH+19]. They recommend to fix the filter size $W = 3$, and $p = 2$, *i.e.*, the minimal possible values, and to set n_2 so that the time dimensionality at the output of the last block is reduced to one. Since each pooling divides the time dimensionality by p, $n_2 \leq \log_p(D)$.[1] Meanwhile, they double the number of filters for each new convolutional block compared to the previous one, without exceeding 256. Unfortunately, using these guidelines is likely to increase n_2 from 10 in Kim *et al.*'s work to at least 17 in our context. First, as explained by He *et al.* [HZRS16], stacking such a number of layers is likely to make the numerical optimization with SGD or its variants harder. That is why an alternative architecture called Resnets has been introduced [HZRS16], and starts to be used in SCA as well [ZS19, GJS20]. This possibility will be discussed in Section 6.4. Second, due to the doubling number of filters at each new block, by transposing the Kim *et al.*'s guidelines, the number of learning parameters would be around 1.86 millions for the mbedTLS dataset and around 2.06 millionsfor the AES 8-bit one. This represents approximately 10 folds more parameters to learn than the number of traces acquired in the profiling set. In such a configuration, the estimation error is likely to be high since it – roughly – depends on the number of learning parameters – see Subsection 4.2.3.

In order to improve the Kim *et al.*'s architecture, Zaid *et al.* [ZBHV19] proposed thumb rules to set the filter size in the convolutional and the pooling layers depending on the maximum temporal amplitude of the de-synchronization. However, it assumes to know the maximum amplitude of the de-synchronization, which is not possible here since it is hard to guess how many times those transformations are applied in the polymorphic instance.

Our Architecture. The drawbacks of Kim *et al.*'s and Zaid *et al.*'s guidelines in our particular context justify why we do not directly use them. Instead, we propose to take the Kim *et al.*'s architecture as a baseline, on which we modify some of the parameters as follows.

First, we set the number of filters in this first block to $K_0 = 10$, we decrease the maximal number of filters from 256 to $K_{\max} = 100$, and we slightly change the way the number of filters is computed in the intermediate convolutional layers, according to Table 6.1. Likewise, we remove the dense block (*i.e.* $n_1 = 0$). This limits the number of learning parameters, which mitigates the issue of the estimation error.

Second, we increase the pooling size to $p = 5$. This mechanically allows to decrease the minimal number of convolutional blocks n_2 from 17 to 6 for the mbedTLS traces and to 7 for the AES 8-bit ones. To be sure that this growth in p does not imply any loss of information in the pooling layers, we set the filter size to $W = 2p + 1 = 11$. Eventually, since with such numbers of convolutional blocks the output dimensionality is not equal to one yet, we set the last pooling layer to be *global*, *i.e.* its stride is set so that the output dimension is collapsed to 1, without adding any extra learning parameter [ZKL+16]. Such an architecture would represent $177,500$ learning parameters, when targeting the mbedTLS implementation and $287,500$ for the AES 8-bit. In other words, the number of learning parameters is lowered by one order of magnitude compared to what would have been required by the Kim *et al.*'s guidelines in our context.[2]

First Convolutional Block. To decrease further the number of learning parameters, one may even tweak the first convolutional block, by exploiting the properties of the input signal. Figure 6.2 sketches an EM trace chunk of about one clock cycle. We make the underlying assumption that the relevant information to extract from the traces is contained in the patterns occurring around each clock, mostly due to the change of states in the memory reg-

[1] We recall that D denotes the dimensionality of the traces, *i.e.* $D = 80,000$ for the mbedTLS implementation ($D = 50$ for the re-aligned data) and $D = 160,000$ for the AES 8-bit.

[2] We would like to point out that the Kim *et al.*'s guidelines were specifically designed to their own context. The authors did not claim to provide general guidelines working in any situation. As such, our conclusions do not question the relevance of Kim *et al.*'s results in their work at CHES 2019 [KPH+19].

isters storing the sensitive variables [MOP07].[3] Moreover, no additional relevant pattern is assumed to be contained in the trace until the next clock cycle, appearing $T = 50$ samples later.[4] By carefully setting W_0 to the size of the EM patterns, and p_0 such that $W_0 \leq p_0 \leq T$, we optimally extract the relevant information from the patterns while avoiding the entanglement between two of them. In our experiments, we arbitrarily set $p_0 = 25$. This first tweaked convolutional block has then the same receptive field than one would have with two normal blocks of parameters ($W = 11, p = 5$). Therefore, we spare one block (*i.e.* the last one), which decreases the number of learning parameters: our architecture now represents $84,380$ (resp. $177,500$) learning parameters for the model attacking the mbedTLS implementation (resp. AES 8-bit). Table 6.1 provides a synthesis of the description of our architecture, along with a comparison with the Kim *et al.* and Zaid *et al.*'s works.

Table 6.1: Our architecture and the recommendations from the literature. In the Zaid *et al.*'s methodology, T denotes the maximum assumed amount of random shift in the traces, and I denotes the assumed number of leakage temporal points in the traces.

	Kim *et al.* [KPH+19]	Our archi.	Zaid *et al.* [ZBHV19]
n_1	1	0	2
n_2	$\log_p(D)$	$\log_p(D)$	3
p	2	$5 (p_0 = 25)$	$2, \frac{T}{2}, \frac{D}{I}$
W	3	$11 (W_0 = 10)$	$1, \frac{T}{2}, \frac{D}{I}$
K_n	$\min(K_0 \times 2^n, K_{\max})$	$10,\ 20,\ 40,\ 40,\ 80,\ 100 (100)$	$K_0 \times 2^n$
K_0	8	10	$8,\ 32$
$K_{\max}$	256	100	-

Training Settings The source code is implemented in Python with the same machine as presented in Section 4.3. For each experiment, the whole dataset is split into a training and a test subsets, containing respectively $95,000$ and $5,000$ traces. The latter ones are used to simulate a key recovery based on the scores attributed to each hypothetical value of the sensitive target variable by the trained model. Moreover, the SH_{100} data augmentation method is applied to the training traces, following the description given in [CDP17]: each trace is randomly shifted of maximum 100 points, which represents the length of 2 clock cycles.[5]

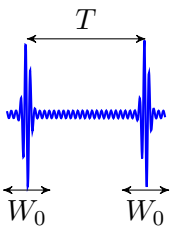


Figure 6.2: Two EM patterns separated by one clock cycle.

The training is done by minimizing the NLL loss, with the Adam optimizer – see Subsection 2.5.2 – during 200 epochs[6] which approximately represents a 16-hour long training for each targeted byte. The learning rate of the optimizer is always set to 10^{-5}.

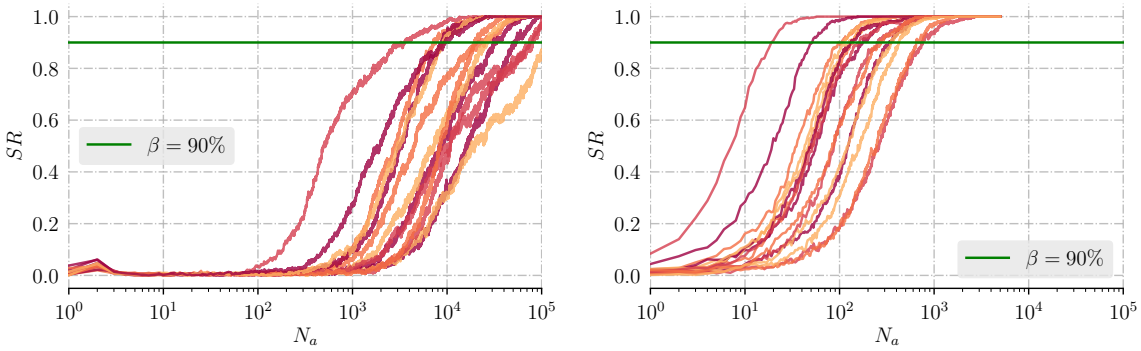
Based on a – eventually partially – trained model, we compute the efficiency of the attack, as defined by Equation 3.4.

[3] This assumption has somehow already been used for the pattern extraction re-alignment in Section 3.8.5.

[4] We recall that despite the effect of code polymorphism, and in absence of hardware jitter, the duration of the clock period, in terms of samples, is roughly constant.

[5] This data augmentation is not applied on the attack traces.

[6] One epoch corresponds to the number of steps necessary to pass the whole training data-set through the optimization algorithm once.



(a) $\mathcal{A}_{\mathsf{CPA}}$ on mbedTLS

(b) $\mathcal{A}_{\mathsf{gT}}$ on mbedTLS

Figure 6.3: Success Rate with respect to the number of attack traces. Attacks on mbedTLS requiring the alignments of the P.o.Is. The different colors denote the different targeted bytes.

6.3 Results

Once the different threat models and their corresponding parameterization have been introduced in Section 6.2, we can now present the results of each attack, also summarized in Table 6.2.

As argued in Section 3.8.5, we can directly conclude from the SNRs given by Figure 3.13 (bottom) and Figure 3.14 (bottom) that the fully automatized attack $\mathcal{A}_{\mathsf{auto}}$ cannot succeed within the maximum amount of collected traces, *i.e.*, $N_a\left(\mathcal{A}_{\mathsf{auto}}\right) > 10^5$, for both implementations.

Figure 6.3a depicts the performances of $\mathcal{A}_{\mathsf{CPA}}$ against the mbedTLS implementation, on each state byte at the output of the SubBytes operation. It can be seen that the the SR intersects the green threshold at $N_a(\mathcal{A}_{\mathsf{CPA}}) \approx 10^3$ for the byte 1, and at $N_a(\mathcal{A}_{\mathsf{CPA}}) \in [\![10^4, 10^5]\!]$ for the other bytes.[7] Those results are in line with the rule of thumb stating that the higher the SNR on Figure 6.1, the faster the success rate convergence towards 1 on Figure 6.3a [MOP07]. Since we argued in Section 3.8.5 that the proposed re-alignment technique was not relevant on the AES 8-bit traces, we conclude that $\mathcal{A}_{\mathsf{CPA}}$ would require more that 10^5 queries on those traces.[8]

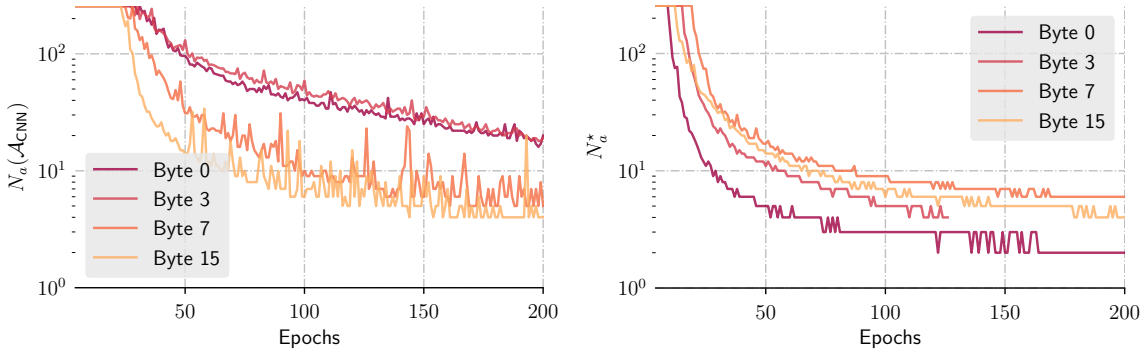
Figure 6.3b summarizes the outcomes of the attack $\mathcal{A}_{\mathsf{gT}}$. One can remark that the SR curves intersect the green threshold at $N_a(\mathcal{A}_{\mathsf{CPA}}) \leq 1,000$, *i.e.* the attack is successful for all the target bytes within $1,000$ queries. This represents an improvement by one order of magnitude as compared to the scenario $\mathcal{A}_{\mathsf{CPA}}$. In other words, the access to an open sample provides a substantial advantage to $\mathcal{A}_{\mathsf{gT}}$ compared to $\mathcal{A}_{\mathsf{CPA}}$. As for the latter one, and for the same reasons, we conclude that the attack $\mathcal{A}_{\mathsf{gT}}$ would fail with 10^5 traces of the AES 8-bit implementation.

Figure 6.4 presents the results of the CNN attack $\mathcal{A}_{\mathsf{CNN}}$. In particular, Figure 6.4a shows that training the CNN for 200 epochs allows to recover a secret byte in less than 20 traces in the case of the mbedTLS implementation. Likewise, Figure 6.4b shows that a successful attack can be done within 10 traces on the AES 8-bit implementation. Moreover, both curves in Figure 6.4 show that the latter observations can be generalized for each byte targeted in the attack $\mathcal{A}_{\mathsf{CNN}}$.[9] Finally, one can remark that training the CNNs during a lower number of epochs (*e.g.*, 100 for mbedTLS, 50 for AES 8-bit), still leads to the same order of magnitude for $N_a^\star$.

[7] Targeting the output of the AddRoundKey operation instead of the output of the SubBytes operation has also been considered without giving better results.

[8] Section 6.4 discusses the possibility of relevant re-alignment techniques for the AES 8-bit implementation.

[9] Additional experiments realized on a setup close to $\mathcal{A}_{\mathsf{CNN}}$ confirm that the results can be generalized to any of the 16 state bytes.



(a) mbedTLS (b) AES 8-bit

Figure 6.4: Evolution of $N_a^\star$ with respect to the number of training epochs during the open sample profiling by the CNN (attack $\mathcal{A}_{\text{CNN}}$).

Table 6.2: Minimal number $N_a^\star$ of required queries to recover the target key bytes.

Scenario	mbedTLS	AES 8-bit
$\mathcal{A}_{\text{auto}}$	$> 10^5$	$> 10^5$
$\mathcal{A}_{\text{CPA}}$	$3.10^3 - 10^5$	$> 10^5$
$\mathcal{A}_{\text{gT}}$	$20 - 10^3$	$> 10^5$
$\mathcal{A}_{\text{CNN}}$	$< \mathbf{20}$	$< \mathbf{10}$

Based on these observations, one can make the following interpretations. First, the attack $\mathcal{A}_{\text{CNN}}$ leads to the best attack among the tested ones, by one or several orders of magnitude. Second, such attacks are reliable, since the results do not differ from one implementation to another, and from one targeted byte to another. Third, the training time for CNNs, currently set to roughly 16 hours for each byte (see Subsection 6.2.4), can be halved or even quartered without requiring too much more queries to succeed the attack. Since in this scenario the profiling phase is here the *critical* (*i.e.* the longest) task, it might be interesting to find a trade-off between the training time and the resulting $N_a(\mathcal{A}_{\text{CNN}})$, depending on the attacker's abilities.

To synthesize, our results are summarized in Table 6.2. In a nutshell, they show that attacks requiring the alignments of the P.o.Is fail due to code polymorphism, but that more elaborated scenarios lead to successful attacks. Compared to the ones conducted by Belleville *et al.* [BCHC18], and depending on the different attack powers considered so far, the number of required queries is lowered by up to several orders of magnitude. In a worst case scenario, our trained CNNs are able to recover every secret key byte in less than 20 traces of dimensionality $160,000$, whereas until now the literature has only demonstrated the relevance of CNN attacks on restricted traces whose size did not exceed $5,000$ samples, *i.e.*, 32 times lower.

6.4 Discussion

So far, Section 6.3 has presented and summarized the results of the attacks, depending on the threat models defined in Subsection 6.2.2, against two implementations of a cryptographic primitive, protected by a given configuration of code polymorphism.

This section proposes to discuss these results, the underlying assumptions behind the attacks, and eventually their consequences.

Choice of our CNN Architecture.

The small amount of queries to succeed the attack $\mathcal{A}_{\mathsf{CNN}}$, conducted on both implementations, shows the relevance of our choice of CNN architecture. This illustrates that an end-to-end attack with CNNs is possible when targeting large scale traces, without necessarily requiring very deep architectures.

We emphasize that there may be other choices of parameters for the convolutional architecture giving relevant results as well, if not better. Yet, we do not find necessary to further investigate this way here. The obtained minimal number of queries $N_a^\star$ was low enough so that any improvement in the CNN performances is not likely to change our interpretations of the vulnerability of the targets against $\mathcal{A}_{\mathsf{CNN}}$.

In particular, the advantage of Resnets [HZRS16] broadly used in image recognition typically relies on the necessity to use deep convolutional architectures in this field [SZ15], as recalled in Remark 5. By the way, promising results have been obtained over the past few months with Resnets in SCA [ZS19, GJS20]. However, following the discussion of Figure 4.2, we empirically verified here that we can take advantage of the distinctive features of side-channel traces to constrain the depth of our model and avoid typical issues related to deep architectures – *i.e.* vanishing gradient – that Resnets are supposed to circumvent [HZRS16].

On the Security of Code Polymorphism

Our study exhibits that the attacks $\mathcal{A}_{\mathsf{auto}}$ and $\mathcal{A}_{\mathsf{CPA}}$ are such that respectively $N_a(\mathcal{A}_{\mathsf{auto}})$ and $N_a(\mathcal{A}_{\mathsf{CPA}})$ are high enough to enable a key refreshing period reasonably high, without compromising the confidentiality of the key. Unfortunately, this does not hold in presence of a stronger attacker having access to an open sample, as emphasized by attacks $\mathcal{A}_{\mathsf{gT}}$ and $\mathcal{A}_{\mathsf{CNN}}$, where a secret key can be recovered within the typical duration of a session key. This may be critical at first sight since massive IoT applications often rely on Commercially available Of The Shelf (COTS) devices, which implies that open samples may be easily accessible to any adversary. Thus, this study claims that though code polymorphism is a promising tool to increase the hardness of SCA against embedded devices, a sound polymorphic configuration, eventually coupled with other counter-measures, is yet to be found, in order to protect against state-of-the-art SCA. Nevertheless, the toolchain used by Belleville *et al.* [BCHC18] allows to explore many configurations beside the one considered here, the exploration of the securing capabilities of the toolchain is then beyond the scope of this thesis, and left as an open question for further works.

More generally, this issue can be viewed from the perspective of the problem discussed by Bronchain *et al.* about the difficulty to prevent side-channel attacks in COTS devices, even with sophisticated counter-measures [BS20]. First, our experimental target is intrinsically highly vulnerable to SCA. Second, the use of software implementations of cryptographic primitives offers a large attack surface, which remains highly difficult to protect especially with a hiding counter-measure alone. This underlines the fact that a component may need to use hiding in combination with other counter-measures, *e.g.*, secret-sharing, to be secured against a strong side-channel attacker model.

6.5 Conclusion

So far, this chapter answers two questions likely to help both developers of secure implementations, and evaluators mounting CNN-based SCA.

From a developer's point of view, this chapter has studied the effect of two implementations of a code polymorphism counter-measure against several side channel attack scenarios, covering a wide range of potential attackers. In a nutshell, code polymorphism as an automated tool, is able to provide a strong protection against threat models considering automated and layman attackers, as the evaluated implementations were secure enough

against our first attacker models. Yet, the implementations evaluated are not sound anymore against stronger attacker models. The soundness of software hiding counter-measures, if used alone, remains to be demonstrated against state-of-the-art attacks, for example by using other configurations of the code polymorphism toolchain, or by proposing new code transformations. All in all, our results underline again, if need be, the necessity to combine the hiding and secret-sharing protection principles in a secured implementation.

From an evaluator's point of view, this work illustrates how to leverage CNN architectures to tackle the problem of large-scale side-channel traces, thereby narrowing the gap between SCA literature and concrete evaluations of secure devices where pattern detection and re-alignment are not always possible. The idea lies in slight adaptations of the CNN architectures already used in SCA, eventually by exploiting the signal properties of the SCA traces. Surprisingly, our results emphasize that, though the use of more complex CNN architectures has been shown to be sound to succeed SCA in the literature, it might not be a necessary condition in an SCA context.

Chapter 7

Gradient Visualization for General Characterization in Profiling Attack

This chapter is inspired from the poster presented at CHES 2018 [MDP18] and the paper published at COSADE 2019 in collaboration with Cécile Dumas and Emmanuel Prouff [MDP19a]. This work has benefited from fruitful discussions with Élie Bursztein and Rémi Audebert.

Contents

7.1 Introduction

We have emphasized in Chapter 3 that when considering a profiled attack scenario, an evaluator can do more than just mounting an attack: he can also build a full diagnosis thanks to the leakage characterization techniques that we presented in Subsection 3.6.1. Those techniques, such as the SNR or the T-test, have the particularity to be based on statistical tools, and so are the GTs. In that sense, they form a sound tandem for the evaluator.

However, we explained in Section 3.7 that in presence of counter-measures such as secret-sharing or hiding, the characterization and attack methods based on those statistical tools are no longer efficient (or at least much less). Likewise, other dimensionality reduction techniques like dedicated variants of PCA [CDP15, SA08, EPW10, CK13, CK14] or KDA [CDP16] can be used, without guarantee that relevant components are extracted.

On the other hand, we have explained in Section 4.4 that ML-based attack methods could circumvent (to a given extent) the issues induced by the most widely used counter-measures, hence their recent hype in the SCA community over the past few years. Unfortunately, the emergence of those methods came with a drawback: no auxiliary characterization method were proposed to the evaluator. Indeed, whereas learning algorithms such as CNNs do not require pre-processing and are at least as efficient as the other state-of-the-art profiling attacks, they act as a black-box. From the evaluator's point-of-view, this is not sufficient. On the one hand he wants to make sure that a CNN attack succeeded for good reasons, *i.e.*, that the learned model can generalize its good performance to new data. On the other hand the evaluator may also want to help the developer to localize and understand where the vulnerability comes from in order to remove or at least reduce it. This issue is part of a more general problematic in Deep Learning based systems, namely their *explainability* and *interpretability*. To address it, a theoretical framework has recently been proposed by Montavon *et al.* [MSM18], and several methods have been tested to tackle the issue. In particular, some computer vision research groups have considered the *Sensitivity Analysis* [SVZ14, SDBR15] framework. It consists in studying how the uncertainty in the output of a mathematical model or system (numerical or otherwise) can be apportioned to different sources of uncertainty in its inputs [Wik19]. This term encompasses many methods from simple ones, such as the computation of the derivatives if the model is differentiable or ablation/occlusion techniques [ZF14], to the study of non-trivial variations, *e.g.*, when building adversarial examples [SZS+14, GSS15].

In this chapter, we propose to apply a particular Sensitivity Analysis method called Gradient Visualization (GV) in order to better understand how a CNN can learn to predict the sensitive variable based on the analysis of a single trace. The main claim is that CNN based models succeed in discriminating P.o.Is from non-informative points, and their localization can be deduced by simply looking at the gradient of the loss function with respect to the input traces for a trained model. We theoretically show that this method can be used to localize P.o.Is in the case of a perfect model. The efficiency of the proposed method does not decrease when counter-measures like secret-sharing or misalignment are applied. In addition, the characterization can be made for each trace individually. We verified the efficiency of our proposed method on simulated data and on experimental traces from both the ASCAD database and the two Polymorphism datasets. We empirically show that Gradient Visualization is at least as good as state-of-the-art characterization methods, in presence or not of different counter-measures.

The chapter is organized as follows. In Section 7.2 we start by considering the optimal model of an ideal attacker may get during profiling, and we deduce some properties of its derivatives with respect to the input traces that can be related to the P.o.Is. In Section 7.3 we use these properties on a model estimated with CNNs and we explain how to practically implement the visualization method. A toy example applied on simulated data is proposed for illustration. Sections 7.4 and 7.5 are eventually dedicated to an experimental validation of the effectiveness of our proposal in realistic attacks scenarios.

7.2 Study of an Optimal Model

In this section, we address the evaluator's interpretation problem in the ideal situation when the conditional distribution $F^\star = \Pr(\mathrm{Z} \mid \mathbf{X})$ is known – *i.e.* when the model is perfect – where $\mathbf{X}$ is the random variable denoting the observed trace and Z is the random variable denoting the sensitive intermediate computation carrying information about the secret key. We will show how the study of the derivatives of such a model with respect to each coordinate of an input trace can highlight information about our P.o.Is. To this end, we need two assumptions.

The first one is Assumption 1 and has already been stated in Section 3.6. Informally, it tells that the leaking information is non-uniformly distributed over the trace, *i.e.*, only a few coordinates contain clues about the attacked sensitive variable. Assumption 1 has been already made *e.g.* by Cagli *et al.* [CDP15]. Depending on the counter-measures implemented into the attacked device, the nature of $\mathcal{I}_\mathrm{Z}$ may be precised. Without any counter-measure, and supposing that the target sensitive variable only leaks once, Assumption 1 states that $\mathcal{I}_\mathrm{Z}$ is only a set of contiguous and constant coordinates, regardless the input traces.

In the case where the target implementation is protected by secret-sharing, $\mathcal{I}_\mathrm{Z}$ will be split into several contiguous and fixed sets whose number is equal to the number of shares in the masking scheme (or at least equal to the number of shares if we relax the hypothesis of one leakage per share). For example if $\mathrm{Z}_1, \mathrm{Z}_2$, leaking respectively at the times samples t_1 and t_2, represent a 2-sharing of Z, then Z_1 and $\mathbf{X}[t]$ with $t \neq t_1$ are independent (resp. Z_2 and $\mathbf{X}[t]$ with $t \neq t_2$ are independent). The conditional probability $\Pr(\mathrm{Z} = s \mid \mathbf{X} = \mathbf{x})$ satisfies for every $s_2 \in \mathcal{Z}$:

$$F^\star(\mathbf{x})[s] = \Pr(\mathrm{Z} = s \mid \mathbf{X} = \mathbf{x}) = \sum_{\substack{s_1, s_2 \in \mathcal{Z} \\ \mathsf{Dec}(s_1, s_2) = s}} \Pr(\mathrm{Z}_1 = s_1 \mid \mathbf{X}[t_1] = \mathbf{x}[t_1]) \Pr(\mathrm{Z}_2 = s_2 \mid \mathbf{X}[t_2] = \mathbf{x}[t_2]) \quad . \tag{7.1}$$

Adding de-synchronization should force $\mathcal{I}_\mathrm{Z}$ to be non-constant between each trace.

The second assumption is the following.

Assumption 2 (Regularity). *The conditional probability distribution $F^\star$ is differentiable over $\mathcal{X}$ and thereby continuous.*

Likewise, Assumption 2 is realistic because it is a direct corollary of a Gaussian leakage assumption for the traces – see Section 3.4. It implies that $\mathbf{x} \mapsto \Pr(\mathbf{X} = \mathbf{x} \mid \mathrm{Z} = s)$ is differentiable and:

$$\nabla_\mathbf{x} \Pr(\mathbf{X} = \mathbf{x} \mid \mathrm{Z} = s) \quad = \quad \mathbf{\Sigma}_s^{-1}(\mathbf{x} - \mathbf{M}_s) \Pr(\mathbf{X} = \mathbf{x} \mid \mathrm{Z} = s) \quad , \tag{7.2}$$

where $\mathbf{M}_s$ and $\mathbf{\Sigma}_s^{-1}$ respectively denote the mean vector and the covariance matrix of the normal probability distribution related to the target sensitive value hypothesis s. Then, from Bayes' theorem – see Subsection 2.2.1, Equation 7.2 and the basic rules for derivatives computation, it gives an analytic expression of $\nabla_\mathbf{x} F^\star(\mathbf{x})$, thereby proving that $F^\star$ is differentiable with respect to the input trace.

Once Assumptions 1 and 2 are stated, we may want to observe their impact on the properties verified by the optimal model derivatives. For such a purpose we start by considering an example on a trace $\mathbf{x}$. Figure 7.1 (left) illustrates such a trace in blue, and the green line depicts a P.o.I, namely a peak of SNR – in other words the set of P.o.Is $\mathcal{I}_\mathrm{Z}$ is reduced to a single time index. The p.m.f. returned by the optimal model $F^\star(\mathbf{x})$ is given at the right of the same figure: it is here represented by the blue histogram over the 256 possible values of a byte. We may fairly suppose that a slight variation on one coordinate not belonging to $\mathcal{I}_\mathrm{Z}$ – dotted in gray in Figure 7.1 (left) – should not radically change the output of the optimal model. The resulting p.m.f., depicted by the gray histogram on Figure 7.1 (right)

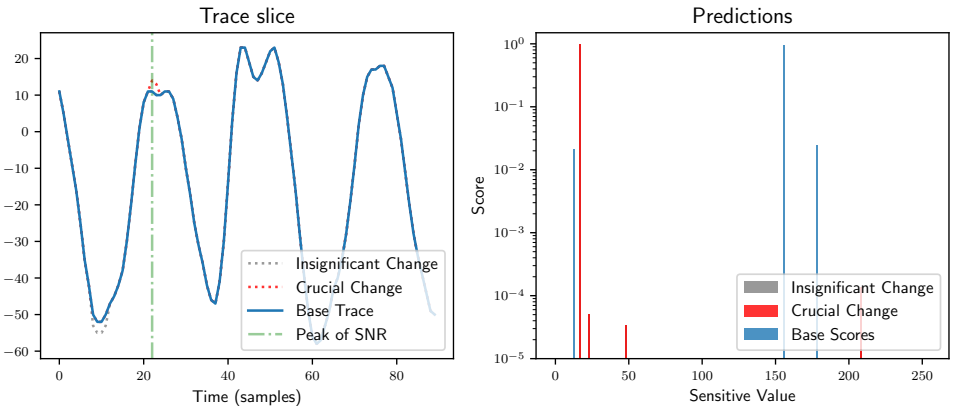


Figure 7.1: Illustration of the Sensitivity Analysis principle. Left: a piece of trace. $t \in \mathcal{I}_Z$ is depicted by the green line, and slight variations dotted in red and gray. Right: predictions of the optimal model.

remains the same, as it is perfectly superposed to the blue histogram. However, applying a slight variation on the coordinate from $\mathcal{I}_Z$ – dotted in red in Figure 7.1 (left) – may radically change the output distribution depicted by the red histogram in Figure 7.1 (right).

This example illustrates the more general idea that small variations applied to the trace at a coordinate $t \in \mathcal{I}_Z$ should radically change the output prediction whereas small variations at $t \notin \mathcal{I}_Z$ should have no impact. As a consequence, if $F^\star$ is differentiable with respect to the input trace (according to Assumption 2), there should exist $s \in \mathcal{Z}$ such that:

$$\frac{\partial}{\partial \mathbf{x}[t]} F^\star(\mathbf{x})[s] \begin{cases} \neq 0 & \text{i.f.f. } t \in \mathcal{I}_Z \\ \approx 0 & \text{i.f.f. } t \notin \mathcal{I}_Z \end{cases}. \tag{7.3}$$

The latter observation can be stated in terms of the Jacobian matrix of the estimator, denoted as $J_{F^\star}(\mathbf{x})$. Its coefficients should be zero almost everywhere, except in columns $t \in \mathcal{I}_Z$:

$$J_{F^\star}(\mathbf{x}) = \begin{pmatrix} \mathbf{0} & \dots & \mathbf{0} & \mathbf{Y}_t & \mathbf{0} & \dots & \mathbf{0} \end{pmatrix}, \tag{7.4}$$

where $\mathbf{Y}_t = \left(\frac{\partial}{\partial \mathbf{x}[t]} F^\star(\mathbf{x})[s_1], \frac{\partial}{\partial \mathbf{x}[t]} F^\star(\mathbf{x})[s_2], \dots, \frac{\partial}{\partial \mathbf{x}[t]} F^\star(\mathbf{x}) \left[s_{|\mathcal{Z}|} \right] \right)^\intercal$ and $\mathbf{0}$ denotes the zero column vector.

The properties verified by the Jacobian matrix matrix in Equation 7.4 form the cornerstone of this chapter, as it implies that we can guess from this matrix whether a coordinate from an input trace belongs to $\mathcal{I}_Z$ or not, *i.e.*, whether a coordinate has been recognized as a P.o.I when designing the optimal model $F^\star$. Moreover, except Assumption 1, no more assumption on the nature of the leakage model is required.

7.3 Proposal for a Characterization Method

So far we have shown that the Jacobian matrix of an optimal model $\mathrm{J}_{F^\star}(\mathbf{x})$ may emphasize P.o.Is. In practice however, the evaluator does not have access to the optimal model, but a trained estimation of it, denoted by $F(\cdot, \theta)$. Here we follow this idea in contexts where the approximation is modeled by training DNNs. Subsection 7.3.1 explains how to compute the gradient visualization and the Jacobian matrix based on a trained DNN. Subsection 7.3.2 illustrates our claim with a toy example. Finally, Subsection 7.3.3 is dedicated to the comparison of our approach with state-of-the-art methods for leakage characterization.

7.3.1 Gradient Approximation with Neural Networks

In Section 4.2.3 we recalled that the universal approximation theorem allows to accurately approximate any function with a MLP, up to a precision depending on the number of neurons in the intermediate layers. It turns out that this theorem also holds for the derivatives of the function to approximate: the latter ones can be accurately approximated by the derivatives of the approximating MLP [Hor91].[1] Hence, the Jacobian matrix of a trained DNN, *i.e.* $\mathrm{J}_{F(\cdot,\theta)}(\mathbf{x})$ represents a sound surrogate to the Jacobian matrix of the optimal model $F^\star$.

To accurately and efficiently compute the Jacobian matrix of a DNN, the backprop algorithm presented in Subsection 4.3.4 can be used. Originally, it applies a reverse mode differentiation in order to compute the gradient of the loss function with respect to the parameter vector θ. Interestingly, we explained that the backward pass of the reverse mode actually computes the derivatives of each layer function with respect to each of its inputs, even if the latter ones are not explicitly part of the learning parameters. This particularly concerns the very first layer which takes as inputs not only some learning parameters but also the input trace $\mathbf{x}$. In other words, given an input trace $\mathbf{x}$, its corresponding sensitive value z, and the loss function used for the optimization $\ell : \mathcal{P}(\mathcal{Z}) \times \mathcal{Z} \to \mathbb{R}_+$,[2] it is possible to compute for free the gradient of $\ell(F(\mathbf{x},\theta), z)$ with respect to the input trace $\mathbf{x}$ whenever the gradient of $\ell(F(\mathbf{x},\theta), z)$ with respect to the learning parameters θ is required. As a consequence, computing such a gradient can be done with a negligible overhead during an iteration of the SGD.

Actually, we justified when presenting the backprop algorithm in Subsection 4.3.4 that the modern DL libraries [PGM+19, AAB+15] are optimized to directly compute the gradient of the loss function without explicitly computing the Jacobian matrix $\mathrm{J}_{F(\cdot,\theta)}(\mathbf{x})$. However, our ultimate goal is still to know the Jacobian matrix of the DNN model. Hopefully, studying either the latter one or the gradient of the loss function is fairly equivalent, as one coordinate of the loss function gradient is a function of elements from the corresponding column in the Jacobian matrix, according to Lemma 1:

$$\nabla_{\mathbf{x}} \ell(F(\mathbf{x},\theta), s) \quad = \quad \mathrm{J}_{F(\cdot,\theta)}(\mathbf{x})^\intercal \cdot \nabla_{\mathbf{y}} \ell(F(\mathbf{x},\theta), s) \quad . \tag{7.5}$$

That is why we propose to visualize the gradient of the loss function, computed with respect to the input trace, to characterize P.o.Is in the context of a DNN attack, instead of the Jacobian matrix – unless explicit mention. More precisely, we visualize the absolute value of each coordinate of the gradient in order to get the sensitivity magnitude. In the following, such a method is named Gradient Visualization (GV). One of the advantage of this method is that the additional source code required to implement this method is very light, as shown in Figure 7.2.

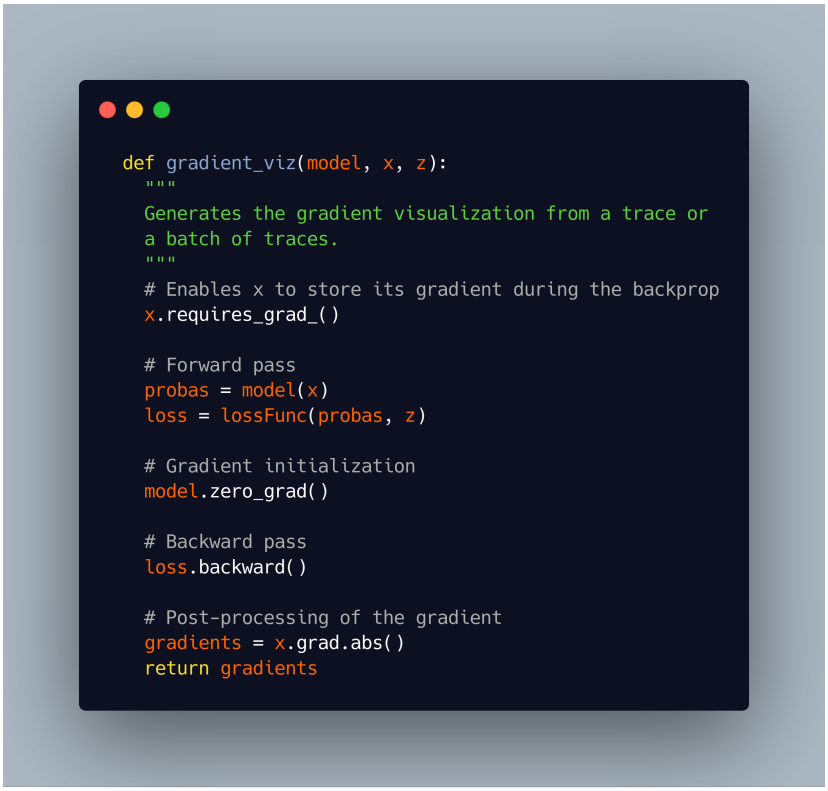
7.3.2 Example on Simulated Data

To illustrate and explain the relevance of the GV method, and before going on experimental data, we here propose to apply it on a toy example, aiming at simulating simple D-dimensional leakages from an n-bit sensitive variable Z. The method follows the same procedure as already explained in Subsection 5.4.1. The traces are defined such that for every $t \in [\![1, D]\!]$:

$$\mathbf{x}_i[t] = \begin{cases} u_i + b_i, \text{ if } t \notin \{t_1, \ldots, t_{d+1}\} \\ \mathrm{hw}(z_{t,i}) + b_i \text{ otherwise} \end{cases} \quad , \tag{7.6}$$

where $(u_i)_i, (b_i)_i$ and all $(z_{t,i})_i$ are i.i.d. draws from the following independent random variables. Respectively, $\mathrm{U} \sim \mathcal{B}(n, 0.5)$, $\mathrm{B} \sim \mathcal{N}(0, \sigma^2)$,[3] where and where $(z_{1,i}, \ldots, z_{d+1,i})$ is a

[1] The interested reader may refer to the survey of Pinkus [Pin99, Sec. 4].
[2] See a definition in Equation 4.2.
[3] It is recalled that hw denotes the Hamming weight function, see Subsection 3.5.1.



```python
def gradient_viz(model, x, z):
  """
  Generates the gradient visualization from a trace or
  a batch of traces.
  """
  # Enables x to store its gradient during the backprop
  x.requires_grad_()

  # Forward pass
  probas = model(x)
  loss = lossFunc(probas, z)

  # Gradient initialization
  model.zero_grad()

  # Backward pass
  loss.backward()

  # Post-processing of the gradient
  gradients = x.grad.abs()
  return gradients
```

Figure 7.2: Source code to implement the GV in Pytorch.

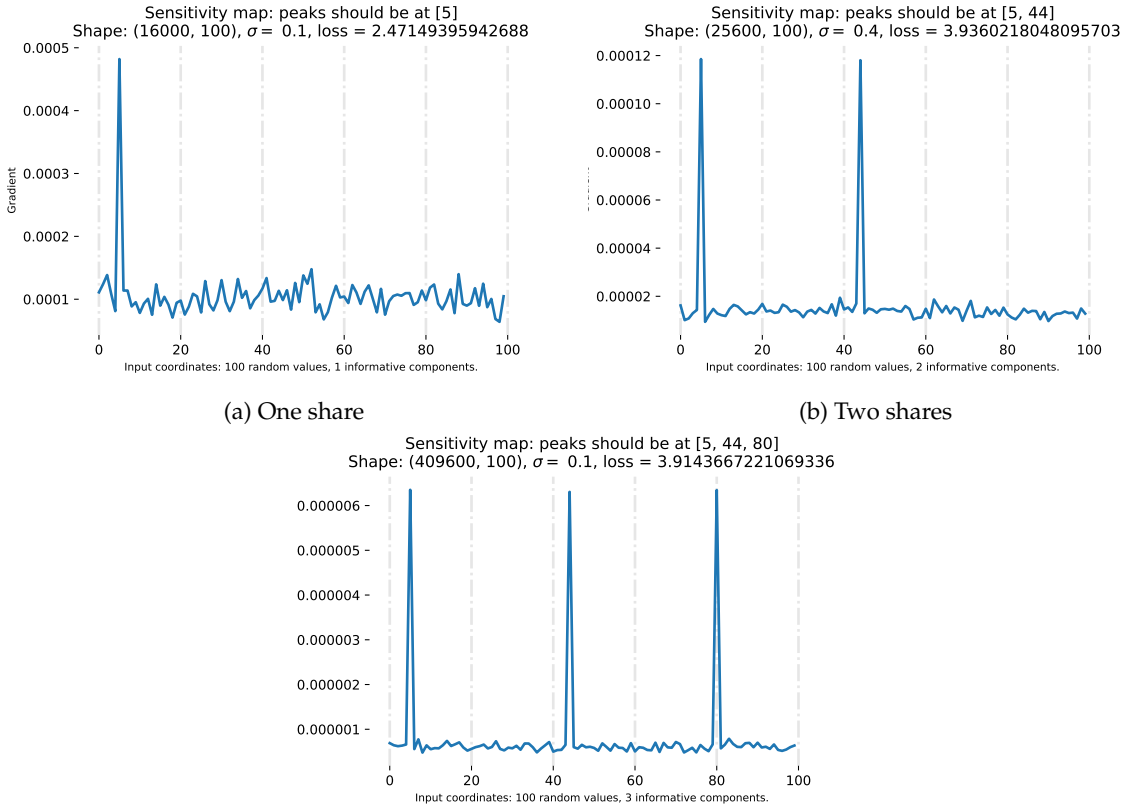
$(d+1)$-sharing of z_i for the bit-wise addition law. This example corresponds to a situation where the leakages on the shares are hidden among values that have no relation with the target.

Every possible combination of the d-sharing has been generated and replicated 100 times before adding the noise, in order to have an exhaustive dataset. Therefore, it contains 100×2^{dn} simulated traces. We ran the experiment for $n = 4$ bits, $d \in \{2, 3\}, D = 100$, and a varying noise $\sigma^2 \in [0, 1]$. Once the data were generated, we trained a neural network with one hidden layer made of D neurons. Figure 7.3 presents some examples obtained for 1 (Figure 7.3a), 2 (Figure 7.3b) and 3 (Figure 7.3c) shares. We clearly see some peaks at the coordinates where the meaningful information have been placed. This confirms that our characterization method is sound when facing leakages protected by secret-sharing, no matter the order though it required 16 times more simulated data and less noised data ($\sigma^2 \geq 0.1$) than for the same experiment against first order secret-sharing.

7.3.3 Comparison with SNR for Characterization

Now we have shown that GV is relevant for characterization on simulated data, one may wonder to what extent this method would be useful compared to other characterization techniques. In this section, we compare our contribution to the SNR used for P.o.Is selection, as presented in Section 3.6.

One has to keep in mind that the SNR is a statistical tool, and produces a single characterization from all the profiling traces; whereas our method gives one map for each trace, though we might average them. This observation has two consequences. First, if an SNR characterization is launched in presence of secret-sharing, every trace coordinate $\mathbf{X}[t]$ is likely to be independent from Z, which will lead the numerator of the SNR (Equation 3.19) to converge towards 0. Secondly, if an SNR characterization is launched in presence of desynchronization, then the denominator of Equation 3.19 is expected to be multiplied by the maximum shift, as argued in Subsection 3.7.2. To sum-up, an SNR characterization cannot



(a) One share

(b) Two shares

(c) Three shares

Figure 7.3: Gradient of the loss function, averaged over the validation traces.

directly highlight higher order leakages when the random material – used for secret-sharing and/or for de-synchronization – is not assumed to be known. Some solutions to deal with this issue have been proposed, *e.g.*, by pre-processing the traces with some re-combination functions – see Section 3.6 – or by applying realignment techniques [vWWB11, NHI+07, DRS+12].

7.3.4 Related Works

The idea of using the derivatives of differentiable models to visualize information is not new. Following the emergence of deep convolutional networks, Simonyan *et al.* [SVZ14] have first proposed the idea of GV to generate a so-called *Sensitivity Map* for image recognition. The approach was motivated by the fact that such a map can be computed for free thanks to the back-propagation algorithm. A derived method, called *Guided Backpropagation* has also been proposed by Springenberg *et al.* [SDBR15]. The latter one slightly modifies the back-propagation rule in a ReLU layer in order to filter the contributions from the upper layers. Actually, Montavon *et al.* [MSM18] state that visualizing the gradient only tracks an explanation to the variation of a final decision – $F(\mathbf{x}, \theta)$ in our context – and not directly the decision itself. To circumvent this, they propose a visualization method called Layerwise Relevance Propagation (LRP). Another method called *Deconvolution* has been proposed by Zeiler *et al.* [ZF14] in order to give insights about the regions of an input data contributing to the activation of a given feature in a model (either in an intermediate layer or in the output layer). In the field of Image Recognition, these methods have been shown to be more relevant than GV.

However, the SCA and Image Recognition fields differ. In the latter one, the decision is highly diluted among lots of pixels, and the decision surface might be locally flat, though we are in an area of interest. Hopefully in an SCA context, Assumption 1 states that it is rea-

sonable to consider that the information is very localized. That is why we are in a particular case where looking at the output sensitivity may be at least or even more interesting than other visualization methods.

In parallel to the publication of our paper at COSADE 2019, Timon has proposed at CHES 2019 the same method, under the name of sensitivity analysis [Tim19].[4] Likewise, Hettwer proposed at SAC'19 a comparison between several techniques such as the GV, the LRP, and some occlusion techniques [HGG19]. The latter ones consist in removing some areas of an input trace, in order to study how a trained model reacts in its predictions. A relevant area should therefore lead to strong dissimilarities in the corresponding predictions when it is removed. Later, Zaid *et al.* proposed the use of *heatmaps* [ZBHV19].[5] It consists in computing the average output over all filters of a given convolution layer. In particular, the heatmap of the first layer is expected to produce a similar map as the GV. Wouters *et al.*, in a paper revisiting the results of Zaid *et al.*, used a variant of the GV called *Gradient* $\times$ *Input* consisting in multiplying the map returned by GV with the input trace itself [WAGP20]. Likewise, Van der Valk *et al.* have investigated the *Singular Vector Canonical Correlation Analysis* provide insights on the layers of a trained MLP [vdVPB19]. Finally, Bursztein *et al.* presented at DEFCON 2020 a tool involving explainability techniques for DL similar to GV [BP20]. By using characterization maps similar to the ones produced by the GV method, they are able to produce a mapping with the assembly instructions yielding the informative leakage. Thanks to a reverse-engineering tool, they are able to map the leaky assembly instructions to the corresponding area in the code, enabling to precisely identify the vulnerability.

7.4 Experimental Verification

So far we have claimed that relevant information can be extracted from the loss gradient of a differentiable model. Following this idea, it has been shown to be efficient to localize P.o.Is on simulated data and we validated that this method might overcome some weaknesses of state-of-the-art techniques. We now plan to experimentally verify these claims on three experimental datasets.

We first conduct comprehensive experimentations on the ASCAD datasets. Before introducing the results in Section 7.5, we first describe our investigations. In Subsection 7.4.1, we present the CNN architecture used for profiling, and Subsection 7.4.2 gives an exhaustive description of all the considered parameters for our experiments.

Then, we also verify the soundness of the GV method over the two polymorphism datasets. The results are reported in Subsection 7.5.5.

7.4.1 CNN Architecture

For these experiments, we consider a VGG-based architecture, that we recall hereafter:

$$F = \mathrm{s} \circ \lambda_{|\mathcal{Z}|} \circ [\sigma \circ \lambda_C]^{n_1} \circ [\delta_p \circ \sigma \circ \mu \circ \gamma_{W,K}]^{n_2} \circ \mu \ , \tag{7.7}$$

where $\gamma_{W,K}$ denotes a convolutional layer made of K filters of size W, μ denotes a batch-normalization layer, σ denotes the ReLU activation function, δ_p denotes an average pooling layer of size p, λ_C denotes a dense layer, and s denotes the softmax layer. Furthermore, the composition $[\delta_p \circ \sigma \circ \mu \circ \gamma_{W,K}]$ is denoted as a convolutional *block*. Likewise, $[\sigma \circ \lambda]$ denotes a dense block. We note n_1 (resp. n_2) the number of dense blocks (resp. convolutional blocks). The details of this architecture have been presented in Subsection 4.2.3. We chose this architecture since it is the baseline used in the works of Benadjila *et al.* [BPS+19] introducing the ASCAD database on which we work – see Subsection 3.8.2.

[4] We prefer using the term "gradient visualization" rather than "sensitivity analysis" which is a metonymy: the latter one encompasses the former one, beside other techniques.

[5] Another technique used by the authors, called *weight visualization* is rather focused on the understanding of the learned weights of the dense layers in a CNN, therefore beyond the scope of this study.

As the ultimate goal is not to get the best architecture as possible, but rather having a simple and efficient one, a lighter baseline has been chosen compared to the original architecture proposed in the authors' work:

- The number of filters in the first layers has been decreased, *i.e.*, $K_0 = 10$ instead of 64, though it is still doubled between each block:

$$K_i = \max(512, K_0 \times 2^i) \ , \tag{7.8}$$

  where K_n denotes the number of filters at the i-th convolutional block for $i \leq n_2$ and the filter size has been set to $W = 5$.

- The dense layers contain less neurons: $C = 1,000$ instead of 4,096.

- The last pooling layer is global – see Subsection 6.2.4 – *i.e.*, its pooling size equals the width of the feature maps in the last convolutional layer, so that each feature maps are reduced to one point. While it drastically reduces the number of neurons in the first dense layer and thereby its number of weights to learn, the global pooling layer also forces the convolutional filters to better localize the discriminative features [ZKL+16].

7.4.2 Settings of the Experiments

Our experiments have been done with the $50,000$ EM traces from the ASCAD database, presented in Subsection 3.8.2. Each trace is made of 700 time samples.[6] Hereafter, the three different configurations investigated in this chapter are presented with the notations taken from [BPS+19]. For each experiment we precise the label to be learned. This label is known during the profiling phase but not during the attack phase:

- **Experiment 1 (no counter-measure)**: the traces are synchronized, the label to learn is $\mathrm{Z} = \mathrm{Sbox}(\mathrm{P} \oplus k^\star) \oplus \mathsf{r}_{\mathsf{out}}$, where $\mathsf{r}_{\mathsf{out}}$ is a random share used to protect the leakage of the Sbox output – see Subsection 3.8.2. In other terms, $\mathsf{r}_{\mathsf{out}}$ is assumed here to be known, like P. The traces correspond to the dataset `ASCAD.h5`, and the labels are recomputed from the `metadata` field of the `hdf5` structure.

- **Experiment 2 (artificial shift)** : the labels are the same as in Exp. 1 but the traces are artificially shifted to the left of a random number of points drawn from a uniform distribution over $[\![0, 100]\!]$. The traces correspond to the dataset `ASCAD_desync100.h5`.

- **Experiment 3 (synchronized traces, with secret-sharing)**: we target $\mathrm{Z} = \mathrm{Sbox}(\mathrm{P} \oplus k^\star)$, *i.e.*, we have no knowledge anymore of the random share $\mathsf{r}_{\mathsf{out}}$ – neither during profiling or attack phase. Concretely, the traces correspond to the dataset `ASCAD.h5` and the labels are directly imported from the field `labels` in the hdf5 structure.

It is noticeable that in every case, as the key is fixed, and both the plaintext and the share $\mathsf{r}_{\mathsf{out}}$ are completely random and independent. Therefore, the labels are always balanced.

Following the results presented in Chapter 5, we use the NLL as a loss function. The settings have been made so that any experiment is reproducible: random seeds are known, and all the settings of the GPU are set to avoid stochastic behavior.[7] For each tested neural network architecture, a 5-fold *cross-validation* strategy has been followed. Namely, the ASCAD database has been split into 5 sets $\mathcal{S}_1, \ldots, \mathcal{S}_5$ of $10,000$ traces each, and the i-th cross-validation, denoted by CV_i, corresponds to a training dataset $\mathcal{S}_p = \cup_{j \neq i} \mathcal{S}_j$ and a validation dataset $\mathcal{S}_v = \mathcal{S}_i$. The given performance metrics and the visualizations are averaged over these 5 folds. The optimization is done with the Adam algorithm – see Subsection 2.5.2.

[6] It corresponds to 26 clock cycles.

[7] Usually, forcing the GPU to have a fully deterministic behavior implies worse runtime performance.

The learning rate is always set to 10^{-4}. Likewise, the batch size has been fixed to 64. For each training, we operate 100 epochs, *i.e.* each couple $(\mathbf{x}_i, z_i)$ is passed 100 times through an iteration of the optimization algorithm, and we keep as the best model the one that has the lowest GE on the validation set.[8]

7.5 Results

This section presents experimentations of the GV in different contexts, namely (Exp. 1) when the implementation embeds no counter-measure, (Exp. 2) when traces are de-synchronized and (Exp. 3) when Boolean secret-sharing is applied. The methods used to train the CNNs, to tune their hyper-parameters and to generate the GV have been presented in Section 7.4.

7.5.1 Application Without Counter-measure

In application context (Exp. 1) – *i.e.* no counter-measure – several CNNs have been trained with the architecture hyper-parameters in Equation 7.7 specified as listed in Table 7.1a. Since the random shares are here directly targeted – *i.e.* the masks are supposed to be known – without re-combination – thereby no dense layer – should be required, according to the study of Benadjila *et al.* [BPS+19, Sec. 4.2.4]. The hyper-parameter n_1 should therefore be null. However, to validate this intuition we let it vary in $\{0, 1, 2\}$. The validation loss corresponding to these values is given in Table 7.1b, where $N_a(F)$ denotes here the minimum number of traces required to have a GE lower than 1. Even if this minimum is roughly the same for the three different configurations, we selected the *best* one – *i.e.* $n_1 = 1$ – for our *best* CNN architecture. Figure 7.4a presents the corresponding GV, and Figure 7.4b depicts the corresponding SNR. It may be observed that the peaks obtained with GV and SNR are identical: the highest point in the SNR is the second highest point in GV, whereas the highest point in GV is ranked 7-th in the SNR peaks. More generally both methods target the same clock cycle (the 19-th). These observations validate the fact that our characterization method is relevant for an unprotected target.

In addition to the GV we also show in Figure 7.5 the Jacobian matrix of the trained model. Around the time coordinate 500 along the x-axis, some blue areas depict a high value in the matrix. One may remark that those blue areas particularly appear for value of the sensitive variable – along the ordinates axis – whose Hamming weight is one. Since a high value in the Jacobian matrix implies a high sensitivity to slight changes in the input trace at the considered time coordinate, we may imagine that the trained CNN is able to give a high confidence when expected to predict those values of the sensitive variable. In other words, they are more distinguishable than the other values. Although not a formal proof, this observation is coherent with a Hamming weight leakage model, where values of the target variable with low – *e.g.* 0 or 1 – or high – *e.g.* 7 or 8 – Hamming weight are more distinguishable than the others. This example hence shows how the Jacobian matrix can bring additional insights compared to the gradient.

[8] Following the discussion in Chapter 5, the other ML metrics are ignored.

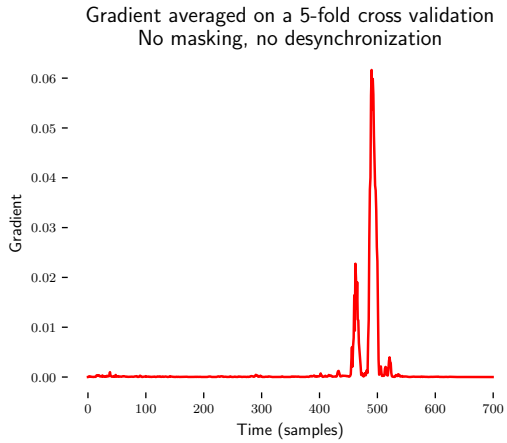
Table 7.1: Settings and results of Exp. 1

(a) Architecture hyper-parameters.

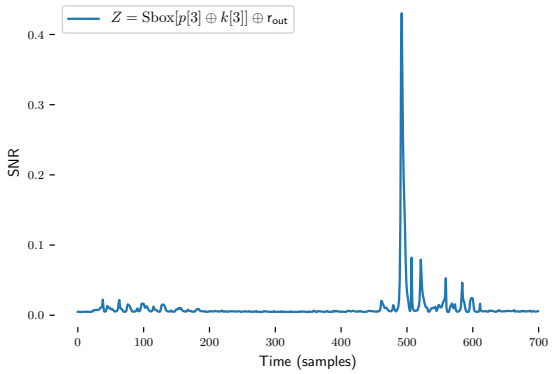
Parameter	Value
n_2	5
n_1	{0, 1, 2}

(b) Performance metrics without counter-measure.

n_1	Loss (bit)	$N_a(F)$
0	6.40	3.25
1	6.15	3
2	6.35	3.25



(a) GV for the trained model with one dense layer, averaged over the validation traces.



(b) The corresponding SNR.

Figure 7.4: Case where no counter-measure is considered.

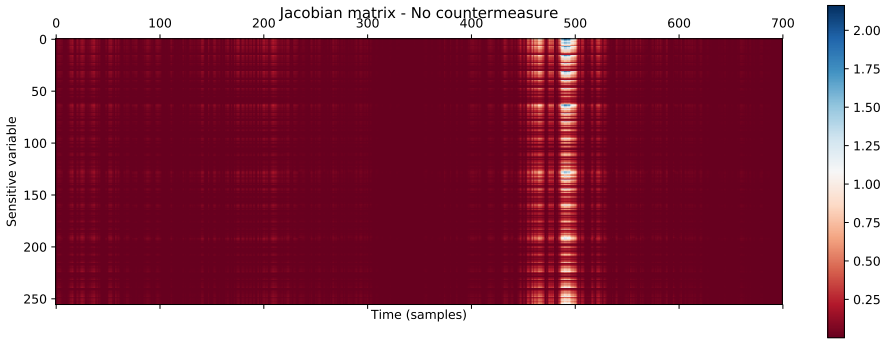


Figure 7.5: Jacobian matrix for the best models in application context Exp. 1.

7.5.2 Application with an Artificial De-synchronization

We now add a new difficulty by considering the case of de-synchronization as described in Subsection 7.4.2. The hyper-parameter grid is exactly the same as in Subsection 7.5.1, and the corresponding loss is given in Table 7.2b. Faced to misalignment, the considered architectures have still good performances, and the attacks succeeded in roughly the same number of traces than before. Interestingly, Figure 7.6 shows that the GV succeeds in recovering the leakage localization while the SNR does not. Actually, the gradient averaged over the profiling traces Figure 7.6a shows that, instead of having a small number of peaks, a band is obtained whose width approximately equals the maximum quantity of shift applied in the traces, namely 100 points. Moreover, individual gradients in Figure 7.6b bring a single characterization for each trace, enabling to guess approximately the shift applied to

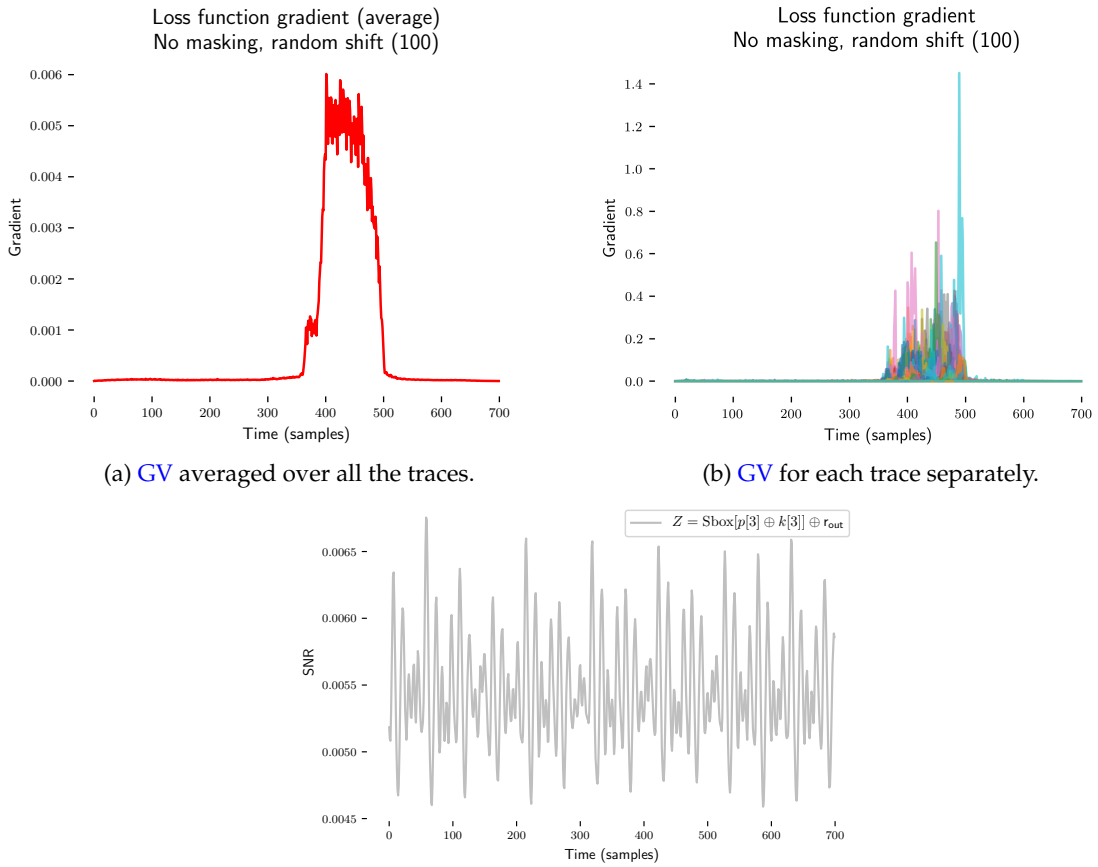
Table 7.2: Settings and results of Exp. 2

(a) Architecture hyper-parameters.

Parameter	Value
n_2	5
n_1	{0, 1, 2}

(b) Performance metrics with de-synchronization.

n_1	Loss (bit)	$N_a(F)$
0	6.64	4.0
1	6.46	3.6
2	6.90	5.4



(a) GV averaged over all the traces.

(b) GV for each trace separately.

(c) The corresponding SNR.

Figure 7.6: Case where de-synchronization is considered.

each trace.

7.5.3 Application with a First Order Secret-Sharing

The next experiment concerns the application of GV in presence of Boolean secret-sharing, namely the one implemented in the ASCAD dataset. Several model configurations have been tested which correspond to the hyper-parameters listed in Table 7.3a. We eventually selected the 8 models that achieved the best efficiency, *i.e.* the model $F(\cdot, \theta)$ with the lowest $N_a(F)$ (Table 7.3b).[9]

For the selected architectures, our first attempt to use GV did not give full satisfaction. As an illustration, Figure 7.7a presents it for one of the tested architectures – averaged over the 5 folds of the cross-validation. Indeed, it looks difficult to distinguish P.o.Is, *i.e.* those identified by our SNR characterization, see Figure 7.8b, from *ghost* peaks, *i.e.* peaks *a priori* independent of the sensitive target. To explain this phenomenon, we decided to study the validation loss of the trained models. Figure 7.7b presents it for one model and for each of the 5 cross-validation folds CV_i, $i \in [\![0, 4]\!]$.

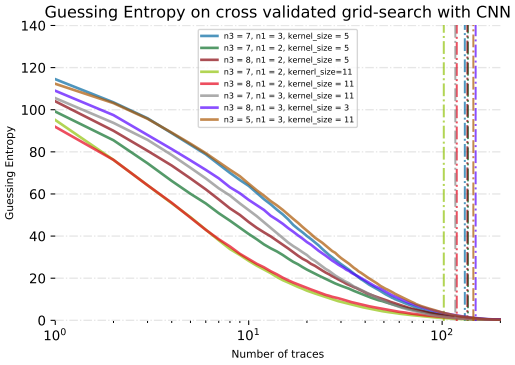
It may be observed in Figure 7.7b that the training and validation loss curves proceeded a fast big decrease after an initial plateau during the first 15 epochs. After that, the validation loss starts increasing while the training loss still decreases. After roughly 50 epochs, the validation loss goes on a regime with unstable results, but still higher than the training loss. These observations are clues of over-fitting.[10] It means that the model exploits

[9] Contrary to the convention taken in Subsection 3.2.2, $N_a(F)$ is here computed with respect to the GE, as defined by Equation 3.7.

[10] We recall the reader that an explanation of over-fitting has been given in Subsection 5.5.2.

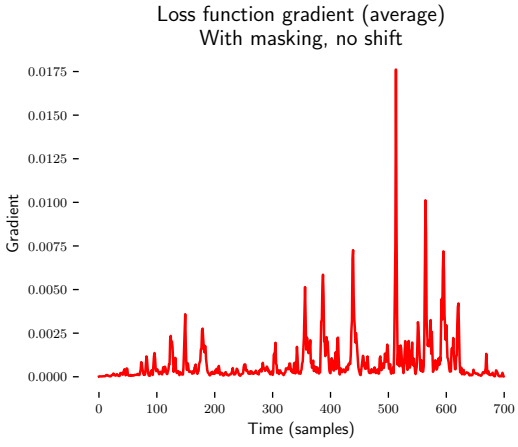
(a) Architecture hyper-parameters – bold values refer to the best choices.

Parameter	Value
n_2	{5, 6, **7**, 8}
n_1	{**2**, 3}
K_0 (first layer)	10
W	{3, 5, **11**}

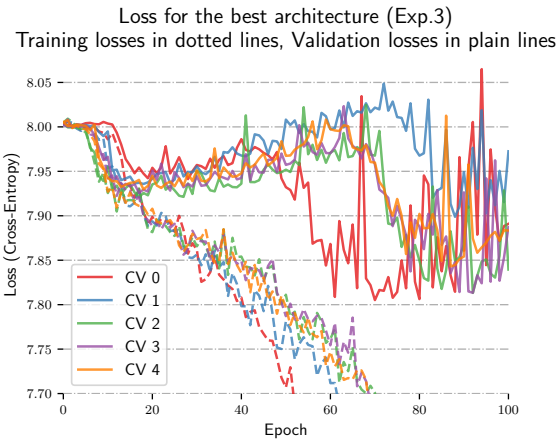


(b) GE for the 8 best architectures.

Table 7.3: Results of Exp. 3 with Boolean secret-sharing.

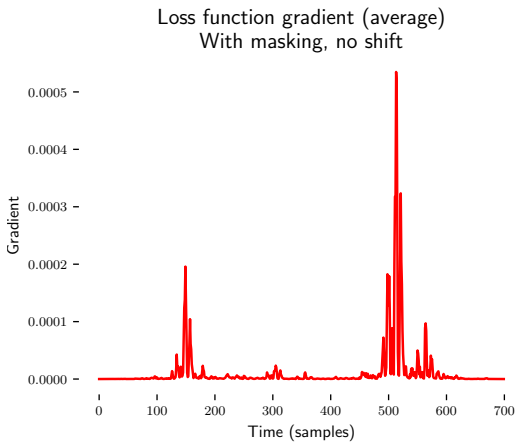


(a) GV in presence of secret-sharing (without early-stopping).

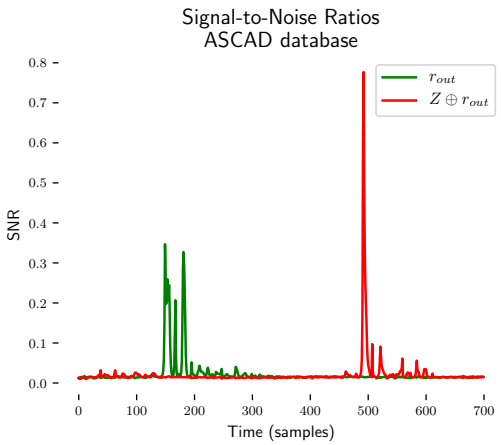


(b) Validation loss for each fold.

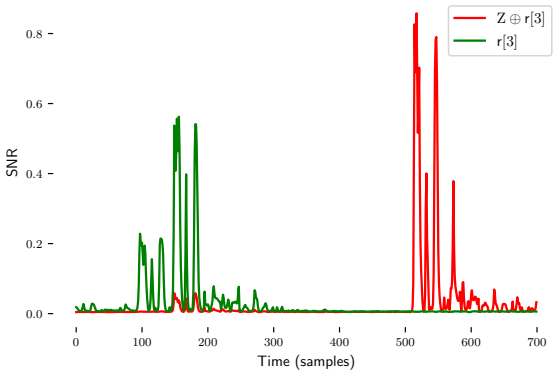
Figure 7.7: Explaining the ghost peaks.



(a) GV.



(b) The corresponding SNR when taking r_{out} as a share.



(c) The corresponding SNR when taking M[3] as a share.

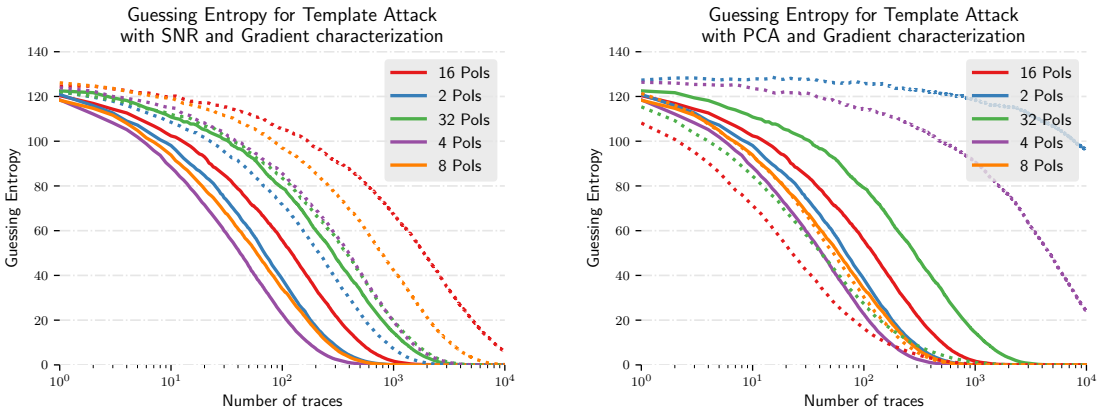
Figure 7.8: Early-stopping is applied.

(non-informative) leakage not localized in the P.o.Is to memorize the profiling data and to improve the training loss. Such a strategy should not generalize well on the validation traces. As we are looking for models that implement a strategy that are generalizable on unseen traces, we propose to use a regularization technique called *early-stopping* [GBC16]: the training is stopped after a number of epochs called *patience* – in our case 10 – if no remarkable decrease – *i.e.* up to a *tolerance term*, 0.25 bits here – is observed in the validation loss. With this slight modification, the previous architectures are trained again from scratch, and a better GV is produced – see Figure 7.8a. As the main peaks are separated enough, an evaluator may conclude that they represent different leakages.

7.5.4 Comparison in the Context of Template Attacks

A careful observation of Figure 7.8 shows that the main peaks given by the GV are not exactly aligned with those given by the SNR characterization – performed under the hypothesis that the shares are known. For GV, the main peak appears at the points corresponding to the 20-th clock cycle, which is one cycle after the one previously targeted by both the GV and the SNR in the previous case where no counter-measure was considered – see Subsection 7.5.1. We validated that this phenomenon occurred for every successful visualization produced by GV. Furthermore, concerning the peaks related to the mask leakage, the GV only emphasizes one clock cycle (the 6-th) whereas the SNR highlights two of them: the 6-th and the 7-th. It implies that the GV should not be taken as an exact equivalent to the SNR.

Actually, we found out a possible track of explanation to justify this slight shift by looking at the pseudo-code sketching the secret-sharing scheme of the ASCAD database [BPS+19,



(a) . . . SNR based attacks.

(b) . . . PCA based attacks.

Figure 7.9: Comparison of the GE for GV based attacks in plain lines and, in dotted lines, . . .

Alg. 1]. Indeed, the latter one emphasizes that another random variable forming a 2-sharing of the output of Sbox, denoted as $\mathsf{r}[3]$, is used in the scheme to protect the sensitive computation before and after applying the SubBytes operation, while the share $\mathsf{r}_{\mathsf{out}}$ considered to compute the SNR in Figures 3.9b and 7.8b is used during the SubBytes operation. By computing the SNRs of $\mathrm{Z} \oplus \mathsf{r}[3]$ and $\mathsf{r}[3]$ in Figure 7.8c, we remark that the peaks of SNR fit better with the peaks of GV previously highlighted in the discussion.

Hence a question through this observation: does it have a sense for the trained CNN to focus more on the leakages of the couple $(\mathrm{Z} \oplus \mathsf{r}[3], \mathsf{r}[3])$ than on the leakages of the couple $(\mathrm{Z} \oplus \mathsf{r}_{\mathsf{out}}, \mathsf{r}_{\mathsf{out}})$? To give an answer, we decided to use our characterization method as a pre-processing for a Template Attack, and compare it to two pre-processing methods: SNR – through P.o.Is selection – and PCA – through dimensionality reduction. The input dimension of the traces are reduced to $2^n, n \in \{1, 2, 3, 4, 5\}$ points, based on the following methods:

- **SNR strategy**: the 2^{n-1} highest P.o.Is from the SNR of $\mathsf{r}_{\mathsf{out}}$ and the 2^{n-1} highest P.o.Is from the SNR of $\mathrm{Z} \oplus \mathsf{r}_{\mathsf{out}}$ are selected;
- **PCA strategy**: the 2^n first components in a decreasing order of contribution are selected;
- **GV strategy**: the 2^{n-1} highest P.o.Is from the GV are selected from the area around the 6-th clock cycle. Likewise, the other half comes from the peaks in the area around the 20-th clock cycle.

Remark 8. *To make a fair comparison in the context of a first order secret-sharing, we assume that we know the shares during the characterization phase for SNR, so that we can localize the corresponding P.o.Is. Notice that we do not assume the mask knowledge neither during the profiling phase nor for the other strategies. Moreover, we do not use any re-combination function as described in Subsection 3.6.1 in none of the different strategies.*

Obviously, this scenario is not realistic as if one has access to the mask during characterization, then the latter one is very likely to be also available during the profiling phase.

Once reduced, the traces are processed with a GT, and the GE is estimated. The results are given on Figure 7.9. The plain curves denote the GE for GV whereas the dotted curves denote either GE obtained with SNR (left) or PCA (right).

From Figure 7.9 we can observe several things:

- Only a few P.o.Is from the GV strategy are needed to get a successful attack. The optimal number of extracted P.o.Is is 4. Beyond that, the other P.o.Is bring more difficulty

in the Template Attack than they bring new information (due to the increasing dimensionality).

- When the SNR strategy is applied, the optimal attack is done with 2 P.o.Is and the more P.o.Is are used, the less efficient are the attacks. This observation confirms that SNR selects relevant P.o.Is as expected. However, when comparing the SNR and GV strategies with a same number of P.o.Is, the latter one appears to be always better, except for 32 P.o.Is where both strategies seem equal.
- The PCA strategy does not work well for the two or four first extracted components. However, when considering eight components and above, it achieves an efficiency as good as the GV strategy, and even sometimes better.
- In any case, the Template Attacks need much more traces to get a GE converging towards zero than the best CNN attack presented in Table 7.3.

Based on the presented experiments, we may draw several conclusions on the GV efficiency. First of all, it seems to be an accurate characterization method, almost always much better than that based on an SNR. This first conclusion enables to answer the question previously asked: the targeted P.o.Is in GV are relevant leakages and the couple of shares $(\mathsf{Z} \oplus \mathsf{r}[3], \mathsf{r}[3])$ leaks more informative clues in the traces about the sensitive variable Z than the couple of shares $(\mathsf{Z} \oplus \mathsf{r}_{\mathsf{out}}, \mathsf{r}_{\mathsf{out}})$. Actually, this finding is not very surprising, since it could have been deduced from the SNRs computed by Benadjila *et al.* when presenting the ASCAD database [BPS+19]. However, since the knowledge of the random shares is not required to train the CNN, the attacker does not have to previously decide which sensitive intermediate computation is the most likely to lead to the most efficient attack.

Secondly, GV can be used as a reliable dimensionality reduction pre-processing in presence of counter-measures, even more reliable than PCA in our cases where one makes a reduction to a very few dimensions (2 or 4). However, this conclusion has a minor interest, as the GV seen as a pre-processing method is done *post-mortem*, and the training of a CNN model did not suffer from a high dimensional input.

Last, but not least, the GV method unfortunately faces a drawback: if the trained CNN overfits, then the GV might suffer from the presence of ghost peaks. That is why the overfitting must be carefully monitored. In this sense, visualizing the gradient can hopefully help to assess whether it is the case or not.

7.5.5 Gradient Visualization on the Polymorphism Dataset

As a final demonstration of the technique, we now apply GV on the trained CNNs from the two attacks $\mathcal{A}_{\mathsf{CNN}}$ described in Subsection 6.2.4, in order to show how this characterization method can provide insights about the acquired traces and the behavior of the target device. For completeness, we additionnaly trained CNNs targeting the remaining bytes of the secret key which have not been investigated yet for the attacks $\mathcal{A}_{\mathsf{CNN}}$ in Chapter 6. The architecture used for those additional trainings remained exactly the same, namely:

$$\mathrm{s} \circ \lambda_{|\mathcal{Z}|} \circ [\sigma \circ \lambda_C]^{n_1} \circ [\delta_p \circ \sigma \circ \mu \circ \gamma_{W,K}]^{n_2} \circ \mu \ , \tag{7.9}$$

where $\gamma_{W,K}$ denotes a convolutional layer made of K filters of size W, μ denotes a batch-normalization layer, σ denotes the ReLU activation function, δ_p denotes an average pooling layer of size p, λ denotes a dense layer, and s denotes the softmax layer. The hyper-parameters are given in Table 6.1.

mbedTLS Figure 7.10 shows the gradient visualizations applied on one trace from the mbedTLS dataset based on the corresponding trained CNN. The top plot shows a trace whereas the bottom plots show the 16 gradients computed from this trace, targeting each

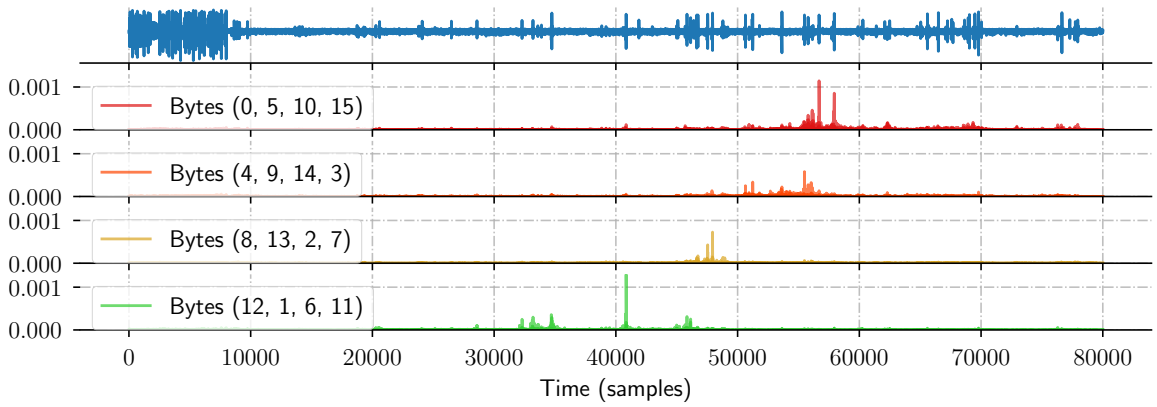
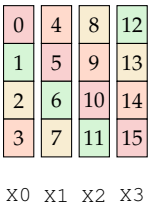
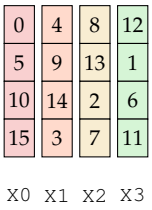


Figure 7.10: GV for one trace of the mbedTLS implementation.



(a) AES state after the first AddRoundKey



(b) AES state at the end of the ShiftRows

Figure 7.11: AES states at two moments in the first round potentially leaking information about the secret key bytes.

byte. Those gradients have been gathered into four pools. First, it can be remarked that contrary to the SNR plotted in Figure 3.13, the GV shows some peaks in the different gradients plotted in Figure 7.10, which shows that the GV is able to emphasize the P.o.Is despite the application of code polymorphism on the target implementation.

More particularly, it may be seen that the peaks of gradient corresponding to the bytes $(12, 1, 6, 11)$ colored in green appear first, followed by those of the bytes $(8, 13, 2, 7)$ in yellow, then those of the bytes $(4, 9, 14, 3)$ in orange and finally the peaks of gradient for the bytes $(0, 5, 10, 15)$ in red. Interestingly, this order can be read in light of the source code of the implementation. The AES state is represented here by four `uint32_t` variables `X0`, ..., `X3`, each one denoting one column of the state array. The repartition of the bytes into the state is represented in Figure 7.11 at two steps of the first round which could potentially be the leakage source. Figure 7.11a denotes the state after the first AddRoundKey while Figure 7.11b denotes the state at the end of the ShiftRows. Since the operations are done column-wise, the bytes belonging to the same column of the state should leak at close time samples to each other in the trace. That being said, we easily remark that the pools of gradient peaks described above coincide with the columns of the AES state at the end of the ShiftRows depicted in Figure 7.11b, which corresponds to the call of the l.u.ts of the T-table implementation, rather than the key addition.

AES 8-bit Figure 7.12 shows the gradient visualizations in the same way as for mbedTLS, but this time, for each key byte separately. A focus on the first peak of each gradient highlights that they appear in increasing order of the byte index: the leakage probably comes from a `for` loop iterating over each byte of the AES state. Thus, the corresponding operation might be either AddRoundKey or SubBytes. Furthermore, a close look at the second peak of each gradient reveals that they are almost aligned, except four of them: $(0, 4, 8, 12)$. A quick look at the ShiftRows operation inside the source code of the AES 8-bit implementation reveals that it never manipulates the latter bytes of the state, contrary to the others. This

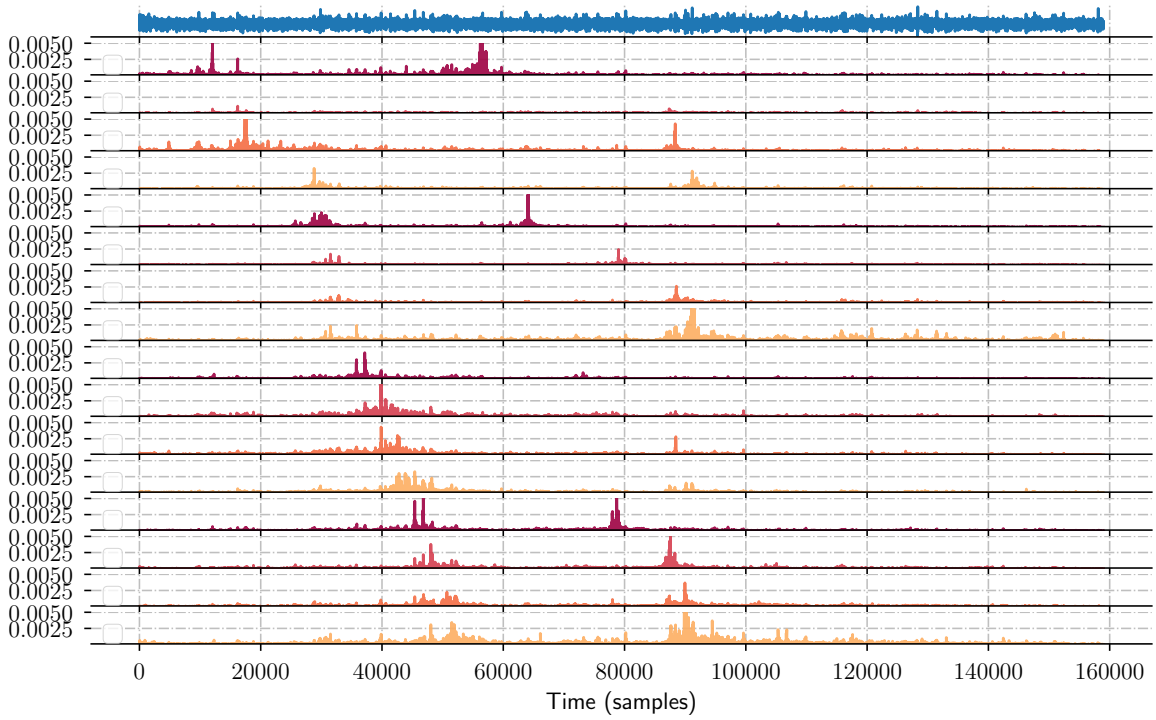


Figure 7.12: GV for one trace of the AES 8-bit implementation. The top plot depicts the considered trace, whereas the bottom plots denote the gradients for each targeted byte.

can also be deduced from Figure 7.11 where the only bytes not moving from Figure 7.11a to Figure 7.11b are those same bytes. We deduce that for the bytes $0, 4, 8, 12$, the CNN exploits the joint leakages of the AddRoundKey and SubBytes operations, whereas for the other bytes the CNN rather exploits the joint leakages of the AddRoundKey and ShiftRows operations.

Anyway, in every cases, two leakages are jointly exploited by the CNN in the AES 8-bit implementation whereas only one leakage seems to be used in the mbedTLS one. This might explain why the attack on the latter implementation is slightly worse than in the former implementation, although the traces seemed less noisy at first sight.

Finally, the gradient visualizations showed in both Figure 7.10 and Figure 7.12 highlight relatively sharp peaks.[11] This means that the CNN model is able to precisely localize the leakages in the traces, despite the application of code polymorphism. In other words, the code transformations applied here did not prevent the CNN model to localize and exploit the leakage. Instead, one would expect sound code transformations to flatten and spread the gradient peaks along a wider zone of the trace, in order to increase uncertainty about the leakage localization. One might imagine other code transformations which could be plugged in the Belleville *et al.*'s tool in order to address this problem, although beyond the scope of this demonstration.

7.6 Conclusion

In this chapter, we have theoretically shown that a method called Gradient Visualization (GV) can be used to localize Points of Interest (P.o.Is). This result relies on two assumptions considered as realistic in an SCA context.

Generally, the efficiency of the proposed method only depends on the ability of the profiling model to succeed in the attack. In the case where counter-measures like secret-sharing or misalignment are considered, CNNs are shown to still build good p.m.f. estimations,

[11] A similar analysis can be done on other traces from the dataset.

and thereby the GV provides a good characterization tool. In addition, such a visualization can be made for each trace individually, and the method does not require more work than needed to perform a profiling with CNNs leading to a successful attack. Therefore, characterization can be done after the profiling phase whereas profiling attacks with Gaussian Templates (GTs) often require to proceed a preliminary characterization phase.

We verified the efficiency of our proposed method on simulated data. It has been shown that as long as a DNN is able to have slightly better performance than randomness, it can localize points containing the informative leakage.

On experimental traces, we have empirically shown that GV is at least as good as the state-of-the-art characterization methods, in different cases corresponding to the presence or not of different counter-measures. Not only it can still localize P.o.Is in presence of desynchronization or secret-sharing but it has also been shown that different P.o.Is can be emphasized compared to the first ones highlighted by SNR. These new P.o.Is have been shown to be at least as relevant as the ones proposed by SNR.

Altogether, the gradient visualization method we proposed here provides tools to the evaluator in order to get a clear understanding of the leakage detected by the DNNs during the profiling phase. We have shown how this characterization could be combined with information on the source code on order to better identify the vulnerability in the code. Therefore, those insights can not only help the evaluator to build its diagnosis, but also help the developer to fix the vulnerability of implementations, no matter they are originally protected or not.

Chapter 8

Conclusion & Perspectives

It is now time to conclude this thesis whose aim was to push the limits of the understanding of deep learning for side-channel analysis. Hereafter, we propose a summary of the works and contributions proposed so far, and we recall to what extent they address the issues raised at the end of Chapter 4. Then, we propose some perspectives, including the description of some seminal works we started during this thesis.

8.1 Summary of the Contributions

After having presented the general framework of side-channel analysis in Chapter 3, we have formalized the use of machine learning in SCA in Chapter 4. Training a model to approximate a true p.m.f. can be seen as solving an optimization problem. More precisely, it consists in selecting from an hypothesis class $\mathcal{H}$ the model F that fits *the most* with a target function $F^\star$, thanks to some pairs of inputs/outputs of the target function, acquired during the profiling phase. Here the target function is a conditional p.m.f. used to feed a distinguisher whose aim is to recover a chunk of secret key. Chapter 4 has reviewed the different advantages and drawbacks of the use of ML, and more particularly the use of DL for SCA. Those observations have lead to raise some issues, summarized at the conclusion of Chapter 4.

The first issue concerned the meaning behind the term "the most" in the previous paragraph. More precisely, it concerns the choice of the loss function quantifying the dissimilarity between the output of a model to learn, and the outputs expected for the optimal model $F^\star$. In particular, what could be the meaning of this loss function, from an SCA point of view? Those are the questions addressed by Chapter 5. We showed that one of the most widely used loss function, namely the NLL, could be used to quantify the quality of the trained model during the attack phase, therefore bridging the gap between the ML metrics and the SCA ones, emphasized first by Cagli *et al.* [CDP17] and Picek *et al.* [PHJ+18]. As a concrete application of our results, it is possible to estimate the efficiency of a key recovery based on the values of the loss function reached at the end of the profiling phase, without having to perform the key recovery as itself. Moreover, we showed that this approach is sound no matter the nature of the counter-measure used to protect the target implementation. Altogether, this study has paved the way towards a better theoretical understanding of DL for SCA.

Chapter 6 provided insights from an evaluation of a software device protected with the code polymorphism counter-measure. From a developer's point-of-view, it demonstrated the necessity to adapt the configuration of the random code generators a.k.a. SGPCs in order to improve the efficiency of the counter-measure against more sophisticated attacks, *e.g.*, based on deep learning. From an evaluator's point-of-view, it emphasized the fact that highly complex DNN architectures with many layers and numerous parameters to fit is not always necessary in SCA, since simpler architectures such as the one suggested in this

work remained sound against traces 32 times larger than what has been tackled so far with deep learning. Although the works presented in Chapter 6 represent a case study rather than a coprehensive one, we hope that it will trigger more thorough discussions about the common belief that complex DNN architectures would be necessary to tackle a DL-based SCA evaluation. Likewise, a natural extension of our works here woul be to investigate the case of large-scale SCA traces on implementations not only protected by hiding, but also protected with secret-sharing.

Finally, Chapter 7 has proposed a method called Gradient Visualization which can be used to localize Points of Interest. This method opens the black-box of deep learning models, making their decisions more transparent in an SCA context. We have shown that this method, requiring a negligible runtime overhead with respect to the one required during the profiling phase, is not particularly dependent on the nature of the counter-measures used to protect a target implementation, as long as the underlying DL model is itself robust to the particular counter-measure. Moreover, the characterization can be done for each trace separately, which may be of particular interest when drawing a precise diagnosis. We have illustrated the relevance of the method on several datasets, and shown of it can be used to identify the origin of the vulnerability in the source code. An automatization of the vulnerability detection, built on a DL-based characterization method such as GV, thereby extending the recent works of Burzstein *et al.*, could be a promising step forward.

8.2 New Tracks of Research in DL-based SCA

We have recalled in Chapter 4 that DNNs are particularly interesting in SCA since they are universal enough to be able to circumvent any counter-measure investigated so far. Actually, Bronchain *et al.* emphasized an intriguing difficulty of DL-based SCA in a paper at CHES 2020: they show through simulated experiments that MLPs encounter difficulties to learn a leakage spanned by an affine secret-sharing scheme, although some of the shares are not noisy. This demonstration is a side experiment of a more general study of a new public dataset which will be soon released by the ANSSI,[1] gathering traces acquired on an MCU protected with an affine secret-sharing scheme – see Subsection 3.7.1. Bronchain *et al.* conclude their paper with this challenge to the proponents of deep learning in SCA.

> Learning this field multiplication in a fully automated manner appears to be a challenging task for existing ML / DL tools (besides being a waste since this part of the attack is trivial to perform manually) [...]. Concretely, we believe our work at least states an interesting challenge to ML / DL research: can the ANSSI implementation be broken [with DL] [...] with similar time complexities and profiling efforts as [their GT-based attacks]? [BS20]

We fully agree with this point of view: it seems obvious that expecting a DNN model to learn how to recombine the informative leakages of several shares is a waste since the nature of the scheme is often known in a profiling attack, according to the Kerckhoff's principle. That is why Bronchain *et al.* could emphasize a simple but efficient attack based on GTs, by leveraging all the knowledge about the implementation. Nevertheless, their proposal implicitly requires additionally to know the values of the random shares used in the secret-sharing scheme during the profiling phase. Most of the time in practical evaluations, this cannot always be assumed, since the developers are rarely willing to give the access to the output of the RNG. This is somehow formalized for example by the model threat proposed by Hoang *et al.* at CHES 2020 [HHO20, Sec. 4.2.1]. Thus, there is a gap between efficient attacks in a worst-case-yet-sometimes-unrealistic scenario – mostly useful from a theoretical point-of-view *e.g.* to discuss the generic soundness of a counter-measure – and automated

[1] The database will be hosted on the `data.gouv.fr` platform.

attacks in a more realistic scenario, which is more relevant for practical evaluations – *e.g.* useful to evaluate the robustness of a particular implementation. I, for one, think that DL may still bring relevant contributions in the latter case, thereby bridging the gap between both scenarios. Hereafter, we propose two main tracks of practical improvements we wish we had time to more deeply explore through this thesis.

8.2.1 Discrete Convolution Layers

We have seen through Equation 5.21 that the conditional p.m.f. of a sensitive random variable protected with a d-th order Boolean scheme could be rephrased as a discrete convolution product with respect to the additive group $(\mathbb{F}_{2^8}, \oplus)$:

$$F^\star(\mathbf{x}) = \Pr(\mathrm{Z} \mid \mathbf{X} = \mathbf{x}) : s \mapsto (h_0 * h_1 * \cdots * h_d)(s) \ , \tag{8.1}$$

where the $h_i = \Pr(\mathrm{Z}_i \mid \mathbf{X} = \mathbf{x})$ are the conditional p.m.f.s of each share Z_i separately. Through this thesis, we have recalled that the profiling phase consisted in substituting the optimal model $F^\star$ by another one $F(\cdot;\theta)$ whose parameters θ are adjusted during the profiling phase. This formulation implies that the attacker must jointly learn not only the leakage models of each share individually but also the way those functions are then recombined to give $F^\star$.

Since the secret-sharing scheme can be assumed to be known by the attacker/evaluator, our proposal would be to directly encode it in the DNN. In other words, we might substitute the learning of $F(\cdot,\theta)$ with the *joint* learning of elementary models $\widetilde{F}(\cdot,\theta_i) : \mathcal{X} \to \mathcal{P}(\mathcal{Z})$, such that:

$$F(\cdot;\theta) = \widetilde{F}(\cdot;\theta_0) * \widetilde{F}(\cdot;\theta_1) * \ldots * \widetilde{F}(\cdot;\theta_d) \ , \tag{8.2}$$

where $\theta = (\theta_0, \theta_1, \ldots, \theta_d)^\intercal$. Here, each elementary model $\widetilde{F}(\cdot,\theta_i)$ would approximate the true leakage h_i of each share Z_i. By "joint learning", we mean that we would still use the values of the sensitive target variable Z as labels during for the training, whereas Bronchain *et al.* trained their elementary leakage models *independently* from each other, *i.e.*, by using the values of the random shares Z_i as labels, which we previously claimed to be hardly feasible in practical evaluations.

Through this new formulation, although the leakage models of each share must still be jointly learned by the attacker, the specific learning of the recombination of the different shares' leakage model could be spared. This would still be harder than independently learning the different elementary leakage models, but would represent a good balance between the assumptions made in academic works and those made in industrial applications.

Moreover, this trick could be extended to any group-based secret-sharing scheme, so the difficulty of learning would not depend anymore on the nature of the scheme. Hence, the difficulty to learn the field multiplication raised by Bronchain *et al.* [BS20] could be circumvented, while not requiring too much unrealistic assumptions in the threat model in practice.

8.2.2 Extending and Generalizing Multi-Task Learning

Maghrebi recently proposed the so-called *multi-labeling* technique – a.k.a. multi-task learning, see Subsection 4.4.5 – in order to target several intermediate sensitive computations at once [Mag20]. However, they claim that their solution only works practically for at most two target variables simultaneously.

During the evaluation of code polymorphism, we have also tested a multi-task learning methodology which is not limited to such a low number of target variables. As an example, we have been able to target the 16 output bytes of the AddRoundKey operation at once during the evaluation of both mbedTLS and AES 8-bit implementations. Not only the architecture used is able to better leverage the computation capacities of a GPU by running more

operations in parallel, but the similarities between the targeted leakages allows to put some of the first layers of the 16 corresponding CNNs in common, into a so-called *stem* block. The spare in the number of parameters can be seen as a way to regularize the training of DL models, by decreasing the ratio between the number of parameters to adjust and the number of labels to predict. The SCA framework is particularly adapted to multi-task learning, since there are many intermediate sensitive computations, yielding similar leakage behavior, that the attacker could target at once.

This could be particularly helpful when tackling the profiling of an implementation protected with secret-sharing. Indeed, most of these implementations may use the same random share to protect many sensitive intermediate computations. As an example, in the ASCAD dataset, the same random share $\mathsf{r_{out}}$ is used to protect all the output bytes of the SubBytes operation. Whereas targeting only one byte – *e.g.* the first one – would require a DL model to localize two intermediate computations, *i.e.* both $\mathrm{Sbox}[p_0 \oplus k_0] \oplus \mathsf{r_{out}}$ and $\mathsf{r_{out}}$, targeting at the same time the 16 output bytes would actually require to localize 17 intermediate operations, *i.e.* the $\mathrm{Sbox}[p_i \oplus k_i] \oplus \mathsf{r_{out}}$ for $0 \leq i \leq 15$ and $\mathsf{r_{out}}$. In other words, the ratio between the number of labels carrying information to feed the training and the number of intermediate computations to localize would increase from $\frac{1}{2}$ to $\frac{16}{17} \approx 0.94$. Intuitively, we expect this trick to make the learning procedure more efficient.

8.2.3 Final Word

We hope that the works presented in this thesis will be of great interest for the SCA community, in order to better understand the way how DL-aided SCA works, while bringing more trust in this approach. The contributions presented so far intended to be not only theoretical, but also practical, and can be easily implemented in the whole workflow of DL-based SCA for security evaluations. Finally, we hope that the last two tracks of research, will give the members of the SCA community some inspiration to guide their future work towards better embracing the full potential of DL in their evaluations.

Bibliography

[AAB+15] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org. 70, 109

[ABI] 8.8 billion smart cards shipped in 2014 driven by growth in the banking and sim card markets. https://www.abiresearch.com/press/88-billion-smart-cards-shipped-in-2014-driven-by-g/. Accessed: 2020-09-11. 3

[ABP12] Giovanni Agosta, Alessandro Barenghi, and Gerardo Pelosi. A code morphing methodology to automate power analysis countermeasures. In Patrick Groeneveld, Donatella Sciuto, and Soha Hassoun, editors, *The 49th Annual Design Automation Conference 2012, DAC '12, San Francisco, CA, USA, June 3-7, 2012*, pages 77–82. ACM, 2012. 46, 94

[ABPS15] Giovanni Agosta, Alessandro Barenghi, Gerardo Pelosi, and Michele Scandale. The MEET approach: Securing cryptographic embedded software against side channel attacks. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 34(8):1320–1333, 2015. 46, 94

[AD86] David Aldous and Persi Diaconis. Shuffling cards and stopping times. *The American Mathematical Monthly*, 93(5):333–348, 1986. III

[AG01] Mehdi-Laurent Akkar and Christophe Giraud. An implementation of DES and aes, secure against some attacks. In Çetin Kaya Koç, David Naccache, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2001, Third International Workshop, Paris, France, May 14-16, 2001, Proceedings*, volume 2162 of *Lecture Notes in Computer Science*, pages 309–318. Springer, 2001. 42

[APSV20] Melissa Azouaoui, Romain Poussier, François-Xavier Standaert, and Vincent Verneuil. Key enumeration from the adversarial viewpoint. In Sonia Belaïd and Tim Güneysu, editors, *Smart Card Research and Advanced Applications*, pages 252–267, Cham, 2020. Springer International Publishing. 27

[ARM19] ARMmbed. 32-bit T-table implementation of AES for mbedTLS. https://github.com/ARMmbed/mbedtls/blob/master/library/aes.c, 2019. 51

[BBCS20] Charles-Henry Bertrand, Olivier Bronchain, Gaëtan Cassiers, and François-Xavier Standaert. How to fool a black box machine learning based side-channel security evaluation. In *Yet Another Conference on CRYPTography and Embedded Devices, YACCRYPTED 2020*, May 2020. 73

[BCH+20a] Nicolas Belleville, Damien Couroussé, Karine Heydemann, Quentin Meunier, and Inès Ben El Ouahma. Maskara: Compilation of a masking countermeasure with optimised polynomial interpolation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2020. 94

[BCH+20b] 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. In *27th Annual Network and Distributed System Security Symposium, NDSS 2020, San Diego, California, USA, February 23-26, 2020*. The Internet Society, 2020. 74

[BCHC18] Nicolas Belleville, Damien Couroussé, Karine Heydemann, and Henri-Pierre Charles. Automated software protection for the masses against side-channel attacks. *ACM Trans. Archit. Code Optim.*, 15(4), November 2018. 45, 46, 51, 52, 94, 95, 101, 102

[BCO04] Eric Brier, Christophe Clavier, and Francis Olivier. Correlation power analysis with a leakage model. In Marc Joye and Jean-Jacques Quisquater, editors, *Cryptographic Hardware and Embedded Systems - CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11-13, 2004. Proceedings*, volume 3156 of *Lecture Notes in Computer Science*, pages 16–29. Springer, 2004. 35, 36

[BCZ18] Luk Bettale, Jean-Sébastien Coron, and Rina Zeitoun. Improved high-order conversion from boolean to arithmetic masking. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(2):22–45, 2018. 42

[BDM+20] Sonia Belaïd, Pierre-Évariste Dagand, Darius Mercadier, Matthieu Rivain, and Raphaël Wintersdorff. Tornado: Automatic generation of probing-secure masked bitsliced implementations. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part III*, volume 12107 of *Lecture Notes in Computer Science*, pages 311–341. Springer, 2020. 45, 94

[BDP06] William Burr, Donna Dodson, and W. Polk. Electronic authentication guideline. Technical report, National Institute for Standard and Technology, 2006. 29

[Bei11] Amos Beimel. Secret-sharing schemes: A survey. In Yeow Meng Chee, Zhenbo Guo, San Ling, Fengjing Shao, Yuansheng Tang, Huaxiong Wang, and Chaoping Xing, editors, *Coding and Cryptology - Third International Workshop, IWCC 2011, Qingdao, China, May 30-June 3, 2011. Proceedings*, volume 6639 of *Lecture Notes in Computer Science*, pages 11–46. Springer, 2011. 40

[BFG+17] Josep Balasch, Sebastian Faust, Benedikt Gierlichs, Clara Paglialonga, and François-Xavier Standaert. Consolidating inner product masking. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I*, volume 10624 of *Lecture Notes in Computer Science*, pages 724–754. Springer, 2017. 43

[BGP+11] Lejla Batina, Benedikt Gierlichs, Emmanuel Prouff, Matthieu Rivain, François-Xavier Standaert, and Nicolas Veyrat-Charvillon. Mutual information analysis: a comprehensive study. *J. Cryptology*, 24(2):269–291, 2011. 34, 76

[BHM+19] Olivier Bronchain, Julien M. Hendrickx, Clément Massart, Alex Olshevsky, and François-Xavier Standaert. Leakage certification revisited: Bounding model errors in side-channel security evaluations. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part I*, volume 11692 of *Lecture Notes in Computer Science*, pages 713–737. Springer, 2019. 76, 78, 79, 80, 81, 92

[BKM+15] Andrey Bogdanov, Ilya Kizhvatov, Kamran Manzoor, Elmar Tischhauser, and Marc Witteman. Fast and memory-efficient key recovery in side-channel attacks. In Orr Dunkelman and Liam Keliher, editors, *Selected Areas in Cryptography - SAC 2015 - 22nd International Conference, Sackville, NB, Canada, August 12-14, 2015, Revised Selected Papers*, volume 9566 of *Lecture Notes in Computer Science*, pages 310–327. Springer, 2015. 25

[BKR11] Andrey Bogdanov, Dmitry Khovratovich, and Christian Rechberger. Biclique cryptanalysis of the full AES. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings*, volume 7073 of *Lecture Notes in Computer Science*, pages 344–371. Springer, 2011. 2, 4

[BLH93] Yoshua Bengio, Yann LeCun, and Donnie Henderson. Globally trained handwritten word recognizer using spatial representation, convolutional neural networks, and hidden markov models. In Jack D. Cowan, Gerald Tesauro, and Joshua Alspector, editors, *Advances in Neural Information Processing Systems 6, [7th NIPS Conference, Denver, Colorado, USA, 1993]*, pages 937–944. Morgan Kaufmann, 1993. 64

[BP20] Elie Burzstein and Jean-Michel Picod. A hacker's guide to reducing side-channel attack surfaces using deep-learning. https://elie.net/talk/a-hacker-guide-to-side-channel-attack-surface-reduction-using-deep-learning/, 2020. 112

[BPRS17] Atilim Gunes Baydin, Barak A. Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: a survey. *J. Mach. Learn. Res.*, 18:153:1–153:43, 2017. 70

[BPS+19] Ryad Benadjila, Emmanuel Prouff, Rémi Strullu, Eleonora Cagli, and Cécile Dumas. Deep learning for side-channel analysis and introduction to ASCAD database. *Journal of Cryptographic Engineering*, November 2019. 48, 65, 67, 71, 72, 73, 95, 96, 97, 112, 113, 114, 119, 120

[BR14] Lejla Batina and Matthew Robshaw, editors. *Cryptographic Hardware and Embedded Systems - CHES 2014 - 16th International Workshop, Busan, South Korea, September 23-26, 2014. Proceedings*, volume 8731 of *Lecture Notes in Computer Science*. Springer, 2014. 138, 140

[BRB+18] Ishmael Belghazi, Sai Rajeswar, Aristide Baratin, R. Devon Hjelm, and Aaron C. Courville. MINE: mutual information neural estimation. *CoRR*, abs/1801.04062, 2018. 92

[Bre01] Leo Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, 2001. 60

[BS20] Olivier Bronchain and François-Xavier Standaert. Side-channel countermeasures' dissection and the limits of closed source security evaluations. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020(2):1–25, Mar 2020. 73, 102, 126, 127

[BV98] Dan Boneh and Ramarathnam Venkatesan. Breaking RSA may not be equivalent to factoring. In Kaisa Nyberg, editor, *Advances in Cryptology - EUROCRYPT '98, International Conference on the Theory and Application of Cryptographic Techniques, Espoo, Finland, May 31 - June 4, 1998, Proceeding*, volume 1403 of *Lecture Notes in Computer Science*, pages 59–71. Springer, 1998. 2

[BV14] Stephen P. Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2014. 21, 22

[C+15] François Chollet et al. Keras. https://keras.io, 2015. 70

[Cag18] Eleonora Cagli. *Feature Extraction for Side-Channel Attacks. (Extraction de caractéristiques pour les attaques par canaux auxiliaires)*. PhD thesis, Sorbonne University, France, 2018. 7, 26, XIII

[CBR+16] Damien Couroussé, Thierno Barry, Bruno Robisson, Philippe Jaillon, Olivier Potin, and Jean-Louis Lanet. Runtime code polymorphism as a protection against side channel attacks. In Sara Foresti and Javier López, editors, *Information Security Theory and Practice - 10th IFIP WG 11.2 International Conference, WISTP 2016, Heraklion, Crete, Greece, September 26-27, 2016, Proceedings*, volume 9895 of *Lecture Notes in Computer Science*, pages 136–152. Springer, 2016. 46, 94

[CCC+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. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2019(2):132–161, Feb 2019. 71, 74

[CCD00] Christophe Clavier, Jean-Sébastien Coron, and Nora Dabbous. Differential power analysis in the presence of hardware countermeasures. In Çetin Kaya Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2000, Second International Workshop, Worcester, MA, USA, August 17-18, 2000, Proceedings*, volume 1965 of *Lecture Notes in Computer Science*, pages 252–263. Springer, 2000. 45

[CDP15] Eleonora Cagli, Cécile Dumas, and Emmanuel Prouff. Enhancing dimensionality reduction methods for side-channel attacks. In Naofumi Homma and Marcel Medwed, editors, *Smart Card Research and Advanced Applications - 14th International Conference, CARDIS 2015, Bochum, Germany, November 4-6, 2015. Revised Selected Papers*, volume 9514 of *Lecture Notes in Computer Science*, pages 15–33. Springer, 2015. 37, 106, 107

[CDP16] Eleonora Cagli, Cécile Dumas, and Emmanuel Prouff. Kernel discriminant analysis for information extraction in the presence of masking. In Kerstin Lemke-Rust and Michael Tunstall, editors, *Smart Card Research and Advanced Applications - 15th International Conference, CARDIS 2016, Cannes, France, November 7-9, 2016, Revised Selected Papers*, volume 10146 of *Lecture Notes in Computer Science*, pages 1–22. Springer, 2016. 37, 106

[CDP17] Eleonora Cagli, Cécile Dumas, and Emmanuel Prouff. Convolutional neural networks with data augmentation against jitter-based countermeasures - profiling attacks without pre-processing. In Wieland Fischer and Naofumi Homma, editors, *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*, volume 10529 of *Lecture Notes in Computer Science*, pages 45–68. Springer, 2017. 9, 45, 62, 67, 70, 71, 72, 74, 95, 96, 97, 99, 125, XIII

[CG09] Christophe Clavier and Kris Gaj, editors. *Cryptographic Hardware and Embedded Systems - CHES 2009, 11th International Workshop, Lausanne, Switzerland, September 6-9, 2009, Proceedings*, volume 5747 of *Lecture Notes in Computer Science*. Springer, 2009. 133, 144

[CGC+21] Wei Cheng, Sylvain Guilley, Claude Carlet, Sihem Mesnager, and Jean-Luc Danger. Optimizing inner product masking scheme by a coding theory approach. *IEEE Trans. Inf. Forensics Secur.*, 16:220–235, 2021. 43

[CHM+15] Anna Choromanska, Mikael Henaff, Michaël Mathieu, Gérard Ben Arous, and Yann LeCun. The loss surfaces of multilayer networks. In Guy Lebanon and S. V. N. Vishwanathan, editors, *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2015, San Diego, California, USA, May 9-12, 2015*, volume 38 of *JMLR Workshop and Conference Proceedings*. JMLR.org, 2015. 68

[CJRR99] Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards sound approaches to counteract power-analysis attacks. In Michael J. Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 398–412. Springer, 1999. 41, 43, I

[CK09] Jean-Sébastien Coron and Ilya Kizhvatov. An efficient method for random delay generation in embedded software. In Clavier and Gaj [CG09], pages 156–170. 45, 49, 95

[CK10] Jean-Sébastien Coron and Ilya Kizhvatov. Analysis and improvement of the random delay countermeasure of CHES 2009. In Stefan Mangard and François-Xavier Standaert, editors, *Cryptographic Hardware and Embedded Systems, CHES 2010, 12th International Workshop, Santa Barbara, CA, USA, August 17-20, 2010. Proceedings*, volume 6225 of *Lecture Notes in Computer Science*, pages 95–109. Springer, 2010. 49

[CK13] Omar Choudary and Markus G. Kuhn. Efficient template attacks. In Aurélien Francillon and Pankaj Rohatgi, editors, *Smart Card Research and Advanced Applications - 12th International Conference, CARDIS 2013, Berlin, Germany, November 27-29, 2013. Revised Selected Papers*, volume 8419 of *Lecture Notes in Computer Science*, pages 253–270. Springer, 2013. 33, 37, 106

[CK14] Marios O. Choudary and Markus G. Kuhn. Efficient stochastic methods: Profiled attacks beyond 8 bits. In Marc Joye and Amir Moradi, editors, *Smart Card Research and Advanced Applications - 13th International Conference, CARDIS 2014, Paris, France, November 5-7, 2014. Revised Selected Papers*, volume 8968 of *Lecture Notes in Computer Science*, pages 85–103. Springer, 2014. 37, 106

[CLM20] Valence Cristiani, Maxime Lecomte, and Philippe Maurine. Leakage assessment through neural estimation of the mutual information. In Jianying Zhou,

Mauro Conti, Chuadhry Mujeeb Ahmed, Man Ho Au, Lejla Batina, Zhou Li, Jingqiang Lin, Eleonora Losiouk, Bo Luo, Suryadipta Majumdar, Weizhi Meng, Martín Ochoa, Stjepan Picek, Georgios Portokalidis, Cong Wang, and Kehuan Zhang, editors, *Applied Cryptography and Network Security Workshops - ACNS 2020 Satellite Workshops, AIBlock, AIHWS, AIoTS, Cloud S&P, SCI, SecMT, and SiMLA, Rome, Italy, October 19-22, 2020, Proceedings*, volume 12418 of *Lecture Notes in Computer Science*, pages 144–162. Springer, 2020. 92

[Cor14] Jean-Sébastien Coron. Higher order masking of look-up tables. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, volume 8441 of *Lecture Notes in Computer Science*, pages 441–458. Springer, 2014. 42

[Cra99] Harald Cramér. *Mathematical methods of statistics*. Princeton University Press, 1999. OCLC: 185436716. 79

[CRR02] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. Template attacks. In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*, volume 2523 of *Lecture Notes in Computer Science*, pages 13–28. Springer, 2002. 32

[CRZ18] Jean-Sébastien Coron, Franck Rondepierre, and Rina Zeitoun. High order masking of look-up tables with common shares. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2018(1):40–72, Feb 2018. 42

[CT06] Thomas M. Cover and Joy A. Thomas. *Elements of information theory (2. ed.)*. Wiley, 2006. 18

[CV95] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Mach. Learn.*, 20(3):273–297, 1995. 60

[dCGRP19] Eloi de Chérisey, Sylvain Guilley, Olivier Rioul, and Pablo Piantanida. Best information is most successful: Mutual information and success rate in side-channel analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2019(2):49–79, Feb 2019. 76, 77, 81

[DDF19] Alexandre Duc, Stefan Dziembowski, and Sebastian Faust. Unifying leakage models: From probing attacks to noisy leakage. *J. Cryptology*, 32(1):151–177, 2019. 43, 77, I

[DDFP20] Gabriel Destouet, Cécile Dumas, Anne Frassati, and Valérie Perrier. Wavelet scattering transform and ensemble methods for side-channel analysis. *IACR Cryptol. ePrint Arch.*, 2020:310, 2020. 37, 72, 73

[DF12] Stefan Dziembowski and Sebastian Faust. Leakage-resilient circuits without computational assumptions. In Ronald Cramer, editor, *Theory of Cryptography - 9th Theory of Cryptography Conference, TCC 2012, Taormina, Sicily, Italy, March 19-21, 2012. Proceedings*, volume 7194 of *Lecture Notes in Computer Science*, pages 230–247. Springer, 2012. 43

[DFS16] Stefan Dziembowski, Sebastian Faust, and Maciej Skórski. Optimal amplification of noisy leakages. In Eyal Kushilevitz and Tal Malkin, editors, *Theory of Cryptography - 13th International Conference, TCC 2016-A, Tel Aviv, Israel, January 10-13, 2016, Proceedings, Part II*, volume 9563 of *Lecture Notes in Computer Science*, pages 291–318. Springer, 2016. 43, I

[DFS19] Alexandre Duc, Sebastian Faust, and François-Xavier Standaert. Making masking security proofs concrete (or how to evaluate the security of any leaking device), extended version. *J. Cryptology*, 32(4):1263–1297, 2019. 43, 77, I

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Trans. Inf. Theory*, 22(6):644–654, 1976. 2

[DLL+19] Simon S. Du, Jason D. Lee, Haochuan Li, Liwei Wang, and Xiyu Zhai. Gradient descent finds global minima of deep neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 1675–1685. PMLR, 2019. 68

[DPRS11] Julien Doget, Emmanuel Prouff, Matthieu Rivain, and François-Xavier Standaert. Univariate side channel attacks and leakage modeling. *J. Cryptographic Engineering*, 1(2):123–144, 2011. 35, 36

[DR02] Joan Daemen and Vincent Rijmen. AES and the wide trail design strategy. In *Advances in Cryptology - EUROCRYPT 2002, Proceedings*, pages 108–109, 2002. 51

[DRS+12] François Durvaux, Mathieu Renauld, François-Xavier Standaert, Loïc van Oldeneel tot Oldenzeel, and Nicolas Veyrat-Charvillon. Efficient removal of random delays from embedded software implementations using hidden markov models. In Stefan Mangard, editor, *Smart Card Research and Advanced Applications - 11th International Conference, CARDIS 2012, Graz, Austria, November 28-30, 2012, Revised Selected Papers*, volume 7771 of *Lecture Notes in Computer Science*, pages 123–140. Springer, 2012. 95, 97, 111

[DV16] Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning, 2016. 65, XIII

[DW19] Liron David and Avishai Wool. Fast analytical rank estimation. In Polian and Stöttinger [PS19], pages 168–190. 27

[DZPS19] Simon S. Du, Xiyu Zhai, Barnabás Póczos, and Aarti Singh. Gradient descent provably optimizes over-parameterized neural networks. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. 68

[EPW10] Thomas Eisenbarth, Christof Paar, and Björn Weghenkel. Building a side channel based disassembler. *Trans. Comput. Sci.*, 10:78–99, 2010. 37, 106

[FMPR10] Guillaume Fumaroli, Ange Martinelli, Emmanuel Prouff, and Matthieu Rivain. Affine masking against higher-order side channel analysis. In Alex Biryukov, Guang Gong, and Douglas R. Stinson, editors, *Selected Areas in Cryptography - 17th International Workshop, SAC 2010, Waterloo, Ontario, Canada, August 12-13, 2010, Revised Selected Papers*, volume 6544 of *Lecture Notes in Computer Science*, pages 262–280. Springer, 2010. 42

[GBC16] Ian J. Goodfellow, Yoshua Bengio, and Aaron C. Courville. *Deep Learning*. Adaptive computation and machine learning. MIT Press, 2016. 21, 22, 60, 62, 63, 118

[GBO19] Joey Green, Tilo Burghardt, and Elisabeth Oswald. Not a free lunch but a cheap lunch: Experimental results for training many neural nets. *IACR Cryptology ePrint Archive*, 2019:1068, 2019. 73

[GGP+15] Cezary Glowacz, Vincent Grosso, Romain Poussier, Joachim Schüth, and François-Xavier Standaert. Simpler and more efficient rank estimation for side-channel security assessment. In Gregor Leander, editor, *Fast Software Encryption - 22nd International Workshop, FSE 2015, Istanbul, Turkey, March 8-11, 2015, Revised Selected Papers*, volume 9054 of *Lecture Notes in Computer Science*, pages 117–129. Springer, 2015. 27

[GHO15] Richard Gilmore, Neil Hanley, and Máire O'Neill. Neural network based attack on a masked implementation of AES. In *IEEE International Symposium on Hardware Oriented Security and Trust, HOST 2015, Washington, DC, USA, 5-7 May, 2015*, pages 106–111. IEEE Computer Society, 2015. 71

[GJS19] Aron Gohr, Sven Jacob, and Werner Schindler. CHES 2018 side channel contest CTF - solution of the AES challenges. *IACR Cryptol. ePrint Arch.*, 2019:94, 2019. 25

[GJS20] Aron Gohr, Sven Jacob, and Werner Schindler. Efficient solutions of the CHES 2018 AES challenge using deep residual neural networks and knowledge distillation on adversarial examples. *IACR Cryptol. ePrint Arch.*, 2020:165, 2020. 25, 65, 98, 102

[GM11] Louis Goubin and Ange Martinelli. Protecting AES with shamir's secret sharing scheme. In Bart Preneel and Tsuyoshi Takagi, editors, *Cryptographic Hardware and Embedded Systems - CHES 2011 - 13th International Workshop, Nara, Japan, September 28 - October 1, 2011. Proceedings*, volume 6917 of *Lecture Notes in Computer Science*, pages 79–94. Springer, 2011. 43

[GMGH19] Christophe Genevey-Metat, Benoît Gerard, and Annelie Heuser. Combining sources of side-channel information. In *CAESAR 2019*, November 2019. 73

[GMO01] Karine Gandolfi, Christophe Mourtel, and Francis Olivier. Electromagnetic analysis: Concrete results. In Çetin Kaya Koç, David Naccache, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2001, Third International Workshop, Paris, France, May 14-16, 2001, Proceedings*, volume 2162 of *Lecture Notes in Computer Science*, pages 251–261. Springer, 2001. 6

[GP99] Louis Goubin and Jacques Patarin. DES and differential power analysis (the "duplication" method). In Çetin Kaya Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems, First International Workshop, CHES'99, Worcester, MA, USA, August 12-13, 1999, Proceedings*, volume 1717 of *Lecture Notes in Computer Science*, pages 158–172. Springer, 1999. 41

[GPAM+14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014. 10

[GPPT15] Daniel Genkin, Lev Pachmanov, Itamar Pipman, and Eran Tromer. Stealing keys from pcs using a radio: Cheap electromagnetic attacks on windowed exponentiation. In Tim Güneysu and Helena Handschuh, editors, *Cryptographic Hardware and Embedded Systems - CHES 2015 - 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings*, volume 9293 of *Lecture Notes in Computer Science*, pages 207–228. Springer, 2015. 6

[GPPT16] Daniel Genkin, Lev Pachmanov, Itamar Pipman, and Eran Tromer. ECDH key-extraction via low-bandwidth electromagnetic attacks on pcs. In Kazue Sako, editor, *Topics in Cryptology - CT-RSA 2016 - The Cryptographers' Track at the RSA Conference 2016, San Francisco, CA, USA, February 29 - March 4, 2016, Proceedings*, volume 9610 of *Lecture Notes in Computer Science*, pages 219–235. Springer, 2016. 6

[GPQ10] Laurie Genelle, Emmanuel Prouff, and Michaël Quisquater. Secure multiplicative masking of power functions. In Jianying Zhou and Moti Yung, editors, *Applied Cryptography and Network Security, 8th International Conference, ACNS 2010, Beijing, China, June 22-25, 2010. Proceedings*, volume 6123 of *Lecture Notes in Computer Science*, pages 200–217, 2010. 42

[GPT15] Daniel Genkin, Itamar Pipman, and Eran Tromer. Get your hands off my laptop: physical side-channel key-extraction attacks on pcs - extended version. *J. Cryptographic Engineering*, 5(2):95–112, 2015. 6

[GR15] Shafi Goldwasser and Guy N. Rothblum. How to compute in the presence of leakage. *SIAM J. Comput.*, 44(5):1480–1549, 2015. 43

[GSS15] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. 106

[GST14] Daniel Genkin, Adi Shamir, and Eran Tromer. RSA key extraction via low-bandwidth acoustic cryptanalysis. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 444–461. Springer, 2014. 6

[GT02] Jovan Dj. Golic and Christophe Tymen. Multiplicative masking and power analysis of AES. In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*, volume 2523 of *Lecture Notes in Computer Science*, pages 198–212. Springer, 2002. 41

[GTdS+18] Rafael Garcia, Alexandru C. Telea, Bruno Castro da Silva, Jim Tørresen, and João Luiz Dihl Comba. A task-and-technique centered survey on visual analytics for deep learning model engineering. *Comput. Graph.*, 77:30–49, 2018. 72

[HGG19] Benjamin Hettwer, Stefan Gehrer, and Tim Güneysu. Deep neural network attribution methods for leakage analysis and symmetric key recovery. In Kenneth G. Paterson and Douglas Stebila, editors, *Selected Areas in Cryptography - SAC 2019 - 26th International Conference, Waterloo, ON, Canada, August 12-16, 2019, Revised Selected Papers*, volume 11959 of *Lecture Notes in Computer Science*, pages 645–666. Springer, 2019. 72, 112

[HGG20] Benjamin Hettwer, Stefan Gehrer, and Tim Güneysu. Applications of machine learning techniques in side-channel attacks: a survey. *J. Cryptogr. Eng.*, 10(2):135–162, 2020. 60, 71

[HGMG20] Annelie Heuser, Christophe Genevey-Metat, and Benoit Gerard. Physical side-channel analysis on stm32f$\{0, 1, 2, 3, 4\}$. https://silm.inria.fr/silm-seminar, 2020. 52

[HHO20] Anh-Tuan Hoang, Neil Hanley, and Maire O'Neill. Plaintext: A missing feature for enhancing the power of deep learning in side-channel analysis? breaking multiple layers of side-channel countermeasures. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020(4):49–85, Aug 2020. 126

[HKPC19] Fred Hohman, Minsuk Kahng, Robert Pienta, and Duen Horng Chau. Visual analytics in deep learning: An interrogative survey for the next frontiers. *IEEE Trans. Vis. Comput. Graph.*, 25(8):2674–2693, 2019. 72

[HLW16] Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. Densely connected convolutional networks. *CoRR*, abs/1608.06993, 2016. 65

[HOM06] Christoph Herbst, Elisabeth Oswald, and Stefan Mangard. An AES smart card implementation resistant to power analysis attacks. In Jianying Zhou, Moti Yung, and Feng Bao, editors, *Applied Cryptography and Network Security, 4th International Conference, ACNS 2006, Singapore, June 6-9, 2006, Proceedings*, volume 3989 of *Lecture Notes in Computer Science*, pages 239–252, 2006. 44

[Hor91] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257, 1991. 109

[HRG14] Annelie Heuser, Olivier Rioul, and Sylvain Guilley. Good is not good enough - deriving optimal distinguishers from communication theory. In Batina and Robshaw [BR14], pages 55–74. 30, 33, 35, 96

[HTF09] Trevor Hastie, Robert Tibshirani, and Jerome H. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, 2nd Edition*. Springer Series in Statistics. Springer, 2009. 33

[HZ12] Annelie Heuser and Michael Zohner. Intelligent machine homicide - breaking cryptographic devices using support vector machines. In Werner Schindler and Sorin A. Huss, editors, *Constructive Side-Channel Analysis and Secure Design - Third International Workshop, COSADE 2012, Darmstadt, Germany, May 3-4, 2012. Proceedings*, volume 7275 of *Lecture Notes in Computer Science*, pages 249–264. Springer, 2012. 71

[HZF+19] Yongbo Hu, Yeyang Zheng, Pengwei Feng, Lirui Liu, Chen Zhang, Aron Gohr, Sven Jacob, Werner Schindler, Ileana Buhan, and Karim Tobich. Machine learning and side channel analysis in a CTF competition. *IACR Cryptol. ePrint Arch.*, 2019:860, 2019. 25

[HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society, 2016. 65, 98, 102, X

[IS15] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis R. Bach and David M. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 448–456. JMLR.org, 2015. 62

[ISW03] Yuval Ishai, Amit Sahai, and David A. Wagner. Private circuits: Securing hardware against probing attacks. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, volume 2729 of *Lecture Notes in Computer Science*, pages 463–481. Springer, 2003. 42

[KB07] Boris Köpf and David A. Basin. An information-theoretic model for adaptive side-channel attacks. In Peng Ning, Sabrina De Capitani di Vimercati, and Paul F. Syverson, editors, *Proceedings of the 2007 ACM Conference on Computer and Communications Security, CCS 2007, Alexandria, Virginia, USA, October 28-31, 2007*, pages 286–296. ACM, 2007. 27

[KB11] Boris Köpf and David A. Basin. Automatically deriving information-theoretic bounds for adaptive side-channel attacks. *J. Comput. Secur.*, 19(1):1–31, 2011. 27

[KB15] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. 22

[KHF+19] Paul Kocher, Jann Horn, Anders Fogh, , Daniel Genkin, Daniel Gruss, Werner Haas, Mike Hamburg, Moritz Lipp, Stefan Mangard, Thomas Prescher, Michael Schwarz, and Yuval Yarom. Spectre attacks: Exploiting speculative execution. In *40th IEEE Symposium on Security and Privacy (S&P'19)*, 2019. 6

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael J. Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999. 6, 35, 36

[Kle13] A. Klenke. *Probability Theory: A Comprehensive Course*. Universitext. Springer London, 2013. 14, 15, 19

[Klo59] B. M. Kloss. Probability distributions on bicompact topological groups. *Teor. Veroyatnost. i Primenen.*, 4(3):255–290, 1959. III

[Koc96] Paul C. Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In Neal Koblitz, editor, *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer, 1996. 6

[KPH+19] 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. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2019(3):148–179, May 2019. 67, 71, 72, 73, 89, 90, 91, 95, 96, 97, 98, 99

[KR19] Yael Tauman Kalai and Leonid Reyzin. A survey of leakage-resilient cryptography. In Oded Goldreich, editor, *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*, pages 727–794. ACM, 2019. 39

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012. 9

[LB94] Yann LeCun and Yoshua Bengio. Word-level training of a handwritten word recognizer based on convolutional neural networks. In *12th IAPR International Conference on Pattern Recognition, Conference B: Patern Recognition and Neural Networks, ICPR 1994, Jerusalem, Israel, 9-13 October, 1994, Volume 2*, pages 88–92. IEEE, 1994. 64

[LBM14] Liran Lerman, Gianluca Bontempi, and Olivier Markowitch. Power analysis attack: an approach based on machine learning. *IJACT*, 3(2):97–115, 2014. 71

[LBM15] Liran Lerman, Gianluca Bontempi, and Olivier Markowitch. A machine learning approach against a masked AES - reaching the limit of side-channel attacks with a learning model. *J. Cryptographic Engineering*, 5(2):123–139, 2015. 71

[LBOM12] Yann LeCun, Léon Bottou, Genevieve B. Orr, and Klaus-Robert Müller. Efficient backprop. In Grégoire Montavon, Genevieve B. Orr, and Klaus-Robert Müller, editors, *Neural Networks: Tricks of the Trade - Second Edition*, volume 7700 of *Lecture Notes in Computer Science*, pages 9–48. Springer, 2012. 68

[LCC+06] Thanh-Ha Le, Jessy Clédière, Cécile Canovas, Bruno Robisson, Christine Servière, and Jean-Louis Lacoume. A proposition for correlation power analysis enhancement. In Louis Goubin and Mitsuru Matsui, editors, *Cryptographic Hardware and Embedded Systems - CHES 2006, 8th International Workshop, Yokohama, Japan, October 10-13, 2006, Proceedings*, volume 4249 of *Lecture Notes in Computer Science*, pages 174–186. Springer, 2006. 35

[LFS18] David Leech, Stacey Ferris, and John Scott. The economic impacts of the advanced encryption standard, 1996-2017. Technical report, National Institute for Standard and Technology, 2018. 3

[LPR+14] Victor Lomné, Emmanuel Prouff, Matthieu Rivain, Thomas Roche, and Adrian Thillard. How to estimate the success rate of higher-order side-channel attacks. In Batina and Robshaw [BR14], pages 35–54. 84, I

[LSJR16] Jason D. Lee, Max Simchowitz, Michael I. Jordan, and Benjamin Recht. Gradient descent only converges to minimizers. In Vitaly Feldman, Alexander Rakhlin, and Ohad Shamir, editors, *29th Annual Conference on Learning Theory*, volume 49 of *Proceedings of Machine Learning Research*, pages 1246–1257, Columbia University, New York, New York, USA, 23–26 Jun 2016. PMLR. 21

[Mag19] Houssem Maghrebi. Deep learning based side channel attacks in practice. *IACR Cryptology ePrint Archive*, 2019:578, 2019. 71, 72, 73

[Mag20] Houssem Maghrebi. Deep learning based side-channel attack: a new profiling methodology based on multi-label classification. *IACR Cryptol. ePrint Arch.*, 2020:436, 2020. 73, 127

[Man04] Stefan Mangard. Hardware countermeasures against DPA ? A statistical analysis of their effectiveness. In Tatsuaki Okamoto, editor, *Topics in Cryptology - CT-RSA 2004, The Cryptographers' Track at the RSA Conference 2004, San Francisco, CA, USA, February 23-27, 2004, Proceedings*, volume 2964 of *Lecture Notes in Computer Science*, pages 222–235. Springer, 2004. 45, 77, 96

[MBC+20] Loïc Masure, Nicolas Belleville, Eleonora Cagli, Marie-Angela Cornelie, Damien Couroussé, Cécile Dumas, and Laurent Maingault. Deep learning side-channel analysis on large-scale traces - A case study on a polymorphic AES. In Liqun Chen, Ninghui Li, Kaitai Liang, and Steve A. Schneider, editors, *Computer Security - ESORICS 2020 - 25th European Symposium on Research in Computer Security, ESORICS 2020, Guildford, UK, September 14-18, 2020, Proceedings, Part I*, volume 12308 of *Lecture Notes in Computer Science*, pages 440–460. Springer, 2020. 93

[MDM16] Zdenek Martinasek, Petr Dzurenda, and Lukas Malina. Profiling power analysis attack based on MLP in DPA contest V4.2. In *39th International Conference on Telecommunications and Signal Processing, TSP 2016, Vienna, Austria, June 27-29, 2016*, pages 223–226. IEEE, 2016. 71

[MDP18] Loïc Masure, Cécile Dumas, and Emmanuel Prouff. Understanding a cnn attack: as crucial as succeeding in it, 2018. 105

[MDP19a] Loïc Masure, Cécile Dumas, and Emmanuel Prouff. Gradient visualization for general characterization in profiling attacks. In Polian and Stöttinger [PS19], pages 145–167. 72, 105

[MDP19b] Loïc Masure, Cécile Dumas, and Emmanuel Prouff. A comprehensive study of deep learning for side-channel analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020(1):348–375, Nov 2019. 47, 73, 75

[Mes00] Thomas S. Messerges. Securing the AES finalists against power analysis attacks. In Bruce Schneier, editor, *Fast Software Encryption, 7th International Workshop, FSE 2000, New York, NY, USA, April 10-12, 2000, Proceedings*, volume 1978 of *Lecture Notes in Computer Science*, pages 150–164. Springer, 2000. 41

[Mit97] Tom M. Mitchell. *Machine learning, International Edition*. McGraw-Hill Series in Computer Science. McGraw-Hill, 1997. 56

[MMO18] Daniel P. Martin, Luke Mather, and Elisabeth Oswald. Two sides of the same coin: Counting and enumerating keys post side-channel attacks revisited. In Nigel P. Smart, editor, *Topics in Cryptology - CT-RSA 2018 - The Cryptographers' Track at the RSA Conference 2018, San Francisco, CA, USA, April 16-20, 2018, Proceedings*, volume 10808 of *Lecture Notes in Computer Science*, pages 394–412. Springer, 2018. 27

[MMOS16] Daniel P. Martin, Luke Mather, Elisabeth Oswald, and Martijn Stam. Characterisation and estimation of the key rank distribution in the context of side channel evaluations. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I*, volume 10031 of *Lecture Notes in Computer Science*, pages 548–572, 2016. 27

[MOBW13] Luke Mather, Elisabeth Oswald, Joe Bandenburg, and Marcin Wójcik. Does my device leak information? an a priori statistical power analysis of leakage detection tests. In *Advances in Cryptology - ASIACRYPT 2013 - Proceedings, Part I*, pages 486–505, 2013. 38, 96

[MOOS15] Daniel P. Martin, Jonathan F. O'Connell, Elisabeth Oswald, and Martijn Stam. Counting keys in parallel after a side channel attack. In Tetsu Iwata and Jung Hee Cheon, editors, *Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part II*, volume 9453 of *Lecture Notes in Computer Science*, pages 313–337. Springer, 2015. 25

[MOP07] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power analysis attacks - revealing the secrets of smart cards*. Springer, 2007. 28, 36, 38, 39, 44, 77, 96, 99, 100

[Mor74] Roland Moreno. Methods of data storage and data storage systems. US3971916A, 1974. 3

[MOS11] Stefan Mangard, Elisabeth Oswald, and François-Xavier Standaert. One for all - all for one: unifying standard differential power analysis attacks. *IET Inf. Secur.*, 5(2):100–110, 2011. 77

[MPP16] Houssem Maghrebi, Thibault Portigliatti, and Emmanuel Prouff. Breaking cryptographic implementations using deep learning techniques. In Claude Carlet, M. Anwar Hasan, and Vishal Saraswat, editors, *Security, Privacy, and Applied Cryptography Engineering - 6th International Conference, SPACE 2016, Hyderabad, India, December 14-18, 2016, Proceedings*, volume 10076 of *Lecture Notes in Computer Science*, pages 3–26. Springer, 2016. 9, 67, 71, 73, 97

[MSM18] Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. Methods for interpreting and understanding deep neural networks. *Digital Signal Processing*, 73:1–15, 2018. 106, 111

[MW20a] Merriam-Webster. "hazardous". In *Merriam-Webster.com dictionary*. August 2020. 8

[MW20b] Merriam-Webster. "safety". In *Merriam-Webster.com dictionary*. August 2020. 8

[MW20c] Merriam-Webster. "security". In *Merriam-Webster.com dictionary*. August 2020. 8

[MZ13] Zdenek Martinasek and Vaclav Zeman. Innovative method of the power analysis. *Radioengineering*, 22:586–594, 06 2013. 9, 71

[Nat01] National Institute of Standards and Technology. Advanced encryption standard (AES). Technical Report NIST FIPS 197, National Institute of Standards and Technology, 2001. 4, 20

[NHI+07] Sei Nagashima, Naofumi Homma, Yuichi Imai, Takafumi Aoki, and Akashi Satoh. DPA using phase-based waveform matching against random-delay countermeasure. In *International Symposium on Circuits and Systems (ISCAS 2007), 27-20 May 2007, New Orleans, Louisiana, USA*, pages 1807–1810. IEEE, 2007. 95, 97, 111

[Nie18] Michael A. Nielsen. Neural networks and deep learning. http://neuralnetworksanddeeplearning.com/, 2018. 67

[NSGD12] Maxime Nassar, Youssef Souissi, Sylvain Guilley, and Jean-Luc Danger. RSM: A small and fast countermeasure for AES, secure against 1st and 2nd-order zero-offset SCAs. In *DATE*, 2012. 95

[OC14] Colin O'Flynn and Zhizhang (David) Chen. Chipwhisperer: An open-source platform for hardware embedded security research. In Emmanuel Prouff, editor, *Constructive Side-Channel Analysis and Secure Design - 5th International Workshop, COSADE 2014, Paris, France, April 13-15, 2014. Revised Selected Papers*, volume 8622 of *Lecture Notes in Computer Science*, pages 243–260. Springer, 2014. 47

[PBP20] Guilherme Perin, Ileana Buhan, and Stjepan Picek. Learning when to stop: a mutual information approach to fight overfitting in profiled side-channel analysis. *IACR Cryptol. ePrint Arch.*, 2020:58, 2020. 71

[Pet98] P. Petrushev. Approximation by ridge functions and neural networks. *SIAM Journal on Mathematical Analysis*, 30(1):155–189, 1998. 63

[PGM+19] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pages 8024–8035, 2019. 70, 109

[PGMP19] Thomas Prest, Dahmun Goudarzi, Ange Martinelli, and Alain Passelègue. Unifying leakage models on a rényi day. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part I*, volume 11692 of *Lecture Notes in Computer Science*, pages 683–712. Springer, 2019. 43, I, III

[PHG17] Stjepan Picek, Annelie Heuser, and Sylvain Guilley. Template attack versus bayes classifier. *J. Cryptographic Engineering*, 7(4):343–351, 2017. 33

[PHG19] Stjepan Picek, Annelie Heuser, and Sylvain Guilley. Profiling side-channel analysis in the restricted attacker framework. *IACR Cryptology ePrint Archive*, 2019:168, 2019. 32, 73

[PHJ+18] 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. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2019(1):209–237, Nov 2018. 50, 67, 70, 71, 72, 74, 91, 125

[Pin99] Allan Pinkus. Approximation theory of the MLP model in neural networks. *Acta Numerica*, 8:143–195, 1999. 64, 109

[Pou18] Romain Poussier. *Key enumeration, rank estimation and horizontal side-channel attacks*. PhD thesis, Catholic University of Louvain, Louvain-la-Neuve, Belgium, 2018. 25

[PR07] Emmanuel Prouff and Matthieu Rivain. A generic method for secure sbox implementation. In Sehun Kim, Moti Yung, and Hyung-Woo Lee, editors, *Information Security Applications, 8th International Workshop, WISA 2007, Jeju Island, Korea, August 27-29, 2007, Revised Selected Papers*, volume 4867 of *Lecture Notes in Computer Science*, pages 227–244. Springer, 2007. 42

[PR10] Emmanuel Prouff and Matthieu Rivain. Theoretical and practical aspects of mutual information-based side channel analysis. *IJACT*, 2(2):121–138, 2010. 76

[PR11] Emmanuel Prouff and Thomas Roche. Higher-order glitches free implementation of the AES using secure multi-party computation protocols. In Bart Preneel and Tsuyoshi Takagi, editors, *Cryptographic Hardware and Embedded Systems - CHES 2011 - 13th International Workshop, Nara, Japan, September 28 - October 1, 2011. Proceedings*, volume 6917 of *Lecture Notes in Computer Science*, pages 63–78. Springer, 2011. 43

[PR13] Emmanuel Prouff and Matthieu Rivain. Masking against side-channel attacks: A formal security proof. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, volume 7881 of *Lecture Notes in Computer Science*, pages 142–159. Springer, 2013. 43, 77, I

[PRB09] Emmanuel Prouff, Matthieu Rivain, and Régis Bevan. Statistical analysis of second order differential power analysis. *IEEE Trans. Computers*, 58(6):799–811, 2009. 39

[PS19] Ilia Polian and Marc Stöttinger, editors. *Constructive Side-Channel Analysis and Secure Design - 10th International Workshop, COSADE 2019, Darmstadt, Germany, April 3-5, 2019, Proceedings*, volume 11421 of *Lecture Notes in Computer Science*. Springer, 2019. 135, 141

[PSK+18] 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 Anupam Chattopadhyay, Chester Rebeiro, and Yuval Yarom, editors, *Security, Privacy, and Applied Cryptography Engineering - 8th International Conference, SPACE 2018, Kanpur, India, December 15-19, 2018, Proceedings*, volume 11348 of *Lecture Notes in Computer Science*, pages 157–176. Springer, 2018. 60

[QS01] Jean-Jacques Quisquater and David Samyde. Electromagnetic analysis (EMA): measures and counter-measures for smart cards. In Isabelle Attali and Thomas P. Jensen, editors, *Smart Card Programming and Security, International Conference on Research in Smart Cards, E-smart 2001, Cannes, France, September 19-21, 2001, Proceedings*, volume 2140 of *Lecture Notes in Computer Science*, pages 200–210. Springer, 2001. 6

[RAD20] Keyvan Ramezanpour, Paul Ampadu, and William Diehl. SCAUL: power side-channel analysis with unsupervised learning. *CoRR*, abs/2001.05951, 2020. 72

[Ren00] M Renaudin. Asynchronous circuits and systems : a promising design alternative. *Microelectronic Engineering*, 54(1):133 – 149, 2000. 44

[RHW86a] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning internal representations by error propagation. In David E. Rumelhart and James L. Mcclelland, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations*, pages 318–362. MIT Press, Cambridge, MA, 1986. 70

[RHW86b] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning Representations by Back-propagating Errors. *Nature*, 323(6088):533–536, 1986. 70

[RP10] Matthieu Rivain and Emmanuel Prouff. Provably secure higher-order masking of AES. In Stefan Mangard and François-Xavier Standaert, editors, *Cryptographic Hardware and Embedded Systems, CHES 2010, 12th International Workshop, Santa Barbara, CA, USA, August 17-20, 2010. Proceedings*, volume 6225 of *Lecture Notes in Computer Science*, pages 413–427. Springer, 2010. 42

[RPD09] Matthieu Rivain, Emmanuel Prouff, and Julien Doget. Higher-order masking and shuffling for software implementations of block ciphers. In Clavier and Gaj [CG09], pages 171–188. 44

[RS09] Mathieu Renauld and François-Xavier Standaert. Algebraic side-channel attacks. In Feng Bao, Moti Yung, Dongdai Lin, and Jiwu Jing, editors, *Information Security and Cryptology - 5th International Conference, Inscrypt 2009, Beijing, China, December 12-15, 2009. Revised Selected Papers*, volume 6151 of *Lecture Notes in Computer Science*, pages 393–410. Springer, 2009. 25

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key crypto-systems. *Commun. ACM*, 21(2):120–126, 1978. 2

[RSV09] Mathieu Renauld, François-Xavier Standaert, and Nicolas Veyrat-Charvillon. Algebraic side-channel attacks on the AES: why time also matters in DPA. In Christophe Clavier and Kris Gaj, editors, *Cryptographic Hardware and Embedded Systems - CHES 2009, 11th International Workshop, Lausanne, Switzerland, September 6-9, 2009, Proceedings*, volume 5747 of *Lecture Notes in Computer Science*, pages 97–111. Springer, 2009. 25

[RSV+11] Mathieu Renauld, François-Xavier Standaert, Nicolas Veyrat-Charvillon, Dina Kamel, and Denis Flandre. A formal study of power variability issues and side-channel attacks for nanoscale devices. In Kenneth G. Paterson, editor, *Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings*, volume 6632 of *Lecture Notes in Computer Science*, pages 109–128. Springer, 2011. 76, 78

[RZC+20] Damien Robissout, Gabriel Zaid, Brice Colombier, Lilian Bossuet, and Amaury Habrard. Online performance evaluation of deep learning networks for side-channel analysis. *IACR Cryptol. ePrint Arch.*, 2020:39, 2020. 71

[SA08] François-Xavier Standaert and Cédric Archambeau. Using subspace-based template attacks to compare and combine power and electromagnetic information leakages. In Elisabeth Oswald and Pankaj Rohatgi, editors, *Cryptographic Hardware and Embedded Systems - CHES 2008, 10th International Workshop, Washington, D.C., USA, August 10-13, 2008. Proceedings*, volume 5154 of *Lecture Notes in Computer Science*, pages 411–425. Springer, 2008. 33, 37, 95, 106

[SB98] Richard S. Sutton and Andrew G. Barto. *Reinforcement learning - an introduction*. Adaptive computation and machine learning. MIT Press, 1998. 26, 27, 57

[SDBR15] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin A. Riedmiller. Striving for simplicity: The all convolutional net. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Workshop Track Proceedings*, 2015. 106, 111

[Sha79] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979. 43

[SHK+14] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014. 62

[Sho94] Peter W. Shor. Polynominal time algorithms for discrete logarithms and factoring on a quantum computer. In Leonard M. Adleman and Ming-Deh A. Huang, editors, *Algorithmic Number Theory, First International Symposium, ANTS-I, Ithaca, NY, USA, May 6-9, 1994, Proceedings*, volume 877 of *Lecture Notes in Computer Science*, page 289. Springer, 1994. 2

[Sin99] Simon Singh. *The Code Book: The Evolution of Secrecy from Mary, Queen of Scots, to Quantum Cryptography*. Doubleday, New York, NY, USA, 1st edition, 1999. 2, 3

[SLJ+15] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 1–9. IEEE Computer Society, 2015. 65

[SLP05] Werner Schindler, Kerstin Lemke, and Christof Paar. A stochastic model for differential side channel cryptanalysis. In Josyula R. Rao and Berk Sunar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2005, 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005, Proceedings*, volume 3659 of *Lecture Notes in Computer Science*, pages 30–46. Springer, 2005. 36, 72, VI

[Sma20] Small portable AES128 in C. https://github.com/kokke/tiny-AES-c, 2020. 52

[SMY09] François-Xavier Standaert, Tal Malkin, and Moti Yung. A unified framework for the analysis of side-channel key recovery attacks. In Antoine Joux, editor, *Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings*, volume 5479 of *Lecture Notes in Computer Science*, pages 443–461. Springer, 2009. 29

[SS06] Daisuke Suzuki and Minoru Saeki. Security evaluation of DPA countermeasures using dual-rail pre-charge logic style. In Louis Goubin and Mitsuru Matsui, editors, *Cryptographic Hardware and Embedded Systems - CHES 2006, 8th International Workshop, Yokohama, Japan, October 10-13, 2006, Proceedings*, volume 4249 of *Lecture Notes in Computer Science*, pages 255–269. Springer, 2006. 44

[SSBD14] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014. 21, 55, 57, 58, 59, 64, 67, 79, 95

[Sta18] François-Xavier Standaert. How (not) to use welch's t-test in side-channel security evaluations. In Begül Bilgin and Jean-Bernard Fischer, editors, *Smart Card Research and Advanced Applications, 17th International Conference, CARDIS 2018, Montpellier, France, November 12-14, 2018, Revised Selected Papers*, volume 11389 of *Lecture Notes in Computer Science*, pages 65–79. Springer, 2018. 38

[STIM18] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*, pages 2488–2498, 2018. 62

[STM] STMicroelectronics. NUCLEO-F303RE. https://www.st.com/content/st_com/en/products/evaluation-tools/product-evaluation-tools/mcu-mpu-eval-tools/stm32-mcu-mpu-eval-tools/stm32-nucleo-boards/nucleo-f303re.html. 52

[SVI+15] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567, 2015. 65

[SVZ14] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Workshop Track Proceedings*, 2014. 106, 111

[SZ15] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. 64, 102

[SZS+14] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. 106

[Tel16] Matus Telgarsky. Benefits of depth in neural networks. In Vitaly Feldman, Alexander Rakhlin, and Ohad Shamir, editors, *29th Annual Conference on Learning Theory*, volume 49 of *Proceedings of Machine Learning Research*, pages 1517–1539, Columbia University, New York, New York, USA, 23–26 Jun 2016. PMLR. 64

[Ter18] Audrey Terras. *Abstract Algebra with Applications*. Cambridge Mathematical Textbooks. Cambridge University Press, 2018. 20, 42, X

[The16] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016. 70

[Tim19] Benjamin Timon. Non-profiled deep learning-based side-channel attacks with sensitivity analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2019(2):107–131, Feb 2019. 67, 72, 73, 95, 112

[Ugo77] Michel Ugon. Portable data carrier including a microprocessor. US4211919A, 1977. 3

[Vap95] V. Vapnik. *The Nature of Statistical Learning Theory*. Information Science and Statistics. Springer New York, 1995. 58, 87

[Vap99] Vladimir Vapnik. An overview of statistical learning theory. *IEEE Trans. Neural Networks*, 10(5):988–999, 1999. 59, 60

[Vap00] Vladimir Vapnik. *The Nature of Statistical Learning Theory*. Statistics for Engineering and Information Science. Springer, 2000. 33, 75

[vdVP19] Daan van der Valk and Stjepan Picek. Bias-variance decomposition in machine learning-based side-channel analysis. *IACR Cryptology ePrint Archive*, 2019:570, 2019. 67

[vdVPB19] 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. *IACR Cryptol. ePrint Arch.*, 2019:1477, 2019. 72, 112

[VGRS12] Nicolas Veyrat-Charvillon, Benoît Gérard, Mathieu Renauld, and François-Xavier Standaert. An optimal key enumeration algorithm and its application to side-channel attacks. In Lars R. Knudsen and Huapeng Wu, editors, *Selected Areas in Cryptography, 19th International Conference, SAC 2012, Windsor, ON, Canada, August 15-16, 2012, Revised Selected Papers*, volume 7707 of *Lecture Notes in Computer Science*, pages 390–406. Springer, 2012. 25

[VGS13] Nicolas Veyrat-Charvillon, Benoît Gérard, and François-Xavier Standaert. Security evaluations beyond computing power. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, volume 7881 of *Lecture Notes in Computer Science*, pages 126–141. Springer, 2013. 27

[VMKS12] Nicolas Veyrat-Charvillon, Marcel Medwed, Stéphanie Kerckhof, and François-Xavier Standaert. Shuffling against side-channel attacks: A comprehensive study with cautionary note. In Xiaoyun Wang and Kazue Sako, editors, *Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings*, volume 7658 of *Lecture Notes in Computer Science*, pages 740–757. Springer, 2012. 44, 45, 84, 85

[VS10] Nicolas Veyrat-Charvillon and François-Xavier Standaert. Adaptive chosen-message side-channel attacks. In Jianying Zhou and Moti Yung, editors, *Applied Cryptography and Network Security, 8th International Conference, ACNS 2010, Beijing, China, June 22-25, 2010. Proceedings*, volume 6123 of *Lecture Notes in Computer Science*, pages 186–199, 2010. 27

[vW01] Manfred von Willich. A technique with an information-theoretic basis for protecting secret data from differential power attacks. In Bahram Honary, editor, *Cryptography and Coding, 8th IMA International Conference, Cirencester, UK, December 17-19, 2001, Proceedings*, volume 2260 of *Lecture Notes in Computer Science*, pages 44–62. Springer, 2001. 42

[vWWB11] Jasper G. J. van Woudenberg, Marc F. Witteman, and Bram Bakker. Improving differential power analysis by elastic alignment. In Aggelos Kiayias, editor, *Topics in Cryptology - CT-RSA 2011 - The Cryptographers' Track at the RSA Conference 2011, San Francisco, CA, USA, February 14-18, 2011. Proceedings*, volume 6558 of *Lecture Notes in Computer Science*, pages 104–119. Springer, 2011. 95, 97, 111

[WAGP20] Lennert Wouters, Victor Arribas, Benedikt Gierlichs, and Bart Preneel. Revisiting a methodology for efficient cnn architectures in profiling attacks. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020(3):147–168, Jun 2020. 112

[WD20] Huanyu Wang and Elena Dubrova. Tandem deep learning side-channel attack against FPGA implementation of AES. *IACR Cryptol. ePrint Arch.*, 2020:373, 2020. 73

[Wik19] Wikipedia. Sensitivity analysis, 2019. 106

[WMCS20] Weijia Wang, Pierrick Méaux, Gaëtan Cassiers, and François-Xavier Standaert. Efficient and private computations with code-based masking. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020(2):128–171, Mar 2020. 43

[WMM19] Felix Wegener, Thorben Moos, and Amir Moradi. DL-LA: deep learning leakage assessment: A modern roadmap for SCA evaluations. *IACR Cryptology ePrint Archive*, 2019:505, 2019. 67

[WP19] Lichao Wu and Stjepan Picek. Remove some noise: On pre-processing of side-channel measurements with autoencoders. *IACR Cryptol. ePrint Arch.*, 2019:1474, 2019. 72

[WPB19] Leo Weissbart, Stjepan Picek, and Lejla Batina. One trace is all it takes: Machine learning-based side-channel attack on EdDSA. In Shivam Bhasin, Avi Mendelson, and Mridul Nandi, editors, *Security, Privacy, and Applied Cryptography Engineering - 9th International Conference, SPACE 2019, Gandhinagar, India, December 3-7, 2019, Proceedings*, volume 11947 of *Lecture Notes in Computer Science*, pages 86–105. Springer, 2019. 71

[WYS+18] Weijia Wang, Yu Yu, François-Xavier Standaert, Junrong Liu, Zheng Guo, and Dawu Gu. Ridge-based DPA: improvement of differential power analysis for nanoscale chips. *IEEE Trans. Inf. Forensics Secur.*, 13(5):1301–1316, 2018. 72

[YEM14] Xin Ye, Thomas Eisenbarth, and William Martin. Bounded, yet sufficient? how to determine whether limited side channel information enables key recovery. In Marc Joye and Amir Moradi, editors, *Smart Card Research and Advanced Applications - 13th International Conference, CARDIS 2014, Paris, France, November 5-7, 2014. Revised Selected Papers*, volume 8968 of *Lecture Notes in Computer Science*, pages 215–232. Springer, 2014. 27

[YLMZ18] Guang Yang, Huizhong Li, Jingdian Ming, and Yongbin Zhou. Convolutional neural network based side-channel attacks in time-frequency representations. In Begül Bilgin and Jean-Bernard Fischer, editors, *Smart Card Research and Advanced Applications, 17th International Conference, CARDIS 2018, Montpellier, France, November 12-14, 2018, Revised Selected Papers*, volume 11389 of *Lecture Notes in Computer Science*, pages 1–17. Springer, 2018. 72

[ZBD+20] Gabriel Zaid, Lilian Bossuet, François Dassance, Amaury Habrard, and Alexandre Venelli. Ranking loss: Maximizing the success rate in deep learning side-channel analysis. Cryptology ePrint Archive, Report 2020/872, 2020. https://eprint.iacr.org/. 92

[ZBHV19] Gabriel Zaid, Lilian Bossuet, Amaury Habrard, and Alexandre Venelli. Methodology for efficient cnn architectures in profiling attacks. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020(1):1–36, Nov 2019. 72, 90, 95, 97, 98, 99, 112

[ZF14] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In David J. Fleet, Tomás Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I*, volume 8689 of *Lecture Notes in Computer Science*, pages 818–833. Springer, 2014. 106, 111

[ZKL+16] Bolei Zhou, Aditya Khosla, Àgata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 2921–2929. IEEE Computer Society, 2016. 98, 113

[ZS19] Yuanyuan Zhou and François-Xavier Standaert. Deep learning mitigates but does not annihilate the need of aligned traces and a generalized ResNet model for side-channel attacks. *Journal of Cryptographic Engineering*, April 2019. 65, 72, 73, 98, 102

[ZZN+20] Jiajia Zhang, Mengce Zheng, Jiehui Nan, Honggang Hu, and Nenghai Yu. A novel evaluation metric for deep learning-based side channel analysis and its extended application to imbalanced data. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020(3):73–96, Jun 2020. 92

Appendix A

Noise Amplification of Secret-Sharing

A long series of papers have been published since the seminal work of Chari *et al.* at CRYPTO 1999 [CJRR99] to show the noise amplification effect of group-based secret-sharing. The latter result has been extended by Prouff *et al.* [PR13] and Duc *et al.* [DDF19, DFS19, DFS16] until the most recent works of Prest *et al.* at CRYPTO 2019 [PGMP19].

All these papers embrace different strategies to show the theoretical soudness of group-based secret-sharing, but most of them rely on the so-called *noise amplification* effect. In a nutshell, it states that one can produce an *artificial* noise in the leaky observations of a sensitive variable Z protected with a d-th order group-based secret-sharing of amplitude $\mathcal{O}\left(\sigma^d\right)$, where σ characterizes the noise of the target device without any counter-measure.

We provide in this appendix a proof sketch of such a result.[1] We first recall that the leakage distribution of a sensitive random variable protected with group-based secret-sharing can be formulated as a convolution product. This observation enables to benefit from the properties of convolutions to reach the result.

A.1 The Link between Noise Amplification and Convolution

We start by remarking that applying group-based secret-sharing can be seen as applying a discrete convolution to the unprotected leakage model.

Proposition 3 ([LPR+14, Sec. 6]). *Let $\mathsf{Z} \in \mathcal{Z}$ be a sensitive target variable, protected by a d-th order secret-sharing with shares $\mathsf{Z}_0, \ldots, \mathsf{Z}_d \in \mathcal{Z}$. Let $\mathbf{X} = (\mathsf{X}_0, \ldots, \mathsf{X}_d)^\intercal$, and $\mathbf{x} = (x_0, \ldots, x_d)^\intercal$ be an observation of $\mathbf{X}$. Let $h : s \mapsto \Pr(\mathsf{Z} = s \mid \mathbf{X} = \mathbf{x})$ be the posterior p.m.f. of the sensitive target variable, while $h_i : s \mapsto \Pr(\mathsf{Z}_i = s \mid \mathsf{X}_i = x_i)$ denotes the posterior p.m.f. associated to the share Z_i. Assume that the following claims hold:*

(a) The random variables Z and $(\mathsf{Z}_i)_{i \in [\![1,d]\!]}$ are i.i.d. uniformly from the group $(\mathcal{Z}, \cdot)$;

(b) Any X_i only depend on Z_i, i.e., X_i denotes the leakage of the share Z_i. In particular, any X_i is independent of the $(\mathsf{X}_j)_{j \neq i}$.

Then, the posterior p.m.f. of Z can be formulated as a discrete convolution product:

$$h(s) = \sum_{s_1} \cdots \sum_{s_d} h_0\left(s \cdot (s_1 \cdot \ldots \cdot s_d)^{-1}\right) h_1(s_1) \ldots h_d(s_d) = (h_0 * h_1 * \ldots * h_d)(s) \ . \tag{A.1}$$

Proof. By applying the Bayes' theorem we get:

$$h(s) \quad = \quad \frac{\Pr(\mathsf{Z} = s)}{\Pr(\mathbf{X} = \mathbf{x})} \Pr(\mathbf{X} = \mathbf{x} \mid \mathsf{Z} = s) \tag{A.2}$$

[1] We only focus here on one piece of the proof, so this appendix does not formally proves the soundness of group-based secret-sharing. We invite the interested reader to refer to the papers cited at the beginning of this appendix.

Using the total probabilities formula d times, we expand the term $\Pr(\mathbf{X} = \mathbf{x} \mid \mathrm{Z} = s)$ as follows:

$$\Pr(\mathbf{X} = \mathbf{x} \mid \mathrm{Z} = s) = \sum_{s_1 \in \mathcal{Z}} \dots \sum_{s_d \in \mathcal{Z}} \Pr(\mathbf{X} = \mathbf{x} \mid \mathrm{Z} = s, \mathrm{Z}_1 = s_1, \dots \mathrm{Z}_d = s_d) \cdot \prod_{i=1}^{d} \Pr(\mathrm{Z}_i = s_i) \quad . \tag{A.3}$$

By noting $s_0 \triangleq s \cdot (s_1 \cdot \ldots \cdot s_d)^{-1}$, and since the mapping $(s, s_1, \dots s_d) \mapsto (s_0, s_1, \dots s_d)$ is invertible we may reformulate the conditional probability as follows:

$$\Pr(\mathbf{X} = \mathbf{x} \mid \mathrm{Z} = s, \mathrm{Z}_1 = s_1, \dots \mathrm{Z}_d = s_d) = \Pr(\mathbf{X} = \mathbf{x} \mid \mathrm{Z}_0 = s_0, \dots \mathrm{Z}_d = s_d) \quad . \tag{A.4}$$

Moreover, according to assumption (b), we have:

$$\Pr(\mathbf{X} = \mathbf{x}) = \prod_{i=0}^{d} \Pr(\mathrm{X}_i = x_i) \quad , \tag{A.5}$$

$$\Pr(\mathbf{X} = \mathbf{x} \mid \mathrm{Z}_0 = s_0, \dots \mathrm{Z}_d = s_d) = \prod_{i=0}^{d} \Pr(\mathrm{X}_i = x_i \mid \mathrm{Z}_i = s_i) \quad . \tag{A.6}$$

Finally, we may use the assumption (a) to remark that:

$$\Pr(\mathrm{Z} = s) = \Pr(\mathrm{Z}_0 = s_0) \quad . \tag{A.7}$$

We may now combine Equations (A.2), (A.3), (A.4), (A.5), (A.6) and (A.7):

$$\begin{aligned} h(s) &= \sum_{s_1 \in \mathcal{Z}} \dots \sum_{s_d \in \mathcal{Z}} \prod_{i=0}^{d} \frac{\Pr(\mathrm{X}_i = x_i \mid \mathrm{Z}_i = s_i) \Pr(\mathrm{Z}_i = s_i)}{\Pr(\mathrm{X}_i = x_i)} \\ &= \sum_{s_1 \in \mathcal{Z}} \dots \sum_{s_d \in \mathcal{Z}} \prod_{i=0}^{d} \Pr(\mathrm{Z}_i = s_i \mid \mathrm{X}_i = x_i) \\ &= \sum_{s_1 \in \mathcal{Z}} \dots \sum_{s_d \in \mathcal{Z}} h_0\left(s \cdot (s_1 \cdot \ldots \cdot s_d)^{-1}\right) h_1(s_1) \cdot \ldots \cdot h_d(s_d) \end{aligned}$$

In other words, $h(s) = (h_0 * h_1 * \ldots * h_d)(s)$. □

Seeing the likelihood p.m.f. of the sensitive variable Z as a convolution product provides an intuitive insight about the soundness of secret-sharing. Indeed, convolutions are well known in signal processing to be *regularizing* operators, *i.e.*, they may transform any *sharp* signal as a *smooth* one. These properties can somehow be translated into the discrete world, as we will see hereafter.

A.2 A Fixed-Point-Like Proof

To show the smoothing effect of discrete convolutions, we first introduce a lemma, stating that the uniform p.m.f. is a fixed point of the convolution operator.

Lemma 3. *Let $h \in \mathcal{P}(\mathcal{Z})$ be a p.m.f. over the set $\mathcal{Z}$, and u the uniform p.m.f. over the same set. Then*

$$h * u = u. \tag{A.8}$$

Proof. For any $s \in \mathcal{Z}$, we have

$$h * u(s) = \sum_{s'} h\left(s \cdot s'^{-1}\right) u\left(s'\right) = \sum_{s'} h\left(s \cdot s'^{-1}\right) \frac{1}{|\mathcal{Z}|} = \frac{1}{|\mathcal{Z}|} \sum_{s'} h\left(s \cdot s'^{-1}\right) \ .$$

By definition of a p.m.f., the latter sum equals one, so $h * u(s)$ does not depend on s: it is the uniform distribution. □

It is well known that sequences recursively defined by the application of an operator having a fixed point may converge to the latter one, under some conditions on the p.m.f.s $(h_i)_{i \in [\![0,d]\!]}$. The fact that the fixed point here is the uniform distribution illustrates the smoothing effect. The theorem we introduce hereafter follows this intuition.

Theorem 3 (Kloss [Klo59]). *Let $\mathcal{Z}$ be a finite group, and $h_0, \ldots, h_d$ be $d+1$ p.m.f.s over $\mathcal{Z}$.*[2] *Let $h = h_0 * \ldots * h_d$. If for any element $s \subset \mathcal{Z}$ we have:*

$$h_i(s) > cu(s),\ c > 0, i \in [\![0, d]\!], \tag{A.9}$$

where u being the uniform distribution over $\mathcal{Z}$, then

$$|h(s) - u(s)| \leq |1 - c|^{d+1} \ . \tag{A.10}$$

In other words, h converges towards the uniform distribution when $d \to \infty$ at an exponential rate, provided that $|1 - c| < 1$. Here, the latter quantity $1 - c$ somehow describes the original noise parameter σ induced by the target device on the leakages: the lower $|1 - c|$, the noisier the leakages from the target device. Hence the noise amplification effect.

Proof. Let $h'_i = h_i - u$, with u being the uniform distribution. Note that h'_i may take negative values. Let us prove that $h'_i * u = u * h'_i = 0$. By using Lemma 3, we known that $h_i * u = u$. It follows that $h'_i * u = (h_i - u) * u = h_i * u - u = u - u = 0$. Moreover, we have $h_0 * h_1 = (h'_0 + u) * (h'_1 + u) = h'_0 * h'_1 + (h'_0 + h'_1) * u + u * u = h'_0 * h'_1 + u$. By induction, it follows that

$$h = h' + u \ , \tag{A.11}$$

where $h' = h'_0 * \ldots * h'_d$. Let $q_i \triangleq h'_i + (1 - c)u = h_i - cu$. The main assumption of the theorem implies that $\forall s \in \mathcal{Z}$, $q_i(s) \geq 0$. By analogy with Equation A.11, we can prove that $q \triangleq q_0 * \ldots * q_d = h' + (1 - c)^{d+1} u$. A convolution of non-negative functions gives a non-negative product, so $\forall s \in \mathcal{Z},\ q(s) \geq 0$.

Therefore, $h = u + h' = \left[1 - (1 - c)^{d+1}\right] u + q$, *i.e.*, $\forall s \in \mathcal{Z}$,

$$h(s) \geq \left[1 - (1 - c)^{d+1}\right] u(s) \ . \tag{A.12}$$

Besides, summing Equation A.12 for every $s' \neq s$ gives: $1 - h(s) \geq \left[1 - (1 - c)^{d+1}\right] (1 - u(s))$, that is:

$$h(s) \leq (1 - c)^{d+1} + \left[1 - (1 - c)^{d+1}\right] u(s) \ . \tag{A.13}$$

By combining Equation A.12 and Equation A.13, we deduce Equation A.10. □

Prest *et al.* proved the same result in their paper at CRYPTO 2019 [PGMP19, Lemma 6] with slightly different arguments based on *random walks*. Theorem 3 is an alternative proof only requiring standard results on probabilities. Indeed, one can then derive the so-called *relative error* defined by Prest *et al.* from Equation A.9 on which relies their noise amplification proof [PGMP19, Thm. 2].

Notice by the way that following a similar reasoning from Aldous and Diaconis [AD86, Thm. 3], this result may be slightly relaxed, by only assuming that the condition stated in Equation A.9 holds starting from a given order $1 < d' < d + 1$. It implies that the bound in Equation A.10 becomes $(1 - c)^{\frac{d+1}{d'}}$. Hence, we get a not only sufficient, but also necessary condition on the initial p.m.f.s $(h_i)_{i \in [\![0,d]\!]}$ to get a noise amplification, at the cost of a lower rate of convergence towards the uniform p.m.f.

[2] Kloss' theorem actually applies on any compact group, possibly uncountable.

Appendix B

List of acronyms

AES-HD AES - Hamming Distance. V, 50, 51, 89–91

AES-RD AES - Random Delay. V, 49, 50, 89, 90

Adam Adaptive Moment Estimation. 22, 82, 87, 90, 99, 113

AES Advanced Encryption Standard. V, VI, 1–4, 9, 19, 20, 26, 27, 29, 41, 42, 44, 45, 48–54, 73, 89–91, 94, 95, 121

ANSSI Agence Nationale de la Sécurité des Systèmes d'Information. iv, 8, 48, 126

API Application Programming Interface. 55, 70

ASCAD ANSSI's SCA Databases. iv, V, 48–50, 89–91, 106, 112, 113, 116, 118, 120, 128

BSI Bundesamt für Sicherheit in der Informationstechnik. 8

C.t.F. Capture the Flag. 25

CC Common Criteria for Information Technology Security Evaluation. 7, 9

CEA Commissariat à l'Énergie Atomique et aux Énergies Alternatives. iv

CEMA Correlation Electro-Magnetic Analysis. 35

CER Cross Entropy Ratio. 92

CESTI Centre d'Évaluation de la Sécurité des Technologies de l'Information. 7

CMOS Complementary Metal Oxide Semiconductor. 35

CNN Neural Networks that implement discrete finite convolutions in place of linear layers. VI, 64, 65, 71, 87, 89, 95–97, 100–103, 106, 112, 114, 119, 120, 122, 123, 128

COTS Commercially available Of The Shelf. 102

CPA Correlation Power Analysis. 34–36, 43, 45, 46, 53, 77, 94, 96, 97

CPU Central Processing Unit. 5, 6, 50, 70

D.F.T. Discrete Fourier Transform. 37, 85

DAG Directed Acyclic Graph. 61

DL Deep Learning. viii, 9–11, 47, 48, 55, 56, 60, 63, 65, 67, 68, 70–74, 76, 92, 109, 112, 125–128

DNN Deep Neural Network. V, 9, 11, 59–64, 66–68, 70–72, 74, 76, 83, 84, 86, 88, 108, 109, 123, 125–127

DPA Differential Power Analysis. 36, 44

EM Electro-Magnetic. 5, 6, 35, 36, 44, 51–53, 73, 96, 113

EMVCo Europay-Mastercard-Visa Consortium. 8

ERM Empirical Risk Minimization. 58–61, 64, 66–68, 71

ETR Evaluation Technical Report. 8

F.I.B. Focused Ion Beamer. 5

FIPS Federal Information Processing Standard. 4, 20

FPGA Field-Programmable Gate Array. 51, 73

GAN Generative Adversarial Network. 10

GE Guessing Entropy. V, VI, 25, 29, 30, 71, 114, 116, 117, 119, 120

GPU (General Purpose) Graphic Processing Unit. 70, 113, 127

GT Gaussian Template. 32, 33, 50, 53, 60, 74, 94, 96, 106, 119, 123, 126

GV Gradient Visualization. VI, 106, 109–112, 114–123, 126

HI Hypothetical Information. 92

i.f.f. if and only if. 2, 14, 18, 25, 92, 108

i.i.d. Independent and Identically Distributed. I, 17, 18, 26, 28, 58, 83, 109

I.S.W. Ishai-Sahai-Wagner; name of the scheme that computes a field multiplication for a Boolean secret-sharing. 42

IC Integrated Circuit. 7

ILSVRC ImageNet Large-Scale Visual Recognition Challenge. 9, 64

IoT Internet of Things. 45, 102

ITSEF Information Technology Security Evaluation Facility. 7–9

JIL Joint Interpretation Library, the reference document describing the state of the art in terms of physical attacks. 9

KDA Kernel Discriminant Analysis. 37, 106

KL Kullback - Leibler. 18, 80

l.r.a. Linear Regression Analysis a.k.a. stochastic attack [SLP05]. 36, 72

l.s.b. Least Significant Bit. 36

l.u.t. Look-Up Table. 20, 42, 45, 51–53, 121

LDA Linear Discriminant Analysis. 33

LRP Layerwise Relevance Propagation. 111, 112

LSTM Long Short-Term Memory. 71, 72

LWE Learning With Errors. 10

m.s.b. Most Significant Bit. 36

MCU Micro-Controller Unit. 5, 47, 126

MI Mutual Information. 18, 19, 34, 45, 75–78, 80, 82–86, 88, 89, 91, 92

MIA Mutual Information Analysis. 34

ML Machine Learning. iv, 9, 16, 34, 56, 57, 59, 60, 63, 66, 70, 71, 74, 76, 82, 83, 106, 114, 125, 126

MLP Multi-Layer Perceptron. V, 63, 64, 84–87, 109, 112, 126

MSE Mean Square Error. 67

NIST National Institute of Standard and Technology. 2–4, 29

NLL Negative Log Likelihood. V, 18, 67, 76, 78–86, 88, 89, 91, 92, 99, 113, 125

OS Operating System. 53

p.d.f. Probability Density Function. 14–16, 19, 24, 30, 38, 39, 85

p.m.f. Probability Mass Function. I–III, 14, 15, 26, 37, 39, 43, 54, 56, 57, 60, 63, 78, 80, 84, 107, 122, 125, 127

P.o.I Point of Interest. VI, 11, 35, 37–39, 49, 53, 54, 87, 88, 94, 100, 101, 106–110, 112, 116, 118–123

PCA Principal Component Analysis. 37, 95, 106, 119, 120

PI Perceived Information. 75, 76, 78–89, 91, 92

QDA Quadratic Discriminant Analysis. 33

RAM Random Access Memory. 47, 52, 53, 70

ReLU Rectified Linear Unit. 63, 97, 111, 112, 120

RNG Random Number Generator. 19, 33, 39, 41, 46, 49, 126

RSA Rivest-Shamir-Adleman. 2, 26, 71

SAT Propositional Satisfiability Problem. 25

SCA Side-Chanel Analysis. iv, viii, V, 9–11, 20, 24–28, 30–35, 37, 39, 40, 43–48, 50–52, 54–61, 65–68, 70–78, 82, 83, 89, 91, 92, 94–98, 102, 103, 106, 111, 122, 125, 126, 128

SGD Stochastic Gradient Descent. 21, 22, 66, 68, 74, 82, 84, 98, 109

SGPC Specialized Generator of Polymorphic Code. 46, 52, 53, 125

SIM Subscriber Identity Module. 3

SNR Signal-to-Noise Ratio. V, VI, 38, 39, 45, 47–54, 77, 87, 95–97, 100, 106, 107, 110, 114–116, 118–121, 123

SR Success Rate. V, 28, 29, 92, 100

SVM Support Vector Machine. 60, 67, 68, 71

T.O.E. Target of Evaluation. 7–9, 56

TA Template Attacks. 32, 37

TEE Trusted Execution Environment. 3

TPM Trusted Platform Module. 3

TVLA Test Vector Leakage Assessment. 94

VC Vapnik-Chervonenkis. 58–60, 64, 89

VGG Visual Geometry Group. 64, 65, 87, 89, 97, 112

VHDL VHSIC Hardware Description Language. 51

WH Walsh-Hadamard. 85

Appendix C

Glossary

σ-algebra A collection of subsets of a global set that is closed under complement, and is closed under countable unions; *Tribu* in French. 14

gradient The vector made of every partial derivative of a function. 20

almost everywhere Which is only not true on a set of measure zero. V, 21, 67

asymmetric cryptography Field of cryptography where the encryption key differs from the decryption one. 2, 26, 71

block cipher symmetric cryptography algorithm that splits a plaintexts message into blocks of fixed size that are then encrypted and decrypted separately. 4

characteristic Least integer p such that $p \cdot 1_{\mathbb{F}} = 0_{\mathbb{F}}$ in a field $\mathbb{F}$. 19

commutative Binary operation $\star$ such that $a \star b = b \star a$. 41, 42

confusion Method making the relation between the simple statistics of ciphertexts and the simple description of the secret key a very ocmplex and involved one [**?**]. 4, 20

consistent Finite proxy approach (*e.g.* estimation, optimization) that converges to the target result. 19

countable Which is in bijection with $\mathbb{N}$. 14

critical Point where the gradient is zero. 20, 21

cryptanalysis The art of guessing the plaintext message behind a ciphertext message. 2

cryptographic primitive Algorithm that is the building block of a crypto-system, *e.g.* processing encryption, decryption, or hach operations. 2–4

cryptography The art of designing encryption and decryption systems. 2, 3, 9, 10, 39, 42

cryptology The union of cryptography and cryptanalysis. 2

cyclic Finite group such that there is one element spanning all the others, by successive application of the inner law $\cdot$. 20

differential cryptanalysis A class of cryptanalysis based on the study of how slight variations in some input messages span variations in the cipher text, depending on the nature of the secret key. 4

diffusion Method to dissipate into long range statistics – *i.e.* into statistical structure involving long combinations of letters in the ciphertext **[?]**. 4, 20

finite Which is in bijection with $[\![1, N]\!]$. N is called the *cardinality* of the set. 14

hyper-parameter A parameter which cannot be optimized using an optimization algorithm, *e.g.* discrete parameters of the architecture. 61–63, 66, 68, 72, 73, 114–117

Jacobian matrix The matrix whose rows are the transposed gradient of each elementary function $\mathbf{x} \mapsto f(\mathbf{x})[i] \in \mathbb{R}$. VI, 21, 69, 70, 108, 109, 114, 115

linear cryptanalysis A class of cryptanalysis based on linear approximations of the target cryptographic primitive. 4

neighbourhood Set of points that are close to a point for a given norm. 20

NP-hard A problem for which there is no algorithm providing a solution in a polynomial time. 67, 70

partial derivative derivative of a function with respect to only one variable. 20

Resnet Deep Residual Neural Networks, equipped with *skip* connections in the architecture [HZRS16]. 61, 65, 98, 102

scalar product A bilinear mapping to $\mathbb{R}_+$ that is symmetric and semi-positive. 20

sparse Containing lots of values (close) to zero. 38

stochastic Which involves randomness. 19

vector space A linear algebraic structure that is stable by linear combinations where scalars are taken from a field [Ter18, Chap. 7.1]. 20, 85

vertical Attacks involving univariate leakages. 46

Appendix D

List of symbols

$\xrightarrow{\mathcal{L}}$ Convergence in law. 16, 19

$\mathcal{P}$ Convergence in probabilities. 16, 58, 59, 79, 81

$\triangleq$ An equality by definition. II, III, 14–20, 24, 28, 29, 31, 35, 37, 38, 43, 56, 58, 63, 66, 67, 78–81, 85

$\mathbb{F}_{p^q}$ Galois Field of polynomials of degree q with coefficients in $\mathbb{F}_2$. 19

$\cdot$ Abstract inner law for a commutative group. I–III, 41, 42

L^2 Quadratic error. 63

List of Figures

List of Tables