Bridging Deep Learning and Classical Profiled Side-Channel Attacks

Gabriel Zaid

THALES

N° d'ordre NNT : 2021LYSES043

THÈSE DE DOCTORAT DE L'UNIVERSITÉ DE LYON
opérée au sein de
l'Université Jean Monnet

École Doctorale 488
Science Ingénierie Santé

Spécialité de doctorat : Informatique

Soutenue publiquement le 30/11/2021, par :
Gabriel Zaid

Bridging Deep Learning and Classical Profiled Side-Channel Attacks

---

Devant le jury composé de :

M. Thierry Artières, Professeur, Ecole Centrale Marseille	Président du Jury
M. François-Xavier Standaert, Professeur, Université Catholique de Louvain	Rapporteur
M. Emmanuel Prouff, HDR, ANSSI & Sorbonne Université	Rapporteur
Mme. Sonia Belaïd, PhD, CryptoExperts	Examinatrice
M. Marc Joye, HDR, Zama & Ecole Normale Supérieure	Examinateur
M. Benoît Gérard, PhD, Direction Générale de l'Armement & IRISA	Examinateur
M. Lilian Bossuet, Professeur, Université Jean-Monnet	Directeur de thèse
M. Amaury Habrard, Professeur, Université Jean-Monnet	Co-directeur de thèse
M. Alexandre Venelli, PhD, NXP Semiconductors	Responsable industriel

Acknowledgments

Trois ans... Cela fait maintenant trois ans que cette thèse a débuté, et, depuis son commencement, il me tarde de rédiger ces quelques lignes de remerciement, et de prendre le temps d'affirmer toute la reconnaissance que j'ai pour l'ensemble des personnes m'ayant accompagné, conseillé ou accordé du temps durant cette magnifique aventure.

Une thèse est faite de rencontre, et la plus importante fût certainement celle d'**Alexandre Venelli**. Cette histoire a débuté par un mail le 5 octobre 2017 à 14h12 lorsqu'il me répondait pour un entretien téléphonique de 30min suite à ma candidature de stage de fin d'étude. A l'époque, je n'imaginais pas l'impact que cette rencontre aurait 4 ans plus tard. A cette époque, ma formation ne laissait pas entrevoir une carrière en cryptographie et pourtant, il a su me faire confiance. Quelques mois plus tard, la même situation s'est présentée lorsqu'il fallait faire un choix pour la thèse, et, une nouvelle fois, il m'a accordé sa confiance alors que, sur le papier, mon profil n'était peut-être pas le plus adapté. C'est simple, s'il n'avait pas été là, je pense que je n'aurai jamais mis un pied dans la cryptographie (domaine qui s'est avéré ensuite être une véritable passion). Son engagement aurait pu s'arrêter là, mais non. Il m'a enseigné sa vision de la recherche où l'open source est un devoir pour tout scientifique (même si cela peut nous jouer des tours). Son esprit critique, sa volonté à toujours comprendre l'intérêt dans tout ce qu'il entreprend, et sa rigueur m'ont un peu plus nourri chaque jour. Il a su rendre cette expérience à la fois scientifiquement et humainement passionnante. Son soutien indéfectible, sa disponibilité sans faille (même dans les moments les plus difficiles), son honnêteté, sa fiabilité, sa patience, son humour et sa volonté de partager ses connaissances font de lui LE mentor dont j'avais besoin ! La qualité de ce travail n'aurait jamais été possible sans son implication. Merci infiniment !

De même, cette thèse n'aurait pas eu la même qualité sans les idées et les remarques, toujours justes et constructives, de mon directeur de thèse **Lilian Bossuet** et de mon co-directeur de thèse **Amaury Habrard**. Diriger une thèse dont 80% s'est fait à distance me semblait être un réel challenge, et pourtant, ils ont su se rendre disponibles à toute heure malgré leurs emplois du temps extrêmement chargés. Leur passion pour l'enseignement, la pédagogie et la recherche est communicative et les innombrables conseils qu'ils ont pu me prodiguer me suivront à vie. La confiance et la liberté, qu'ils m'ont accordé dans les sujets que je souhaitais traiter, a été essentiel pour m'exprimer et pour mon épanouissement. Ça a été un immense plaisir de partager ces 3 années à leurs côtés et je me sens extrêmement chanceux de les avoir eus comme directeur et co-directeur de thèse. Merci pour tout !

Je voudrais également remercier **Monsieur Emmanuel Prouff** et **Monsieur François-Xavier Standaert** pour m'avoir fait l'honneur de rapporter mon travail. C'est pour moi un réel privilège d'avoir des références incontestables du domaine pour juger mes travaux de thèse. Merci pour le temps consacré à la relecture minutieuse de mon mémoire et pour l'ensemble des améliorations qu'ils y ont apporté. Je leur en suis d'autant plus reconnaissant que je sais leurs emplois du temps chargés. Je souhaite également remercier **Monsieur Thierry Artières**, **Madame Sonia Belaïd**, **Monsieur Benoît Gérard** et **Monsieur Marc Joye** pour avoir accepté de faire partie de mon jury. Vos travaux et la façon dont vous menez vos carrières scientifiques respectives sont une grande source d'inspiration et j'espère que je saurai me montrer digne de votre confiance.

En parallèle, je souhaite présenter mes sincères remerciements à **Monsieur Hervé Debar**, **Monsieur David Naccache** et **Monsieur Mathieu Ciet**, dont les précieux conseils, qu'ils m'ont prodigués avant de débuter cette thèse, m'ont énormément apporté durant cette aventure. Je les garderai en moi tout au long de ma carrière scientifique. Merci également à **Monsieur Matthieu Rivain** et à **CryptoExpert** de m'avoir donné l'opportunité d'exploiter leurs travaux qui m'ont grandement servi tout au long de ce périple.

Outre mes trois encadrants, deux personnes ont officieusement grandement contribués aux résultats de cette thèse. Dans un premier temps, je souhaite exprimer ma profonde gratitude auprès de **François Dassance** (le chevalier d'argent), mon compatriote de l'ESA (un jour peut-être), qui a passé d'innombrables heures à écouter mes théories les plus étranges les unes que les autres. Merci pour tout ce qu'il a pu m'apporter, pour les discussions passionnantes autour de la physique avec pour paroxysme une expérimentation INCROYABLE visant à démontrer les effets du centre de gravité, pour m'avoir fait prendre conscience qu'une génie de 5 ans avait probablement plus de réflexion logique que moi, pour son esprit critique sur mes travaux (et nos nombreux désaccords souvent dû à ma mauvaise foi), pour tous ses conseils et pour son infinie gentillesse: 1000 mercis ! Milesker Anitz !
Merci également à **Mathieu Carbone** avec qui je partage l'amour de la nourriture un peu trop grasse, des restaurants un peu trop chers et des histoires un peu trop bien racontées de Christophe Hondelatte. Ces années de collaboration ont considérablement enrichi cette thèse. Son optimisme pour la recherche et le développement de nouvelles idées est une motivation constante et une source d'échange et de débats sans fin. Son parcours et ses envies professionnels m'ont fait prendre conscience qu'on ne devait pas rester cantonner dans une case, et que notre plus grande richesse est la pluridisciplinarité. Merci pour sa patience, ses remarques toujours justes et constructives, son honnêteté, ses boulettes (le Shaman, on s'en souviendra), son écoute et son partage d'expérience et de connaissances. Je m'estime extrêmement chanceux d'avoir pu partager cette aventure avec lui.

Cette thèse a eu la particularité de se dérouler au sein de deux équipes. J'ai notamment eu la chance de passer une grande partie de mon temps au sein du CESTI de Thales où j'ai pu faire la rencontre de personnes extraordinaires. En 1er lieu me vient ma "Dream Team du SCA", dont le noyau dur est composé d'Alexandre, François, **Guillaume Dufresne** (dit Caramel, ou le Minot), **Benjamin Gamblin** (dit le Castrais) et **Luc Ye** (dit le Blagueur du vendredi midi). Les soirées jeux de société, les chorbas entre kouf', les anecdotes démentielles de GDU ("Vous saviez que je m'étais fait livrer un colis qui est parti d'Angleterre et qui [...]", je ne finirais pas cette histoire car elle fera l'objet d'un livre bibliographique. Restez connecter !), les heures de jeux sur Starcraft II (je suis très mauvais, mais je reste sur 2 victoires d'affilée, mes 2 seules d'ailleurs...), les challenges, les restos, les "Welcome back !" de notre sensai Andrew et j'en passe, tant de bons moments qui resteront parmi les meilleurs de ces 3 dernières années. Ils ont été un soutien sans doute plus important qu'ils ne l'imaginent. En particulier, merci à Guillaume pour tous ses enseignements (sauf la fois où [...], ça, c'est notre petit secret) et pour tout le temps qu'il m'a accordé alors qu'il en manquait déjà. Merci à Benji pour sa gentillesse, sa bienveillance, son altruisme et sa bonne humeur quotidienne même s'il faut sans cesse lui rappeler qu'on ne met pas de "g" à "pain". Merci à Luc pour avoir été le 1er à m'accueillir et à m'intégrer au CESTI. Si j'ai pu m'épanouir aussi rapidement, c'est en grande partie grâce à lui !

Dans le même esprit, j'aimerai adresser de chaleureux remerciement à mon voisin de bureau, **Julien Badoules**, qui me supporte depuis plus de 2 ans. Merci pour son soutien quotidien, pour ses nombreux conseils lorsque je devais faire un choix pour mon futur et pour son amitié. Merci également à la légende du CESTI, **Franck Salvador**, pour l'ensemble de ses enseignements sur des sujets que je maîtrise peu. Son humilité et son exigence pour le travail bien fait sont de réelles sources d'inspiration. Merci à l'influenceur phare du CESTI, **Grégoire Lewis**, qui a toujours su rester le même, malgré ses millions de followers. Clémentine peut être fière de

lui. Je souhaite également remercier **Jérémie Delpech** pour son soutien inconditionnel, pour ses vidéos hilarantes sorties des profondeurs d'internet, et pour ses lectures régulières des faits divers pittoresques dans la région de Toulouse. Je remercie aussi l'ensemble des responsables du CESTI : **Joan Mazenc** pour la confiance qu'il m'accorde depuis 4 ans, **Suzel Lavagne**, **Emmanuel Doche** et **Rodolophe Favrel** pour votre accueil au laboratoire. Et évidemment (les meilleurs pour la fin), je souhaite remercier l'ensemble de mes collègues présents et passés sans qui, cette aventure n'aurait pas été aussi épanouissante: les khouyas **Riyad Amrani** et **Philippe Roussel**, mon acolyte de la première heure **Léo Larsson**, **Maryline Olle** pour ses innombrables services, **Laurent Castelnovi** pour nos nombreux échanges même après son départ, l'homme à tout faire **Ludovic Tyack**, **Réda Lounis** pour sa joie de vivre communicante, **Philippe Rose** (promis, tu n'es pas obligé de m'appeler docteur tout le temps), **François Linant** dont la date d'anniversaire m'a porté chance durant ma soutenance, **Oriane Agostini**, **Anne-Sophie Rivemale**, **Carine Therond**, **Pierre Bappel**, **Thierno Barry**, **Thomas Ben**, **Romain Bergé**, **Théo Cusnir**, **Cyril Mansour**, **Georges-Bastien Michel**, **Erwan Pillet**, **Cyril Proch**, **Samuel Ricou**, **José Soler**, **Romain Veriato** et **Martin Verdier** (dit l'intouchable). Ça a été un réel plaisir de me lever tous les matins pour venir travailler à leurs côtés.

Bien qu'il fasse bon vivre dans le sud-ouest de la France, mon passage à Saint-Étienne fût d'une grande richesse. En particulier, je souhaite remercier **Damien Robissout** qui m'a accompagné pendant 3 ans. Partager le même sujet de thèse avec un autre doctorant est assez rare pour le souligner. Nos échanges par mail, nos heures passées au téléphone pour partager nos idées, nos aventures à Wrac'h ou à Varsovie (lorsqu'on n'avait pas encore fait la connaissance de la Covid) et les nombreux évènements, qu'il a organisés (ski à Chalmazel, bowling, escape game, raclette, ...), resteront sans aucun doute comme les meilleurs de cette thèse. Ce fut un réel plaisir de partager autant de bons moments avec lui et je lui souhaite tout le meilleur du monde pour la suite. Cette thèse aurait sans nul doute été moins qualitative, et moins passionnante, sans la présence de **Vincent Grosso**. Son exigence scientifique, ses connaissances vertigineuses, son esprit critique et son humilité inégalée en font, sans aucun doute, un exemple à suivre (sauf en ce qui concerne le foot, mais je ne suis pas sûr d'être le mieux placé pour en parler). Je tiens à le remercier pour ces merveilleux moments passés en sa compagnie (sauf pour la fois où il m'a emmené à la piscine). Je remercie les anciens doctorants et postdoc passés durant mon séjour, et qui, par leurs innombrables conseils, m'ont permis de démarrer cette thèse dans les meilleures dispositions. Merci à **Brice Colombier** qui nous a pris sous son aile avec Damien afin de nous enseigner toutes les erreurs à ne pas commettre durant une thèse. Merci également au coureur de l'extrême, **Ugo Mureddu**, pour ses encouragements, dont j'avais bien besoin, avant ma 1ère présentation officielle à Phisic. Merci au prince du Maroc **El Mehdi Benhani** pour nos après-midi de discussion autour du monde de la recherche. Merci infiniment d'avoir fait le déplacement pour venir assister à ma soutenance de thèse. Merci à ma star **Elie Noumon Allini**, pour ses enseignements sur le bruit blanc, la variance d'Allan et les PLLs. Son sourire radieux et son rire communicatif ont été de grands manques à mon départ de Saint-Étienne. Enfin, je souhaite remercier chaleureusement **Pierre-Antoine Tissot** sans qui les figures 3.3 et 3.4 de ce manuscrit n'auraient jamais vu le jour. Je remercie également l'ensemble des permanents et chercheurs associés du laboratoire **Alain Aubert**, **Florent Bernard**, **Nathalie Bochard**, **Pierre-Louis Cayrel** et **Viktor Fischer** pour leurs disponibilités, leur gentillesse et leurs simplicités.

J'aimerai aussi remercier l'ensemble des personnes, avec qui, j'ai eu la chance d'échanger et/ou de rencontrer lors de conférences/workshops lorsque cela était encore possible. En particulier, merci à **Eleonora Cagli**, **Loïc Masure**, **Rémy Daudigny**, **Houssem Maghrebi**, **Annelie Heuser**, **Benoît Gérard**, **Pierre-Alain Moëllic** et **Nicolas Bruneau** pour les différents échanges d'extrêmes qualités et plein d'enseignements que nous avons pu avoir.
J'ai également eu la chance de prendre part à la journée des "Nouvelles Avancées en Sécurité des Systèmes d'Information" organisée par l'INSA Toulouse et le LAAS-CNRS. En particulier, merci à **Mohamed Kaâniche** et **Vincent Nicomette** pour leur invitation. Ce fût un honneur d'être le 1er doctorant à venir présenter mes travaux lors de cette journée qui mérite à prendre encore

plus ampleur.

Je souhaite finalement conclure cette partie en remerciant les personnes qui compte le plus pour moi: ma famille, à qui, je dédie ce manuscrit. Je remercie de tout coeur l'ensemble de ma famille en France et au Maroc pour toutes ces belles années passées à vos côtés. Merci à ma Mima koubida ("Aaaaaaaah oui !") dont l'amour indéfectible et le soutien insatiable me poussent à m'améliorer un peu plus chaque jour. J'ai également une pensée pour mes grands-parents qui n'ont pas pu assister à cette réussite et dont l'héritage, qu'ils ont laissé, a façonné la personne que je suis devenu. Merci à mes parents pour l'amour qu'ils me portent, pour avoir toujours cru en moi et pour les innombrables sacrifices qu'ils ont faits pour que j'en sois là aujourd'hui. Je leur dois tout, et j'espère que je serai toujours me montrer digne de leur éducation. Je ne les remercierai jamais assez de m'avoir offert de si beaux exemples. **Je vous aime !** Merci à mes 3 frères avec qui nous formons un quatuor digne des Tortues Ninja. Je ne serais tout simplement pas la même personne s'ils n'avaient pas été là. Merci pour leur joie de vivre, leur exemplarité et pour m'avoir fait prendre conscience qu'avec un peu de volonté et beaucoup de travail, on pouvait soulever des montagnes. Merci d'être les rayons de soleil qui m'anime depuis 26 ans. Je leur dois une reconnaissance éternelle. Merci à la dernière arrivée dans la famille, Pô. Comme tu ne me comprendras malheureusement jamais, je vais essayer de traduire pour toi "Miaou miaou, miaaaaou !".

* * * * * * * * * * *

Abstract

In our information age, the needs of security increase. Therefore, the application of cryptographic algorithms is essential to provide strong theoretical security guarantees on data privacy. Proved secure against "black-box" adversaries, these algorithms can be implemented into embedded systems (*e.g.* smart cards). While those systems are physically accessible, adversaries can exploit their physical properties, such as *physical leakages* (*e.g.* power consumption), in order to retrieve sensitive information about the cryptographic computation. This attack scenario, called *side-channel attack*, uses some statistical tools to highlight the dependence between the physical leakages and the secret key. To reduce the adversary's scope, some countermeasures have been proposed (*e.g.* desynchronization, masking scheme) in order to make the attack more difficult to perform. However, the benefits of those countermeasures can be questioned by the recent development of Machine Learning and Deep Learning techniques because they automatically extract the sensitive information included in the physical leakages. Difficult to configure and optimize, the deep learning techniques can be considered as "black-box" models. In particular, no theoretical result bridges deep learning models and classical side-channel attacks. Thus, this new application can be challenged as no link is proposed with theoretical optimal attacks. This industrial thesis, based on the collaboration between the "*CNRS Laboratoire Hubert Curien UMR 5516 F-42023*" and the Thales ITSEF, contributes to reducing these issues in order to assess the robustness of an embedded system against side-channel attacks.

Firstly, we bridge the deep learning and the side-channel paradigms through the identification of a similar approach, namely the generative approach, and correlate our results with well-known issues in side-channel context (*i.e.* dimensionality reduction, synchronization, points of interest selection). Then, we design a new neural network architecture, called cVAE-ST, and demonstrate that the generative approach is an important step towards explainability and interpretability. This result is notably useful to reduce the limitation provided by the "black-box" properties and enhance the security assessment of cryptographic algorithms' implementations. However, our work also illustrates the practical limitations of the generative approach. A concrete alternative is to consider the discriminative approach. The second part of this thesis consists in the development of a methodology for constructing low complexity neural network architecture. The related empirical results validate this methodology by providing the best state-of-the-art result on all the dataset publicly available. Reducing as much as possible the neural network complexity is highly beneficial from an ITSEF perspective as it reduces the *elapsed time* criterion which non-negligibly influences the conclusion about the robustness of an embedded system. However, for being confident in the evaluation process, it is essential to generate models which converge towards the theoretical optimal solution.

Then, this thesis focuses on the attack optimality and introduces a new loss function that is specified to the side-channel context. This new metric, called *Ranking Loss*, enhances the model performance by maximizing the widely known success rate metric. Using the ranking loss is beneficial to select the model which converges towards the theoretical optimal attack. Finally, we develop an additional loss function for the *Ensembling* approach. This new metric, called *Ensembling Loss*, generates interactions between multiple algorithms in order to enhance their complementarity and improve even more the resulted attack. In side-channel context, this work demonstrates that negligible the gain of accuracy highly influences the resulted side-channel attack and can be useful to question the conclusion regarding the robustness of an embedded system.

Résumé

À l'heure où l'information prend de plus en plus d'ampleur dans le monde qui nous entoure, la nécessité de sécuriser nos données croît. Par conséquent, l'utilisation des algorithmes cryptographique est essentielle. Prouvés comme mathématiquement sûrs contre des attaquants en "boite noire", ces algorithmes peuvent faire l'objet d'attaques physiques lorsque ces derniers sont implémentés au sein de systèmes embarqués. En ce sens, les *attaques par canaux auxiliaires* peuvent être appliquées afin d'exploiter les *fuites physiques* (*e.g.* la consommation électrique) relatives aux calculs d'une implémentation cryptographique. Afin de limiter le périmètre de l'attaquant, de nombreuses contre-mesures ont été proposées (*e.g.* désynchronisation, schéma de masquage) afin de complexifier la mise en place de telles attaques. Cependant, les récents progrès en apprentissage profond remettent en cause ces contre-mesures et permettent d'extraire, de manière automatique, l'information sensible inclue dans les fuites physiques. Difficiles à configurer et à optimiser, les techniques d'apprentissage profond, très exploratoires, sont considérées comme des "boites noires". En particulier, aucun résultat théorique n'a permis d'établir de liens concrets entre les attaques par canaux auxiliaires, connues pour leur efficacité, et l'apprentissage profond. Par conséquent, cette nouvelle application peut être remise en cause, car la question de l'optimalité des attaques reste en suspens. L'objectif cette thèse CIFRE, menée au sein du laboratoire Hubert Curien et du centre d'évaluation de Thales (CESTI), est de réduire ces limitations dans un contexte d'évaluation sécuritaire visant à assurer la robustesse d'un système embarqué.

Dans un premier temps, nous lions les paradigmes d'apprentissage profond et d'attaques par canaux auxiliaires *via* l'identification d'une approche commune, dite générative. Notre travail permet de corréler un ensemble de problématiques connues par la communauté (réduction de dimensionnalité, resynchronisation, sélection de points d'intérêt). Par la construction d'une nouvelle architecture, appelée cVAE-ST, nous montrons que l'adaptation d'approches génératives est une première étape vers une meilleure compréhension et interprétabilité des résultats. Cela permet de réduire les limitations de "boites noires" et ainsi, de résulter en une meilleure évaluation de sécurité. Cependant, nous montrons que ces approches ont encore quelques limitations pratiques. Considérées comme une alternative concrète aux cVAE-ST, nous nous intéressons dans un second temps aux approches discriminatives. En particulier, nous définissons une méthodologie de construction d'architecture à faible complexité, dont les résultats empiriques obtenus sur l'ensemble des bases de données publiques, sont, aujourd'hui, les meilleurs de l'état de l'art. Leur faible complexité permet de réduire considérablement la phase de conception des algorithmes, souvent exploratoire, et ainsi, réduire le temps nécessaire d'une évaluation de sécurité. Cependant, dans le cadre d'une évaluation, il est important de générer des algorithmes convergeant vers une solution optimale.

Dans un troisième temps, nous nous focalisons donc sur l'optimalité des attaques et introduisons une nouvelle métrique d'apprentissage spécifique au contexte des attaques par canaux auxiliaires. Appelée *Ranking Loss*, cette métrique vise à améliorer les performances des algorithmes en maximisant le taux de succès d'une attaque et de se rapprocher des performances théoriques optimales. Finalement, nous caractérisons une nouvelle métrique d'apprentissage spécifique au contexte d'*Ensembling*. Appelée *Ensembling Loss*, cette métrique combine plusieurs algorithmes afin d'accroître leur complémentarité et ainsi, améliorer les performances d'une attaque par canaux auxiliaires. D'après les critères d'évaluation sécuritaire, cette dernière étude permet de démontrer qu'un faible gain en performance peut remettre en cause les conclusions relatives à la robustesse d'un système embarqué contre des attaques par canaux auxiliaires.

Contents

List of Figures

List of Tables

List of abbreviations

Adam Adaptive Moment Estimation. 72, 73, 98, 115, 150, 171, 176

AES Advanced Encryption Standard. xv, 6, 7, 9, 13, 28, 29, 31, 32, 40, 41, 44, 46–48, 76, 152, 170

ANSSI Agence Nationale de la Sécurité des Systèmes d'Information. 10

BSI Bundesamt für Sicherheit in der Informationstechnik. 10

CC Common Criteria for Information Technology Security Evaluation. 10, 160

CEMA Correlation Electro-Magnetic Analysis. 37

CER Cross-Entropy Ratio. 152, 153

CHES Conference on Cryptographic Hardware and Embedded Systems. 14–16, 109, 137, 159

CIFRE Convention Industrielle de Formation par la Recherche. ix, 12

CMOS Complementary Metal Oxide Semiconductor. xv, 26, 27

CNN Convolutional Neural Network. 13, 15, 63, 64, 74, 76, 109, 110, 114, 121, 152, 156, 182, 188

COSADE International Workshop on Constructive Side-Channel Analysis and Secure Design. 15

CPA Correlation Power Analysis. xv, 36, 37, 40, 41, 190

CRT Chinese Remainder Theorem. 162

CTCPEC Canadian Trusted Computer Product Evaluation Criteria. 10

cVAE-ST Conditional Variational AutoEncoder based on Stochastic Attacks. vii, ix, xvi, xix, 15, 91, 95, 99, 104, 130–132, 187–190

CVPR Conference on Computer Vision and Pattern Recognition. 14

DES Data Encryption Standard. 7, 170

DL Deep Learning. 57

DLSCA Deep Learning-based Side-Channel Analysis. xv, 13–16, 53, 190

DoM Difference of Means. 36

DSCA Differential Side-Channel Analysis. xv, 35, 36, 73

EAL Evaluation Assurance Level. 10, 11

ECC Elliptic Curve Cryptography. xv, 16, 33, 34, 43, 46, 49, 50, 77, 158, 170, 175, 176, 180, 182, 184

ECML PKDD European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases. 14

EL Ensembling Loss. xvi, xvii, 170, 174, 179, 181

ELU Exponential Linear Unit. 115

ERM Empirical Risk Minimization. 57, 78

ETR Evaluation Technical Report. 11

FC Fully-Connected. 115, 178

FCNN Fully-Connected Neural Network. 63, 121

FIPS Federal Information Processing Standard. 10

GAN Generative Adversarial Network. 88, 189

GPU Graphic Processing Unit. 12

HO-SCA High-Order Side-Channel Attack. 45, 189, 191

IACR International Association for Cryptologic Research. 13, 15, 16, 109, 137, 159

ICML International Conference on Machine Learning. 14

ITSEC Information Technology Security Evaluation Criteria. 10

ITSEF Information Technology Security Evaluation Facility. vii, xv, 10–13

KL Kullback-Leibler. xvi, 22, 90, 91, 93–96, 231

LSB Least Significant Bit. 18

MAC Message Authentication Code. 6

MIA Mutual Information Analysis. 37

MSB Most Significant Bit. 18

MSE Mean Square Error. 94

NeurIPS Neural Information Processing Systems. 14

NIST National Institute of Standard and Technology. 6, 10

NLL Negative Log-Likelihood. xvi, xvii, 70, 153, 174, 181

NMOS N-channel Metal Oxide Semiconductor. 26

PC Personal Computer. 27

PCA Principal Component Analysis. 76

PDF Probability Density Function. 37–39, 45, 74, 75, 82, 131, 149

PMF Probability Mass Function. 55, 61, 111

PMOS P-channel Metal Oxide Semiconductor. 26

PoI Point of Interest. 29, 36, 38, 67, 73, 87, 95–97, 99, 101, 102, 116–118, 126, 127, 152–154, 156, 189, 227

ReLU Rectified Linear Unit. 61, 62, 115

RERM Regularized Empirical Risk Minimization. xv, 59, 60

ResNet Residual Neural Network. 67, 76

RGS Référentiel Général de Sécurité. 10

RkL Ranking Loss. xvi, xvii, 137, 141, 153, 174, 181

RNN Recurrent Neural Network. 76

RSA Rivest-Shamir-Adleman. xv, 8, 16, 33, 34, 43, 46, 49, 77, 158, 160, 162, 170, 171, 175–177, 180, 183, 184

SA Stochastic Attacks. 38

SCA Side-Channel Attacks. xix, 9, 13, 33, 86, 189

SELU Scaled Exponential Linear Unit. 61, 62, 115, 118, 176

SGD Stochastic Gradient Descent. 68, 71, 72, 115

SNR Signal-to-Noise Ratio. xx, 30–32, 38, 44, 46–50, 99, 101, 103, 111, 114, 117, 122–127, 152–154, 190, 227

SOG-IS Senior Officials Group Information Systems. xix, 10, 160, 161, 177, 179–182, 184

SPA Simple Power Attack. xv, 33, 34, 43

SR Success Rate. 40

SSCA Simple Side-Channel Analysis. 73

ST Security Target. 11

TA Template Attacks. 37

TCHES IACR Transactions on Cryptographic Hardware and Embedded Systems. 15, 16, 109, 137, 159

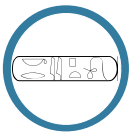
TCSEC Trusted Computer System Evaluation Criteria. 10

TOE Target of Evaluation. 10–13, 110

VC Vapnik-Chervonenkis. 58

Part I

Preliminaries

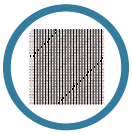


1900 B.C. — Hieroglyph

Skytale — **500 B.C.**

850 A.C. — Publication of the first book on cryptanalysis by Al-Kindi

Publicatiion of the Vigenère polyalphabetic cipher — **1585 A.C.**



1883 A.C. — Definition of the Kerckhoffs' Principles [Ker83]

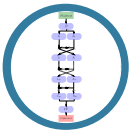
First provably secure algorithm: One-Time pad — **1917 A.C.**

WWII — Cryptanalysis of The Enigma

Publication of "Communication Theory of Secrecy Systems", Claude Shannon [Sha49] — **1949 A.C.**

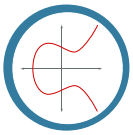
1976 A.C. — Creation of the public-key paradigm [DH76]

The Data Encryption Standard is published as a FIPS standard [Des77] — **1977 A.C.**



1978 A.C. — Ronald Rivest, Adi Shamir and Leonard Adleman introduce the RSA algorithm [RSA78]

Elliptic Curve Cryptography [Mil86, Kob87] — **1986 A.C.**

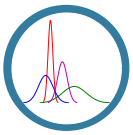


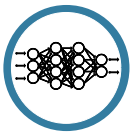
1996 A.C. — Application of the first side-channel attack [Koc96]

Proposition of the first unprofiled attack: Differential Power Analysis [KJJ99] — **1999 A.C.**

2000 A.C. — Standardization of the Rijndael algorithm (AES)

Proposition of the first profiled attack: Template Attacks [CRR03] — **2002 A.C.**





2016 A.C. — First application of deep learning techniques for performing side-channel attacks [MPP16]

Chapter 1

Introduction

This thesis is positioned at the cutting edge of *Cryptography* and *Machine Learning* with a particular focus on the application of *Deep Learning* techniques to enhance the performance of *Side-Channel Attacks*. In this chapter, we give an overview of the thesis' problematic and the contributions we have made. After providing a general introduction of *Cryptography* and its application in real life, we highlight the need of certification which ensures the robustness of cryptographic implementations against some attacks we present. Finally, we motivate this thesis by describing the benefits of using deep leaning to perform side-channel attacks.

Contents

1.1 The Art of the Secret

A secret is a piece of information that is only known by one person or a few people and should not be told to others. The field studying the science of the secret is called *Cryptology* and comes from the *Greek kryptós* (hidden, secret) and *logia* (study). More precisely, it refers to the protection of communications from an unexpected evil party called *attacker* or *adversary*. This field can be decomposed into two closely and complementary subfields: *Cryptography* (*kryptós* and *graphein* (to write)) and *Cryptanalysis* (*kryptós* and *analýein* (to analyze)).
Cryptography is the study of mathematical techniques tackling the problem of protecting some secret information through four fundamental mainstays, namely *integrity*, *authentication*, *confidentiality* and *non-repudiation*. The confidentiality is a property ensuring to keep the content of information from all but those authorized to have it. To guarantee this property some *cryptographic algorithms* are constructed. From a message, known as a *plaintext*, and a secret, referred as a *secret key*, a cryptographic algorithm returns an unintelligible message, namely a *ciphertext*, for any person that does not know the secret. A cryptographic process taking a pair (plaintext, secret) as input and returning a ciphertext is called an *encryption*. The inverse process generating a plaintext from a pair (ciphertext, secret) is the *decryption*. In this setting, the secret key is mandatory to perform the encryption and/or the decryption. Depending on how does the secret key is manipulated, the cryptographic algorithms can be categorized into two classes: *symmetric*

and *asymmetric* algorithms. The cryptographic algorithms are designed to resist against adversaries. Following the Kerckhoffs' Principles [Ker83], we generally assumed that the adversary has a perfect knowledge of these algorithms. Hence, they should be secure based on their mathematical foundations only. A classical way to introduce cryptographic algorithms is to consider a communication between two parties known as *Alice* (sender) and *Bob* (receiver).

Symmetric Cryptography. Symmetric cryptography is a general term referring to the protection of a communication between a sender and a receiver who share the same secret key. An algorithm that derives from symmetric cryptography is called a *symmetric algorithm*. Typically, a symmetric algorithm can be useful to ensure confidentiality through encryption mechanisms and to guarantee the data integrity as well as its authenticity with the application of *message authentication code* (MAC). Formally, a symmetric algorithm can be expressed as follows. Assuming two characters Alice and Bob such that Alice wants to send a message to Bob through a communication channel. To ensure the confidentiality of the message, both protagonists have to share a common secret key k^*. Then, before sending its message x to Bob, Alice has to encrypt it using the secret key k^* and an *encryption function* denoted $enc_{k^*}(x)$. The result of this encryption defines the ciphertext. Once the ciphertext is constructed, Alice sends it to Bob through a communication channel. Once Bob receives the ciphertext *ciph*, he uses the secret key k^* and a *decryption function* denoted as $dec_{k^*}(ciph)$ in order to retrieve the readable message that Alice sent. This communication process is described in Figure 1.1.

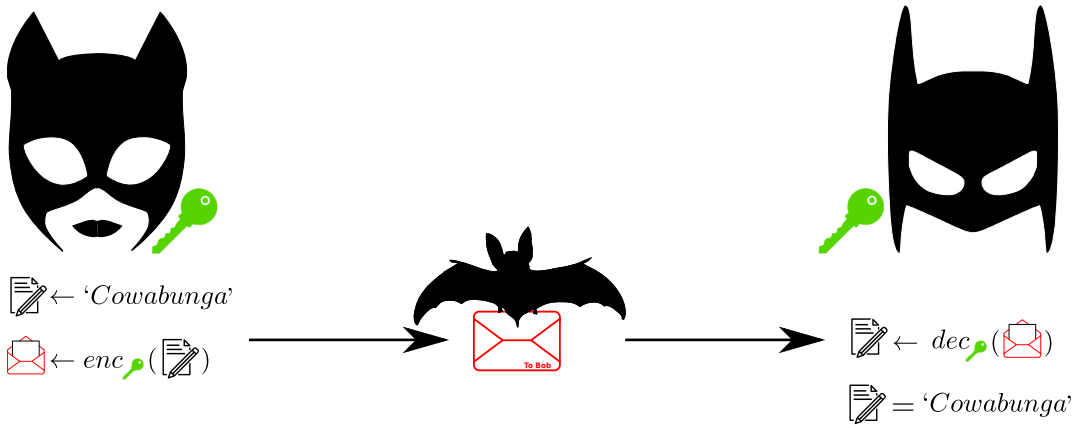


Figure 1.1: Secure communication using a symmetric algorithm.

To convert a message to a ciphertext, an encryption function can be viewed as a *stream cipher* or a *block cipher* combined with *modes of operation* most of the time. Firstly, a stream cipher decomposes a message (resp. ciphertext) into single bits, which then are individually converted in order to construct a binary representation of the related ciphertext (resp. message) knowing k^*. On the other hand, a block cipher combined with modes of operation decomposes a message (resp. ciphertext) into several blocks of identical size (*e.g.* 64 bits, 128 bits, 256 bits) and performs an encryption (resp. decryption) one block at a time. To determine a block cipher as secure, it has to ensure that two encryptions of the same block of plaintext return two different blocks of ciphertext. Thus, it prevents the risk of extracting information on the global message. More formally, a block cipher has to respect two properties [Sha49]: the *confusion* and the *diffusion*. The confusion property hides the dependency between the plaintext and the ciphertext while the diffusion ensures the dependence between one bit of the ciphertext and many bits of the plaintext and the secret key.
One commonly used block cipher algorithm is called the *Advanced Encryption Standard* (AES) [DR02], also known as Rijndael, designed by Joan Daemen and Vincent Rijmen. Standardized by the *National Institute of Standard and Technology* (NIST) in 2000, the AES algorithm operates on blocks of 128 bits. As illustrated in Figure 1.2, a round of AES is configured by multiple

cryptographic primitives, namely *SubBytes*, *ShiftRows*, *MixColumns* and *AddRoundKey*. The confusion property is ensured by the substitution functions (*i.e.* SubBytes and AddRoundKey) while the diffusion property is ensured by the ShiftRows and the MixColumns. Deeper details on the *AddRoundKey* and *SubBytes* functions will be provided in Chapter 2. Depending on the considering key size, the number of rounds N_r varies such that 10 (resp. 12/14) rounds are performed when the key size equals 128 (resp. 192/256). Other symmetric algorithms (*e.g. Data Encryption Standard* (DES) [Des77]) are deprecated, because too weak security-wise, and hence are out of the scope of this thesis.

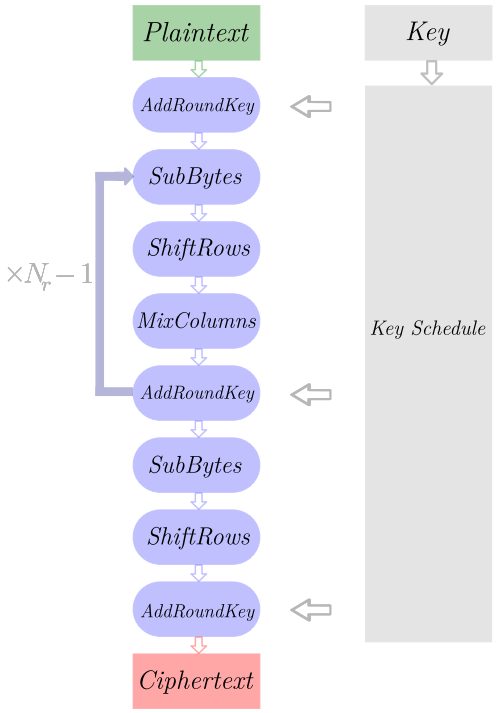


Figure 1.2: Structure of AES.

Asymmetric Cryptography. In 1976, Diffie and Hellmann propose a new approach to securely communicate between multiple parties [DH76]. Indeed, asymmetric cryptography considers multiple keys (each dedicated to a party) in order to ensure confidentiality and authentication. While the confidentiality is provided by the encryption function, the authentication is guaranteed by the *electronic signature* which provides the same benefits as handwritten signature. This manuscript only considers the latter one. If two parties are considered, a pair of keys has to be generated for each protagonist. The first key, called *private key*, is only known by its owner and it is used to *sign* a message. On the other hand, the second key, called *public key*, is known by everyone and can be used by any party who wants to verify the signature provided by its owner. The main difference between symmetric cryptography and asymmetric cryptography holds in the number of keys that are needed to perform an encryption and a decryption when a couple (sender, receiver) is considered. Formally, a digital signature algorithm can be expressed as following. Let Alice be a sender and Bob be a receiver. First, Alice constructs a public key (resp. private key) denoted $p_{k^*,A}$ (resp. $s_{k^*,A}$). In this process, Alice wants to electronically sign a message x. Hence, she signs her message[a] with her private key in order to obtain the related signature (*i.e.* $sign = enc_{s_{k^*,A}}(x)$). Then, Alice sends the message with its signature to Bob through a communication channel. Once Bob receives the pair (x, $sign$), he uses Alice's public key $p_{k^*,A}$ to verify the signature $dec_{p_{k^*,A}}(sign)$ and checks if the result corresponds to x. This process is

[a]In practice, the signature is not perfomed on the message itself. This simplification has been made for educational reasons. Indeed, for security reason, a signature scheme often applies a hash function to the message before applying $enc_{s_{k^*,A}}$.

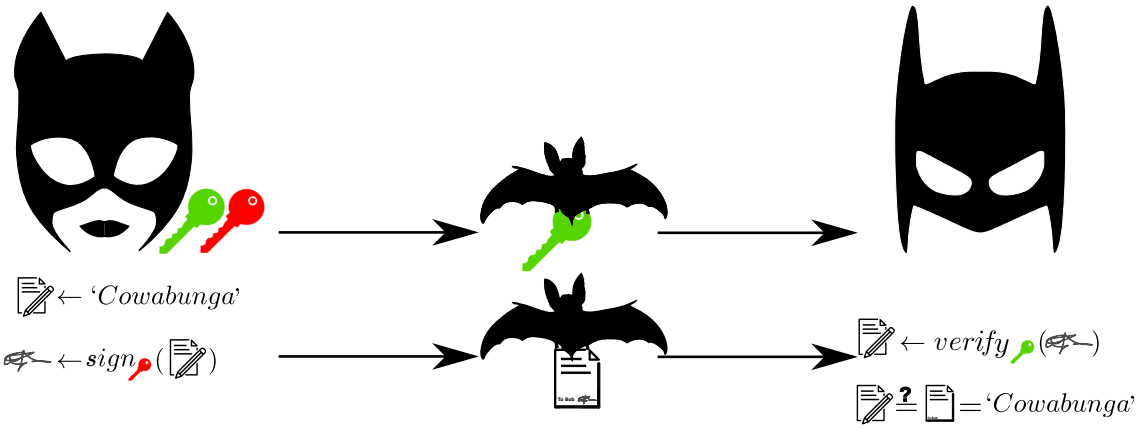


Figure 1.3: Digital signature mechanism.

described in Figure 1.3. The security of asymmetric cryptographic algorithms is based on hard mathematical problems which means that even a lot of computing power will not defeat such algorithm. Classically, two main mathematical problems are considered for constructing asymmetric algorithm: the difficulty to factorize the product of two large prime numbers and the discrete logarithm problem (DLP).
The first problem is the foundation of the RSA system, introduced by Rivest, Shamir and Adleman [RSA78]. Given two large prime numbers p and q, the public key is composed by a modulus $N = p \times q$ and an integer e co-prime with $(p-1)(q-1)$. Then, the private key, defined by an integer d, has to respect the restriction $e \cdot d \ mod \ (p-1)(q-1) = 1$. To sign a message, Alice performs the following modular exponentiation $sign = x^d \ mod \ N$ before sending the signature through the communication channel. Once Bob receives the signature and the related message, he computes $sign^e \ mod \ N$ to retrieve its content and then, verifies that it corresponds to x. This algorithm is known as a hard problem because retrieving p, q and d knowing the public key $p_{k^*,B} = (N, e)$ is computationally expensive. An alternative to the RSA signature algorithm is Elliptic Curve Digital Signature Algorithm (ECDSA) [ANS05] that is based on Elliptic Curve Cryptography [Mil86, Kob87].

Over the years, cryptography arises as a fundamental science for ensuring secure communication over insecure channels. Thus, its application in our modern world appears as a natural consequence.

1.2 Embedded Cryptography

1.2.1 Physical Attacks

The proliferation of computers and communications systems in the 1960s and a growing demand from the private sector to provide security services play a major role on the cryptography's development. Hence, cryptography-based security systems may be utilized in various applications (*e.g.* data storage, access control, network communications). These cryptography-based security systems, known as *cryptographic modules*, are defined in [FIP09]: a cryptographic module "*a set of hardware, software and/or firmware that implements approved or allowed security functions* (e.g. *cryptographic algorithms and key establishment) and encompasses the perimeter*". One common cryptographic module is the so-called *microprocessor card*. A microprocessor card consists in a small physical device which is composed by a *microchip* and has the particularity to host its own operating system. Its ability to store and modify information in its own non-volatile memory makes this card a powerful and practical tool in various applications (*e.g.* mobile telephony, banking, healthy cards, *etc*). However, the numerous constraints of size, consumption, and usability

make the microprocessor card a sensitive target to the attacks described in this thesis.
The security of a cryptographic algorithm is based on the secrecy of the key. Indeed, if the key is exposed then all the security features may be violated. Therefore, it is crucial to store the secret key in an unwavering environment. Typically, standardized cryptographic algorithms can be considered as secure against *black-box* attacks. In such configuration, the adversary only knows the algorithm structure, the plaintext and the ciphertext used during the communication process. Based on this knowledge, the adversary aims at extracting the secret key by exploiting the intrinsic properties of the targeted cryptographic algorithm. For example, the AES algorithm has been designed to be robust against classical cryptanalysis attacks, namely *differential* [BS90] and *linear* [Mat94] cryptanalysis. However, this scenario does not fit anymore when cryptographic module is considered. Indeed, implementing cryptographic algorithm into a physical system requires to store the secret key into the device. Hence, these embedded systems can be defeated by some known *physical attacks*. Much more powerful than classical cryptanalysis techniques, the physical attacks target the implementation of cryptographic algorithms. Depending on the adversary's capabilities, we can decompose these attacks into different categories [MOP07].

Active vs. Passive Attacks. The *active attack* refers to a perturbation that affects permanently or temporarily the device behavior. Typically, an adversary can exploit this abnormal behavior to retrieve the secret key. The most common example has been introduced by Boneh *et al.* [BDL97] and is known as the *fault injection* (FI) attack. It induces erroneous results being exploitable to recover the targeted secret.
As opposed, the *passive attack* does not interact with the cryptographic module. It aims at eavesdropping the leaking information of the targeted device under a normal computational behavior. By observing the physical emanations, the adversary tries to extract some sensitive information in order to retrieve the secret key. These attacks, known as *side-channel attacks* (SCA), were firstly proposed by Kocher in 1996 [Koc96]. The source of information can be large: power consumption [KJJ99], electromagnetic emanations [GMO01, QS01], computational time [Koc96], acoustic emanations [Tro04, GST16], *etc.*

Invasive vs. Non-Invasive Attacks. Cryptographic modules such as microprocessor cards can be physically prepared in order to enhance the resulted attack. Depending on the level of intrusion, these attacks can be categorized into *invasive*, *semi-invasive* or *non-invasive* attacks.
Firstly, the invasive attack is the strongest attack that can be performed against a cryptographic module. Commonly, this attack requires to irremediably alter the physical device. For example, a microprocessor card can be depackaged [KK99] in order to get a physical access to the related microchip. Thus, the adversary can establish a direct contact with the cryptographic module in order to recover the secret key. Even if this configuration is beneficial for the adversary, this requires a lot of skills, expensive equipment and the security flaws' exploitation highly depends on the targeted cryptographic module (*e.g.* reverse engineering).
In the semi-invasive attack, the adversary also removes the package of the cryptographic module (*e.g.* depackaging of a microchip). It can be useful to capture the electromagnetic emanations from the system [GMO01, QS01] or modifying its behavior by faults injection [SA03]. This attack differs from the invasive attack as it does not require a direct contact to the targeted system for being performed.
Finally, the non-invasive attack only exploits unintentional leakage information such as running time, power consumption. Hence, no tampering are processed on the cryptographic module to reveal information on the secret. For example, measuring the power consumption [KJJ99] and injecting faults using clock glitches or temperature variations, which does not alter the cryptographic module, [BECN+06] are non-invasive attacks.

In this manuscript, a particular focus will be made on side-channel attacks that are considered as passive and semi-invasive/non-invasive attacks. The sources of information we consider are the power consumption and the electromagnetic emanations.

1.2.2 A Step Towards the Certification

The emergence of side-channel attacks as credible threats against cryptographic modules forces the governments and the industries to ensure the security of the cryptographic modules by providing some rules. These rules are defined by the *certification schemes* which are not limited to the side-channel threats.

Scheme. The most common scheme, namely the *Common Criteria for Information Technology Security Evaluation* (CC) (standard ISO/IEC 15408 [ISO]), was created in 1999. It unifies three previous standards known as the *Information Technology Security Evaluation Criteria* (ITSEC, led by European countries), the *Canadian Trusted Computer Product Evaluation Criteria* (CTCPEC, led by Canada) and the *Trusted Computer System Evaluation Criteria* (TCSEC, led by the United States). The CC introduces the concept of *Target of Evaluation* (TOE) that characterizes the system under evaluation. In particular, the certification of the TOE is made following different levels of security assurance. The *Evaluation Assurance Level* (EAL) starts to EAL1 (lowest security grade) and goes to EAL7 (highest security grade). This level does not measure the security of the TOE itself but states that the system is evaluated under some conditions. Indeed, they determine the complexity of the tasks that has to be made to simulate the adversary's ability. Consequently, to ensure the security and the robustness of the TOE, the certification is given for a finite period of time at the end of which it has to follow a new certification process. This ensures the robustness of the TOE against new security threats. However, the CC does not provide any details on how the cryptographic implementation should be made within a TOE. Hence, national standards release some rules that have to be respected in order to certify the specifications of cryptographic modules. The *Senior Officials Group Information Systems Security*[b] (SOG-IS) agreement defines a set of requirements and evaluation procedures related to cryptographic aspects of CC security evaluations of IT products. Participants are government organizations or government agencies from countries of the European Union (*i.e.* Austria, Belgium, Croatia, Denmark, Estonia, Finland, France, Germany, Italy, Netherlands, Luxembourg, Norway, Poland, Slovakia, Spain, Sweden, United Kingdom). In France, these recommendations can be substitute by the *Référentiel Général de Sécurité* (RGS) [ANS14], that is introduced by the *Agence Nationale de la Sécurité des Systèmes d'Information* (ANSSI) and provides some rules and guidelines that have to be respected by the Developper. In the United States, these requirements are provided by the *Federal Information Processing Standard Publication 140-2* (FIPS 140-2 [FIP01]).

Actors. Depicted in Figure 1.4, the CC defines three entities that are involved in the certification process:

- Developer – This entity conceives a product destined for sale. To ensure the reliability of the product, the developer can enter into a certification process in order to reduce the security flaws. This certification can be made for different purposes such as international recognition, commercial advantages or it can be needed for selling some specific products (*e.g.* the credit cards have to be certified before its use by banking companies).

- Certification body – Often represented by governmental organizations like the *Agence Nationale de la Sécurité des Systèmes d'Information* (ANSSI, France), the *Bundesamt für Sicherheit in der Informationstechnik* (BSI, Germany) or the *National Institute of Standard and Technology* (NIST, United States), it delivers the certificate to the Developer and ensures the suitability of the evaluation provided by the Evaluator.

- Evaluator – It acts as a third independent party. Licensed by the certification body, this laboratory, also known as an *Information Technology Security Evaluation Facility* (ITSEF), must be impartial and is responsible of the security assessment of the TOE. Based on this evaluation, the Certification body decides to deliver or not the certificate to the Developer.

[b] The interested readers may find useful information in `https://www.sogis.eu/index_en.html`

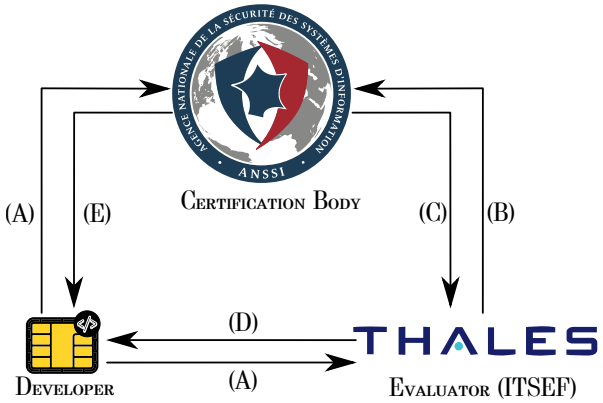


Figure 1.4: Example of the French Certification Process with Thales considered as the ITSEF.

Evaluation Process. The certification process can be decomposed into different phases. This manuscript deliberately chooses to omit some interactions between the entities in order to simplify the process and get a global overview on it. All the entities and the exchanges we considered are summarized in Figure 1.4.

- **(A)** – The Developer aims at certifying one of its products. Hence, he sends a request to the Certification body and the chosen Evaluator in order to start the certification process. First, the Developer has to provide the *Security Target* (ST) that defines the TOE and lists the features it claims as secure.

- **(B)** – Once the Evaluator receives the ST, he is charged to validate the claims provided by the Developer. To this end, he produces to a vulnerability assessment that depends on the desired EAL grade. This process is performed by deeply analyzing the TOE and verifying the expected functionalities. If vulnerabilities are detected, the Evaluator determines some scenarios in order to exploit it under real conditions. Indeed, to evaluate the feasibility of these attacks, the Evaluator has to estimate a metric, called the *cotation*, that is based on some criteria [SI13]: *elapsed time*, *expertise*, *knowledge of the TOE*, *access to TOE*, *equipment and tools*. These terms reflect the difficulty of performing an attack. Based on his investigations, the Evaluator writes an *Evaluation Technical Report* (ETR), defines if the TOE is secure or not, and transmits the report to the Certification body.

- **(C)** – From the ST and the ETR, the Certification body emits a judgment on the robustness of the TOE. If the Certification body cannot make a decision from the ETR, he can ask the Evaluator to conduct additional tests in order to facilitate its decision-making. The Evaluator should go back to step **(B)** in order to satisfy the requirements.

- **(D)** – Once the ETR is confirmed by the Certification body, it can be transmitted by the Evaluator to the Developer.

- **(E)** – Finally, if the features claimed by the TOE are not met, the Certification body informs the Developer that some countermeasures should be provided in order to reduce the risks exploited during the evaluation. Consequently, the Developer has to consider the ETR's recommendations provided by the Evaluator at step **(D)**. Once the modifications are considered, the certification process goes back to step **(A)**. On the other hand, if the security claims are respected, the Certification body supplies the *certificate* with the related EAL grade and can then be sold as a certified product.

Based on the certification process, the following section describes the context as well as the motivations of the thesis and defines the main criterion considered in our contributions.

1.3 Motivations & Goal of the thesis

Context. The thesis originates from the collaboration between the "*CNRS Laboratoire Hubert Curien UMR 5516 F-42023*", part of the University of Saint-Etienne (France) and University of Lyon (France), and the *Thales ITSEF* (Toulouse, France). The contributions presented in this thesis have been developed in the context of a *CIFRE* convention (Industrial Agreements for Training through Research), a type of contract launched in 1981 to increase research activities in French enterprises by promoting the interaction with public research institutions. Conducting in an ITSEF, this manuscript falls within the Evaluator point of view (see Figure 1.4). More precisely, we focus our interest on the step **(B)** of the evaluation process only. To efficiently evaluate the robustness of a cryptographic module, the Evaluator has to conduct the evaluation under real conditions. As previously mentioned, the time required to perform an attack, namely *elapsed time*, has to be considered during the certification process. The longer the attack, the highest the cotation rate. A solution to reduce the elapsed time is to enhance the side-channel attacks by modeling the optimal solution. If this threshold is reached, the Evaluator achieves the worst-case scenario from a security point of view. This optimal solution helps him to assess the security of the TOE with a high confidence. Even if the certification process is not considered in the scope of this manuscript, the elapsed time and the optimality of a side-channel attack are crucial considerations in our contributions.

Typically, a side-channel attack consists in the construction of a model correlating a set of data (*e.g.* power consumption and/or electromagnetic emanations) with the secret key manipulated by the targeted cryptographic module. To build this model, the Evaluator uses statistical tools (*e.g.* correlation) but needs to preprocess the data in order to enhance the attack performance. This preprocessing phase is based on the expertise of the Evaluator and highly impacts the performance of the related attack as well as the elapsed time of an evaluation. Finding an alternative model which automatically maps a set of data with the correct unknown secret key without any preprocessing phase is crucial in order to reduce the elapsed time attribute while preserving the ability of the Evaluator to recover the unknown secret key. These models can be designed from *Machine Learning* approach which is an application of the *Artificial Intelligence*[c] paradigm. It provides to algorithms the ability to automatically learn how a given task should be solved from successive experiences and by the use of data. A subfield of Machine Learning, namely *Deep learning* and defined as [DY14] "*in the intersections among the research areas of neural networks, artificial intelligence, graphical modeling, optimization, pattern recognition, and signal processing*", sounds a good solution to automatically learn the model we expected in side-channel context.

Brief history of Deep Learning. In 1943, McCulloch and Pitts defined the first simplified model of *biological neuron* that considered one or multiple inputs and returned a binary output [MP43]. This concept was then extended by Rosenblatt to introduce the *perceptron* [Ros58] (deeper details on the perceptron will be provided in Subsection 4.2.1). In 1986, Rumelhart *et al.* propose a new way to interact with a multilayer perceptrons that automatically learns its configuration for a specific task [RHW86]. Deep learning is a class of machine learning algorithms that extract higher-level features from a data by the application of multiple layers (*e.g.* multiple perceptrons layers) [DY14, p.199-200]. After the first application of GPUs to parallelize the computational cost of deep learning models [RMN09], this field growth in popularity in 2012, when Krizhevsky *et al.* [KSH12] wins Imagenet's image classification contest [RDS+15] and drastically outperforms all the previous results. Indeed, deep learning algorithms automatically learn how does the relevant information induced in the data should be combined in order to correctly classify it.

[c] *Artificial Intelligence* is a wide-ranging branch of computer science concerned with building algorithms capable of performing tasks that typically require human intelligence (*e.g.* image classification, speech recognition, computer vision, *etc*).

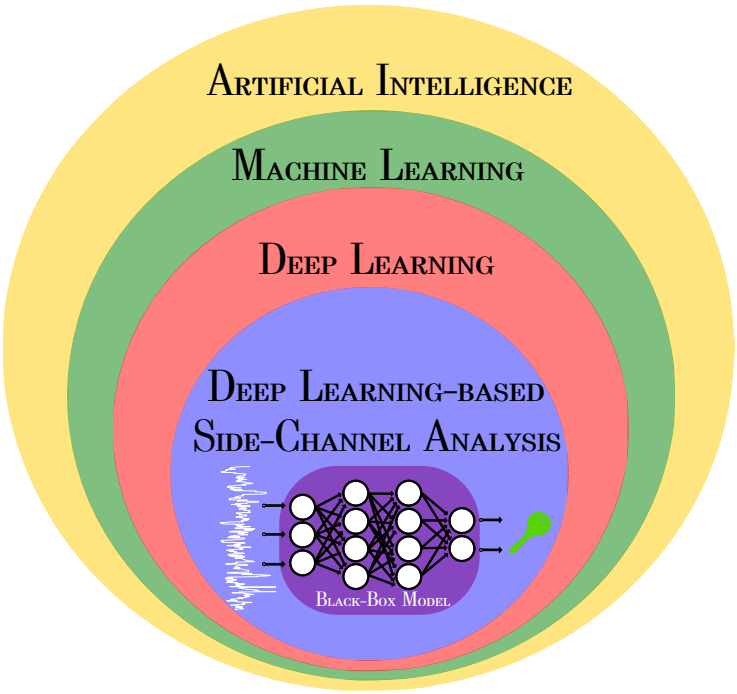


Figure 1.5: Position of Deep Learning-based Side-Channel Analysis regarding Artificial Intelligence.

Emergence of DLSCA. Similarly, this approach can be considered in side-channel attacks for extracting the secret key from a set of power consumption and/or electromagnetic emanations and limiting the preprocessing phase. In 2013, Martinasek and Zeman introduce the very first work using deep learning approach in SCA [MZ13]. In [MPP16], Maghrebi *et al.* extend this work by showing that *deep neural networks* are beneficial for defeating an AES's implementation protected with *Boolean masking* (details in Subsection 3.4.1). Then, Cagli *et al.* demonstrate the suitability of using a particular network, namely *convolutional neural networks* (CNNs), to break software and hardware implementations protected with desynchronization countermeasures [CDP17a] without any preprocessing phase. Those results convinced the side-channel community and the Evaluator to deeper investigate the benefits of the deep learning techniques to perform side-channel attacks. This phenomenon is illustrated in Figure 1.6. This figure depicts the evolution of the number of papers dealing with DLSCA issues since 2015. The database we use lists the deep learning-based side-channel papers we observed on *ePrint*[d] and *arXiv*[e] during this period of time. From the Evaluator point of view (*e.g.* ITSEF), the deep learning-based side-channel analysis can be highly recommended. While a large part of an evaluation process is to define the adequate preprocessing for limiting the impact of some countermeasures, this new direction can be favorable to reduce the effort of the Evaluator for assessing the TOE's security. However, while the side-channel attacks are based on strong mathematical foundations, the deep learning models can be difficult to fully understand and are oftenly seen as *black-boxes* (see Figure 1.5). Thus, their interpretation and their configuration are considered as non-trivial tasks for the Evaluator.

Goal. In this thesis, we want to tackle the problem of black-box issues as well as side-channel attack optimality in order to limit as much as possible the preprocessing phase without altering the

[d]The *Cryptology ePrint Archive* indexes the recent research in cryptology the authors want to make publicly available. This open-access archive was started by *International Association for Cryptologic Research* (IACR) in 2000 (url: `https://eprint.iacr.org/`)

[e]arXiv is an open-access archive that was founded in 1991 by Paul Ginsparg. Maintained by Cornell Tech, this platform concerns a wide range of fields and is not exclusive to the cryptology community (url: `https://arxiv.org/`)

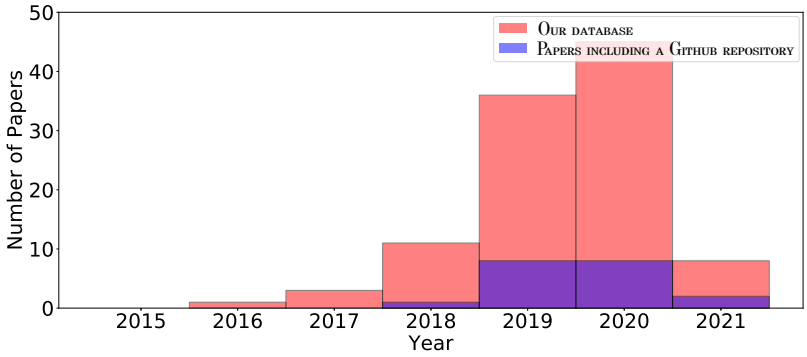


Figure 1.6: Evolution of the number of DLSCA papers released (April 21st, 2021). Among the papers we report, only 18% include a *GitHub* repository.

efficiency of the models. In order to reach these objectives, we first aim at bridging the gap between deep learning and classical side-channel attacks by investigating new kinds of models based on different strategies. This direction is beneficial to reduce the black-box issues and concretely illustrate the benefits and the limitations of the deep learning techniques applied in side-channel context as well as defining some insights to construct an effective model. Furthermore, while the state-of-the-art considers classical deep learning metrics to train a model, we aim to construct new metrics that are specific to the side-channel context in order to approximate the optimal solution. In parallel, to facilitate the reproducibility of our results, we decide to follow one common approach considered by the machine learning community which deeply encourages the papers to provide the code and/or the dataset accompanying their submission. In the major conferences, reproducibility of results and easy availability of code is taken into account in the decision-making process (*e.g. Neural Information Processing Systems* (NeurIPS), *International Conference on Machine Learning* (ICML), *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases* (ECML PKDD), *Conference on Computer Vision and Pattern Recognition* (CVPR), *etc.*). For example, to ease the code publication format, NeurIPS defines a clear policy[f] to follow and suggests a template[g] the authors can use in order to standardize the submission process. Following this approach can be beneficial when deep learning techniques are applied in side-channel context. By pursuing this ideology, we are one of the first authors in the DLSCA field (see Figure 1.6) to open our research by making publicly available the source code[h] [ZBHV19a, ZBD+20a, ZBHV21a, ZBC+21a] of our contributions we describe in the following section.

1.4 Contributions

This section summarizes the contributions we propose in this manuscript and describes the outline of the thesis.

Bridging deep learning and classical profiled side-channel attacks. While the first papers in DLSCA were released in 2013, no particular investigations were made to connect the side-channel and deep learning's worlds. In Chapter 5, we reduce this theoretical gap by first, linked the stochastic models proposed by *Schindler et al.* [SLP05] with the generative approach commonly used by the deep learning community. This rapprochement, proposed in Chapter 5,

[f] `https://neurips.cc/Conferences/2020/PaperInformation/CodeSubmissionPolicy`

[g] `https://github.com/paperswithcode/releasing-research-code`

[h] To reduce this issue, the *Conference on Cryptographic Hardware and Embedded Systems* (CHES) (edition 2021) proposes the so-called *Artifact Evaluation* which supports the open-source and reproducible research within the embedded cryptography's field. Even if this process is currently not taken into account during the submission, this new proposition is a new step towards an open-source research in the side-channel community.

helps us to construct the first generative model that serves to understand the benefits and the limitations of the deep learning-based side-channel attacks. While classical deep learning models are seen as black-box tools, this new network, called *Conditional Variational AutoEncoder-based Stochastic Models* (cVAE-ST), can be fully explained from a side-channel perspective. Presented in Chapter 5, this network is derived from the stochastic models and can easily argue the benefits of classical preprocessing techniques considered by the side-channel community (*i.e.* dimensionality reduction, point of interest selection, synchronization, *etc*). To deal with the Evaluator point of view, we focus a particular interest on the ease of network configuration in order to limit the *elapsed time* criterion. Hence, we define a theoretical complexity bounds of the network and suggest its minimal complexity given a set of point of interests. Then, we demonstrate the ability of the cVAE-ST to approximate a leakage trace and extended its application to a wide range of case study. Finally, we define the benefits and the limitations of this new proposition and suggest some improvements that can be provided by the discriminative approach. The solutions proposed in Chapter 5 have been presented at [ZBC+21b]. In addition, all the experiments provided in this chapter are reproducible [ZBC+21a].

Construction of discriminative models. While the cVAE-ST are beneficial from an evaluation perspective as it is fully interpretable, the classical DLSCA approach, namely discriminative models, are suggested as more powerful than generative models [NJ02]. However, as the design of such neural network architecture is an arduous task, it is crucial to deeply understand how does the Evaluator have to configure them in order to exploit the security flaws of cryptographic module. Hence, in Chapter 6, we propose to decompose the most widely use network, namely a *Convolutional Neural Network* (CNN), into a convolutional and a classification part in order to deeply characterize the impact of some hyperparameters that compose the convolutional part. Assessing its impact to retrieve the point of interests helps us to construct a new convolutional part that drastically reduces the network complexity without altering the point of interest detection and the resulted performance. In a nutshell, we propose to reduce the desynchronization effect by focusing the network on the point of interests only and reducing the effect of the uninformative time samples. This result is in contrast to the DLSCA literature that is mostly inspired from the computer vision field. While our contribution outperforms the results obtained by the state-of-the-art, it questions the need of complex networks to perform efficient side-channel attacks. The solutions proposed in Chapter 6 have been presented at *CHES* and published in the journal *IACR Transactions on Cryptographic Hardware and Embedded Systems* (TCHES) [ZBHV19b]. In addition, all the experiments provided in this section are reproducible [ZBHV19a].

To obtain the most powerful models, the Evaluator has to optimize the discriminative approach in order to converge towards the optimal Adversary. From an evaluation perspective, converging towards the optimal Adversary is beneficial in order to fully assess the robustness of the cryptographic module against side-channel attacks. Part III is decomposed into two categories: performance/learning metrics and network combination.

Assessing the efficiency of a generative/discriminative model. To evaluate the relevance of the network, the Evaluator monitors the related performance metric for estimating its efficiency to perform a side-channel attacks. As illustrated in [CDP17a, PHJ+18], the classical deep learning metrics do not fit with the side-channel paradigm. To mitigate this issue, we propose a new performance metric which derives from a well-known side-channel performance metric: *Success Rate*. This proposition indicates the number of iterations the network has to be updated in order to attain its best side-channel performance. Hence, during the training process, the Evaluator can assess the efficiency of the generated model. The solutions proposed in [RZC+21] have been presented at *International Workshop on Constructive Side-Channel Analysis and Secure Design* (COSADE) and published in the proceedings of the international conference. This contribution will not be presented in this thesis as it was mainly handled by another co-author

Damien Robissout.

Convergence towards the optimal models. Then, in Chapter 7, we extend the work provided by Masure *et al.* [MDP19b] which determines how far a DLSCA model converges towards the optimal side-channel solution. Actually, the most classical loss function used for training models in DLSCA, namely the *Negative Log Likelihood*, is not optimal. To overcome this drawback, we propose a new loss function which derives from the *Success Rate* described in [SMY09]. We demonstrate the benefits of this metric from a side-channel and a deep learning perspective in order to assess its suitability for the Evaluator. Considering our contribution maximizes the success rate of a side-channel attack by minimizing the ranking error of the secret key in comparison with all other key hypotheses. It reduces the errors provided by the cross-entropy loss function and generate a model converging towards the optimal solution. From an evaluation perspective, it is therefore possible to precisely assess the robustness of cryptographic module against side-channel attacks. Published in the journal IACR TCHES [ZBD+20b], this new proposition is finally experimentally compared with the most classical metric used in DLSCA context. In addition, all the experiments provided in this section are reproducible [ZBD+20a].

Benefits of diversity to enhance side-channel attacks. While a single network can be limited to retrieve relevant information related to the secret key, the combination of multiple networks can handle this problem. Known as *Ensembling* approach, it consists in mixing of several discriminative models in order to reduce its overall error. For being effective, this approach has to aggregate the result of *diverse* models, *i.e.* with uncorrelated errors. In Chapter 8, we propose a new loss generating interactions between the networks in order to reduce as much as possible the correlated errors. Hence, from the Evaluator perspective, this proposition can be useful to increase the performance of the related side-channel attack. While most of the literature focuses its interest on the symmetric implementations only, the Evaluator also has to cope with asymmetric implementation. Hence, we propose to evaluate the benefits of our contribution on cryptographic modules implementing asymmetric algorithms (*i.e.* RSA and ECC). During an evaluation, a side-channel attack against asymmetric implementations is rarely sufficient to retrieve the entire bit of the secret key. Typically, the Evaluator has to combine side-channel and partial attacks to extract the secret key from the cryptographic module. In order to assess the suitability of the proposed loss in the evaluation process, we evaluate the impact of the performance gain on the remaining partial attack under different scenarios. The solutions proposed in Chapter 8 have been presented at *CHES* and published in the journal IACR TCHES [ZBHV21b]. In addition, all the experiments provided in this section are reproducible [ZBHV21a].

To introduce these contributions, Chapter 3 proposes a general introduction to *Side-Channel Analysis* and presents some classical models and metrics used during an evaluation. A general survey of the main attacks and countermeasures is defined. Then, Chapter 4 describes the general notion of *Deep Learning* as well as the statistical learning theory which defines the problem of finding a predictive function based on a set of data. These chapters are preceded by a formal description of the mathematical background we consider in this manuscript and is introduced in Chapter 2 which follows.

Chapter 2

Mathematical Background

In this chapter, some technical backgrounds are recalled. This chapter also fixes the notation used in the rest of this thesis.

Contents

2.1 Notations

Let calligraphic letters $\mathcal{X}$ denote sets such that, if $\mathcal{X}$ is finite, its cardinality, denoted $|\mathcal{X}|$, defines its number of elements. The corresponding capital letters X (resp. bold capital letters) denote random variables (resp. random vectors $\mathbf{T}$). The lowercase x (resp. $\mathbf{t}$) denote the realization of X (resp. $\mathbf{T}$). The i^{th} entry of a vector $\mathbf{t}$ is defined as $\mathbf{t}[i]$. Finally, $\mathbf{t}_{[i:j]}$ denotes a subvector of $\mathbf{t}$ such that it is characterized by the values between the i^{th} entry and the j^{th} entry of $\mathbf{t}$.
A side-channel measurement will be constructed as a random vector $\mathbf{T} \in \mathbb{R}^D$ where D defines the dimension of the related leakage trace. The targeted sensitive variable, denoted $Y = f(X, k^*)$, depends on a cryptographic primitive $f : \mathcal{X} \times \mathcal{K} \to \mathcal{Y}$, a public variable $X \in \mathcal{X}$ (*e.g.* plaintext or ciphertext) and a part of the secret key $k^* \in \mathcal{K}$ (*e.g.* byte) that the Evaluator tries to retrieve. We will denote $Y_k = f(X, k)$ a hypothetical sensitive variable using a hypothetical key $k \in \mathcal{K}$ such that $Y_k = Y$ if $k = k^*$.

2.2 Basics on Finite Field & Boolean Algebra

This section recalls the main definitions of finite field and Boolean algebra.

Definition 2.2.1 (Field)**.** A field is a set $\mathbb{F}$ with two operations $+$ and $\times$ satisfying the following properties:

- $\mathbb{F}$ is an Abelian group under $+$ with identity element 0,
- The non-zero elements of $\mathbb{F}$ form an Abelian group under $\times$, with identity element 1,
- Distributivity of multiplication over addition, *i.e.* $a \times (b + c) = a \times b + a \times c$ for any a, b, $c \in \mathbb{F}$.

The number of elements in a field $\mathbb{F}$ is called the *order* of $\mathbb{F}$. A *finite field* is simply a field whose underlying set has a finite number of elements. In the following, a finite field of order p is denoted $\mathbb{F}_p$, and, for any positive n, we denote $\mathbb{F}_2^n$ the vector space over $\mathbb{F}_2$ of dimension n.

Definition 2.2.2 (Binary representation of an integer)**.** Let a be an integer such that its binary representation is $(a[n-1]a[n-2]\dots a[0])_2$ with:

$$a = \sum_{i=0}^{n-1} 2^i \cdot a_i,$$

with $a_i \in \{0,1\}$. We define $a[0]$ as the *least significant bit* (or *LSB*) value while $a[n-1]$ denotes the *most significant bit* (or *MSB*) value.

The exclusive or (XOR) operator is denoted $\oplus$ and is defined for every $(a,b) \in \{0,1\}$ as follows:

$$a \oplus b = \begin{cases} 1 & \text{if} \quad a \neq b, \\ 0 & \text{if} \quad a = b. \end{cases}$$

The AND operator, denoted $\wedge$, is defined for every $(a,b) \in \{0,1\}$ as follows:

$$a \wedge b = \begin{cases} 1 & \text{if} \qquad a = b = 1, \\ 0 & \text{otherwise.} \end{cases}$$

For every $a = (a[n-1]a[n-2]\dots a[0])_2 \in \mathbb{F}_2^n$ and $b = (b[n-1]b[n-2]\dots b[0])_2 \in \mathbb{F}_2^n$, the vectorial XOR (resp. AND), often called *bitwise addition* (resp. *bitwise multiplication*), can be expressed as:

$$a \bigstar b = (a[n-1] \bigstar b[n-1] \dots a[0] \bigstar b[0])_2,$$

with $\bigstar$ denoting $\oplus$ or $\wedge$.

2.3 Basics on Linear Algebra & Vector Analysis

As the linear algebra is essential for understanding and working with many machine learning algorithms, this section introduces the notions used throughout this manuscript.

Definition 2.3.1 (Matrix)**.** A matrix A is a $2D$-array defined in rows and columns such that $\mathcal{M}_{n,m}(\mathbb{R})$ denotes the matrix space of n rows and m columns so that each coefficient $(a_{ij})_{0 \leq i < n, 0 \leq j < m}$ is in $\mathbb{R}$.

One specific operation on matrices is the *transpose*. It characterizes the mirror image of a matrix across its main diagonal axis. The transpose operation of a matrix $A \in \mathcal{M}_{n,m}(\mathbb{R})$ is denoted $A^T \in \mathcal{M}_{m,n}(\mathbb{R})$ such that each of its coefficient at the i^{th} row and the j^{th} column is defined as $a_{ij}^T = a_{ji}$. A matrix A is known as *symmetric* if $A^T = A$. In other words, A is symmetric if the mirror elements with respect to the main diagonal axis are equal (*i.e.* $a_{ij} = a_{ji}$).

One special case matrix is the *square diagonal matrix*, which is defined to have all of its elements equal to zero except those on the main diagonal. Thus, a square diagonal matrix $A \in \mathcal{M}_{n,n}(\mathbb{R})$ can be written as follows:

$$A = \underbrace{\left.\begin{bmatrix} a_{00} & 0 & 0 & \cdots & 0 \\ 0 & a_{11} & 0 & \cdots & 0 \\ 0 & 0 & a_{22} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & a_{n-1\ n-1} \end{bmatrix}\right\} n \text{ rows}}_{n \text{ columns}}$$

The *trace* of a square diagonal matrix A is an operator that gives the sum of all the diagonal entries of A such that:

$$tr(A) = \sum_{i=0}^{n-1} a_{ii}.$$

A particular square diagonal matrix is the *square identity matrix* $\mathbf{I}$, which is defined to have all of its coefficients $(a_{ii})_{0 \leq i < n}$ equal to 1 (*i.e.* $tr(A) = n$). In the rest of this manuscript, $\mathbf{I}_n$ denotes a square identity matrix in $\mathcal{M}_{n,n}(\mathbb{R})$.

While elementary matrix addition and matrix multiplication are defined, the matrix division does not hold. However, for a square matrix, multiplication by its inverse may be thought as an analogous operation. Let A be a square matrix in $\mathcal{M}_{n,n}(\mathbb{R})$, we denote A^{-1} its inverse if the following relationship hold:

$$A^{-1}A = AA^{-1} = \mathbf{I}_n.$$

The readers should notice that the matrix multiplication is not commutative, *i.e.* the condition $AB = BA$, such that $A \in \mathcal{M}_{m,n}(\mathbb{R})$ and $B \in \mathcal{M}_{n,p}(\mathbb{R})$, is not always true. The importance of the inverse matrix can be seen from the solution of a set of algebraic linear equations such as:

$$Ax = b,$$

If the inverse A^{-1} exists, then the previous equation can be rewritten as follows:

$$\begin{aligned} A^{-1}Ax &= A^{-1}b, \\ \mathbf{I}x &= A^{-1}b, \\ x &= A^{-1}b. \end{aligned}$$

The inverse of a matrix does not always exist. If a matrix $A \in \mathcal{M}_{n,n}(\mathbb{R})$ is invertible, it is defined to be *non-singular*. If A^{-1} does not exist, the matrix is *singular*.
One common solution to verify if A is a singular matrix consists in computing its *determinant*, denoted $|A|$. If $|A| = 0$, then space is completely reduced along at least one dimension and thus, induces a singular matrix.

Norm. Sometimes we will need to measure the distance of a vector $\mathbf{x} \in \mathbb{R}^n$, from the origin, through the computation of the $L_p -$ norm. The related distances are also called *Minkowski distances*.

Definition 2.3.2 ($L_p -$ norm)**.** Given a vector $\mathbf{x} \in \mathbb{R}^n$, the $L_p -$ norm is defined as follows:

$$||x||_p = \left(\sum_{i=0}^{n-1} |x[i]|^p \right)^{\frac{1}{p}},$$

for $p \in \mathbb{R}^+$.

Different norms on a vector space can give rise to different geometrical and analytical structures. In Figure 2.1, we can see the shape of the $L_p -$ norm for various values of p such that $||x||_p = 1$. In particular, the following three specific norms are frequently used:

- $L_1 -$ **norm** – $||x||_1 = \sum_{i=0}^{n-1} |x[i]|$.
- **Euclidean Norm** – $||x||_2 = \sqrt{\sum_{i=0}^{n-1} |x[i]|^2}$.
- $L_\infty -$ **norm** – $||x||_\infty = \lim_{p \to \infty} \left(\sum_{i=0}^{n-1} |x[i]|^p \right)^{\frac{1}{p}}$.

While the Euclidean norm is classically used in Machine Learning, it can be undesirable because it increases very slowly near the origin. While it is important to discriminate the very small values from those that exactly equal zero, the application of the $L_1 -$ norm can be beneficial. Thus depending on the scenario, the p value should differ.

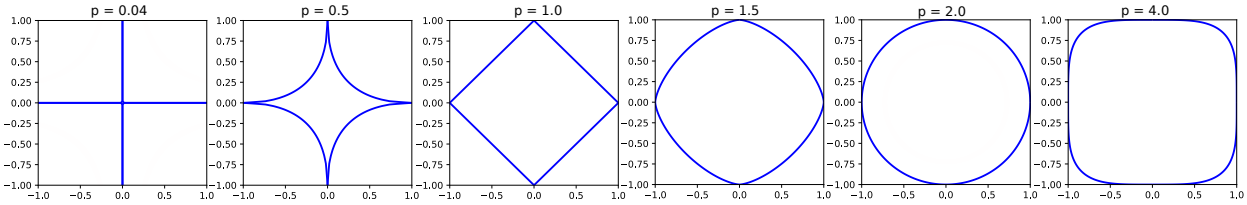


Figure 2.1: L_p- norm visualization for two dimensional vector such that $||x||_p = 1$.

Gradient and Jacobian matrix. Given a function $f : \mathbb{R}^n \to \mathbb{R}$, $\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}[i]}$ defines the partial derivative of f with respect to the i^{th} entry of a random vector $\mathbf{x} \in \mathbb{R}^n$. This partial derivative measures how the function f is affected as only the element $\mathbf{x}[i]$ varies from $\mathbf{x}$. The *gradient*, denoted $\nabla_{\mathbf{x}} f(\mathbf{x})$, generalizes this notion to a vector $\mathbf{x}$ such that:

$$\nabla_{\mathbf{x}} f(\mathbf{x}) = \left[\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}[0]}, \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}[1]}, \ldots, \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}[n-1]} \right]. \tag{2.1}$$

When we move to derivatives of a function $f : \mathbb{R}^n \to \mathbb{R}^m$, then, the matrix containing all the partial derivatives is known as the *Jacobian matrix* and it is denoted $J_f \in \mathcal{M}_{m,n}(\mathbb{R})$ such that:

$$J_f(\mathbf{x}) = [\nabla_{\mathbf{x}} f(\mathbf{x})[0], \nabla_{\mathbf{x}} f(\mathbf{x})[1], \ldots, \nabla_{\mathbf{x}} f(\mathbf{x})[m-1]]^T. \tag{2.2}$$

The Gradient and the Jacobian matrix are essentially used to solve optimization problems in Section 4.3.

2.4 Basics on probability theory

In this section, we introduce all the needed notations related to the probability theory.

Recall on probability. The probability of observing an event X is denoted by $\mathsf{Pr}[X]$ such that a conditional probability of observing an event X knowing an event Y is denoted $\mathsf{Pr}[X|Y]$. Besides, the indicator function, denoted by $\mathbb{1}_X$, equals 1 if the event X is realized and 0 otherwise.

In probability theory and statistics, a *probability distribution* is a statistical function that describes all the possible values and likelihoods that a random variable X can take within a given range. If X denotes a continuous random variable, the related probability distribution is computed from a *Probability Density Function* (PDF), denoted by f_X, and that satisfies, for every $a \in \mathcal{X}$ and $b \in \mathcal{X}$:

$$\mathsf{Pr}\left[a \leq X \leq b\right] = \int_{[a,b]} f_X(x)dx,$$

and such that $\int_{\mathcal{X}} f_X(x)dx = 1$ and for every $x \in \mathcal{X}$, $f_X \geq 0$.
Otherwise, every discrete random variable X is associated with a *Probability Mass Function* (PMF) $f_X : x \to \mathsf{Pr}[X = x]$. In the rest of the manuscript, the set of every PMF is denoted by $\mathcal{P}(\mathcal{X})$. The notation $x \to \mathsf{Pr}[x]$ for a continuous random variable X may be further used without ambiguity to denote the PDF of X.

Sometimes, it can be useful to compute the *joint probability distribution* of two, or more, events such that it characterizes the probability distribution of the intersection of those events. Typically, given two discrete random variables X and Y, the related joint probability mass function can be expressed as:

$$\mathsf{Pr}\left[X = x, Y = y\right] = \mathsf{Pr}\left[Y = y | X = x\right] \cdot \mathsf{Pr}\left[X = x\right] = \mathsf{Pr}\left[X = x | Y = y\right] \cdot \mathsf{Pr}\left[Y = y\right],$$

for every $x \in \mathcal{X}$, $y \in \mathcal{Y}$ and such that $\mathsf{Pr}[Y = y|X = x]$ (resp. $\mathsf{Pr}[X = x|Y = y]$) denotes the *conditional probability* of observing the event Y (resp. X) knowing the random variable X (resp. Y). In particular, $\mathsf{Pr}[X = x, Y = y] = \mathsf{Pr}[X = x] \cdot \mathsf{Pr}[Y = y]$ if and only if X and Y are *independent* and the joint probability is then the product of the two marginal probabilities. If both random variables are *conditionally independent* given a random variable Z, then, $\mathsf{Pr}[X = x, Y = y|Z = z] = \mathsf{Pr}[X = x|Z = z] \cdot \mathsf{Pr}[Y = y|Z = z]$.

Given the joint probability distribution of two discrete random variables X and Y, it is possible to compute the conditional probability $\mathsf{Pr}[X = x|Y = y]$ following *Bayes' Theorem*:

$$\mathsf{Pr}\left[X = x|Y = y\right] = \frac{\mathsf{Pr}\left[Y = y|X = x\right]\mathsf{Pr}\left[X = x\right]}{\mathsf{Pr}\left[Y = y\right]}.$$

In many occasions, we may know the joint probability distribution of two (continuous or discrete) random variables X et Y and we want to characterize the probability distribution of each random variable independently (*e.g.* $\mathsf{Pr}[X = x]$ for every $x \in \mathcal{X}$). This probability distribution is known as the *marginal probability distribution* and can be computed for every $x \in \mathcal{X}$ as follows:

$$\mathsf{Pr}\left[X = x\right] = \sum_{y \in \mathcal{Y}} \mathsf{Pr}\left[X = x, Y = y\right].$$

Moments. The moments of a random variable X are quantities providing information about the shape and location of its PMF (discrete case) or its PDF (continuous case). $\mathbb{E}[X]$ is used to denote the *expected value* of a random variable X such that, if X is a discrete random variable, then:

$$\mathbb{E}[X] = \sum_{x \in \mathcal{X}} x \cdot \mathsf{Pr}\left[X = x\right],$$

while for continuous random variables, it is computed as follows:

$$\mathbb{E}[X] = \int_{\mathcal{X}} x \cdot \mathsf{Pr}\left[x\right] dx.$$

The expected value is notably used to measure the location or the central tendency of $\mathcal{X}$. $\mathbb{E}_{X \sim \mathcal{P}}$ defines under which probability distribution it is computed.
In addition, the second central moment, also known as the *variance*, of a random variable X is defined as:

$$\mathbb{V}[X] = \mathbb{E}\left[\left(X - \mathbb{E}\left[X\right]\right)^2\right] = \mathbb{E}\left[X^2\right] - \mathbb{E}\left[X\right]^2.$$

The symbol $\mathbb{V}_X[f(X)]$ (resp. $\mathbb{E}_X[f(X)]$) denotes the variance (resp. expected value) of a function f of the random variable X, over the distribution of X.
The *standard deviation* of a random variable X, denoted σ_X, is defined as the square root of its variance:

$$\sigma_X = \sqrt{\mathbb{V}\left[X\right]}.$$

The *covariance* $Cov[X, Y]$ between two random variables X and Y is defined by:

$$Cov\left[X, Y\right] = \mathbb{E}\left[\left(X - \mathbb{E}\left[X\right]\right)\left(Y - \mathbb{E}\left[Y\right]\right)\right] = \mathbb{E}\left[XY\right] - \mathbb{E}\left[X\right]\mathbb{E}\left[Y\right].$$

This equation implies $Cov[X, X] = \mathbb{V}[X]$. If $\mathbf{X}$ denotes a D-dimensional random variable $\mathbf{X} = (X_0, X_1, \ldots, X_{D-1})$, its expectation is defined as the vector composed of its coordinate expectations. The related *covariance matrix* $\Sigma_{\mathbf{X}}$ of a random vector $\mathbf{X}$ is an extension of the variance to a multivariate random variable that is defined as follows:

$$\Sigma_{\mathbf{X}} = \left(Cov\left[X_i, X_j\right]\right)_{0 \leq i < D, 0 \leq j < D}.$$

Probability distributions. Through this thesis two probability distributions are essentially used:

- **Discrete Uniform Distribution** – A random variable X follows a *discrete uniform distribution* over $\mathcal{X}$, if, each element x of the finite set $\mathcal{X}$ has an equal probability $\mathsf{Pr}[X = x] = \frac{1}{|\mathcal{X}|}$.

- **Gaussian Distribution** – Also known as the *normal distribution*, we use the symbol $X \sim \mathcal{N}_D(\mu, \sigma^2)$ to denote a random variable X that follows an *univariate Gaussian distribution* of parameters $\mu \in \mathbb{R}$ and $\sigma \in \mathbb{R}^+$ such that the corresponding PDF is defined by:

$$f_X(x) = \frac{1}{\sqrt{2\pi \cdot \sigma^2}} \exp\left\{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right\}.$$

  The *multivariate Gaussian distribution* of dimension D with expectation vector $\boldsymbol{\mu} \in \mathbb{R}^D$ and the covariance matrix $\Sigma \in \mathcal{M}_{D,D}(\mathbb{R})$ is denoted $\mathcal{N}_D(\boldsymbol{\mu}, \Sigma)$ such that the related PDF f_X can be defined as follows:

$$f_{\mathbf{X}}(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^D \cdot |\Sigma|}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x}-\boldsymbol{\mu})\right\}. \tag{2.3}$$

  We define $\mathbf{X} \sim \mathcal{N}_D(\boldsymbol{\mu}, \Sigma)$ to denote a random variable X that follows a multivariate Gaussian distribution of parameters $\boldsymbol{\mu} \in \mathbb{R}^D$ and $\Sigma \in \mathcal{M}_{D,D}(\mathbb{R})$.

2.5 Information theory

In this section, we define the information theory quantities needed in the rest of this thesis.

In information theory, the *Shannon entropy* of a discrete random variable X measures the amount of information contained by a realization of X.

Definition 2.5.1 (Entropy)**.** Given a discrete random variable X, the entropy of X, denoted $H(X)$, can be computed as follows:

$$H(X) = -\sum_{x \in \mathcal{X}} \mathsf{Pr}\left[X = x\right] \cdot \log_2\left(\mathsf{Pr}\left[X = x\right]\right).$$

From this notation, we can extend the Shannon entropy to the conditional entropy of a random variable X knowing Y is defined by:

$$\begin{aligned} H(X|Y) &= \sum_{\mathbf{y} \in \mathcal{Y}} \mathsf{Pr}\left[Y = y\right] \cdot H\left(X|Y = y\right) \\ &= -\sum_{\mathbf{y} \in \mathcal{Y}} \mathsf{Pr}\left[Y = y\right] \cdot \sum_{x \in \mathcal{X}} \mathsf{Pr}\left[X = x|Y = y\right] \cdot \log_2\left(\mathsf{Pr}\left[X = x|Y = y\right]\right). \end{aligned}$$

To quantify how much information can be extracted about X by observing a second random variable Y, the notion of *Mutual Information* can be used.

Definition 2.5.2 (Mutual Information)**.** Let (X, Y) be two discrete random variables with values over the space $\mathcal{X} \times \mathcal{Y}$. The mutual information, denoted by $MI\left(X;Y\right)$, between those two random variables can be expressed as follows:

$$\begin{aligned} MI\left(X;Y\right) &= H\left(Y\right) - H\left(Y|X\right) \\ &= H\left(Y\right) + \sum_{y \in \mathcal{Y}} \mathsf{Pr}\left[Y = y\right] \cdot \sum_{x \in \mathcal{X}} \mathsf{Pr}\left[X = x|Y = y\right] \cdot \log_2\left(\mathsf{Pr}\left[Y = y|X = x\right]\right) \\ &= D_{KL}(\mathsf{Pr}[X,Y] || \mathsf{Pr}[X]\mathsf{Pr}[Y]), \end{aligned}$$

where the D_{KL} function is known as the Kullback-Leibler (KL-) divergence [KL51]. The KL-divergence is always non-negative and equals zero if and only if $\mathsf{Pr}[X, Y] = \mathsf{Pr}[X]\mathsf{Pr}[Y]$.

From this result, we can introduce the conditional mutual information that quantifies the amount of information Y reveals on X given a random variable Z as follows:

$$\begin{aligned} MI(X;Y|Z) &= H(Y|Z) - H(Y|X,Z) \\ &= \sum_{z\in\mathcal{Z}} \mathsf{Pr}[Z=z] \sum_{(x,y)\in\mathcal{X}\times\mathcal{Y}} \mathsf{Pr}[X=x, Y=y|Z=z] \ \log\left(\frac{\mathsf{Pr}[X=x,Y=y|Z=z]}{\mathsf{Pr}[X=x|Z=z]\mathsf{Pr}[Y=y|Z=z]}\right). \end{aligned}$$

Introduced by McGill [McG54], *interaction information* is a multivariate generalization of mutual information for measuring dependence among multiple random variables. The interaction information $MI(\{X_0, X_1, \cdots, X_n\})$ between $n+1$ random variables $\{X_0, X_1, \cdots, X_n\}$, denoted as $\{X_{0:n}\}$ in the following sections, and the conditional interaction information $MI(\{X_{0:n}\}|Y)$ are respectively defined as:

$$MI(\{X_{0:n}\}) = \begin{cases} MI(X_0; X_1) & \text{if} \quad n = 1, \\ MI(\{X_{0:n-1}\}|X_n) - MI(\{X_{0:n-1}\}) & \text{for} \quad n \geq 2. \end{cases}$$

$$MI(\{X_{0:n}\}|Y) = \mathbb{E}_Y\left[MI(\{X_{0:n}\})|Y\right].$$

Those notions of information theory are essentially used in Chapter 7 and Chapter 8.

In the next chapter, we continue to introduce the notions required for this thesis by focusing on the *Side-Channel Analysis* field.

Chapter 3

Side-Channel Analysis

In this chapter, we describe the link between physical leakage and data dependence. Then, we present the entire process the Evaluator conducts during the security evaluation. This includes a description of the measurement setup as well as statistical tools allowing the detection of time samples where the targeted data leak. This chapter continues with an introduction of the optimal side-channel attack and describes the classical approaches considered to retrieve a secret information. Finally, classical countermeasures, which make the cryptographic module more robust against side-channel attacks, are presented. This chapter is an overview of the side-channel literature, hence, readers who are already familiar with this field can skip it, although the material may be useful to get familiar with the notations and terminologies used in this manuscript.

Contents

3.1 Power Consumption & Data Dependency

3.1.1 CMOS Circuit

To extract the information from a cryptographic module, the Evaluator exploits the security flaws induced by the physical component. Most of the chips are implemented using the *Complementary Metal Oxide Semiconductor* (CMOS) technology to perform logical operation. One of the simplest examples of the CMOS is the inverter with lumped-C (see Figure 3.1). Composed by two transistors, namely a *P-channel Metal Oxide Semiconductor* (PMOS) and a *N-channel Metal Oxide Semiconductor* (NMOS), the power consumption of the CMOS inverter depends on a static and a dynamic power consumption, respectively denoted as P_{stat} and P_{dyn}. P_{dyn} is characterized by the switching of logic cells while P_{stat} denotes the circuit consumption when no process of switching occurs. For a given moment of time, four input transitions can be observed (*i.e.* $0 \to 0$, $0 \to 1$, $1 \to 0$, $1 \to 1$). In two cases (*i.e.* $0 \to 0$, $1 \to 1$), the input of the CMOS circuit stays constant and its consumption only depends on P_{stat}, while in the other two cases (*i.e.* $0 \to 1$, $1 \to 0$), the combination of P_{stat} and P_{dyn} is consumed. Hence, the consumption of the CMOS circuit depends on the transition it performed. If an input transition $1 \to 0$ is observed, PMOS (resp. NMOS) turns to *ON* (resp. *OFF*) and charges a capacitor C_L. In this setting, a direct path exists between V_{out} and V_{DD}, resulting in a steady-state value of 1. On the other hand, if an input transition $0 \to 1$ is observed, NMOS (resp. PMOS) turns to *ON* (resp. *OFF*) and discharges C_L. Thus, V_{out} is modified to output a 0 value. This short introduction highlights the dependence between the data manipulated by the cryptographic module and its related power consumption [MOP07]. If the Evaluator captures a power consumption resulting from a cryptographic function (*e.g.* encryption, decryption), he can correlate it with the data processed and thus, guess the secret key. To perform this correlation, he has to model the expected behavior of the targeted cryptographic model in order to enhance its ability to extract the secret key.

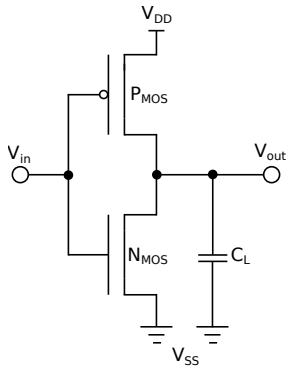


Figure 3.1: Structure of a CMOS inverter.

3.1.2 Leakage Model

Performing a side-channel attack consists in mapping data values, which are processed by the cryptographic module, to the related power consumption. For exploiting the dynamic power consumption, the Evaluator has to consider a metric to characterize the following transitions: $0 \to 1$ and $1 \to 0$.

Definition 3.1.2.1 (Hamming Distance)**.** Let $a \in \mathbb{F}_2^n$ and $b \in \mathbb{F}_2^n$, the Hamming Distance between a and b is defined as follows:

$$\mathrm{HD}(a,b) = \sum_{i=0}^{n-1} \mathbb{1}_{a[i] \neq b[i]},$$

where $a[i]$ (resp. $b[i]$) denotes the i^{th} bit of a (resp. b).

The basic idea of the Hamming Distance is to count the number of transitions that occurs when $a \rightarrow b$. Hence, for two consecutive values, this metric correlates this number with the related power consumption. However, even if this metric is suited to describe the leakage consumption, it requires the knowledge of the preceding and the succeeding data values. Consequently, if the Evaluator does not have enough knowledge to predict the transition, he has to consider a simpler solution.

Definition 3.1.2.2 (Hamming Weight)**.** Let $a \in \mathbb{F}_2^n$, the Hamming Weight of a is defined as:

$$\mathrm{HW}(a) = \sum_{i=0}^{n-1} \mathbb{1}_{a[i]=1},$$

where $a[i]$ denotes the i^{th} bit of a.

Remark 3.1.2.1. Let $a \in \mathbb{F}_2^n$ and $b \in \mathbb{F}_2^n$, $\mathrm{HD}(a, b) = \mathrm{HW}(a \oplus b)$.

As defined for the Hamming Distance, the Hamming Weight describes the number of transitions that occur when $a \rightarrow b$ such that a equals 0. Thus, the Evaluator assumes that the power consumption and the processed data are proportional to the number of bits set to 1. Due to the assumption on a, this metric does not entirely characterize the power consumption of the CMOS circuit because it does not describe the transitions involved. Hence, depending on its knowledge, the Evaluator has to adequately choose the best *leakage model*. Typically, the leakage model produced by the computation of a data in $\mathbb{F}_2^n$ is modelled by a D-size random vector over $\mathbb{R}^D$. If $D = 1$, the leakage model is said *univariate*, otherwise if $D > 1$, the leakage model is said *multivariate*.

Definition 3.1.2.3 (Leakage Model)**.** A leakage model is a function $\psi : \mathbb{F}_2^n \rightarrow \mathbb{R}^D$ characterizing the dependence between the data manipulated by the cryptographic module and the related power consumption.

Even if the most classical leakage models are defined by the Hamming Distance and the Hamming Weight, a plethora of solutions exists:

- *Mono-bit* leakage model – instead of considering the entire data, the Evaluator can describe the activity induced by a single bit of the processed data.
- *Linear* leakage model – in [SLP05], Schindler *et al.* propose a custom leakage model that is a linear combination of bits. The particularity of this solution is to capture the bits' interaction of the processed data in order to find a leakage model with a lowest granularity level. Notably, the Hamming Weight can be seen as a solution of the linear leakage model.

All these propositions will be considered in this manuscript. However, how does the Evaluator capture the power consumption from a chip implemented with the CMOS technology? How can he exploit the security flaws we revealed based on a leakage model ψ? The following section tries to answer these questions.

3.2 Side-Channel Evaluation

3.2.1 Measurements

Figure 3.2 presents a typical setup used by the Evaluator to assess the robustness of cryptographic modules against side-channel attacks. To perform this evaluation, at least, an oscilloscope, a personal computer (PC), a device under test (*i.e.* cryptographic module) and one probe capturing the power consumption or the electromagnetic emanations are needed. First, it is assumed that the Evaluator (PC's owner) has a physical access to the cryptographic module. Then, he sends a request to the device under test in order to perform a cryptographic function (*e.g.* encryption,

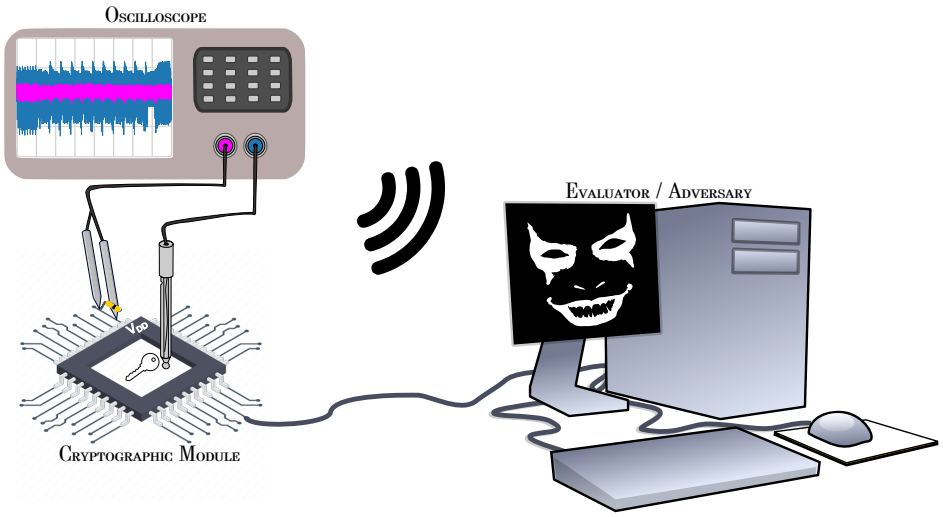


Figure 3.2: Measurement setup to acquire power consumption and electromagnetic emanations.

decryption) and obtains the related output (*e.g.* ciphertext, signature). During the processed of the cryptographic function, the oscilloscope captures the corresponding signal with the help of a probe. If the Evaluator wants to capture the power consumption of the cryptographic module, a solution consists in the insertion of a small resistor in series with the power supply or the ground. Connected to the oscilloscope, the current probe measures the voltage difference across the resistor. This channel captures the entire power consumption of the device under test. The resulted signal is not only dependent on the targeted cryptographic function but also on the other operations performed simultaneously. This noise, namely *switching noise*, refers to the variations induced by cells that do not depend on the targeted cryptographic function. Depending on its prominence, the quality of the signal can be highly impacted. A solution to reduce this risk is to precisely locate the cryptographic function's activity. To this end, the Evaluator can use a probe that captures the electromagnetic fields of the cryptographic module in order to reveal more information about the targeted cryptographic function. To measure an isolated signal emitted by the device under test (*e.g.* encryption, decryption), the Evaluator has to position its probe near to the cryptographic core. However, as the signal highly varies with the position of the probe, an incorrect probe's location can significantly decrease the effectiveness of a side-channel attack. While the performance of a side-channel attack highly depends on the noise induced in the exploited signal, it is essential to adequately select the channel.
To validate the probe position, the Evaluator can visualize the related signal and assess its suitability regarding the targeted cryptographic function. In Figure 3.3, the Evaluator captures the signal corresponding to an AES-128 encryption. As illustrated, the 10 rounds of the cryptographic algorithm can be clearly observed as well as the cryptographic primitive, namely *AddRoundKey*, *SubBytes* and *MixColumns*. Since *ShiftRows* do not handle arithmetic operations, it cannot be visualized on the power consumption. With the help of the signal, the Evaluator reveals the different operations occurring during the encryption of a message. Based on this representation, he extracts some information that depends on the secret key manipulated by the cryptographic module. This extraction procedure will be defined in Section 3.3. In the following, no particular distinction will be made between electromagnetic emanations and power consumption.

Definition 3.2.1.1 (Leakage Trace)**.** A leakage trace, denoted $\mathbf{T}$, defines a physical measurement (*e.g.* power consumption, electromagnetic emanation) which depends on the secret key manipulated by the cryptographic module under test.

Given a variable Y and an independent noise $\mathbf{Z}$, a leakage traces $\mathbf{T}$ is a D-dimensional random vector $\{\mathbf{T}[0], \ldots, \mathbf{T}[D-1]\}$ where $\mathbf{T}[i]$ represents the leakage of time sample i (for $0 \leq i < D$)

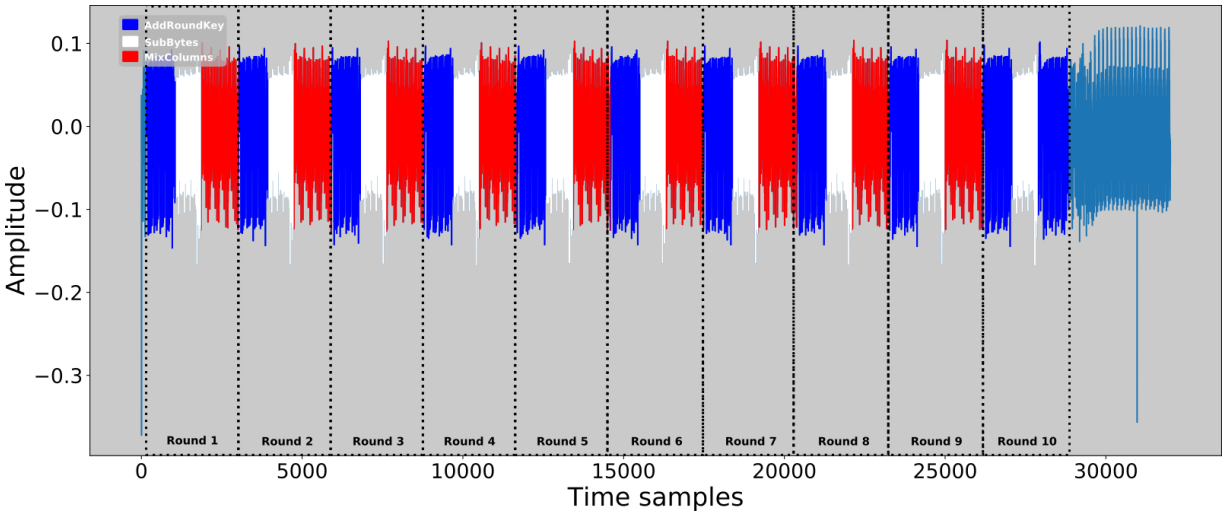


Figure 3.3: A power consumption of an AES-128 encryption.

satisfying the *Gaussian Indepedent Noise Assumption*[a]:

$$\mathbf{T} = \psi(Y) + \mathbf{Z}, \tag{3.1}$$

where $\psi(.)$ denotes the leakage model, Y depends on the targeted secret and we assume that the noise $\mathbf{Z}$ follows a multivariate Gaussian distribution $\mathcal{N}_D(\boldsymbol{\mu}, \Sigma)$ with unknown parameters $(\boldsymbol{\mu}, \Sigma)$. Thus, $\mathbf{T}$ can be decomposed into a deterministic (*i.e.* $\psi(Y)$) and a random (*i.e.* $\mathbf{Z}$) part such that the deterministic part is non-negligible if the secret key is manipulated. During the evaluation process, the Evaluator aims at extracting the deterministic part from the leakage trace in order to retrieve the secret used by the cryptographic function. However, how does the Evaluator retrieve the time sample which depends on the secret key? The following section presents the tools the Evaluator considers to perform this analysis.

3.2.2 Leakage Assessment

To assess the security of a cryptographic module, the Evaluator has to estimate its ability to extract the deterministic part defined in Equation 3.1.

Definition 3.2.2.1 (Sensitive variable)**.** A sensitive variable, denoted $Y = f(x, k^*)$, is an internal state of the cryptographic function which depends on a value the Evaluator targets (*e.g.* secret key, secret random value) and a cryptographic primitive $f : \mathcal{X} \times \mathcal{K} \to \mathcal{Y}$ (*e.g.* output of the Sbox, a XOR between a plaintext $x \in \mathcal{X}$ and a secret key $k^* \in \mathcal{K}$).

In that purpose, the Evaluator uses *leakage metrics* which measure the information contained in a leakage trace $\mathbf{T}$ regarding a processed sensitive variable Y.

Definition 3.2.2.2 (Point of interest)**.** Given a leakage trace $\mathbf{T}$, a point of interest is a time sample with a non-negligible deterministic part (*i.e.* $\psi(Y)[i] \neq 0$).

From the Evaluator perspective, the leakage assessment is helpful to find the points of interest (PoIs) and evaluate the security flaws induced in a cryptographic module. Furthermore, it can also be beneficial to validate the observed processing of a leakage trace and the detected point of interest. As an example, in Figure 3.3, if the Evaluator considers the output of the Sbox in the first round as his sensitive variable, he expects to obtain one or multiple PoIs between time samples $1,000$ and $1,700$. Typically, to detect these PoIs, the Evaluator can apply two types of leakage metrics: the *non-specific* and the *specific* leakage metrics.

[a] This means that the Gaussian noise $\mathbf{Z}$ is independent of the variable Y.

Non-Specific leakage metric. Given a set of leakage traces, this metric does not require any prior knowledge on the cryptographic module under test (*i.e.* sensitive variable) or any assumption about how it leaks (*i.e.* leakage model). Mostly applied under black-box restrictions, it gives a level of confidence related to the presence of leakages without specifying its dependence with the secret key. Thus, it does not provide any information about the robustness of the device against physical attacks. However, using this metric is highly beneficial from the Evaluator point of view in order to get a rapid overview of the cryptographic module's behavior (*e.g.* flaws in the implemented countermeasures). A common example considers the *Welch's T-test* [Wel47] to perform this assessment [GJJR11, BCD+13]. Let $\mathcal{T}_A$ and $\mathcal{T}_B$ be two sets of leakage traces respectively associated to fixed and randomly generated sensitive variable (*i.e.* fixed vs. random)[b]. Let also $\mu_A[i]$ (resp. $\mu_B[i]$) and $\sigma_A^2[i]$ (resp. $\sigma_B^2[i]$) be the mean and the variance of the i^{th} time sample of $\mathcal{T}_A$ (resp. $\mathcal{T}_B$). For targeted cryptographic modules, the Welch's T-test associated with the i^{th} time sample is defined as[c]:

$$Ttest(\mathcal{T}_A, \mathcal{T}_B)[i] = \frac{\mu_A[i] - \mu_B[i]}{\sqrt{\frac{\sigma_A^2[i]}{N_A} + \frac{\sigma_B^2[i]}{N_B}}},$$

where N_A (resp. N_B) denotes the number of leakage traces in $\mathcal{T}_A$ (resp. $\mathcal{T}_B$).
The result determines if the sets $\mathcal{T}_A$ and $\mathcal{T}_B$ are drawn from the same distribution (*i.e.* *null hypothesis*). If two sets do not have similar means and variances, the null hypothesis is rejected and the Evaluator can assess that the time samples of $\mathcal{T}_A$ differ from $\mathcal{T}_B$. Following [SM15], this decision can be made following a threshold. Indeed, if $|Ttest| > 4.5$, the null hypothesis can be rejected without any other consideration. This threshold was questioned in [DZD+18] such that it can lead to erroneous results for high-dimensional leakage traces. However, running independent Welch's T-test multiple times can reduce this risk by only considering the leakages that appear at the same time samples. Analyzing the evolution of $Ttest$ over the time samples is highly recommended to visualize the difference induced in both sets of leakage traces. However, even if this solution is quite useful, the observed points of interest are not necessarily beneficial to perform a side-channel attack. While no correlations are made between these leakages and the targeted sensitive variable, the related exploitation can result in a poor performance. To precisely identifying some correlations between the targeted sensitive variable and the leakage traces, the Evaluator has to apply specific leakage metrics.

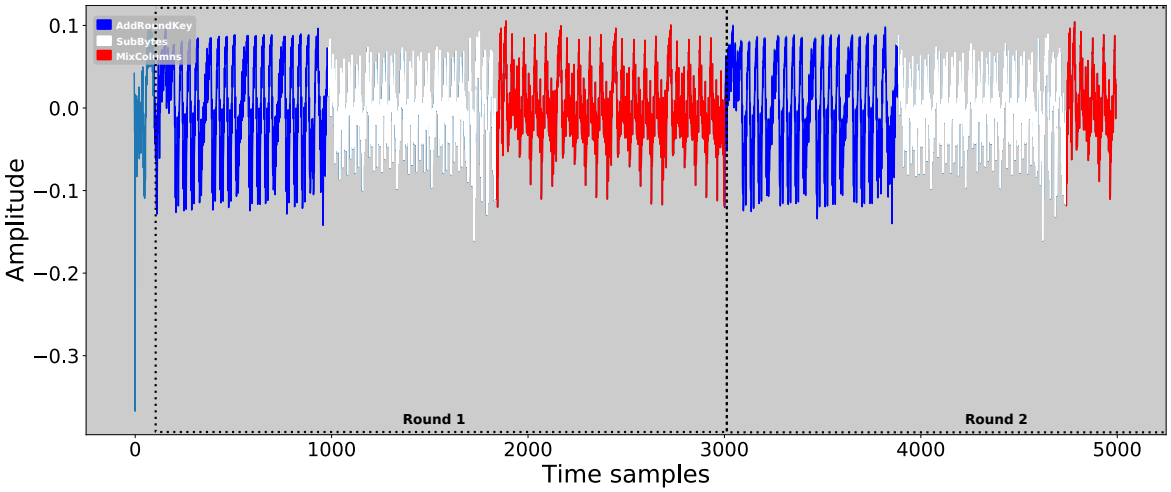
Specific leakage metric. This metric aims at identifying the time samples dependent on a targeted sensitive variable. To perform this assessment a *specific* partition of the sensitive variable has to be made given a chosen leakage model. An example is the application of a classical signal processing metric, namely *Signal-to-Noise Ratio* (SNR). Given a set of leakage traces and its related sensitive variable Y, the Evaluator can compute the signal-to-noise ratio such that the i^{th} sample is defined as follows:

$$SNR[i] = \frac{\mathbb{V}_Y\left[\mathbb{E}\left[\mathbf{T}[i]|Y\right]\right]}{\mathbb{E}_Y\left[\mathbb{V}\left[\mathbf{T}[i]|Y\right]\right]}. \tag{3.2}$$

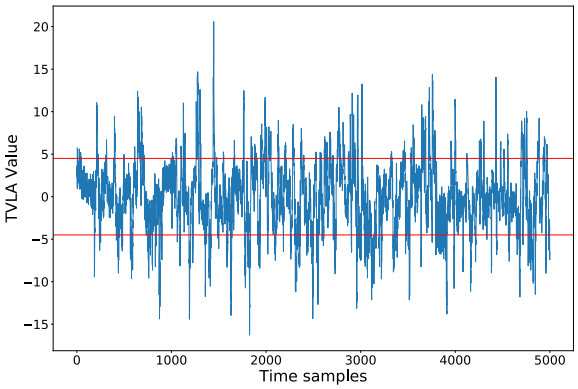
Following Equation 3.1, for every y, $\mathbb{E}\left[\mathbf{T}|Y=y\right] = \mathbb{E}\left[\psi(y)\right] + \mathbb{E}\left[\mathbf{Z}\right]$, and thus, $\mathbb{V}_Y\left[\mathbb{E}\left[\mathbf{T}|Y\right]\right]$ can be simplified to $\mathbb{V}_Y\left[\psi(Y)\right]$ if the noise $\mathbf{Z}$ is independent from the targeted sensitive variable Y. Then, for every y, we can verify $\mathbb{V}\left[\mathbf{T}|Y=y\right] = \mathbb{V}\left[\mathbf{Z}\right]$. Consequently, under Gaussian independent noise assumption, Equation 3.2 can be rewritten as $SNR[i] = \frac{\mathbb{V}_Y[\psi(Y)[i]]}{\mathbb{V}[\mathbf{Z}[i]]}$. In other words, this metric defines the ratio between the variance related to the targeted sensitive information and the variance of the noise. Thus, using the specific leakage metric is helpful for the Evaluator in order to directly assess the occurrence of points of interest for a given sensitive information.

[b] In [DS16], Durvaux and Standaert propose an alternative by applying the Welch's T-test on fixed vs. fixed sets. This solution is beneficial from a sampling complexity perspective.

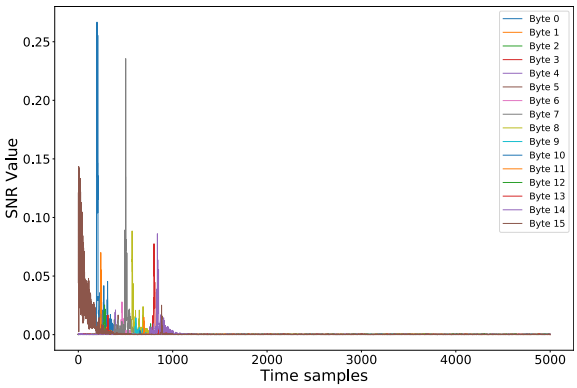
[c] In [Sta19], Standaert extends this metric to protected implementations.



(a) Power consumption of an AES-128 encryption



(b) Welch's T-test



(c) Signal-to-Noise Ratio (Targeted operation: *AddRoundKey*)

Figure 3.4: Leakage assessment of AES-128 encryption function.

In a classical evaluation process, the Evaluator collects one or multiple leakage traces (see Figure 3.3) and identifies the different operation processed during the cryptographic function (*e.g.* encryption). In Figure 3.4a, a focus was made on the first AES's round. Hence, we can clearly evaluate all the operations performed during an AES encryption (*i.e.* SubBytes, ShiftRows, MixColumns and AddRoundKey). Once the Evaluator visualizes these operations, he can use a non-specific leakage metric to get a global overview of the behavior of the cryptographic module. In Figure 3.4b, the Evaluator observes that several leakages occur but cannot assess those depending on the sensitive variable. Furthermore, the Welch's T-test can sometimes be irrelevant for detecting some peaks. Indeed, for some informative samples, the mean values of the fixed and the random set of leakage traces can provide similar values which makes the detection of sensitive peaks impossible. Applying the statistical tool on large number of traces can be an effective solution to prevent this issue. During the evaluation process, the Evaluator can still consider the non-specific leakage metrics as an initial step to get a first insight of the cryptographic module's behavior. However, from the Evaluator point of view, the non-specific leakage metrics cannot be considered as a unique solution because it does not highlight the leakages that depend on the targeted sensitive information. Even if a SNR computation is more expensive than a non-specific approach [DS16], its application is mandatory to label the observed peaks and ease the resulted attacks. Indeed, as illustrated Figure 3.4c, we perform a leakage assessment on the *AddRoundKey* operation of an AES-128 encryption function. The results we obtain are in accordance with the *AddRoundKey* processing observed in Figure 3.4a. Hence, if the Evaluator wants to target the *AddRoundKey* operation, he can only focus its side-channel attack on the first $1,000$ time samples. This preprocessing can be highly beneficial to reduce the impact of irrelevant time samples

regarding the targeted sensitive information and reducing the elapsed time of the related attack (see Section 3.3). The Welch's T-test can also be considered as a specific leakage metric [GJJR11] but the SNR will be the only specific leakage metric we mention in this manuscript.

Once the Evaluator detects the dependence between the processed data and the resulted leakage trace, he can exploit them through different attack scenarios detailed in the following section.

3.3 Side-Channel Attacks

3.3.1 Optimal Attack

Scenario. To perform a side-channel attack, the Evaluator tries to exploit the dependence between a D-dimensional leakage trace $\mathbf{T}$ and a sensitive variable Y in order to extract the secret information brought by the deterministic part of Equation 3.1 (*i.e.* $\psi(\mathbf{Y})$). The targeted sensitive variable $y = f(x, k^*)$ depends on a cryptographic primitive f (*e.g.* the output of the AES Sbox induced during the encryption of a plaintext x), on a public variable $x \in \mathcal{X}$ (*e.g.* plaintext or ciphertext) and on a part of the key $k^* \in \mathcal{K}$ (*e.g.* a particular byte) that the Evaluator wants to retrieve. For such a purpose, he collects a set $\mathcal{T}$ of N leakage traces $(\mathbf{t}_i)_{0 \leq i < N}$ resulting from the computation of $(f(x_i, k^*))_{0 \leq i < N}$ such that $(x_i)_{0 \leq i < N}$ is uniformly distributed over $\mathcal{X}$. Then, each trace is labeled with its related sensitive variable such that he obtains the following set of labeled leakage traces $\mathcal{T} = \{(\mathbf{t}_0, y_0), (\mathbf{t}_1, y_1), \ldots, (\mathbf{t}_{N-1}, y_{N-1})\}$.

DIVIDE & CONQUER STRATEGY

Because the Evaluator has access to the internal process of the targeted cryptographic function, the majority of side-channel attacks can follow the "*Divide & Conquer*" strategy which consists in targeting small independent part, known as sub-part, of the secret key (*e.g.* a byte of k^*). For example, if an AES-128 is considered, the Evaluator targets 16 sequences of 8 bits (*i.e.* $(k_i^*)_{0 \leq i < 16}$) instead of 1 sequence of 128 bits (*i.e.* k^*). It is beneficial to drastically reduce the key search space from 2^{128} (black-box setting) to 2^8. Hence, once the Evaluator recovers one secret's byte, he can replicate his attack on the other fifteen sequences in order to retrieve the whole secret key. This reduction makes the side-channel attacks particularly efficient performance-wise.

Definition 3.3.1.1 (Partial Key Recovery)**.** The *partial key recovery* defines the ability of the Evaluator to recover a sub-part of the secret key k^*.

Definition 3.3.1.2 (Global Key Recovery)**.** The *global key recovery* defines the ability of the Evaluator to recover the whole secret key k^*.

The Evaluator's goal is to assess the security of a cryptographic module under side-channel attacks. Hence, he has to perform a global key recovery in order to fully evaluate the robustness of the device under test. However, while the goal of this thesis is to enhance the existing deep learning-based side-channel attacks, this manuscript will only consider the partial key recovery strategy assuming that our propositions can also be applied under a global key recovery strategy. In the rest of the manuscript, the secret key k^* will interchangeably refer the whole secret as well as its subparts $(k_i^*)_{0 \leq i < 16}$. It is typically considered in the side-channel literature as well as the security lab that a partial key recovery can be extrapolated to other parts of the secret key k^*.

Based on this knowledge, the Evaluator tries to exploit the data dependence of the leakage trace with the help of statistical tools.

Definition 3.3.1.3 (Distinguisher)**.** A distinguisher, denoted $\mathcal{D}$, is a function identifying the dependence between a set of leakage traces and the related sensitive information Y. Furthermore, it outputs an ordered score vector defining the likelihood of observing each key hypothesis $k \in \mathcal{K}$.

Objective 3.3.1.1 (Optimal Adversary)**.** *Given a cryptographic module, the optimal Adversary aims at finding the distinguisher that recovers the secret key with a minimum number of attack leakage traces.*

MAIN RESULT IN SCA CONTEXT

During the evaluation process, the Evaluator finds solutions to converge towards the optimal Adversary in order to strongly assess the security of the targeted device. In other words, he has to find the distinguisher suggesting the optimal Adversary.

Definition 3.3.1.4 (Optimal Distinguisher [HRG14])**.** Given a set $\mathcal{T}$ of N leakage traces and a conditional probability distribution of observing a leakage $\mathbf{T}$ knowing a sensitive cryptographic primitive Y, denoted as $\mathsf{Pr}[\mathbf{T}|Y]$, we define the optimal distinguisher as:

$$\mathcal{D}_{opt}\left(\mathcal{T}, k\right) = \sum_{i=0}^{N-1} \log\left(\mathsf{Pr}\left[\mathbf{T} = \mathbf{t}_i | Y = f\left(x_i, k\right)\right]\right).$$

If the chosen distinguisher is well suited, the Evaluator can compute the *Distinguisher Rule* $\arg\max_{k\in\mathcal{K}}(\mathcal{D}_{opt}(\mathcal{T}, k))$ in order to output the most likely key candidate k. If $k = k^*$, then, the optimal distinguisher succeeds to extract the targeted information from the cryptographic module. The main issue of Definition 3.3.1.4 is that the optimal distinguisher requires the real unknown leakage model in order to be computed. While this knowledge highly depends on the setup environment, one solution is to find the adequate estimation of $\mathsf{Pr}\left[\mathbf{T}|Y\right]$. In the literature, this distinguisher can be defined by multiple tools.

Attack Categories. Classically, side-channel attacks can be decomposed into different categories depending on the targeted cryptographic implementation (*i.e.* symmetric vs. asymmetric algorithms) and the knowledge of the Evaluator:

- *Simple* side-channel attacks – The *simple* side-channel attacks consist of analyzing the operation flow in runtime of a cryptographic algorithm in order to retrieve the secret key. This strategy, called *horizontal attack* [Wal01, CFG+10, BJPW13], consider a single trace to extract the manipulated sensitive variable. Mainly designed to defeat the asymmetric algorithms' implementation, the leakage trace is decomposed into several segments characterizing similar operations (*e.g.* modular exponentiation for RSA, scalar multiplication for ECC).
- *Differential* side-channel attacks – The *differential* side-channel attacks target the processed data. Based on several computations, the Evaluator performs *vertical attacks* which consist in collecting the related leakage traces and makes a guess about the manipulated secret key. From this hypothesis, he can construct the corresponding sensitive variables and assigns a score to the guessed secret key which characterizing its statistical relation with the acquired leakage traces. The higher the score, the most likely the candidate (*i.e.* secret key).

The following sections explain the difference between those approaches in order to clarify these categories.

3.3.2 Simple Side-Channel Analysis

Originally described in [Koc96], the Simple Power Analysis (SPA) is known as the simplest side-channel attack. It consists in exploiting the security flaws of a cryptographic module by observing one or very few number of leakage traces. As mentioned in Subsection 3.1.1, the physical leakage of the device under test depends on the manipulated data. If the Evaluator is able to capture

a key-dependent leakage trace, he may retrieve information about the secret. Such example is illustrated in Figure 3.5. Depending on the operation processed by the cryptographic module, the Evaluator identifies a difference on the resulted patterns. Based on this observation, he can guess the value of the manipulated secret's bit.

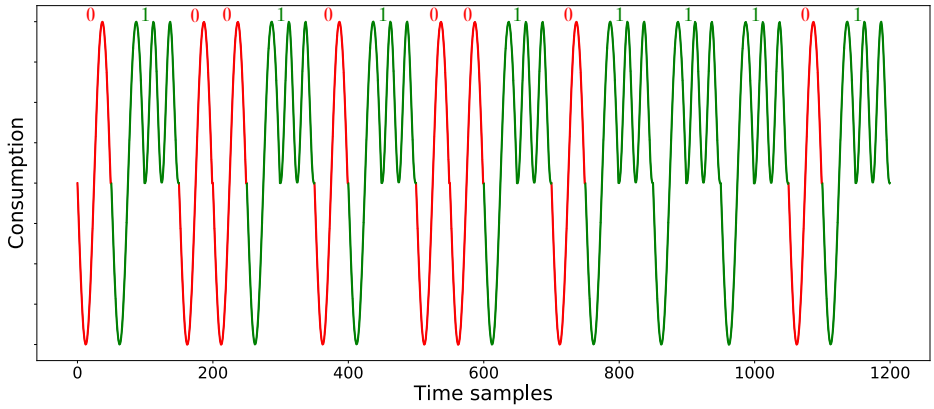


Figure 3.5: Simple Power Analysis (SPA) ($k^* = 0100101001011101$).

This is a typical example of a SPA against RSA implementation based on the "*Square & Multiply*" algorithm[d] (see Algorithm 1). In this algorithm, the processing differs depending on the value of the secret's bit. While a bit value 0 considers a single operation (*i.e.* a modular square), a bit 1 value causes two operations (*i.e.* a modular square followed by a modular multiplication). Thus, the Evaluator can easily observe this difference on the resulted leakage trace such that the pattern related to the bit value 1 should be longer than the pattern related to the bit value 0. This attack was then extended to electromagnetic emanations [QS01, GMO01]. Similarly in ECC, the execution of a "*Double & Add*" algorithm involved in the point multiplication can be exploited by a SPA. Mostly dedicated to attack the asymmetric cryptographic algorithms, the SPA can be prevented by removing the dependence between computations and key bit values (see Algorithm 2). However, robust countermeasures exist to reduce the risk of SPA. Hence, the targeted cryptographic module is rarely vulnerable against this threat.

Algorithm 1: Square & Multiply (Modular exponentiation, Left-to-Right)

Data: message: x, private key: $d \in \mathbb{F}_2^n$, modulus: N
Result: signature: $x^d \ mod \ N$
$sign \leftarrow 1$;
for $i \leftarrow n-1$ **to** 0 **do**
  $sign \leftarrow sign^2 \ mod \ N$;
  if $d[i] = 1$ **then**
    $sign \leftarrow (sign \times x) \ mod \ N$;
  end
end
return $sign$;

Algorithm 2: Multiply Always (Modular exponentiation, Left-to-Right)

Data: message: x, private key: $d \in \mathbb{F}_2^n$, modulus: N
Result: signature: $x^d \ mod \ N$
$R_0 \leftarrow 1$, $R_1 \leftarrow x$;
$j \leftarrow n-1$, $i \leftarrow 0$;
while $j \geq 0$ **do**
  $R_0 \leftarrow R_0 \times R_i \ mod \ N$;
  $i \leftarrow i \oplus d[j]$;
  $j \leftarrow j - 1 + i$;
end
return R_0;

In most of the cases, the Evaluator has to deal with a large number of leakages traces (up to million leakage traces) in order to retrieve the secret key. In this setting, he has to conduct another strategy, namely *Differential Side-Channel Analysis*, consisting in the combination of leakage traces.

[d] Among the numerous references an interested reader can refer for instance to [PP10] for details.

3.3.3 Differential Side-Channel Analysis

The basic idea of differential side-channel analysis is to gather the information of multiple leakage traces constructed from different inputs. As shown in Figure 3.6, the *Differential Side-Channel Analysis* (DSCA) attack can be described in four phases:

- *Construction* – Given a set of N plaintexts $\mathcal{X} = \{x_0, x_1, \ldots, x_{N-1}\}$, the Evaluator emits a hypothesis on the true unknown secret key k^*. This key hypothesis, denoted k, is used to construct a set of label hypotheses $\{f(x_0, k), \ldots, f(x_{N-1}, k)\}$.
- *Modeling* – Based on the set of label hypotheses, the Evaluator models the expected leakage model for each sensitive variable $(f(x_i, k))_{0 \leq i < N}$. In some cases (*e.g.* [CRR03]), the modeling phase can be done to capture both part of the leakage trace, namely the deterministic (*i.e.* ψ) and the random (*i.e.* Z) parts.
- *Observation* – In parallel of the *Modeling* phase, the Evaluator collects the related physical traces from the targeted cryptographic module considering $\mathcal{X}$ and an unknown fixed secret key k^* as inputs.
- *Exploitation* – From the hypothetical leakage models and the related leakage traces, the Evaluator applies a chosen distinguisher to estimate if k equals k^*.

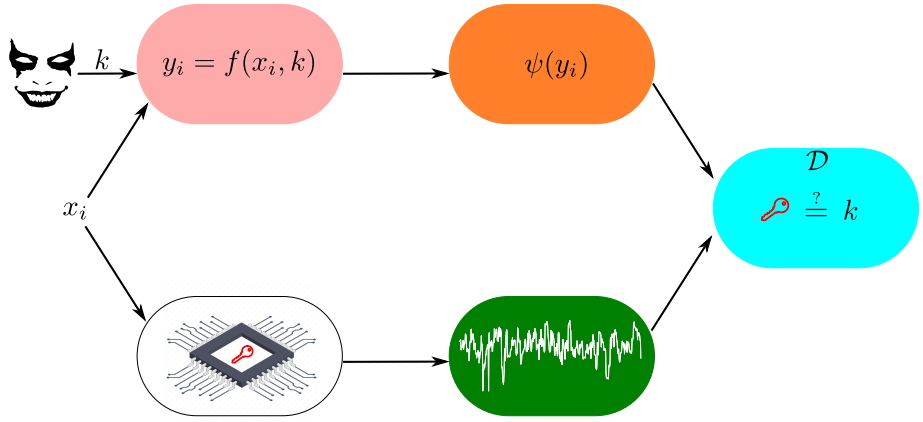


Figure 3.6: Scenario of a differential side-channel attack (DSCA).

To retrieve k^*, the Evaluator has to process these phases for each key hypothesis $k \in \mathcal{K}$. Typically, to perform a differential side-channel attack, various scenarios can be considered by the Evaluator:

- *Non-profiled* vs. *Profiled* attacks – To perform a differential side-channel attack, the Evaluator targets a cryptographic module containing a secret key he wishes to retrieve. In the *non-profiled* scenario, he has the ability to send a request to the cryptographic module (*e.g.* a message) in order to acquire the resulting output (*e.g.* a ciphertext). From the physical properties of the device under test (*e.g.* power consumption, electromagnetic emanations), the Evaluator uses some statistical tools to extract its dependence with a targeted sensitive variable. In this scenario, the Evaluator does not have prior information on the leakage traces (*i.e.* leakage model). On the other hand, in the *profiled* scenario, it is assumed that the Evaluator has access to an open cryptographic module which is identical to the targeted one. In this configuration, the Evaluator decomposes the process into two phases: a *profiling* and an *attack* phase. In profiling phase, the evaluator gathers leakage information from the open copy of the cryptographic module in order to characterize Equation 3.1. Once the Evaluator *learns* how the leakage trace behaves, he gathers leakage information from the targeted cryptographic module focusing on the identical, but now unknown Y. However,

based on the knowledge he acquired in the profiling phase, the Evaluator can emit some assumptions on the sensitive variable Y processed. This scenario is considered as the most critical from an evaluation point of view.

- *Univariate* vs. *Multivariate* attacks – Given a D-dimensional leakage trace $\mathbf{T}$, the Evaluator can assess the security flaws of the targeted cryptographic module based on leakage metrics defined in Subsection 3.2.2. Hence, he extracts a vector of index where the impact of $\psi(Y)$ is non-negligible. This vector lists all the points of interest induced in the leakage trace. If the Evaluator chooses to exploit the information from a single element of this list, it aims at performing an *univariate* side-channel attack. This time sample can be selected depending on the result the Evaluator obtained from a specific leakage metric. In a *multivariate*[e] side-channel attack, the Evaluator exploits multiple coordinates from the PoIs' vector in order to retrieve information about the targeted variable Y. The strategy can be chosen depending on the attack scenario (*i.e.* profiled vs. non-profiled attack) or the countermeasures implemented (see Section 3.4).

Non-profiled Attacks. As mentioned, a non-profiled side-channel attack relies on the ability of the Evaluator to extract the sensitive information from one or multiple leakage traces without any prior knowledge on the related leakage model. They are mostly univariate (*i.e.* $D = 1$) and take into account all the time samples independently. Hence, the selected distinguisher is performed successively on each of those samples and select the one that facilitates the key extraction.

The first non-profiled attack proposed in the literature is the *difference of means* (DoM) [KJJ99]. Kocher *et al.* selected the mono-bit leakage model to extract the sensitive information from the leakage traces. Given a key hypothesis $k \in \mathcal{K}$, the Evaluator partitions the leakage traces into two subsets according to the value of the j^{th} bit of Y (denoted as $Y[j]$). Then, he estimates the difference between the subsets for each $k \in \mathcal{K}$ and defines the argmax as the secret key.

Definition 3.3.3.1 (Difference of Means Distinguisher)**.** Given a set $\mathcal{T}$ of 1-dimensional leakage traces labeled from a key hypothesis $k \in \mathcal{K}$, a set of plaintext $\mathcal{X}$ and a targeted bit index j, the difference of means distinguisher is defined as:

$$\mathcal{D}_{DoM}\left(\mathcal{T}, \mathcal{X}, k, j\right) = \left| \frac{\sum_{i=0}^{N-1} f(x_i, k)[j] \cdot t_i}{\sum_{i=0}^{N-1} f(x_i, k)[j]} - \frac{\sum_{i=0}^{N-1} (1 - f(x_i, k)[j]) \cdot t_i}{\sum_{i=0}^{N-1} (1 - f(x_i, k)[j])} \right|,$$

where $f : \mathcal{X} \times \mathcal{K} \to \mathcal{Y}$ denotes the targeted cryptographic primitive.

For correct key guessed $\arg\max_{k \in \mathcal{K}}(\mathcal{D}_{DoM}(\mathcal{T}, \mathcal{X}, k, j)) = k^*$, $\mathcal{D}_{DoM}(\mathcal{T}, \mathcal{X}, k, j)$ converges towards the true unknown $\mathbb{E}\left[\mathbf{T} | f(X, k^*)[j] = 0\right] - \mathbb{E}\left[\mathbf{T} | f(X, k^*)[j] = 1\right]$ while a wrong key guess generates two subsets with random predictions. This lead $\mathcal{D}_{DoM}$ to converge towards 0 as the number of measurements growths.
A generalization of the DSCA was introduced by Brier, Clavier and Oliver in 2003 [BCO03, BCO04]. This solution, named *correlation power analysis* (CPA), is the most widely used distinguisher in DSCA.

Definition 3.3.3.2 (Pearson's Correlation Distinguisher)**.** Given a set $\mathcal{T}$ of 1-dimensional leakage traces labeled from a key hypothesis $k \in \mathcal{K}$, a set of plaintext $\mathcal{X}$ and a predicted leakage model $\hat{\psi}$, the Pearson's correlation distinguisher is defined as:

$$\mathcal{D}_{CPA}(\mathcal{T}, \mathcal{X}, k, \psi) = \frac{Cov\left[\mathcal{T}, \hat{\psi}(f(X, k))\right]}{\sigma_{\mathcal{T}} \cdot \sigma_{\hat{\psi}(f(X,k))}},$$

where $f : \mathcal{X} \times \mathcal{K} \to \mathcal{Y}$ denotes the targeted cryptographic primitive.

[e] For the sake of simplicity, the described approach focuses on attack targeting a single sensitive variable. However, more sophisticated attacks, such as *higher-order attacks* (see Section 3.5), may exploit simultaneously multiple leakages related to several sensitive variables.

A value close to -1 or 1 indicates a strong linear dependence between the measurements and the related leakage model of Y. An alternative to the CPA is the *correlation electromagnetic analysis* (CEMA). It results from a substitution between power and electromagnetic emanations channels. In opposition, a correlation coefficient close to 0 reveals the absence of dependence between these data. Consequently, the Evaluator considers the key hypothesis k maximizing the amplitude of the coefficient correlation as the secret key k^*. While the CPA perfectly fits for detecting linear dependence between the measurements and the chosen leakage model, it does not capture the non-linear dependencies.
To circumvent this issue, Gierlichs *et al.* introduce the *mutual information analysis* (MIA) [GBTP08] as an alternative distinguisher exploiting any kind of dependencies between the leakage traces and the leakage model.

Definition 3.3.3.3 (Mutual Information Distinguisher)**.** Given a set $\mathcal{T}$ of 1-dimensional leakage traces labeled from a key hypothesis $k \in \mathcal{K}$, a set of plaintext $\mathcal{X}$ and a predicted leakage model $\hat{\psi}$, the mutual information distinguisher is defined as:

$$\mathcal{D}_{MIA}(\mathbf{T}, \mathcal{X}, k, \hat{\psi}) = H\left[\mathbf{T}\right] - H\left[\mathbf{T}|\hat{\psi}(f(X,k))\right].$$

The key guess maximizing the mutual information analysis is the one minimizing the conditional entropy $H\left[\mathbf{T}|\hat{\psi}(Y)\right]$ such that $Y = f(X,k)$. Hence, an estimation of the *Probability Density Functions* (PDFs) of the conditional leakages (*i.e.* $(\mathbf{T}|\hat{\psi}(f(X,k)))$) should be computed for each key hypothesis k in order to perform the attack. However, this estimation needs to choice an appropriate methods such as histograms, kernel density function, or parametric estimation. However, selecting the appropriate method can be very challenging. This reason leads the side-channel community to mainly consider the CPA to perform non-profiled attacks.

Profiled Attacks. As mentioned, the profiled attacks aim at characterizing the behavior of the cryptographic module under test. Such a setting requires the ability to acquire a clone of the device such that the data and the key can be fully configured by the Evaluator. While the *Modeling* phase of non-profiled attacks consists in selecting the correct unknown leakage model ψ, in profiled attacks, this phase suggests the estimation (or profiling) of the PDFs $(\mathsf{Pr}\left[\mathbf{T} = \mathbf{t}|Y = y\right])_{y\in\mathcal{Y}}$, *i.e.* estimates the likelihood of a key guess $k \in \mathcal{K}$ such that, from the Bayes' Theorem, we have:

$$\mathsf{Pr}\left[\mathbf{T} = \mathbf{t}|Y = f(x,k)\right] = \frac{\mathsf{Pr}\left[Y = f(x,k)|\mathbf{T} = \mathbf{t}\right] \cdot \mathsf{Pr}\left[\mathbf{T} = \mathbf{t}\right]}{\mathsf{Pr}\left[Y = f(x,k)\right]}. \tag{3.3}$$

If k is assumed to be uniformly distributed, Equation 3.3 can be simplified as:

$$\mathsf{Pr}\left[\mathbf{T} = \mathbf{t}|Y = f(x,k)\right] = \epsilon \cdot \mathsf{Pr}\left[Y = f(x,k)|\mathbf{T} = \mathbf{t}\right], \tag{3.4}$$

where ϵ denotes a constant independent of k.

Definition 3.3.3.4 (Log-Likelihood Distinguisher)**.** Given a set $\mathcal{T}$ of N leakage traces labeled from a key hypothesis $k \in \mathcal{K}$ and a set of plaintext $\mathcal{X}$, the Log-Likelihood distinguisher can be computed from a set of estimated PDFs, denoted $(\widehat{\mathsf{Pr}}\left[\mathbf{T}|Y = y\right])_{y\in\mathcal{Y}}$, as:

$$\mathcal{D}_{LL}(\mathcal{T}, \mathcal{X}, k) = \sum_{i=0}^{N-1} \log\left(\widehat{\mathsf{Pr}}\left[\mathbf{T} = \mathbf{t}_i|Y = f(x_i,k)\right]\right) - \log\left(\epsilon\right).$$

While ϵ does not depend on k, it is usually ignored by the log-likelihood distinguisher.

The first profiled side-channel attack, namely *template attack* (TA), was introduced by Chari *et al.* in 2002 [CRR03]. In the original paper, it is assumed that the PDFs $(\mathsf{Pr}\left[\mathbf{T} = \mathbf{t}|Y = y\right])_{y\in\mathcal{Y}}$ are the density of a (multivariate) Gaussian distribution $\mathcal{N}_D(\boldsymbol{\mu}_Y, \Sigma_Y)$ such that the pair $(\boldsymbol{\mu}_Y, \Sigma_Y)$

is dependent on the targeted sensitive variable Y. During the profiling phase, the Evaluator estimates the empirical mean and the empirical covariance matrix for each $y \in \mathcal{Y}$. For a set of D-dimensional leakage traces $\mathcal{T}$, the estimation of the $D \times D$ covariance matrix Σ_y for each $y \in \mathcal{Y}$ can be impracticable. More precisely, if N leakage traces are needed to adequatly approximate Σ_y for each $y \in \mathcal{Y}$, the total amount of physical measurements that is required to compute all covariance matrices is $N \cdot |\mathcal{Y}|$. Thus, the Evaluator has to find the best trade-off between the number of acquired leakage traces, his storage capacity and the optimal estimation of $(\Sigma_y)_{y \in \mathcal{Y}}$. To circumvent it, Choundary and Kuhn propose to *pool* the covariance matrix into a single solution in order to ease its estimation [CK14]. The *pooling* consists in considering a unique covariance matrix, denoted Σ for all $y \in \mathcal{Y}$. Consequently, whatever the manipulated sensitive variable, the related multivariate Gaussian distribution can be reduced to $\mathcal{N}_D(\boldsymbol{\mu}_Y, \Sigma)$ for each $y \in \mathcal{Y}$. Once the profiling phase is performed, the Evaluator can apply the log-likelihood distinguisher on the PDFs' estimation in order to guess the secret key manipulated by the cryptographic module.
If the estimation of the PDFs are equal to the true unknown PDFs (*i.e.* $\widehat{\mathsf{Pr}}[\mathbf{T}|Y] = \mathsf{Pr}[\mathbf{T}|Y]$), then, the template attacks maximize the probability of attack success. From a side-channel perspective, the template attacks are known as the most powerful solution. An alternative to the template attacks have been introduced in 2005 by Schindler *et al.* [SLP05]. This proposition, called *stochastic attacks* (SA), will be addressed in Section 5.1. All the attacks introduced are summarized in Table 3.1 and linked with their related configurations.

The curse of Dimensionality. The practical issue of the profiled and non-profiled attacks is the selection of the points of interest. Indeed, side-channel measurements acquired with an oscilloscope consist of thousands or even millions of time samples. These huge dimensional leakage traces can be problematic from a practical perspective as the risk of exploiting non-informative time samples increases. Consequently, the larger the dimension D, the larger this risk. Thus, from a practical perspective, the Evaluator has to find solution to focus its attack only on the informative time samples (*i.e.* PoIs). To conduct such approach, he can reduce the dimensionality of the leakage traces through multiple scenarios. Furthermore, reducing the amount of time samples can be highly beneficial to accelerate the attack process and thus, to minimize the *elapsed time* (see Subsection 1.2.2) attribute.

DIMENSIONALITY REDUCTION

One solution to perform dimensionality reduction is based on the specific leakage metrics. In this scenario, the Evaluator can assess the time samples influenced by a targeted sensitive variable. For example, once a SNR computation is performed (see Figure 3.4c), the Evaluator is able to only select the time samples where the most of relevant information leaks. Based on his selected points of interest, a profiled/non-profiled attack can be applied on this subset of time samples. Another solution does not locate the points of interest but compresses the leakage traces while preserving a maximum amount of relevant information. Several techniques are introduced in the literature: the *Principal Components Analysis* [RO05, APSQ06, SA08, CDP16], the *Linear Discriminant Analysis* [SA08, BGH+15] or the *Kernel Discriminant Analysis* [CDP17b].

In this manuscript, we mainly focus on the vertical multivariate profiled attack because it corresponds to one of the highest risks from an attack perspective. However, how does the Evaluator can evaluate if the performed attack retrieves the correct secret key? Which solutions should be considered? In which scenario? The following section tries to answer these questions.

3.3.4 Performance Metrics

To assess the security of the cryptographic module, the Evaluator has to evaluate the robustness of the device against side-channel attacks. For such purpose, the literature introduced some *performance metrics* to define the feasibility of a *successful attack*. In this manuscript, the definition

Table 3.1: Summary of the most classical side-channel attacks.

Side-Channel Attacks						
	Simple Power Analysis [Koc96]	**Differential Power Analysis** [KJJ99]	**Correlation Power Analysis** [BCO04]	**Mutual Information Analysis** [GBTP08]	**Template Attacks** [CRR03]	**Stochastic Attacks** [SLP05]
Operation Flow	✓					
Processed Data		✓	✓	✓	✓	✓
Horizontal*	✓					
Vertical		✓	✓	✓	✓	✓
Profiled					✓	✓
Non-profiled	✓	✓	✓	✓		✓**
Univariate		✓	✓	✓	✓	✓
Multivariate				✓	✓	✓

* Most of the side-channel attacks defined in Table 3.1 can be performed in horizontal setting. However, as the original papers do not consider this scenario, we do not mention them as horizontal attacks.

** The interested readers may refer to [DPRS11, DDP13] for deeper details on the application of non-profiled stochastic attacks, also known as *Linear Regression Analysis*.

of a successful attack differs depending on the cryptographic algorithm considered as well as the performance metrics.

Symmetric cryptographic implementations. For symmetric cryptographic implementations, we define a key recovery attack using a distinguisher $\mathcal{D}$ as successful if, for a given set of N_a attack traces, the Evaluator retrieves the entire value of the secret key $k^* \in \mathbb{F}_2^n$. From a distinguisher, the Evaluator *predicts* the sensitive variables manipulated by the cryptographic module. Knowing the inputs used during the cryptographic process (*e.g.* encryption), he can use a set of N_a attack traces and compute a score vector, based on $\mathcal{D}$ (*e.g.* difference of means, Pearson's correlation, mutual information, log-likelihood).
Given the score associated with each key hypothesis $k \in \mathcal{K}$, the Evaluator can classify all the key candidates into a vector of size $|\mathcal{K}|$, denoted $g_{N_a} = \left[g^1_{N_a}, g^2_{N_a}, \ldots, g^{|\mathcal{K}|}_{N_a}\right]$, such that:

$$g_{N_a}(\mathcal{D}, k') = \sum_{k \in \mathcal{K}} \mathbb{1}_{s_{N_a}(\mathcal{D},k) \geq s_{N_a}(\mathcal{D},k')},$$

where $s_{N_a}(\mathcal{D}, k)$ denotes the score related to key candidate $k \in \mathcal{K}$ for a given number N_a of attack leakage traces and a given distinguisher $\mathcal{D}$.

Example 3.3.4.1 (Computation of $s_{N_a}(\mathcal{D}_{LL}, k)$.). Given a set $\mathcal{T}$ of N_a attack leakage traces, a set of plaintext $\mathcal{X}$ and the Log-Likelihood distinguisher $\mathcal{D}_{LL}$, the score $s_{N_a}(\mathcal{D}_{LL}, k)$ related to a key hypothesis $k \in \mathcal{K}$ can be computed as follows:

$$s_{N_a}(\mathcal{D}_{LL}, k) = \sum_{i=0}^{N_a - 1} \log\left(\widehat{\mathsf{Pr}}\left[\mathbf{T} = \mathbf{t}_i | Y = f(x_i, k)\right]\right), \tag{3.5}$$

where $\left(\widehat{\mathsf{Pr}}\left[\mathbf{T}|Y = y\right]\right)_{y \in \mathcal{Y}}$ denotes a set of estimated PDFs (see Definition 3.3.3.4).

The variable $g_{N_a}(\mathcal{D}, k')$ defines the position of a given hypothetical key k', in g_{N_a}, amongst all hypotheses. We consider $g^1_{N_a}$ as the most likely candidate and $g^{|\mathcal{K}|}_{N_a}$ as the least likely one.

Definition 3.3.4.1 (Rank)**.** The *rank*, defined by $g_{N_a}(\mathcal{D}, k^*)$, measures the position of the secret key k^* in g_{N_a}.

The rank of the correct key gives to the Evaluator an insight into how well the selected distinguisher performs. Broadly, the rank decreases as the attack becomes better and reaches one if it

can recover the correct key hypothesis.

The first used evaluation metric introduced in the literature was the *stability criterion* [SBdMVC08] that consists in determining the minimum number of leakage traces that are needed to continuously retrieve k^* (*i.e.* rank equals 1) when accumulating the attack traces. However, this metric can be questioned. Indeed, if the distinguisher rule outputs the correct key hypothesis 99% of the time, the targeted cryptographic module is considered as secure while it is not indisputable. To fix this issue, Standaert *et al.* propose two new metrics [SMY09]: the *Success Rate* and the *Guessing Entropy*.

More precisely, the success rate is a probability measure quantifying the number of attack traces that are needed for retrieving the secret key stored into a cryptographic module.

Definition 3.3.4.2 (Success Rate)**.** Given a number of traces N_a, the *Success Rate* (SR) is a metric that defines the probability that an attack succeeds in recovering the secret key k^* amongst all hypotheses.

A success rate of β means that β attacks succeed in retrieving k^* over 100 realizations. In [SMY09], Standaert *et al.* propose to extend the notion of success rate to an arbitrary order o such that:

$$\mathsf{SR}^o(\mathcal{D}, N_a) = \mathsf{Pr}\left[g_{N_a}(\mathcal{D}, k^*) \leq o\right]. \tag{3.6}$$

In other words, the o^{th} order success rate is defined as the probability that the targeted secret k^* is ranked amongst the o first key guesses in the score vector g_{N_a}. From an exhaustive search perspective, the o^{th} order success means that the attacker has, at most, o key candidates to test after the attack in order to recover the secret key k^*. In side-channel attacks, the Evaluator wants to find a distinguisher $\mathcal{D}$ such that the condition $\mathsf{SR}^o(N_a) > \beta$ is verified with the minimum number N_a of attack traces (see Objective 3.3.1.1).

Definition 3.3.4.3 (Guessing Entropy)**.** The *guessing entropy* is defined as the expected rank of the secret key k^* such as:

$$GE(\mathcal{D}, N_a) = \mathbb{E}\left[g_{N_a}(\mathcal{D}, k^*)\right].$$

In other words, the guessing entropy measures the average number of key guesses that have to be tested before finding the secret key. In [MMOS16], Martin *et al.* propose to take into account the orders of magnitude of the rank in order to quantify the remaining brute-force budget. This solution can be employed when the Evaluator has to deal with key enumeration algorithms [VCGRS13, VCGS13, PSG16].

Definition 3.3.4.4 (Ranking Entropy)**.** The *ranking entropy* is defined as the expected logarithm of the rank of the secret key k^* such as:

$$RE(\mathcal{D}, N_a) = \mathbb{E}\left[\log_2\left(g_{N_a}(\mathcal{D}, k^*)\right)\right].$$

In the security laboratories, the ranking entropy is also known as the *remaining entropy*. These metrics, *i.e. Success Rate*, *Guessing Entropy* and *Ranking Entropy*, are often defined in an empirical way by computing them several times over different sets of leakage traces collected from the same targeted cryptographic module in order to construct *security graphs* [VCGS13]. A special feature of the security graphs is the indication of the probability that a key will be found, depending on the amount of keys that the Evaluator can enumerate.
However, as mentioned in Subsection 3.3.1, this thesis only considers the partial key recovery strategy, and consequently, cannot fully exploit the benefits of the security graphs introduced in [VCGS13]. Instead, we consider *partial security graph* plotting the rank evolution of a targeted key byte. An example of such partial security graph is given in Figure 3.7. Given an unprotected AES-128 implementation, we perform 10 successive CPA against the *AddRoundKey* operation considering sets of different leakage traces and we report the rank evolution of the correct secret

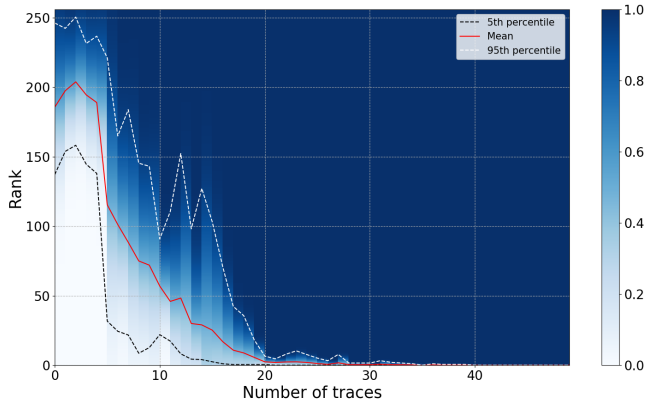


Figure 3.7: Partial security graph resulting of a CPA which targets one byte of an unprotected AES-128 with known inputs.

key byte into 10 vectors (one vector for each attack). In Figure 3.7, the red curve illustrates the mean rank evolution of this byte on 10 attacks while the dotted black (resp. dotted white) curve indicates the 5^{th} (resp. 95^{th}) percentile of the rank evolution. The area around the red curve indicates a non-negligible probability of successful attack.
In this manuscript, the partial security graphs will be considered as the most likely performance metric when the targeted cryptographic module implements a symmetric algorithm.

Asymmetric cryptographic implementations. For asymmetric cryptographic implementations, we define a key recovery attack using a distinguisher $\mathcal{D}$ as successful attack if, for one or few attack traces, the Evaluator retrieves the whole private key $s_{k^*} \in \mathbb{F}_2^n$. Indeed, due to a careful combination of countermeasures (see Subsection 3.4.1), the Evaluator must be able to recover most of the private bits with one or a few leakage traces capturing the cryptographic process (*e.g.* signature). Consequently, his strategy differs from the symmetric cryptographic implementations. While an attack against symmetric cryptographic implementations computes the score from a distinguisher over multiple traces, the Evaluator cannot consider the performance metrics previously designed. Let $\mathcal{T}$ be a set of leakage traces labeled from the same private key $s_{k^*} \in \mathbb{F}_2^n$. Hence, each trace included in $\mathcal{T}$ is related to a single bit of $(s_{k^*}[i])_{0 \leq i < n}$.

Definition 3.3.4.5 (Accuracy)**.** Given a set $\mathcal{T}$ of n leakage traces, a distinguisher $\mathcal{D}$ defining a score related to each leakage trace $(\mathbf{t}_i)_{0 \leq i < n}$ and a private key $s_{k^*} \in \mathbb{F}_2^n$, the *accuracy* of $\mathcal{D}$ is defined as:

$$Acc(\mathcal{T}, \mathcal{D}, s_{k^*}) = \frac{1}{n} \sum_{i=0}^{n-1} \mathbb{1}_{\substack{\arg\max(\mathcal{D}(\mathbf{t}_i, k)) = s_{k^*}[i] \\ k \in \{0,1\}}}$$

The accuracy describes the ability of the distinguisher $\mathcal{D}$ to correctly map a leakage trace to its related private key bit $(s_{k^*}[i])_{0 \leq i < n}$. If the resulted accuracy is less than 100%, the Evaluator has to perform additional operations to retrieve the full private key.

Definition 3.3.4.6 (Remaining Operations)**.** Given a set $\mathcal{T}$ of n leakage traces and a distinguisher $\mathcal{D}$ resulting in an accuracy of α, the remaining operations $N_{op,\alpha}$ defines the number of operations the Evaluator has to perform in order to retrieve the remaining bits of the private key s_{k^*}.

The European SOG-IS scheme considers that a maximum brute-force complexity of around 2^{100} remaining operations is practical. Hence, this manuscript considers this threshold to evaluate if an attack becomes feasible against asymmetric cryptographic implementations. Note that the notion of time complexity is independent of the computational power available to the attacker. In addition, even if no theoretical results link the accuracy with the remaining operations, it sounds

evident that a correlation between both metrics exist. Chapter 8 experimentally validates this intuition. Hence, increasing the accuracy of a side-channel attack is crucial in order to reduce the remaining operations required to find the entire private key.

For ensuring the robustness of the cryptographic modules against physical attacks, the Developer has to implement some countermeasures in order to harden the leakage exploitation. Hence, the side-channel community designs countermeasures in order to reduce the relation between the leakage traces and the processed data.

3.4 Side-Channel Countermeasures

Following [Shi00], a countermeasure can be defined as:

> *"An action, device, procedure, or technique that reduces a threat, a vulnerability, or an attack by eliminating or preventing it, by minimizing the harm it can cause, or by discovering and reporting it so that corrective action can be taken."*

In the following, we review the most common countermeasures against side-channel attacks. Typically, these countermeasures can be decomposed into two categories: *data randomization* and *hiding*.

3.4.1 Data Randomization

Randomization for block ciphers. For block cipher implementations, one solution consists in using *masking schemes*. This countermeasure consists in breaking the dependency between the targeted sensitive variable $Y \in \mathbb{F}_2^n$ and the secret key by integrating multiple *shares*. The shares' integration is beneficial to make the sensitive variable independent from the secret key. Based on the d-probing secure circuit of Ishai *et al.* [ISW03], the masking schemes are designed such that during the computation, the d shares $\{Y^0, Y^1, \ldots, Y^{d-1}\}$ occurring reveal no information about the secret. The main idea is to randomly split a secret into several shares such that the Evaluator needs all of them to reconstruct the secret. Hence, given a group operation $\perp$, the following relation should be satisfied:

$$Y^0 \perp Y^1 \perp \ldots \perp Y^{d-1} = Y, \tag{3.7}$$

where the $d-1$ shares $\{Y^1, Y^2, \ldots, Y^{d-1}\}$, known as *masks*, are independent and randomly distributed over $\mathbb{F}_2^n$ while Y^0, known as the *masked variable* is processed such that the masking scheme is satisfied (*i.e.* Equation 3.7).

Definition 3.4.1.1 (d^{th}-Order Masking)**.** A *d^{th}-order* masking describes a masking scheme inducing d masks.

In the literature, different operations $\perp$ have been introduced, notably, the *XOR* operation [GP99, CJRR99], the *modular addition* [Mes01], *multiplicative* [GT03] or the affine scheme [vW01, FMPR11].

In this manuscript, a particular focus will be made on the *Boolean Masking Scheme*. In this scheme, the invertible operation $\perp$ is defined by a XOR operation in $\mathbb{F}_2^8$, denoted $\oplus$. Proposed by Goubin and Patarin [GP99], and Chari *et al.* [CJRR99], this is the most classical masking scheme used in practice due to its simple hardware/software implementation. The computation performed on the masks and the masked variable should be computed in order to satisfy Equation 3.7. An example of 1^{st} order Boolean masking scheme is presented in Figure 3.8.

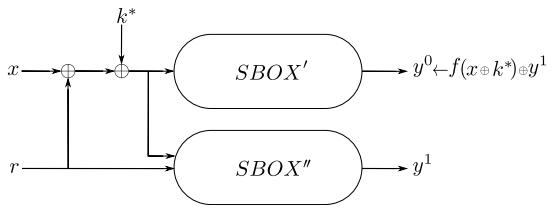


Figure 3.8: Example of 1^{st}-order Boolean masking scheme.

The soundness of the masking schemes was formally demonstrated by Chari *et al.* [CJRR99]. They establish that the number of leakage traces that are needed for defeating such implementations increases exponentially with the number of shares. More precisely, for a d-order masking scheme, they assume that each share is modeled as $\mathbf{T}^i = Y^i + Z^i$ such that the noise $(Z^i)_{0 \leq i < d}$ follows a Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$. Hence, the number of attack traces is proportional to σ^d. Many papers have been proposed as an extension of this seminal work [PR13, DDF14, DFS15, DFS18]. As a result, the side-channel attacks exploiting a single sensitive variable cannot be performed anymore. A solution suggests the application of *high-order side-channel attacks* targeting multiple sensitive variables simultaneously (see Section 3.5).

Randomization for asymmetric algorithms. As illustrated in Subsection 3.3.2, the asymmetric algorithms' implementation can be vulnerable to SPA due to the conditional branch induced in the modular exponentiation (*e.g.* RSA implementation, see Algorithm 1) or in the scalar multiplication (*e.g.* ECC implementation). In order to reduce this limitation, the literature proposes some countermeasures. A first countermeasure consists of preventing the conditional branch dependence by performing the same operations independently of the value of the current bit. This solution can be brought by the *Multiply Always* algorithm (see Algorithm 2), the Montgomery Ladder or the *Double & Add Always* algorithm. However, such countermeasure is still vulnerable against differential attacks [Cor99]. Hence, as for the symmetric case, the side-channel community proposes to randomize the manipulated data in order to reduce the dependence between the sensitive variable and the leakage trace by using *blinding schemes*. However, depending on the targeted asymmetric algorithm, the countermeasure differs. In RSA algorithm, let $sign = x^{s_{k^*}} \ mod \ N$ be the signature of a message x given a private key s_{k^*} and a modulus N. To make a RSA-CRT implementation robust against side-channel attacks, the Developer can implement the following blinding countermeasures:

- *Exponent blinding* [Koc96] – If the same private key s_{k^*} is used to compute several signatures, the Developer can generate any random positive integer r such that:

$$sign = x^{s_{k^*}} \ mod \ N = x^{s_{k^*} + r \cdot (q-1)(p-1)} \ mod \ N.$$

  Hence, for each signature computation, a new random value r is picked in order to reduce the dependence between the private key s_{k^*} and the related leakage trace. A similar solution the Developer can consider is the *exponent splitting* [CJ01].

- *Message blinding* [MDS99] – The idea is to transform the message x in such a way that the Evaluator cannot perform, for example, a *multiple-exponent, single-data* (MESD) a *zero-exponent, multiple-data* (ZEMD) attack [MDS99] or a *doubling attack* [FV03]. Given r_1 and r_2, two random position integers, it can be computed based on the following equation:

$$sign = x^{s_{k^*}} \ mod \ N = ((x + r_1 \cdot N)^{s_{k^*}} \ mod \ r_2 \cdot N) \ mod \ N.$$

Combination of these countermeasures can be beneficial for reducing the risks of security flaws. For the ECC algorithms, an analogy can be made with the *point blinding* and the *scalar blinding* as fruitful countermeasures. The list introduced in this section represents a non-exhaustive enumeration of the possible attacks and countermeasures against RSA and ECC. The interested readers may refer to [FV12, DGH+13, AFSW13, FGI+16] for additional information.

3.4.2 Hiding

In addition to the data randomization, the hiding mechanisms are widely used as countermeasures which can be included in hardware or in software cryptographic implementations. As defined in Subsection 3.2.2, the Evaluator aims at identifying the time samples where the sensitive variable leaks in order to extract the information related to the manipulated secret key. The goal of hiding is to distribute the execution of the sensitive variables over different periods of time. The distribution should neither be predictable nor be observable by the Evaluator. If the leakage traces are perfectly synchronized, the probability of observing the execution of a given cryptographic operation for a period of time is nearly 1 while the hiding mechanisms reduce this probability. The higher the hiding level, the lesser the probability of observing the execution of a given cryptographic operation for a certain period of time. Therefore, it significantly reduces the correlation between the targeted sensitive variable and the time samples where it leaks. This countermeasure reduces the Signal-to-Noise Ratio (see Equation 3.2) by increasing the related noise without eliminating the informative signal itself. It differs from the masking countermeasures which reduce correlation between the leakage traces and the sensitive variables by splitting it over multiple shares. In the literature, several hiding techniques have been introduced:

- *Adding Noise* – One classical countermeasure generates some noise in the leakages trace in order to increase the SNR value. The cryptographic module performs unrelated operations when the relevant computation is being conducted. The sensitive information included in the leakage trace is thus polluted by the irrelevant operations. This countermeasure is considered as a moderate way for reducing the security flaws because the electromagnetic emanation analysis can capture a very local signal that is not affected by the unrelated operations.

- *Random Process Operations* [CCD00] – It is based on the insertion of *dummy* instructions so that the corresponding operation cycles do not match between the leakage traces. A straightforward method consists in randomly picking up a number of dummy instructions in an interval $[0, \alpha]$, with $\alpha \in \mathbb{N}$. Some methods were proposed in order to efficiently insert random delay in software implementations [TB07, CK09] such as the insertion of *NOP* instructions. In hardware level, the random process operation can be formulated as a *jitter* generated through unstable clock frequency.

- *Dummy Operations* – This approach randomly inserts meaningless operations, which do not influence the targeted cryptographic function (*e.g.* encryption, decryption), amongst sensitive operations. The *1-amongst-n* countermeasure is a classical example. It consists of executing the computation of the sensitive variables randomly amongst $(n-1)$ fake operations. In opposition to the random process operations, the Evaluator cannot distinguish the dummy operations from the real one.

- *Shuffling* [HOM06] – It consists in randomizing the order of independent operations. This countermeasure limits the ability of the Evaluator to assess the time samples where the sensitive variable is manipulated. As an example, the *SubBytes* operations of an AES implementation are independent and thus, can be permuted in an arbitrary order.

- *Balancing* – This countermeasure aims at making the activity of the physical implementation independent of the sensitive variables. Applied at the logic gate level, it adds complementary logic to the existing one in order to make the activity of the cryptographic module constant [MAM+03, TV03, CZ06].

- *Shielding* – To prevent hide the internal behavior of the chip, this countermeasure proposes to insert circuit and wire shielding, such as a Faraday cage, in order to reduce and filter the signal. Hence, the Evaluator cannot exploit the dependence between the data manipulated by the cryptographic module and the related power consumption.

Hiding countermeasures do not make the attack theoretically unfeasible but increase the number of attack leakage traces considerably. Indeed, from an evaluation perspective, these countermeasures highly impact the time requires to perform a successful attack, *i.e. elapsed time*, and thus, makes a cryptographic module robust regarding the certification schemes. However, these countermeasures can be mitigated with a preprocessing phase consisting in applying some signal processing tools such as filtering, pattern detection, signal transformation, synchronization, *etc.* While most of the Evaluator's time consists in optimizing the preprocessing phase in order to perform successful attacks, some tools have been designed to limit the desynchronization effect, *e.g. phase-only correlation* [HNI+06], the *amplitude-only correlation* [GKLD11], *dynamic time warping* [vWWB11].

Most of the real-life protected cryptographic modules implement a mixture of multiple hiding and masking countermeasures in order to make the side-channel attacks challenging. In this thesis, we only consider the random delay interrupt as a source of hiding while the masking scheme that we have to deal with is the Boolean masking scheme.

3.5 High-Order Side-Channel Analysis

In Subsection 3.4.1, the masking countermeasures have been introduced to split a sensitive variable into d shares $\{Y^0, Y^1, \ldots, Y^{d-1}\}$. To counteract the effectiveness of the masking schemes, the Evaluator can perform a *High-Order Side-Channel Attack* (HO-SCA) for retrieving the secret key manipulated by the cryptographic module. Let the masking scheme be Boolean, $\mathcal{M} = \{m_1, m_2, \ldots, m_{d-1}\}$ be a set of $d-1$ random masks and Y the targeted sensitive variable such that:

$$Y^i = \begin{cases} Y \oplus m_1 \oplus \ldots \oplus m_{d-1} & \text{if} \quad i = 0, \\ m_i & \text{otherwise.} \end{cases}$$

Definition 3.5.1 (d^{th}-Order Attack)**.** A d^{th}-Order Attack refers the order of the statistical moment related to the leakage distribution that is exploited during the attack.

If no masking schemes are implemented, the sensitive variable is unprotected. Thus, a 1^{st}-order attack exploits the mean (*i.e.* the first moment) of the leakage distribution to extract the sensitive variable. Similarly to a 1^{st}-order side-channel attack, when the Evaluator wants to perform a high-order attack, he aims at targeting the unmasked sensitive variable Y. However, Y is not directly observable through side-channel measurements because of the masking scheme contrary to its related PDFs. If Y is split into d shares $\{Y^0, \ldots, Y^{d-1}\}$, the Evaluator has to perform a $(d+1)^{\text{th}}$ order attack in order to reveal the sensitive information Y. If the modeling of the shares $(\mathbf{T}^i = Y^i + Z^i)_{0 \leq i < d}$ is independent, then the Evaluator must combine the leakages of different shares and he has to estimate a high-order moment of the leakage distribution. Typically, classical approaches considered in profiled side-channel attacks use some *combining functions* as preprocessing. This approach involves the combination of the d shares in order to reveal the dependence of the leakage traces and the related sensitive variable Y. To apply this proposition, various combining functions $C(Y^0, Y^1, \ldots, Y^{d-1})$ are introduced:

- *Product combining* [CJRR99] – Proposed by Chari *et al.*, this combination function multiplies the shares such that:

$$C_{prod}(Y^0, \ldots, Y^{d-1}) = \mathbf{T}^0 \times \ldots \times \mathbf{T}^{d-1}. \tag{3.8}$$

- *Absolute difference combining* [Mes00] – Introduced by Messerges, it computes the absolute difference between the shares and can be expressed as follows:

$$C_{abs.diff}(Y^0, \ldots, Y^{d-1}) = |\mathbf{T}^0 - \mathbf{T}^1| - \mathbf{T}^2| \ldots | - \mathbf{T}^{d-1}|. \tag{3.9}$$

- *Optimal product combining* [PRB09] – Prouff *et al.* expressed it as the centered product of the shares:

$$C_{opt.prod}(Y^0, \ldots, Y^{d-1}) = (\mathbf{T}^0 - \mathbb{E}[\mathbf{T}^0]) \times \ldots \times (\mathbf{T}^{d-1} - \mathbb{E}[\mathbf{T}^{d-1}]). \tag{3.10}$$

In [PRB09], the authors demonstrate the benefits of using the optimal product combining function if the leakage model ψ of the processed sensitive variables is the Hamming Weight. In addition, they suggest that under a very noisy model, the number of side-channel measurements required to retrieve the secret key is greater when the absolute difference combining function is considered. Other combining functions have been proposed in the literature [JPS05, OM06] but stay out of the scope of this manuscript. Once these combinations functions are used as preprocessing, the Evaluator can perform the classical side-channel attacks as defined in Subsection 3.3.3.

To assess the benefits of new contributions, the side-channel community mainly applies their related propositions on a wide range of implementations that we describe in the next section. Those datasets are also considered in the rest of this manuscript.

3.6 Presentation of the Datasets

The datasets, introduced in this section, are beneficial to experimentally validate the theoretical observations we will make. We use eight different implementations with a wide range of use cases. Six datasets correspond to symmetric cryptographic implementations related to the *Advanced Encryption Standard*. These datasets offer a wide range of use cases: simulations, high-SNR[f] unprotected implementation on a smart card (*i.e.* Chipwhisperer and DPA contest-v4), low-SNR unprotected implementation on a FPGA (*i.e.* AES_HD), low-SNR protected implementation with first-order masking (*i.e.* ASCAD) and desynchronization with a random delay (*i.e.* AES_RD). The other datasets implements protected public-key algorithms, namely RSA and ECC.

Simulations. First, this dataset considers simulations of D-dimensional leakage traces from a 8-bit sensitive variable Y. In this manuscript, the simulated leakage traces are built such that for every time sample $i \in [0, D-1]$, the j^{th} leakage trace $\mathbf{t}_j$ is defined as follows:

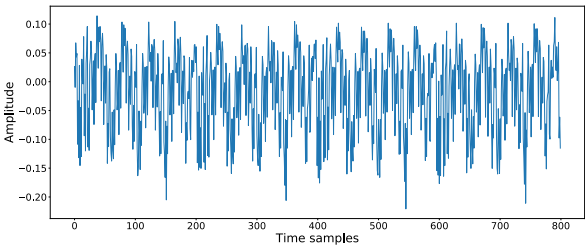
$$\mathbf{t}_j[i] = \begin{cases} \psi(y_j)[i] + \mathbf{z}_j[i] & \text{if} \quad i \in \{l_0, \ldots, l_{u-1}\}, \\ \mathbf{z}_j[i] & \text{otherwise,} \end{cases}$$

where $\{l_0, \ldots, l_{u-1}\}$ defines a set of indices related to each point of interest, $\psi(y_j)[i]$ denotes the leakage model constructed from $y_j \in \mathbb{F}_2^n$ that depends on the secret key k_j^*. Finally, $\mathbf{z}_j$ defines the noise following a Gaussian distribution $\mathcal{N}(0, \sigma^2)$. The noise as well as the leakage model considered in this manuscript will be specified each time the simulation leakage traces will be applied.

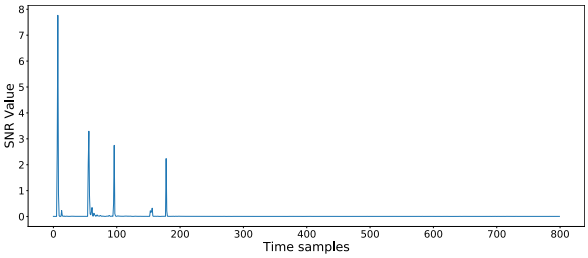
ChipWhisperer is an unprotected emulation of AES-128 implemented in software on a Chipwhisperer board [OC14] which corresponds to a ATxMega-128D4, which is composed by a 8-bit microcontroller[g]. Due to the lack of countermeasures, the Evaluator can recover the secret key directly. In this experiment, we attack the first round S-box operation. We identify each leakage trace with the sensitive variable $Y = \text{Sbox}[X[0] \oplus k^*]$ where $X[0]$ denote the first byte of the plaintext such that the measured SNR equals 7.767 (see Figure 3.9b). The experiences introduced in this manuscript are conducted with $100,000$ leakage traces of 800 time samples (see Figure 3.9a).

[f] We recall that the signal-to-noise ratio (see Equation 3.2) quantifies the dependence between the time samples and the targeted sensitive information.

[g] The interested readers may refer to `https://newae.com/tools/chipwhisperer/`.

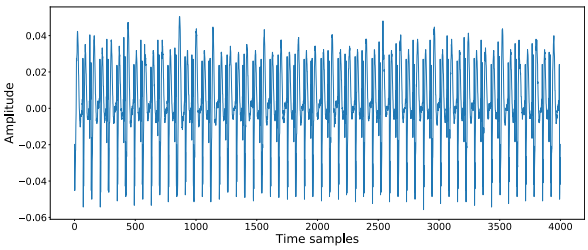


(a) Example of a leakage trace from ChipWhisperer.

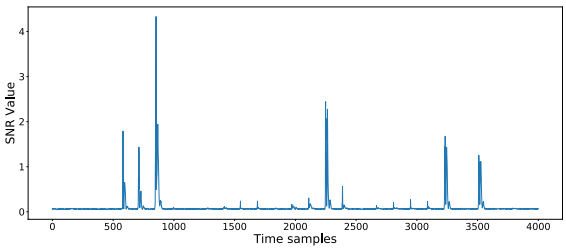


(b) Signal-to-Noise Ratio.

Figure 3.9: ChipWhisperer dataset.



(a) Example of a leakage trace from DPA contest-v4.



(b) Signal-to-Noise Ratio.

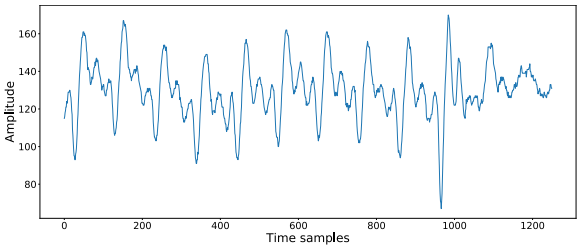
Figure 3.10: DPA contest-v4 dataset.

DPA contest-v4 is a protected AES-128 implemented in software on an Atmel ATMega-163 smart card, which is composed by a 8-bit AVR microcontroller [BBD+14][h]. In order to prevent first-order attack, the Developer protects the implementation with a first-order Boolean masking scheme. However, for this dataset, we assume that the Evaluator knows the mask manipulated by the targeted cryptographic module. Hence, this strong assumption reduces the protection of the implementation such that a first order attack can be performed. Thus, this implementation can be considered as unprotected. In this manuscript, we attack the first round Sbox operation. Hence, each leakage trace $\mathbf{T}$ is labeled with its corresponding sensitive variable $\text{Sbox}[X[0] \oplus k^*] \oplus M$ where M denotes the known mask and $X[0]$ the first byte of the plaintext manipulated by the cryptographic module. The related dataset is composed by $5,000$ leakage traces. Each leakage trace has $4,000$ time samples (see Figure 3.10a) and the highest SNR targeting the sensitive variable with known mask equals 4.33 (see Figure 3.10b).

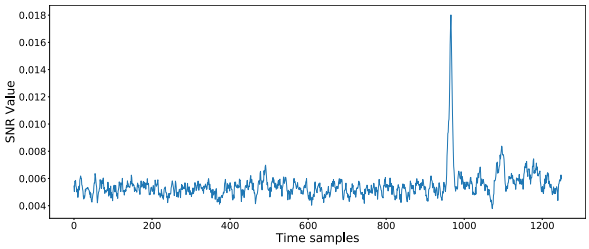
AES_HD is an unprotected AES-128 implemented on Xilinx Virtex-5 FPGA. Introduced in [PHJ+18][i], the authors decided to attack the register writing in the last round such that the label of a leakage trace $\mathbf{T}$ can be described as $\text{Sbox}^{-1}[C[j] \oplus k^*] \oplus C[j']$ where $C[j]$ and $C[j']$ are two ciphertext bytes associated with the leakage trace $\mathbf{T}$, and the relation between j and j' is given by the ShiftRows operation of AES. The authors use $j = 12$ and $j' = 8$. This dataset is composed by a set of $75,000$ leakage traces. Each leakage trace has $1,250$ time samples (see Figure 3.11a) and the highest SNR targeting the sensitive variable equals 0.01554 (see Figure 3.11b).

AES_RD is an AES-128 obtained from an 8-bit Atmel AVR microcontroller, protected by a random delay countermeasure that inserts a random number of dummy operations which is determined by a RNG [CK09][j]. This countermeasure shifts each trace following a random variable.

[h] `http://www.dpacontest.org/v4/42_traces.php`
[i] `https://github.com/AESHD/AES_HD_Dataset`
[j] `https://github.com/ikizhvatov/randomdelays-traces`

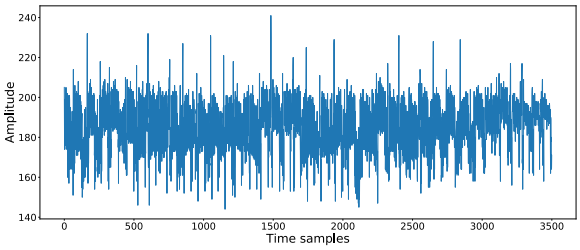


(a) Example of a leakage trace from AES_HD.

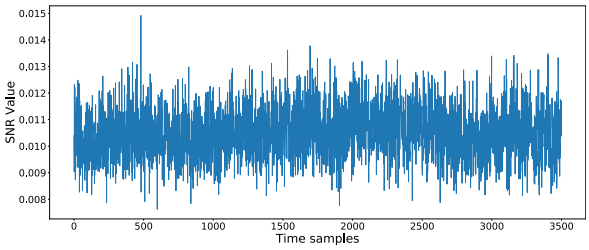


(b) Signal-to-Noise Ratio.

Figure 3.11: AES_HD dataset.



(a) Example of a leakage trace from AES_RD.



(b) Signal-to-Noise Ratio.

Figure 3.12: AES_RD dataset.

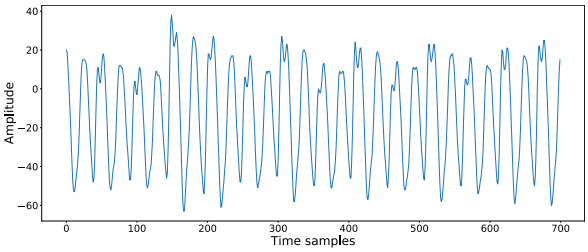
This renders the attack more difficult because of the misalignment. Like DPA-contest v4, the sensitive variable is the first round Sbox operation where each leakage trace $\mathbf{T}$ is labeled as such $\text{Sbox}[X[0] \oplus k^*]$. This dataset contains $50,000$ leakage traces of $3,500$ time samples (see Figure 3.12a) and the SNR targeting the sensitive variable does not identify any relevant leakages (see Figure 3.12b). This phenomenon can be explained by the implementation of the random delay.

ASCAD-v1 is introduced in [BPS+20] and is the first open database[k] that has been specified to serve as a common basis for further works on the application of deep learning techniques in the side-channel context[l]. The target platform is an 8-bit AVR microcontroller (ATmega8515) where a masked AES-128 is implemented. In order to prevent first-order attack, the Developer protects the implementation with a first-order Boolean masking scheme. The targeted sensitive variable is the third byte of the first round Sbox operation. Consequently, each leakage trace is labeled as follows: $Y = \text{Sbox}[X[3] \oplus k^*]$. While no leakages characterizing Y are observed (see Figure 3.13b), the Evaluator or the neural network has to combine an unknown mask, denoted $r3$ and the related masked value, denoted $\text{Sbox}[X[3] \oplus k^*] \oplus r3$, in order to perform a high-order side channel attack. The SNR peaks of the mask, the masked value and the unmasked value are illustrated in Figure 3.13b. This dataset is interesting to assess the ability of a neural network to retrieve Y without any assumption on the unknown mask. In order to make the attack even more difficult, two additional ASCAD datasets have been proposed with an artificial shift of maximum amplitude of 50 and 100 time samples. The ASCAD-v1 dataset contains $60,000$ such that each of them has 700 time samples[m] (see Figure 3.13a).

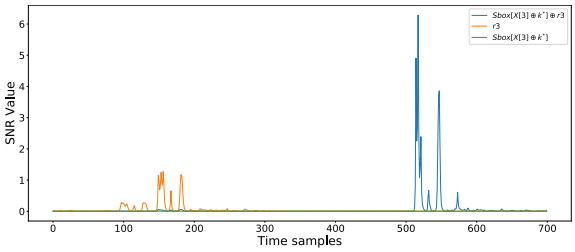
[k] `https://github.com/ANSSI-FR/ASCAD`

[l] Recently, a new dataset known as ASCAD-v2 has been released [MS21]. This implementation is out of the score of this document.

[m] A new version randomizes the keys used in the training set. The resulted leakage traces are twice larger. However, this version is not investigated in this thesis.

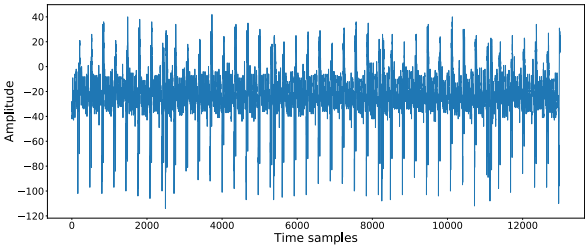


(a) Example of a leakage trace from ASCAD-v1.

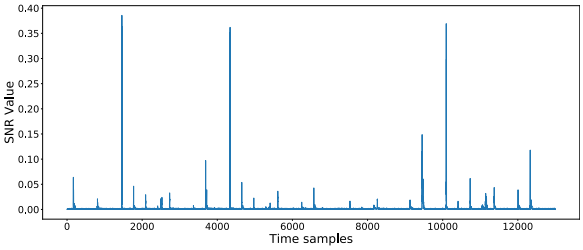


(b) Signal-to-Noise Ratios related to the mask ($r3$), the masked value (Sbox$[X[3] \oplus k^*] \oplus r3$) and the unmasked value (Sbox$[X[3] \oplus k^*]$).

Figure 3.13: ASCAD dataset.



(a) Example of a leakage trace from the secure RSA implementation.



(b) Signal-to-Noise Ratio.

Figure 3.14: Secure RSA dataset.

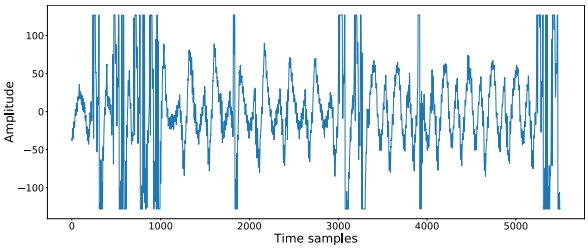
Secure RSA. Introduced in [CCC+19], the targeted RSA implementation is based on a *Left-to-Right Square & Multiply Always* exponentiation algorithm (see [CCC+19, Algorithm 1]) combined with exponent and message blinding. This implementation runs on a 0.13um 32-bit contact Smartcard IC. The software part of the targeted RSA implementation does not provide specific security mechanisms to defeat horizontal or address-bit side-channel attacks. This choice has been done deliberately by CryptoExperts' team[n] who was responsible for the development of the RSA software part. In [CCC+19], the authors highlight that the application of advanced deep learning-based side-channel attacks makes security mechanisms against horizontal and address-bit attacks mandatory to reduce the Evaluator's scope.
For two 512 bits primes p and q, the combination of the three masking countermeasures corresponds to the following equation:

$$m^d \bmod N = ((m + k_1 \cdot N)^{s_{k^*} + k_2 \cdot \phi(N)} \bmod (k_0 \cdot N)) \bmod N,$$

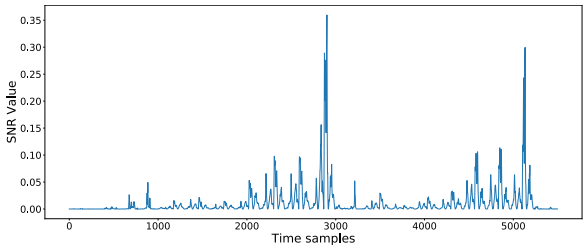
with k_0, k_1, k_2 three random values of bit-length 64, $N = p \times q$ the modulus of $1,024$ bits.
In the *Square & Multiply Always* algorithm (see [CCC+19, Algorithm 1]), Carbone *et al.* identify a vulnerability related to the manipulation of an index named seg_{free}. Indeed, this index stays unchanged for two consecutive exponentiations if the related exponent bit equals 1. If the Evaluator retrieves the value of this index, he can gradually learn the entire exponent bits except for the last one. For each leakage trace, this index value is defined in 0, 1, 2 (see [CCC+19, Equation 5]). For a complete overview of the device under test, we suggest that the readers refer to [CCC+19]. The related dataset is composed by $43,880$ leakage traces of $13,000$ time samples (see Figure 3.14a) and the related SNR computation is defined in Figure 3.14b.

Secure ECC. Defined in [NCOS17], the secure ECSM (Elliptic Curve Scalar Multiplication) algorithm has been implemented in an 8-bit ATmega328P microcontroller [Chm20]. This implementation is constructed from Curve25519 [Ber06] which is a 255-bit elliptic curve. More precisely,

[n] `https://www.cryptoexperts.com/`



(a) Example of a leakage trace from the secure ECC implementation.



(b) Signal-to-Noise Ratio.

Figure 3.15: Secure ECC dataset.

this curve is defined by the equation $y^2 = x^3 + 486662 \cdot x^2 + x$ over the prime field defined by the prime number $2^{255} - 19$ and it uses the base point $x = 9$. It generates a cyclic subgroup whose order is the prime $2^{252} + 27742317777372353535851937790883648493$ and cofactor 8. To protect its implementation, the Developer employs the Montgomery Ladder with randomized projective coordinates and a *conditional swap* (cswap) (see [NCOS17, Algorithm 1]). Starting from two (or more) curve points, the cswap countermeasure performs the scalar multiplication algorithm on one of these points depending on a mask value. Hence, if the Evaluator learns all the conditional swap bits from one side-channel trace, he retrieves the secret key (*i.e.* 256 bits) [NCOS17]. To be successful, the secret bits have to be recovered from a single side-channel trace. In the dataset, each leakage trace represents a single iteration of the Montgomery Ladder scalar multiplication (see Figure 3.15a) and the related label corresponds to the cswap condition bit value. For deeper information on the device under test, we suggest that the readers refer to [NCOS17, Chm20]. The related dataset contains a set of $24,560$ leakage traces of $5,500$ time samples (see Figure 3.15a) and the related SNR computation is defined in Figure 3.15b.

3.7 Conclusion

This chapter introduces the side-channel attacks that are considered in this thesis. The scenario of such attack is summarized in Figure 3.16. From the physical consumptions (*e.g.* power consumption, electromagnetic emanations) of a cryptographic module, the Evaluator identifies the time samples where a sensitive variable is manipulated (see Subsection 3.2.2). Based on this knowledge, he can use statistical tools to approximate the optimal distinguisher (see Definition 3.3.1.4). Typically, two strategies can be considered: non-profiled vs. profiled side-channel attacks. While the first approach consists in targeting a cryptographic module without any prior knowledge on the related leakage model, it is classically considered as less efficient than profiled side-channel attacks which assume a stronger Evaluator. In the latter scenario, the Evaluator has access to an open copy of the device in order to characterize the behavior of the leakage traces defined in Equation 3.1. This characterization is helpful to approximate the optimal distinguisher through the log-likelihood distinguisher (see Definition 3.3.3.4).

However, from a practical perspective, this solution has some limitations. Indeed, the characterization of the leakage traces requires a preprocessing phase that reduces the dimensionality of the leakage traces in order to ease the computation process. In addition, to reduce the risk of side-channel attacks, the community introduces different countermeasures, namely data randomization (see Subsection 3.4.1) and hiding (see Subsection 3.4.2). The first proposition inserts randomness to the sensitive variables in order to uncorrelate the sensitive variable from the secret key, and thus, from the leakage traces. In addition, the hiding countermeasures propose to uncorrelate one leakage trace from another by inserting irrelevant information during the cryptographic process (*e.g.* encryption).
To circumvent these issues, some solutions have been introduced such as the high-order side-

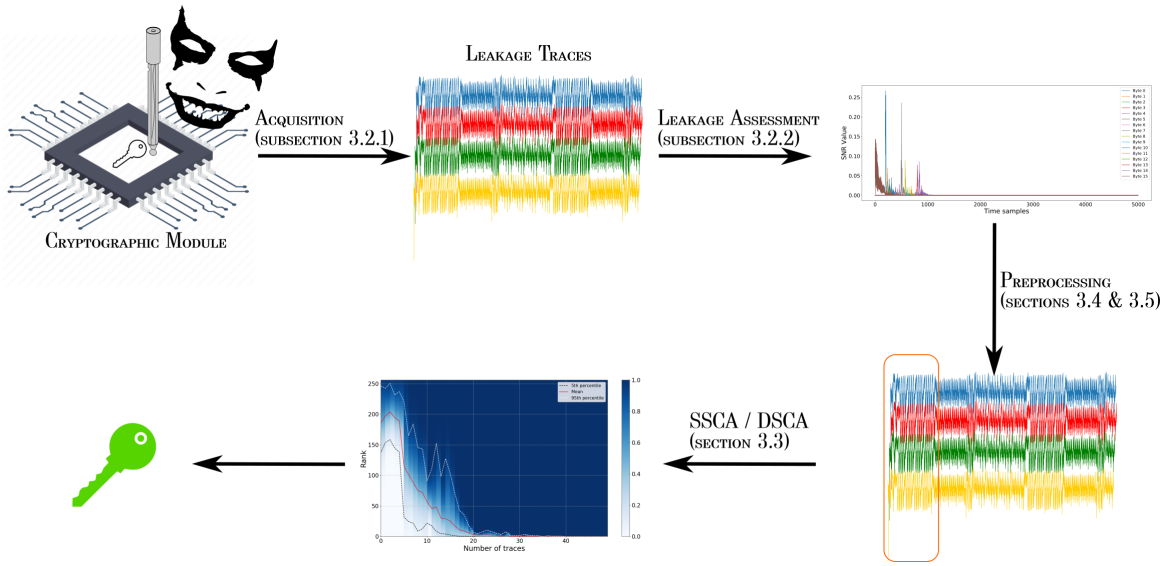


Figure 3.16: Scenario of a side-channel analysis.

channel attacks (see Section 3.5) or the synchronization tools (see Subsection 3.4.2). But the efficiency of those solutions, and consequently the attack performance, highly depends on the Evaluator's expertise.

WHAT'S NEXT?

Hence, while some decision-making are eased by the attack scenario (*e.g.* simple vs. differential side-channel attacks, univariate vs. multivariate side-channel attacks, profiled vs. non-profiled side-channel attacks), others are very challenging to define (*e.g.* selection of points of interest, synchronization tools, the choice of the most effective combining functions) depending on the targeted cryptographic module and the implemented countermeasures. While the decisions provided on the latter challenges highly influence the efficiency of classical side-channel attacks, the Evaluator has to make a suited decision based on his own expertise or, in some worst case (*e.g.* selection of points of interest), following a rule of thumb. A solution suggests in automatically finding the setting that converges towards the optimal attack solution. As a natural consequence, investigating the benefits of the Deep Learning can be useful in order to reach this solution. This following chapter introduces this field and characterizes its beneficits from the Evaluator point of view.

Chapter 4

Deep Learning for Side-Channel Attacks

In this chapter, we describe the basic principles of Deep Learning that are applied throughout the rest of the manuscript. Hence, the first three sections are generic and not specific to the side-channel context. This choice is motivated by the need of understanding all the concepts induced by the deep learning field. With this aim in mind, we begin this chapter with a definition of what a deep learning algorithm is and differentiate the supervised and the unsupervised problems. After presenting the different manners to approximate a function *via* the construction of neural networks, we describe how the training process is performed and define the challenges that occur when the Evaluator aims at approximating an unknown function. Finally, a specific DLSCA case study is proposed and an overview of the DLSCA literature is made. We conclude this chapter by presenting the datasets we consider in this thesis.

Contents

4.1 What is a Deep Learning Model?

4.1.1 Principles of Learning Algorithm

The goal of deep learning is to build algorithms which automatically learn how to solve a task that generally consists in processing the data for addressing a given problem.

Definition 4.1.1.1 (Learning)**.** [Mit97, p. 2] A computer program is said to learn from experience Exp with respect to some class of tasks and performance measure Perf, if its performance for solving tasks, as measured by Perf, improves with experience Exp.

Based on Definition 4.1.1.1, the Evaluator can follow the machine learning paradigm in order to automatically learn mathematical functions (also known as *predictive model*) that captures some properties of the leakage traces in order to make a decision, or return a prediction, related to the secret key manipulated by the cryptographic module. However, no universal approximator has been defined until now. Consequently, some assumptions have to be made before performing the learning phase. More precisely, a *model space* $\mathcal{F}$ (or *hypothesis space*), which defines a set of predictive models, has to be selected based on prior knowledge of the data (*e.g.* leakage traces).

Definition 4.1.1.2 (Learning algorithm)**.** A learning algorithm aims at selecting the most suitable predictive model $F \in \mathcal{F}$ for solving a given task.

Hence, to select this predictive model, a learning algorithm needs experiences represented by a collection of many leakage traces.

Definition 4.1.1.3 (Input)**.** The input of a learning algorithm is defined by a model space $\mathcal{F}$ and a collection of data (or examples), denoted as $\mathcal{I}$, from which a learning algorithm learns to complete some class of tasks.

Definition 4.1.1.4 (Output)**.** The output of a learning algorithm denotes the element F belonging to some model space $\mathcal{F}$ from a set of inputs $\mathcal{I}$, which *best* solves a given task.

Typically, we say that a learning problem is *realizable* if the hypothesis space $\mathcal{F}$ contains the true unknown function which perfectly solves the given task. As classical algorithms, a learning algorithm is no more than a mathematical function $F(.)$ that automatically learns to solve one or several tasks. In this manuscript, the task is characterized by the ability of the learning algorithm to automatically select $F \in \mathcal{F}$ that maps a leakage trace to the correct secret key k^*.

To model such algorithm, the machine learning paradigm is based on the *statistical learning theory*, introduced by Vapnik and Chervonenkis [VC71]. This theory, detailed in Subsection 4.1.2, consists of studying the problem of inference by constructing a model from a set of data. In 2003, Bousquet *et al.* [BBL04] describe it as a mathematical framework which can be roughly summarized in three steps. The transposition to the side-channel context can be presented as follows:

1. Collecting of a set $\mathcal{T}$ of leakage traces involved during a phenomenon we want to characterize (*i.e.* a cryptographic function, *e.g.* encryption, decryption),

2. Estimating of a model $F \in \mathcal{F}$ describing this phenomenon,

3. Making predictions using this model and verifying its suitability following the ground truth Y in order to retrieve information related to the secret key k^* manipulated by the cryptographic module.

From a set of inputs, two phases are needed in order to consider the learning algorithm during an evaluation process. First, the *profiling* (or *training*) phase, consists in monitoring it to complete one or several tasks given a set of profiling inputs $\mathcal{I}_p$. From $\mathcal{I}_p$ and the related predictions, a penalization term is computed in order to correct the decision-making of the learning algorithm and improve its performance for resolving tasks. Then, once the learning algorithm is trained, the *inference* phase uses the resulted F to make predictions against previously unseen and unlabeled set of data $\mathcal{I}_a$. This scenario is typically what the Evaluator does when he performs a *profiled* side-channel attack (see Subsection 3.3.3). Hence, the deep learning-based side-channel attacks can be filed in this category.

To consider such scenario, the Evaluator has to differentiate the multiple experiences exposed by the machine learning community. Typically, these experiences Exp can be categorized into two branches: *supervised learning* and *unsupervised learning*.

Supervised learning. This approach uses a set of inputs (*i.e.* leakage traces) to teach a model to yield a set of labels (*i.e.* $\mathcal{Y}$). In this setting, we assume that the Evaluator has access to a set of N_p labeled (or annotated) leakage traces which can be defined as $\mathcal{I}_p = \{(\mathbf{t}_0, y_0), \ldots, (\mathbf{t}_{N_p-1}, y_{N_p-1})\}$ with $(\mathbf{t}_i, y_i)_{0 \leq i < N_p} \in (\mathcal{T} \times \mathcal{Y})$. Mainly considered for resolving *classification* or *regression* tasks, the supervised learning approach builds a model which predicts a targeted sensitive variable Y from a leakage trace $\mathbf{T}$. The term "supervised" refers the annotated data that the Evaluator uses to correct the decision-making of the learning algorithm by comparing the predicted output $\hat{Y}$ with the real expected label Y.
In supervised learning, the learning algorithm is trained to produce a function $F : \mathcal{T} \to \mathcal{P}(\mathcal{Y})$ that takes a data as input and outputs a PMF over a finite domain $\mathcal{Y}$. This learning process can be split into two tasks: *regression* task and *classification* task. The distinction between these approaches lies in the definition of $\mathcal{Y}$. Typically, in regression tasks, the resulted model F aims at predicting a continuous value in $\mathcal{Y} \subseteq \mathbb{R}$ which characterizes the correlations between labels and D-dimensional inputs. On the other hand, in classification tasks, the learning algorithm is trained to select a model F which assigns a specific *class* $Y \in \mathcal{Y}$ to any input leakage trace. This assignment is usually processed by estimating the conditional probability $\mathsf{Pr}[Y|\mathbf{T}]$ for each class $Y \in \mathcal{Y}$ such that the prediction $\hat{Y}$ is considered as the most likely conditional probability (*i.e.* $\hat{Y} = \arg\max_{y \in \mathcal{Y}} F(\mathbf{T})[y]$). Many algorithms have been presented in the literature for resolving classification tasks: naives Bayes [FGG97], k-nearest neighbors [CH67], kernel support vector machines [BGV92, CV95], decision tree classifiers [BFOS84], *etc.*

Unsupervised learning. This learning algorithm differs from the supervised learning as it handles with an unlabeled training set (*i.e.* $\mathcal{I}_p = \mathcal{T} = \{\mathbf{t}_0, \mathbf{t}_1, \ldots, \mathbf{t}_{N_p-1}\}$). Hence, contrary to the supervised learning approach which produces functions that map an input $\mathbf{T}$ to a class or a continuous value in $\mathcal{Y}$, the unsupervised learning observes several inputs $\mathbf{T}$ and attempts to learn the entire probability distribution $\mathsf{Pr}[\mathbf{T}]$ that generates a dataset. These learning algorithms automatically discover patterns or data groupings without the Evaluator's intervention. Based on similarity measures, it is highly beneficial to perform data analysis, clustering [Llo82, CM02] or dimensionality reduction [Jol86, vdMH08].

The classical profiled side-channel attacks usually approximate the conditional probability distribution $\mathsf{Pr}[\mathbf{T}|Y]$, thus, it can be assigned to a supervised learning experience as it requires to label each leakage trace $\mathbf{T}$. Similarly, deep learning-based side-channel attacks consider the classification task $\mathsf{Pr}[Y|\mathbf{T}]$ as suited to retrieve the secret key involved in the targeted cryptographic module. Hence the leakage traces need to be labeled in order to perform both approaches. However, some differentiation can be highlighted. These specifications will be deeper investigated in Part II. As previously mentioned, two phases have to be conducted for considering a model as operational. The first one, namely profiling phase, aims at reducing the error the model made from several leakage traces. While this process can differ between supervised and unsupervised

approaches, the following section will be specifically oriented for resolving classical side-channel tasks (*i.e.* supervised problems).

The main idea of the following section is to introduce the concept of learning theory which describes the notion of *true risk* and *empirical risk*. These metrics are mandatory to explain how a model F is selected from a finite model space $\mathcal{F}$.

4.1.2 Learning Theory

To train a learning algorithm in supervised setting, the Evaluator has to deal with a training set of labeled D-dimensional leakage traces, such that, the pairs $(\mathbf{T}, Y) \in \mathcal{T} \times \mathcal{Y}$ are independently and identically distributed (*i.i.d*) according to an unknown joint distribution $\mathsf{Pr}[\mathbf{T}, Y]$. Based on a sequence of N_p *i.i.d* pairs $(\mathbf{T}, Y)$, the Evaluator aims at constructing a function $F : \mathcal{T} \to \mathcal{P}(\mathcal{Y})$ which can be used to predict the most likely Y from $\mathbf{T}$. In order to correctly choose F, a criterion has to be set in order to assess the quality of the generated model.

Definition 4.1.2.1 (Loss)**.** A loss is a function $\mathcal{L} : \mathcal{F} \times \mathcal{Y} \to \mathbb{R}^+$ measuring the degree of disagreement between a selected predictive model $F \in \mathcal{F}$, considering $\mathbf{T}$ as input, and the expected Y.

The most natural loss for classification is the $0-1$ loss which can be defined as follows:

$$\mathcal{L}_{0-1}(F(\mathbf{T}), Y) = \begin{cases} 0 & \text{if} \quad \arg\max_{y \in \mathcal{Y}} F(\mathbf{T})[y] = Y, \\ 1 & \text{otherwise.} \end{cases}$$

This loss corresponds to the proportion of incorrectly predicted values $\hat{Y} = \arg\max_{y \in \mathcal{Y}} F(\mathbf{T})[y]$. Thus, considering the $0-1$ loss function is beneficial to measure the error rate of the learning algorithm. Then, to completely assess the quality of the selected model F, the notion of *true risk* has to be clearly introduced.

Definition 4.1.2.2 (True risk)**.** Given a loss $\mathcal{L} : \mathcal{F} \times \mathcal{Y} \to \mathbb{R}$ and a model $F : \mathcal{T} \to \mathcal{P}(\mathcal{Y})$, the true risk $\mathcal{R}$ defines the expected loss over the distribution $\mathsf{Pr}[\mathbf{T}, Y]$:

$$\mathcal{R}(\mathcal{L}, F) = \int_{\mathcal{T} \times \mathcal{Y}} \mathcal{L}\left(F\left(\mathbf{t}\right), y\right) \mathsf{Pr}[\mathbf{t}, y] \, dt \, dy.$$

The smaller the true risk (also known as *generalization error*), the better the model F. Hence, the supervised learning aims at finding a model minimizing this criterion. However, this metric cannot be computed since the joint distribution $\mathsf{Pr}[\mathbf{T}, Y]$ is unknown. Thus, the task of learning algorithm is to minimize the true risk without being able to evaluate it directly. An alternative is to compute an *empirical risk* (also known as *empirical error*) from a labeled training set $\mathcal{I}_p$ of finite size.

Definition 4.1.2.3 (Empirical risk)**.** Given $\mathcal{I}_p = \{(\mathbf{t}_0, y_0), \ldots, (\mathbf{t}_{N_p-1}, y_{N_p-1})\}$, with $(\mathbf{t}_i, y_i)_{0 \leq i < N_p} \in \mathcal{T} \times \mathcal{Y}$, a labeled training set of size N_p, a loss $\mathcal{L} : \mathcal{F} \times \mathcal{Y} \to \mathbb{R}$ and a model $F : \mathcal{T} \to \mathcal{P}(\mathcal{Y})$, the empirical risk $\hat{\mathcal{R}}(\mathcal{L}, F)$ defines the expected loss over $\mathcal{I}_p$:

$$\hat{\mathcal{R}}(\mathcal{L}, F) = \frac{1}{N_p} \sum_{i=0}^{N_p-1} \mathcal{L}\left(F(\mathbf{t}_i), y_i\right). \tag{4.1}$$

The larger the size of the training set is, the closer to the true risk the empirical risk is. Indeed, following the *law of large number* theorem, we immediately obtain that:

$$\mathsf{Pr}\left[\left(\mathcal{R}(\mathcal{L}, F) - \lim_{N_p \to \infty} \frac{1}{N_p} \sum_{i=0}^{N_p-1} \mathcal{L}\left(F(\mathbf{t}_i), y_i\right)\right) = 0\right] = 1. \tag{4.2}$$

This result indicates that with enough leakage traces, the Evaluator can compute an empirical risk which is a good approximation of the true unknown risk.

MAIN RESULT IN DL CONTEXT

Minimizing this empirical risk is thus highly beneficial to generate a suited learning algorithm for completing a given task.

Empirical risk minimization. The principle of *Empirical Risk Minimization* (ERM) is to pick a model F from a finite model space $\mathcal{F}$ that minimizes the empirical risk as follows:

$$F = \underset{F_\Theta \in \mathcal{F}}{\arg\min}\, \hat{\mathcal{R}}(\mathcal{L}, F_\Theta). \tag{4.3}$$

Although the ERM rule seems very natural, this approach may miserably fail. Indeed, the next section discusses the situation where the true risk is much larger than its empirical risk. This behavior can be exposed through an error decomposition inducing a generalization gap.

4.1.3 Generalization Learning Boundary

For constructing a good model, we would like to learn an algorithm which minimizes the empirical risk as a surrogate of the true unknown risk $\mathcal{R}$. More importantly, we expect to select a model F which performs similarly on unseen labeled leakage traces $\mathcal{T}$ drawn from $\mathsf{Pr}[\mathbf{T}, Y]$. In statistical learning theory, this notion is often referred to as the *generalization gap* and it is denoted as $|\mathcal{R}(\mathcal{L}, F) - \hat{\mathcal{R}}(\mathcal{L}, F)|$ for a given loss $\mathcal{L}$ and a given model F included in a finite model space $\mathcal{F}$. For the sake of simplicity, this subsection is focused on binary classification tasks (*e.g.* $\mathcal{Y} = \{-1, 1\}$) but this notion can be extended to multi-classification tasks. When the Evaluator designs and trains a model, we can reasonably hope that the following probability condition holds:

$$\mathsf{Pr}\left[|\mathcal{R}(\mathcal{L}, F) - \hat{\mathcal{R}}(\mathcal{L}, F)| \geq \epsilon\right] \leq \delta, \tag{4.4}$$

with $\epsilon \geq 0$ and $\delta \in [0, 1]$. Bounds of this form are generally derived from the *probability approximately correct* (PAC) learning framework and are usually independent of the unknown distribution $\mathsf{Pr}[\mathbf{T}, Y]$ from which leakage traces are drawn. This solution is useful in order to bound the gap between the true risk and the empirical risk given a model F. One common tool for deriving generalization bounds is the theory of uniform convergence of empirical quantities [VC71]. It means that, for any model $F \in \mathcal{F}$, the true risk $\mathcal{R}$ is bounded by its empirical risk $\hat{\mathcal{R}}$ and a penalty term depending on the number of training leakage traces N_p, the size (also known as *complexity*) of $\mathcal{F}$ and the probability δ.

Theorem 4.1.3.1 (Uniform convergence bound - Finite case)**.** *[BBL04] Let $\mathcal{F}$ be a finite model space, $\mathcal{I}_p$ be a labeled training set of size N_p i.i.d from an unknown probability distribution $\mathsf{Pr}[\boldsymbol{T}, Y]$ and $\delta > 0$. For any $F \in \mathcal{F}$, a generalization bound states that we have:*

$$\mathcal{R}(\mathcal{L}, F) \leq \hat{\mathcal{R}}(\mathcal{L}, F) + \sqrt{\frac{\log |\mathcal{F}| + \log\left(\frac{1}{\delta}\right)}{2 \cdot N_p}}, \tag{4.5}$$

with probability $1 - \delta$.

Theorem 4.1.3.1 highlights the impact of some variables in the generalization gap. First, as illustrated in Equation 4.2, increasing the size of the labeled training set is beneficial to converge the empirical risk towards the true risk by the law of large number property. However, the model complexity (*i.e.* $|\mathcal{F}|$) can badly influence the generalization gap. Intuitively, the larger the value $|\mathcal{F}|$ is, the more models $F \in \mathcal{F}$ exist that will fit the training data to resolve the underlying

problem (*i.e.* classification task). However, considering too complex model class can also be harmful for the generalization gap. Even if Theorem 4.1.3.1 is valuable for finite model space $\mathcal{F}$, this notion has to be extended to continuous cases. In [VC71], the *Vapnik-Chervonenkis* dimension of a model space $\mathcal{F}$, denoted $VC(\mathcal{F})$, has been defined as the cardinality of the largest dataset that can be shattered by $\mathcal{F}$. If arbitrarily large finite sets of leakage traces $\mathcal{T}$ can be shattered by $\mathcal{F}$, then $VC(\mathcal{F}) = \infty$. Hence, using the VC-dimension is helpful to extend the bound defined in Equation 4.5.

Theorem 4.1.3.2 (Uniform convergence bound with VC-dimension)**.** *[BBL04] Let $\mathcal{F}$ be a continuous model space with VC-dimension $VC(\mathcal{F})$, $\mathcal{I}_p$ be a labeled training set of size N_p i.i.d from an unknown probability distribution $\mathsf{Pr}[\boldsymbol{T}, Y]$ and $\delta > 0$. For any $F \in \mathcal{F}$, a generalization bound states that we have:*

$$\mathcal{R}(\mathcal{L}, F) \leq \hat{\mathcal{R}}(\mathcal{L}, F) + 2 \cdot \sqrt{2 \cdot \frac{VC(\mathcal{F}) \left(\log \left(\frac{2 \cdot N_p \cdot e^1}{VC(\mathcal{F})} \right) + 1 \right) - \log \left(\frac{\delta}{2} \right)}{N_p}}, \tag{4.6}$$

with probability $1 - \delta$.

The bound defined in Equation 4.6 is in accordance with the previous observation. Indeed, the general idea is that the model space must be complex enough to achieve a low empirical risk $\hat{\mathcal{R}}$, but not too complex in order to limit the rise of $VC(\mathcal{F})$ (or $|\mathcal{F}|$ for finite case) which leads to high bounds on the true risk $\mathcal{R}$. This phenomenon can be observed in Figure 4.1. More precisely, three phases can be observed.

Remark 4.1.3.1. As highlighted throughout this section, the theoretical frameworks used to bound the generalization gap assume, on the one hand, that the data instances tend to the true unknown joint distribution $\mathsf{Pr}[\mathbf{T}, Y]$ if the amount of data tends to infinity and, on the other hand, that the complexity of the task does not influence the gap between the true risk and the empirical risk. While the first assumption cannot be validated in practice as the Evaluator does not capture an infinite number of leakage traces, the latter issue may fail to describe certain tasks. For instance, in deep learning, the resulted models are often over-parametrized and the complexity-based boundaries failed to describe the generalization capabilities of such deep neural networks [ZBH+17, NLB+19, ZBH+21]. As this research direction remains open, we consider it as out of the scope of this thesis. Consequently, the rest of the manuscript follows the classical generalization bounds defined in the machine learning literature.

UNDERFITTING vs. OVERFITTING

While the Evaluator's goal is to construct a learning algorithm such that the empirical risk is as close as possible to the true unknown risk (see Equation 4.3), he has to find a good tradeoff between model complexity and the upper bound on the true risk in order to generalize its knowledge on unseen datasets. If the model complexity is low, the penalty term can be almost negligible (see Equation 4.5 and Equation 4.6) while the empirical risk is high because of the poor representativity of F which is caused by the low dimensionality of $\mathcal{F}$. Hence, the models included in $\mathcal{F}$ can be not complex enough to fit with its input. For example, considering only linear functions in $\mathcal{F}$ is not suitable if the task that has to be resolved is a polynomial function. This issue is called *underfitting* and it occurs when the model complexity is not able to obtain a sufficiently low empirical risk in comparison with the true risk. On the other hand, if the model complexity increases, the difference between the empirical risk and the true risk decreases unlike the penalty term. Unfortunately, this situation, called *overfitting*, leads to an increase of the generalization gap such that the true risk is much larger than the empirical risk. In other words, the resulted model learns by "*heart*" the labeled training sample but has poor performance on unseen samples drawing from the same unknown distribution $\mathsf{Pr}[\mathbf{T}, Y]$. Hence, the Evaluator has to find a good tradeoff between the minimization of the empirical risk and the ability of the related model to generalize its knowledge on unseen data.

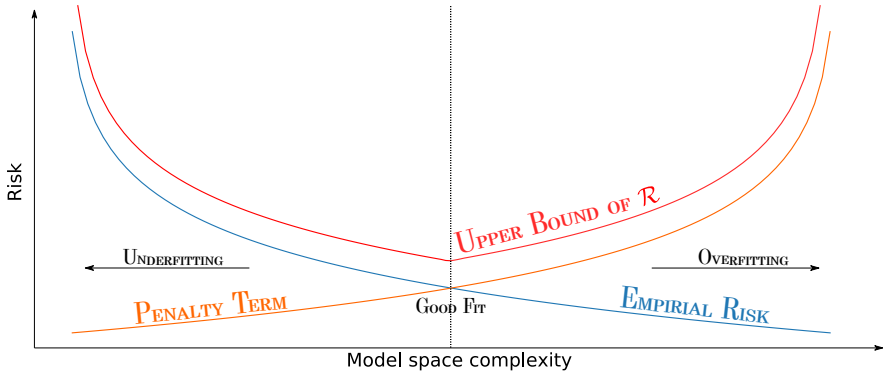
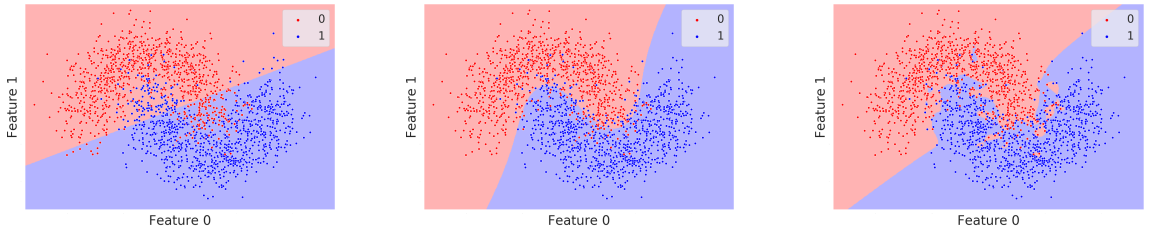


Figure 4.1: Evolution of the Risk over the Model space complexity.



(a) Underfitting **(b)** Good fit **(c)** Overfitting

Figure 4.2: Example of underfitting - overfitting effect on a binary classification task.

An example of underfitting-overfitting effect on binary classification task is provided in Figure 4.2. Finding a good tradeoff is crucial to ensure the construction of a learning algorithm which is able to perform similarly on labeled training sample and unseen data. The tradeoff between the model space complexity and the generalization of the model behavior is the core problematic we want to solve in Chapter 6 for deep learning-based side-channel attacks.

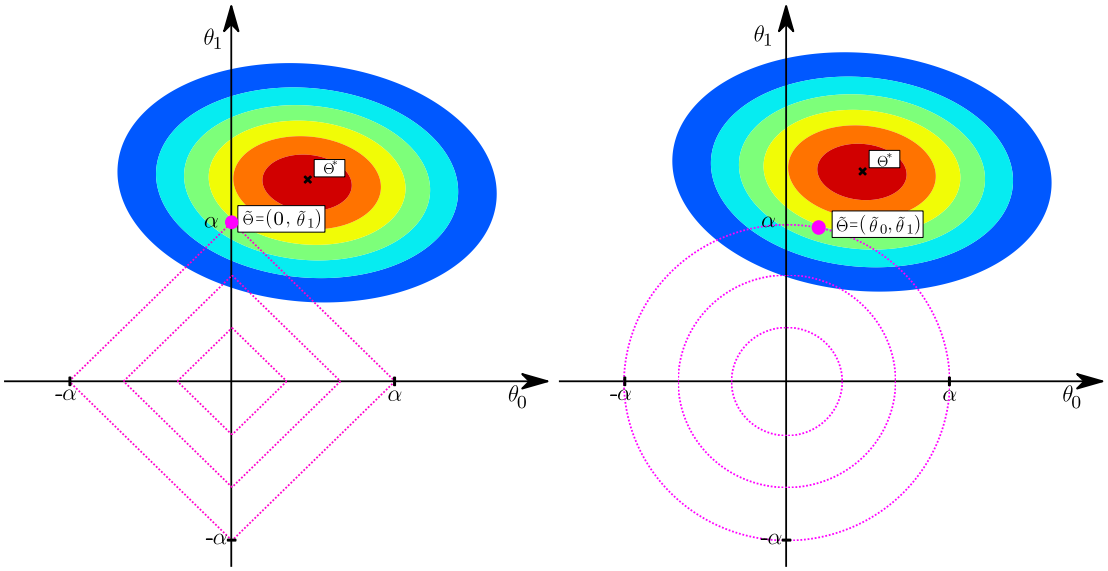
However, even if the model space complexity is too large, the Evaluator can mitigate it through a penalization process that will be introduced in the following section.

4.1.4 Regularization

If the Evaluator chooses its model based on the empirical risk minimization, he can encounter overfitting issues when the model complexity is large. In order to reduce this problem, he can consider the principle of *Regularized Empirical Risk Minimization* (RERM) which consists in picking a model F from a complex space $\mathcal{F}$ and a parameter λ that minimizes the empirical risk as follows:

$$F = \underset{F_\Theta \in \mathcal{F}}{\arg\min} \left(\hat{\mathcal{R}}(\mathcal{L}, F_\Theta) + \lambda \cdot C(F_\Theta) \right), \tag{4.7}$$

with $C(.)$, called *regularizer*, is a model complexity measure. A common practice is to select the p-norm $||.||_p$ on the parameters of the model F_Θ, *i.e.* Θ, as regularizer. The free parameter $\lambda \geq 0$, called the *regularization parameter*, allows choosing the right tradeoff between generalization and model complexity. The role of the RERM is to consider the model complexity in the selection process of a model in order to reduce the overfitting issue.
Typically, the choice of regularizer is important and depends on the considered task and the desired effect. Let us assume that the Evaluator wants to generate a model $F \in \mathbb{R}^2$ such that



(a) L_1-norm **(b)** L_2-norm

Figure 4.3: Geometric interpretation of regularizer using L_1-norm and L_2-norm with $||F_\Theta||_p \leq \alpha$ constraint which minimizes the empirical risk represented by the solid ellipses. The optimal RERM solution is defined by Θ^* but can lead to overfitting issues.

$||F||_p$ is bounded by a constant α. Thus, $\mathcal{F}$ characterizes all the model defined by 2 parameters, denoted θ_0 and θ_1 (*i.e.* $\Theta = [\theta_0, \theta_1]$), at most such that the RERM principle aims at finding the one minimizing Equation 4.7. Figure 4.3 plots the most common norm-based regularizers considered by the deep-learning community, namely the L_1-norm and the L_2-norm. To satisfy Equation 4.7, we have to reach an equilibrium configuration $\tilde{\Theta}$ $(= [\tilde{\theta}_0, \tilde{\theta}_1])$ between the regularizer and the empirical risk. This point is illustrated by a purple circle in Figure 4.3. As we can observe, depending on the regularizer we consider, we do not regularize the empirical risk in the same way. While the L_1 regularizer tends to cut off some parameters by turning their coefficients to zero, the L_2 regularizer tends to shrink these coefficients to a tiny value, while avoiding too large parameters, in order to keep some of their influence on the prediction. This observation can be made in Figure 4.3. Indeed, through Figure 4.3a, the Evaluator forces to choose a model $F \in \mathcal{F}$ such that the following equilibrium point $(0, \tilde{\theta}_1)$ is reached. Hence, considering the L_1 regularizer aims at reducing the complexity of the selected model F by shrinking a parameter to 0. On the other hand, Figure 4.3b slightly differs from L_1 regularizer by choosing a model $F \in \mathcal{F}$ with a non-zero (but relatively small) parameter $\tilde{\theta}_0$.

SUM UP...

To adequately select the most suited predictive model F from $\mathcal{F}$, the Evaluator has to choose the one respecting the following conditions:

1. Minimize the empirical risk defined in Equation 4.3,
2. Generalize its behaviour on unseen data. In such purpose, the generalization gap $|\mathcal{R}(\mathcal{L}, F) - \hat{\mathcal{R}}(\mathcal{L}, F)|$ has to be minimized.
3. Reduce the underfitting and overfitting issues by adequately configuring the finite model space $\mathcal{F}$. Thus, $\mathcal{F}$ should be complex enough to solve the related task (*i.e.* extracting sensitive variable from a leakage trace). However, if the model space is too complex, the Evaluator can consider some regularization tools in order to reduce the overfitting issue.

While the regularization process is suitable to mitigate the model complexity, the Evaluator has to define the space $\mathcal{F}$ which can be characterized through different approaches (*e.g.* kernel support vector machines [BGV92, CV95], decision tree classifiers [BFOS84], *etc*). The following section extends the notion of model estimation through the application of neural networks.

4.2 Model Estimation through Neural Networks

4.2.1 Function Approximation Tools

So far we have introduced the Learning Theory principle as a tool selecting a model F from a generic model space $\mathcal{F}$. As mentioned in Subsection 4.1.1, the aim of supervised deep learning approach is to select a model $F : \mathcal{T} \to \mathcal{P}(\mathcal{Y})$ from $\mathcal{F}$ which takes a D-dimensional leakage trace $\mathbf{T} \in \mathbb{R}^D$ as input and outputs a PMF[a] over a finite domain $\mathcal{Y}$. One solution to construct the related learning algorithm is to design parametric functions such that the parameters that minimize Equation 4.1 are automatically learned. A *parametric model* F_Θ defines a set of functions indexed by a vector Θ of scalar called *weights*. A simple example is a set of linear functions $F_\Theta : \mathbb{R} \to \mathbb{R}$ defined by:

$$F_\Theta(\mathbf{T}) = \theta_0 + \theta_1 \cdot \mathbf{T},$$

with $\Theta = [\theta_0, \theta_1] \in \mathbb{R}^2$ such that θ_0 is also known as the *bias*. A graphical representation of such linear function is illustrated in Figure 4.4a. This figure is a typical example of a *feedforward neural network structure*. The first feedforward neural network, namely *perceptron*, was developed in 1958 [Ros58] and can be expressed as the following parametric model $F_\Theta(\mathbf{T}) = \varrho\left(\left\langle \Theta_{[1:D]}, \mathbf{T} \right\rangle + \Theta[0]\right)$ where ϱ, called *activation function*, is a non-linear function and $\Theta \in \mathbb{R}^{D+1}$. In [Ros58], it is characterized by a threshold activation function which outputs 1 if, for a threshold $\tau \in \mathbb{R}$, $\left\langle \Theta_{[1:D]}, \mathbf{T} \right\rangle + \Theta[0] > \tau$. Since then, many activation functions have been explored by the machine learning community [MHN13, CUH16, HG16]. Given a vector $\Theta \in \mathbb{R}^{D+1}$ and a vector $v \in \mathbb{R}^D$, this thesis will only consider the following ones:

- ReLU – the *rectified linear unit* activation function outputs a vector of non-negative value respecting the following equation:

$$\varrho_{relu}\left(\langle \Theta_{[1:D]}, v\rangle + \Theta[0]\right) = \left\{\max\left(0, \langle \Theta_{[1:D]}, v\rangle + \Theta[0]\right)\right\}.$$

- SELU – the *scaled exponential linear unit* activation function [KUMH17] is an alternative to the ReLU and recommended for its self-normalizing properties. The SELU is defined as follows:

$$\varrho_{selu}\left(\langle \Theta_{[1:D]}, v\rangle + \Theta[0]\right) = \lambda \begin{cases} \langle \Theta_{[1:D]}, v\rangle + \Theta[0] & \text{if} \quad \langle \Theta_{[1:D]}, v\rangle + \Theta[0] > 0, \\ \alpha(exp(\langle \Theta_{[1:D]}, v\rangle + \Theta[0]) - 1) & \text{if} \quad \langle \Theta_{[1:D]}, v\rangle + \Theta[0] \leqslant 0. \end{cases} \tag{4.8}$$

  The SELU function pushes neurons towards zero mean and unit variance in order to prevent vanishing and exploding gradient problems discussed in Subsection 4.3.2.

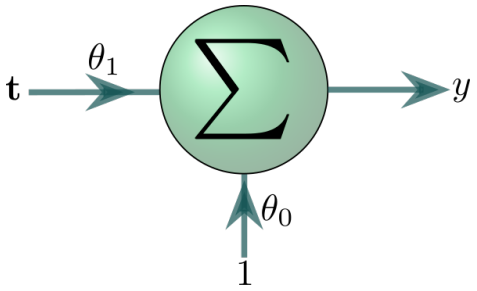
- Softmax/Sigmoid – this activation function normalizes a vector into a probability distribution over $|\mathcal{Y}|$ different possible outputs. It can be computed as follows:

$$\varrho_{softmax}\left(\langle \Theta_{[1:D]}, v\rangle + \Theta[0]\right) = \left\{\frac{e^{\langle \Theta_{[1:D]}, v\rangle + \Theta[0][j]}}{\sum_z e^{\langle \Theta_{[1:D]}, v\rangle + \Theta[0][z]}}, \text{for } j \in [0, |\mathcal{Y}| - 1]\right\}.$$

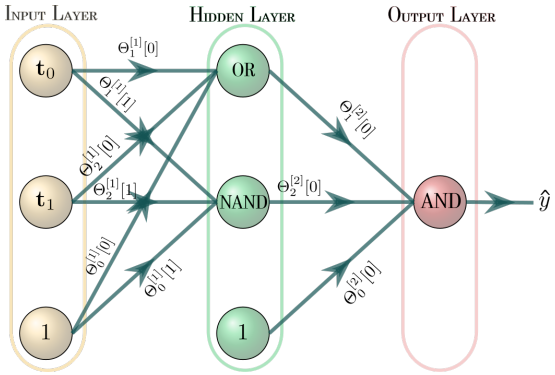
  The softmax is a generalization of the *sigmoid* activation function applied to multiple classes. The sigmoid function, considered to deal with binary classification problems, is characterized as follows:

$$\varrho_{sigmoid}\left(\langle \Theta_{[1:D]}, v\rangle + \Theta[0]\right) = \frac{1}{1 + e^{-\langle \Theta_{[1:D]}, v\rangle + \Theta[0]}}.$$

[a] This notion is defined in Section 2.4.



(a) Linear function considering an input, denoted $\mathbf{t}$, a bias, two parameters (θ_0, θ_1) and an output y.



(b) XOR function.

Figure 4.4: Graphical representations of diverse functions considering the neural network structure.

Even if a perceptron is useful for constructing a parametric model, it considers low model space complexity $\mathcal{F}$ because it can only generate models classifying linearly separable sets of vectors. While the approximation of linear functions can be suited in many scenarios, the Evaluator can suggest the construction of a more complex parametric learning model in order to reduce the gap between the true risk $\mathcal{R}$ and the empirical risk $\hat{\mathcal{R}}$. The Evaluator can then combine multiple *perceptrons* in order to construct a *neural network* which can be represented as a directed acyclic graph such that the weight values are characterized by the edges.

Example 4.2.1.1 (The XOR function.). To illustrate the directed acyclic graph idea, we construct a parametric model approximating the XOR function. As defined in Section 2.2, the XOR function is an operation on two binary values $\mathbf{t}_0$ and $\mathbf{t}_1$ such that it returns 1 if and only if exactly one of these bits equals 1. It returns 0 otherwise. In other words, the XOR operation between two bits can be defined as follows:

$$\mathbf{t}_0 \oplus \mathbf{t}_1 = (\mathbf{t}_0 \vee \mathbf{t}_1) \wedge (\overline{\mathbf{t}_0 \wedge \mathbf{t}_1}).$$

Hence, this function is non-linearly separable by a single decision boundary line: a perceptron cannot generate a suitable parametric model. Indeed, a XOR function can be decomposed into three operations which are non-linearly separable, namely an OR and a NAND which are aggregated into a AND gate. Hence, the resulted directed acyclic graph can be decomposed into three *nodes* (also called *neurons*) such that each of them is modeled by a perceptron. In this example, OR and NAND operations are independent and thus, can be simultaneously approximated such that the weight vector $\Theta_1^{[1]}$ (resp. $\Theta_2^{[1]}$) expresses the importance of the bits $\mathbf{t}_0$ and $\mathbf{t}_1$ to perform the OR (resp. NAND) operation. More precisely, the OR neuron outputs 1 if, at least, one of its inputs equals 1 while the NAND neuron returns 1 if and only if neither input is not activated at the same time. To express the independence between these nodes, the neurons are gathered under the same *layer*. Typically, three types of layers can be defined: the *input layer* receives the input of the parametric model while the last layer, denoted *output layer*, is mostly responsible for the decision-making related to the prediction $\hat{y}$. All the layers in between are called *hidden layers*. Hence, once the OR and NAND are processed in the hidden layer, the related results are put in the last neuron which computes the AND operation (see Figure 4.4b). Depending on the output of the last layer, the Evaluator makes a decision on the estimated $\hat{y}$ characterizing the XOR operation between $\mathbf{t}_0$ and $\mathbf{t}_1$. In addition, while the combination of linear functions is also linear, each neuron has to consider a non-linear activation function (*e.g.* ReLU, SELU, *etc*) in order to correctly approximate the XOR function.

As illustrated, a neural network is no more than a parametric function F_Θ that aims at resolving a given task in $\mathcal{Y}$. However, the Evaluator can question how well an arbitrary neural network approximates his underlying problem.

Universal Approximation Theorem. In [Cyb89], Cybenko proves that a particular neural network, namely *fully-connected neural network* (FCNN), considering one hidden layer and a finite number of neurons can approximate any bounded and regular function if ϱ is sigmoidal (*i.e.* $\varrho_{sigmoid}$). This result was extended to a wide range of non-polynomial activation functions [Hor91, Pin99] but does not ensure that all activation functions perform equally well in specific problems. More formally, the FCNNs respect the *universal approximation theorem* which can be stated as follows.

Theorem 4.2.1.1 (Universal Approximation Theorem)**.** *[Cyb89, Hor91, Pin99] Let $\varrho : \mathbb{R} \to \mathbb{R}$ be any continuous non-polynomial activation function. Let I_D be a compact subset $\mathbb{R}^D$ such that $C(I_D)$ denotes the set of continuous function on I_D. Then, given any $\epsilon > 0$, there exists $N \in \mathbb{N}$, two vectors $v = [v_0, \ldots, v_{N-1}]$, $b = [b_0, \ldots, b_{N-1}] \in \mathbb{R}^N$ and a matrix $\Theta \in \mathcal{M}_{D,N}(\mathbb{R})$ such that we may define:*

$$F_\Theta(\boldsymbol{T}) = \sum_{i=0}^{N-1} v_i \varrho\left(\langle \Theta_i, \boldsymbol{T} \rangle + b_i\right),$$

as an approximation of any function $f \in C(I_D)$,

$$|F_\Theta(\boldsymbol{T}) - f(\boldsymbol{T})| < \epsilon.$$

IN SHORT ...

The Universal Approximation Theorem is interesting from a theoretical point of view because it suggests that a neural network with a single hidden layer can be sufficient to approximate a wide range of functions. However, from a practical perspective, the number of neurons in the hidden layer can exponentially grow, inter alia, with the generalization gap introduced in Subsection 4.1.3.

Fortunately, some alternative solutions can be considered in order to reduce the number of weights induced in a neural network. Indeed, in [ES16], Eldan and Shamir reduce the limitations brought by the universal approximation theorem by suggesting that neural networks with 3 layers can be designed to approximate some functions which cannot be estimated with shallow and wide neural networks. In addition, they show that even if the depth is increased by 1, it can be exponentially more valuable than width. This suggestion was confirmed in [Tel16] for any positive number of layers.

While this section introduces the concept of parametric model, the following one proposes to extend this notion to particular neural networks, namely *fully-connected neural networks* (FCNNs) and *convolutional neural networks* (CNNs).

4.2.2 Neural Network Architectures

To approximate a complex parametric model F_Θ, the Evaluator can extend the notion we briefly introduced in Subsection 4.2.1, namely neural network. More formally, a neural network can be defined as follows.

Definition 4.2.2.1 (Neural Network)**.** Given a D-dimensional input $\mathbf{T}$, a neural network characterizes a parametric model $F_\Theta : \mathbb{R}^D \to \mathbb{R}^{|\mathcal{Y}|}$ which can be defined as the following composition:

$$F_\Theta(\mathbf{T}) = \varrho^{[L]} \circ g^{[L]}_{\Theta^{[L]}} \circ \varrho^{[L-1]} \circ \ldots \circ \varrho^{[2]} \circ g^{[2]}_{\Theta^{[2]}} \circ \varrho^{[1]} \circ g^{[1]}_{\Theta^{[1]}} \circ \mathbf{T},$$

where L defines the number of layers that forms the neural network, $\varrho^{[l]} : \mathbb{R}^{N^{[l]}} \rightarrow \mathbb{R}^{N^{[l]}}$ is a non-linear activation function related to the l^{th} layer which includes $N^{[l]}$ neurons, $g^{[l]}_{\Theta^{[l]}} : \mathbb{R}^{N^{[l-1]}+1} \rightarrow \mathbb{R}^{N[l]}$ denotes the operation induced in the l^{th} layer and involving a matrix $\Theta^{[l]} = \{\Theta^{[l]}_0, \ldots, \Theta^{[l]}_{N^{[l-1]}+1}\} \in \mathcal{M}_{N^{[l-1]}+1, N^{[l]}}(\mathbb{R})$.

Definition 4.2.2.2 (Network Architecture)**.** The network architecture is defined by the shape of the related neural network.

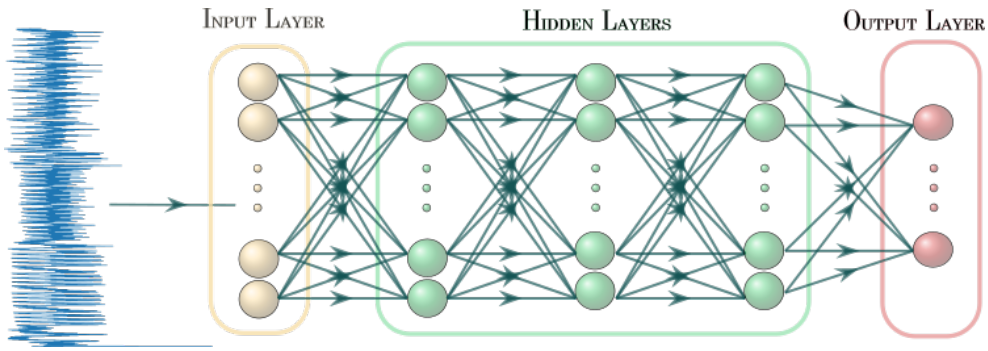
Definition 4.2.2.3 (Network Complexity)**.** The network complexity denotes the number of weights that constitutes the related parametric model F_Θ.

One classical example of neural networks is the *multilayer perceptrons* and refers to neural networks composed by multiple layers of numerous neurons considering ϱ as a threshold activation function (*i.e.* perceptron). An extension of the multilayer perceptrons is the *fully-connected neural network* that defines a neural network such that all the neurons induced in one layer are connected to the neurons in the next layer. In other words, Definition 4.2.2.1 substitutes $g^{[l]}_{\Theta^{[l]}}$ with the dot product between a matrix $\Theta^{[l]} \in \mathcal{M}_{N^{[l-1]}+1, N^{[l]}}(\mathbb{R})$ and a vector v in $\mathbb{R}^{N^{[l-1]}+1}$, denoted $\langle \Theta^{[l]}, v \rangle$. An example of fully-connected neural network applied in side-channel context is provided in Figure 4.5a. The *structure agnostic* property of this architecture is one of its main advantages. Indeed, to consider it, no particular assumption has to be made on the data structure (*e.g.* image, video, leakage trace, *etc*). In side-channel context, this is beneficial to automatically learns the time samples that have to be combined in order to recover the secret key manipulated by the targeted cryptographic module. However, from a practical point of view, the fully-connected neural networks can be difficult to consider due to their extremely high complexity. Thus, following Subsection 4.1.3, this type of neural network can lead to overfitting issues such that the upper bound of the true risk $\mathcal{R}$ increases with its complexity. To circumvent this issue, a solution consists of only combining nearby neurons instead of all of them. In such configuration, the network complexity can be drastically reduced while preserving variables that are temporally and spatially highly correlated. A neural network respecting this restriction, called *convolutional neural network*, has been proposed by LeCun *et al.* in 1989 [LBD+89]. While this manuscript is mainly focused on this neural network structure, a detailed description of the convolutional neural network is proposed.

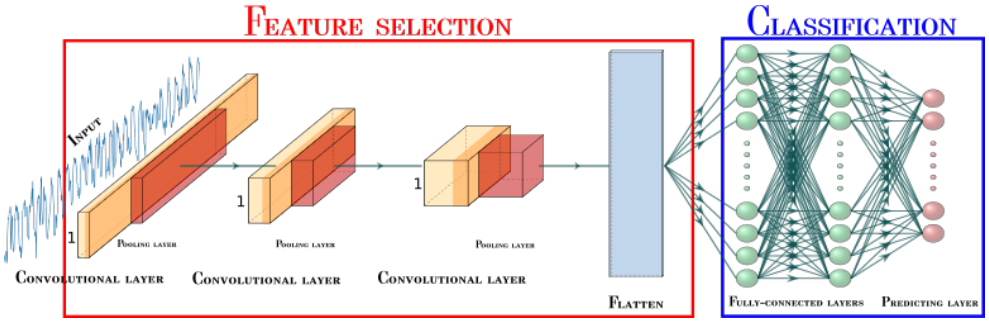
Convolutional Neural Network. In this manuscript, we choose to simplify its representation by considering that it can be decomposed into two parts: a *feature selection* part, which extracts information from a leakage trace $\mathbf{T}$ to help the decision-making (*i.e.* points of interest)[b], and a *classification* part which combines these relevant features in order to correctly retrieve the targeted secret key k^* (see Figure 4.5b). To select features, a CNN is composed of n_3 stacked *convolutional blocks* that correspond to n_2 *convolutional layers* (denoted γ), combined with an activation function ϱ, and one *pooling layer* (denoted δ) [ON15]. One interesting property of the convolutional block is to combine three central ideas, namely *locality*, *shared weights* and *temporal subsampling* [LBBH98]. This is strongly beneficial to reduce the problem of performance loss caused by distortion techniques as those introduced in Subsection 3.4.2. This motivates Cagli *et al.* [CDP17a] to investigate the suitability of such neural network in side-channel context in order to reduce the hiding countermeasures. Then, the feature recognition part is plugged into the classification part of n_1 *Fully-Connected layers* (denoted λ) that are similar to those used in fully-connected neural networks. Finally, a *predicting layer*, denoted $\lambda_{|\mathcal{Y}|}$, composed of $|\mathcal{Y}|$ classes is considered in order to compute $\mathsf{Pr}[Y|\mathbf{T}]$ which is mandatory to perform the log-likelihood distinguisher (see Definition 3.3.3.4). This conditional probability is measured *via* the softmax activation function $\varrho_{softmax}$. To sum up, a common convolutional network can be characterized by the following formula:

$$\varrho_{softmax} \circ \lambda_{|\mathcal{Y}|} \circ [\varrho \circ \lambda]^{n_1} \circ [\delta \circ [\varrho \circ \gamma]^{n_2}]^{n_3}.$$

[b] This assumption is verified in Chapter 6.



(a) Fully-connected neural network structure.



(b) Convolutional neural network structure.

Figure 4.5: Graphical representations of diverse architectures classically considered in deep learning-based side-channel attacks.

The rest of this section deeply describes the different layers of the architecture presented above.

Convolutional layer (γ)**.** The convolutional layer performs a series of convolutional operations on a leakage trace $\mathbf{T}$, used as input, to facilitate the identification of patterns containing a non-negligible deterministic part $\psi(Y)$. In that purpose, each leakage trace $\mathbf{T}$ is convolved with a filter (also called *kernel*). The output of the convolution reveals temporal instants that influence the classification. These time samples are called *features*. To build a convolutional layer identifying suitable features, some parameters have to be configured by the Evaluator, namely *length of filters*, *number of filters*, *stride* and *padding*.

- Filters – They are designed to identify features that increase the efficiency of the classification task, *i.e.* retrieving k^*. However, depending on their size, filters reveal local or global features. Smaller filters tend to identify local features while larger filters focus on global features. Figure 4.6 gives an example in which the length of filters is set to 3.

- Stride – Stride refers to the step between two consecutive convolutional operations. Using a small stride corresponds to the generation of an overlap between different filters while a longer stride reduces the output dimension. By default, the stride is set to 1 (see Figure 4.6).

- Padding – Let $\mathbf{a}$ and $\mathbf{b}$ be two vectors such that $dim(\mathbf{a}) \neq dim(\mathbf{b})$, the dimension of the convolution between these two vectors will be $dim(\mathbf{a} \circledast \mathbf{b}) = \left\lfloor \frac{dim(\mathbf{a})-dim(\mathbf{b})}{stride} \right\rfloor + 1$ where $\circledast$ refers to the convolution operator. In some cases, a subsample may be generated. To avoid this phenomenon and losing information, a padding can be used in order to add a "border" on the related leakage trace $\mathbf{T}$. This processing ensures that the input dimension is retained after the convolutional operation. By default, two kinds of padding are used: *valid padding* and *same padding*. *Valid padding* means "no-padding" while *same padding* refers to a zero-padding (the output has the same dimension as the input). Figure 4.6 gives an example

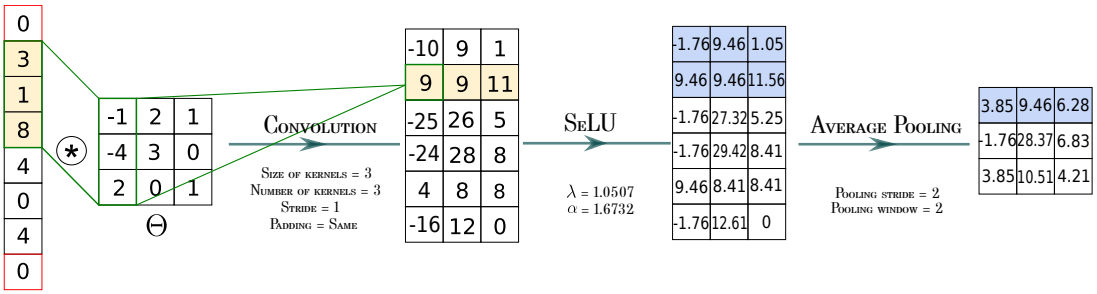


Figure 4.6: Example of computations performed in a convolutional block.

in which same padding is selected. Indeed, two 0 values are added at the endpoints of the vector in order to obtain an output vector of dimension 6.

After each convolutional operation, an activation function is performed in order to only keep the features identified as relevant for the classification.

Pooling layer (δ)**.** The pooling layer is a non-linear layer that divides the dimension of the leakage traces such that the most relevant information is preserved. To apply its down sampling function, a pooling window and a pooling stride have to be configured. Usually, these variables have the same value to avoid overlapping. The window slides through the leakage trace in order to select a segment where the pooling function can be applied. In deep learning, two pooling functions are commonly used:

- MaxPooling – The output of the pooling operation is defined as the maximum value contained in the pooling window.
- AveragePooling – The output of the pooling operation is defined as the average of the values contained in the pooling window. Figure 4.6 shows an example of this function applied to a 1-D vector with pooling_window = pooling_stride = 2.

Flatten layer. Once the set of n_3 convolutional blocks (*i.e.* $[\delta \circ [\varrho \circ \gamma]^{n_2}]^{n_3}$) has been designed, the flatten layer concatenates each intermediate trace of the final convolutional block in order to reduce the 2-D space, which corresponds to the dimension at the end of the convolutional part, into a 1-D space to fit with the expected dimension of the classification part. Let us denote M the input of the flatten layer such that $M \in \mathcal{M}_{n,d}(\mathbb{R})$ where n denotes the number of outputs after the last convolutional block and d denotes the output dimension such that:

$$M = \begin{bmatrix} \mathbf{t}_0[0] & \mathbf{t}_0[1] & \mathbf{t}_0[2] & \cdots & \mathbf{t}_0[d-1] \\ \mathbf{t}_1[0] & \ddots & \cdots & \cdots & \vdots \\ \mathbf{t}_2[0] & \cdots & \ddots & \cdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{t}_{n-1}[0] & \cdots & \cdots & \cdots & \mathbf{t}_{n-1}[d-1] \end{bmatrix} \tag{4.9}$$

where $(\mathbf{t}_i)_{0 \leqslant i < n}$ is the i^{th} intermediate trace, which is defined by the i^{th} output of the last convolutional block, and $(\mathbf{t}_i[j])_{0 \leqslant j < d}$ the j^{th} sample of the i^{th} trace.
The output of the flatten layer is a concatenated vector C that can be constructed following two approaches:

- Column-wise – $C = [\mathbf{t}_0[0], \mathbf{t}_1[0], \mathbf{t}_2[0], ..., \mathbf{t}_{n-1}[d-1]]$,
- Row-wise – $C = [\mathbf{t}_0[0], \mathbf{t}_0[1], \mathbf{t}_0[2], ..., \mathbf{t}_{n-1}[d-1]]$.

Fully-connected layers (λ)**.** Once the convolutional part has selected the relevant features, one or several fully-connected layers are set in order to recombine each neuron and accurately classify each leakage trace $\mathbf{T}$ provided as input of the convolutional neural network. Fully-connected layers can be compared to a fully-connected neural network where each neuron has full connections to all activation functions of the previous layer (see Figure 4.5).

Predicting Layer ($\varrho_{softmax} \circ \lambda_{|\mathcal{Y}|}$)**.** Finally, a convolutional neural network is concluded by a predicting layer which is a fully-connected layer composed by $|\mathcal{Y}|$ neurons such that a softmax activation function $\varrho_{softmax}$ is computed in order to turn a vector of $|\mathcal{Y}|$ real values into a vector of $|\mathcal{Y}|$ real values that sum to 1. Hence, the softmax activation function is useful to convert the output of $\lambda_{|\mathcal{Y}|}$, namely *scores*, to a normalized probability distribution, which can then be considered to compute the log-likelihood distinguisher (see Definition 3.3.3.4).

Designing Issue. To construct an effective convolutional neural network, the Evaluator has to deal with a plethora of parametric variables that he has to configure.

Definition 4.2.2.4 (Model hyperparameters)**.** The model hyperparameters refer to parametric variables whose values are defined by the Evaluator in order to design the neural network architecture.

Depending on the choice of the model hyperparameters (*e.g.* number of layers, number of neurons per layer, length of filters, number of convolutional blocks), the performance of the neural network varies considerably. In convolutional neural networks, these hyperparameters are defined by number of convolutional layers, number of convolutional blocks, number of filters per convolutional layer, length of filters per convolutional layer, number of fully-connected layers, number of neurons per fully-connected layer, *etc.*

OPEN QUESTION

For a neural network, a basic question is how to trade off between its width and depth: Should the Evaluator uses convolutional neural networks that are narrow and deep (many layers, with a small number of neurons per layer), or shallow and wide? Hence, choosing correct model hyperparameters is the first step towards obtaining an optimal neural network.

One classical solution is to consider the convolutional neural networks already proposed by the deep learning community: LeNet [LBD+89], AlexNet [KSH12], VGGNet [SZ15], InceptionNet [SVI+16], ResNet [HZRS16] *etc.* While these convolutional neural networks were specifically designed to fit with the image classification problem, the side-channel community started to investigate their suitability in deep learning-based side-channel attacks [CDP17a, PSK+18, KPH+19, ZS19, BPS+20, JZHY20, GJS20, MS21]. In Chapter 6, we propose to investigate the impact of multiple hyperparameters (*i.e.* length of filters, number of convolutional blocks, pooling layers) in order to design a convolutional neural network specific to the side-channel context such that the convolutional part focuses its interest only on the PoIs.

Once the Evaluator designs a neural network that characterizes a parametric model F_Θ, he has to determine the value of the weights, included in Θ, such that F_Θ is able to retrieve the secret key manipulated by the targeted cryptographic module. To find this solution, he has to perform a so-called *training process* that is based on an *optimization problem* we describe in the following section.

4.3 A Step Towards the Optimal Parametric Model

4.3.1 Optimization Problem

Recall that the goal of a learning algorithm is to find a model $F \in \mathcal{F}$ which minimizes the empirical risk $\hat{\mathcal{R}}$ (see Definition 4.1.2.3) such that $F = \underset{F_\Theta \in \mathcal{F}}{\arg\min}\, \hat{\mathcal{R}}(\mathcal{L}, F_\Theta)$ and respecting a tight generalization gap with the true risk $\mathcal{R}$. Hence, given a parametric model F_Θ and a loss function $\mathcal{L}$, the Evaluator has to find the weights Θ that minimizes the empirical risk as suggested in Equation 4.3. This requirement can be provided by some optimization tools such as the *gradient descent*.

Gradient Descent [C$^+$47]. Gradient descent is an iterative minimization algorithm which consists in computing the gradient of the empirical risk in order to monitor the trainable parameters such that $\hat{\mathcal{R}}$ is minimized. Given a parametric model F_Θ and a differentiable loss function $\mathcal{L}$, the Evaluator has to compute the vector of partial derivatives of $\hat{\mathcal{R}}$ at the weights of the l^{th} layer (*i.e.* $\Theta^{[l]}$), denoted $\nabla_{\Theta^{[l]}} \hat{\mathcal{R}}(\mathcal{L}, F_\Theta)$ (see Equation 2.1). These partial derivatives give the direction in which the empirical risk $\hat{\mathcal{R}}$ has the steepest ascent. Taking the opposite direction is helpful to find the weight values minimizing the loss function such that, from a random starting point $\Theta^{[l]}_{(0)}$, the weights are updated as follows at each iteration:

$$\Theta^{[l]}_{(i+1)} = \Theta^{[l]}_{(i)} - \eta \cdot \nabla_{\Theta^{[l]}_{(i)}} \hat{\mathcal{R}}(\mathcal{L}, F_{\Theta_{(i)}}), \tag{4.10}$$

where $\Theta^{[l]}_{(i)}$ denotes the value of the trainable parameters at the i^{th} iteration and $\eta > 0$, called *learning rate*, is a parameter that the Evaluator has to configure. More precisely, it quantifies the size of the *step* to reach a minimum. After n iterations, the Evaluator can update the weight values following multiple strategies: averaging the weight values $\bar{\Theta}^{[l]} = \frac{1}{n}\sum_{i=1}^{n} \Theta^{[l]}_{(i)}$, retaining the weights induce by the last iteration $\Theta^{[l]}_{(n)}$, or keeping the weight values providing the best performance, *i.e.* $\underset{i \in [1,n]}{\arg\min}\, F_{\Theta_{(i)}}$.

If $\hat{\mathcal{R}}(\mathcal{L}, F_\Theta)$ is a *convex-Lipschitz* function, the number of iterations, needed for reaching the unique optimal solution Θ^* which minimizes $\hat{\mathcal{R}}(\mathcal{L}, F_\Theta)$, such that $\hat{\mathcal{R}}(\mathcal{L}, F_\Theta) - \hat{\mathcal{R}}(\mathcal{L}, F_{\Theta^*}) \leq \epsilon$ is achieved, growths with $\frac{1}{\epsilon^2}$ [SSBD14, Corollary 14.2]. Hence, the number of iterations can be large as ϵ decreases. In addition, while the classical gradient descent algorithm considers the entire training set $\mathcal{I}_p$ to update the weight values Θ, each iteration can be expensive from a computational point of view. Hence, some alternative solutions have to be considered.

One common solution to perform such gradient descent optimization is to apply the *Stochastic Gradient Descent* (SGD) algorithm which approximates the most suitable parametric model F_Θ on a single leakage trace $(\mathbf{t}_j)_{0 \leq j < N_p}$ instead of $\mathcal{I}_p$. Hence, from a random starting point $\Theta^{[l]}_{(0)}$, the weight values are monitored as follows:

$$\Theta^{[l]}_{(i+1)} = \Theta^{[l]}_{(i)} - \eta \cdot \nabla_{\Theta^{[l]}_{(i)}} \hat{\mathcal{R}}(\mathcal{L}, F_{\Theta_{(i)}}(\mathbf{t}_j)[y_j]).$$

This equation is repeated over the entire set $\mathcal{I}_p$ until $\hat{\mathcal{R}}(\mathcal{L}, F_\Theta)$ converges. While only one random leakage trace is processed at a time, the SGD is beneficial from a computational perspective. In addition, for large datasets, the resulted parametric model can converge faster towards the minima of the loss function because the parameters are updated more frequently. From an optimization point of view, the SGD has some limitations. Indeed, due to the multiple iterations performed on a single leakage trace, a high variance can be observed from an iteration to another. This phenomenon leads to unstable convergence to the loss minima. One solution to reduce this issue is to compute the stochastic gradient descent over *mini-batch* that characterizes a set of few training leakage traces included in $\mathcal{I}_p$. This gradient descent algorithm, called *Mini-Batch Gradient*

Descent, is useful to reduce the variance of the parameter updates and leads to stable convergence. To compute the gradient over Θ, the Evaluator has to consider the *backward propagation* described below.

Backward propagation. This algorithm, also called *backpropagation*, was introduced by Rumelhart *et al.* in 1986 [RHW86]. In 1989, Yann LeCun uses backpropagation to train convolutional neural network to recognize handwritten digits [LBD+89]. The backpropagation is based on the chain rule which is used to compute the derivative of parametric models formed by composing of functions.

Definition 4.3.1.1 (Chain rule)**.** [GBC16, Section 6.5.2] Let two vectors $a \in \mathbb{R}^m$, $b \in \mathbb{R}^n$. Let g be a function which maps from $\mathbb{R}^m$ to $\mathbb{R}^n$ such that $b = g(a)$. Let f maps an input from $\mathbb{R}^n$ to $\mathbb{R}$ such that $f(b) = f \circ g(a)$. Then, by the *chaining* rule, the partial derivative of $f(g(a))$ at a can be decomposed as:

$$\nabla_a f(g(a)) = J_g(a)^T \cdot \nabla_{g(a)} f(g(a)), \tag{4.11}$$

where $J_g(a)$ is the Jacobian matrix of g at a (see Equation 2.2).

From this definition, we see that this principle can be used to compute the partial derivative of parametric models as those introduced in Definition 4.2.2.1. The backpropagation algorithm has been proposed to iteratively compute the partial derivatives on each vector composing $\Theta^{[i]} = \{\Theta_0^{[i]}, \ldots, \Theta_{N^{[i-1]}+1}^{[i]}\} \in \mathcal{M}_{N^{[i-1]}+1, N^{[i]}}(\mathbb{R})$ in order to reduce the storage issue of Jacobian matrix.

COMPUTATIONAL TRICK

Even if Definition 4.3.1.1 is focused on vectors, the chain rule can be applied on tensors of arbitrary dimensionality. Indeed, as mentioned in Definition 4.2.2.1, neural networks deal with matrices. Hence, to extend the chain rule principle to any arbitrary dimension, the Evaluator has to flatten the related matrix into a vector before its application. Once the gradient is computed following Equation 4.11, the result can be reshaped in order to fit with the related neural network.

Consequently, given a parametric neural network F_Θ (see Definition 4.2.2.1), the backpropagation algorithm follows Definition 4.3.1.1 in order to facilitate the computation of partial derivative $\nabla_{\Theta^{[l]}} \hat{\mathcal{R}}(\mathcal{L}, F_\Theta)$ at a given layer indexed at l. Indeed, if $a = \mathbf{T}$, $g(a) = F_\Theta = \varrho^{[L]} \circ g^{[L]}_{\Theta^{[L]}} \circ \ldots \circ \varrho^{[1]} \circ g^{[1]}_{\Theta^{[1]}} \circ \mathbf{T}$ and $f \circ g(a) = \hat{\mathcal{R}}(\mathcal{L}, F_\Theta)$, then, we can rewrite Equation 4.11 as follows:

$$\nabla_{\Theta^{[l]}} \hat{\mathcal{R}}(\mathcal{L}, F_\Theta) = \sum_{j=0}^{N^{[l]}} J_{(\varrho^{[L]} \circ g^{[L]}_{\Theta^{[L]}} \circ \ldots \circ \varrho^{[l]} \circ g^{[l]}_{\Theta^{[l]}})[j]} \left(\Theta^{[l]}\right)^T \cdot \nabla_{(\varrho^{[L]} \circ g^{[L]}_{\Theta^{[L]}} \circ \ldots \circ \varrho^{[l]} \circ g^{[l]}_{\Theta^{[l]}})[j]} \hat{\mathcal{R}}(\mathcal{L}, F_\Theta).$$

In other words, the backpropagation algorithm computes the gradient of $\hat{\mathcal{R}}(\mathcal{L}, F_\Theta)$ with respect to the trainable parameters induced in the layer indexed at l (*i.e.* $\Theta^{[l]}$). While the neural network F_Θ is a composition of functions, the chain rule can be performed to iteratively determine $(\nabla_{\varrho^{[L]} \circ g^{[L]}_{\Theta^{[L]}} \circ \ldots \circ \varrho^{[l]} \circ g^{[l]}_{\Theta^{[l]}}} \hat{\mathcal{R}}(\mathcal{L}, F_\Theta))_{1 \leq l \leq L}$. The term *backpropagation* can be explained by the need of computing $\nabla_{\varrho^{[L]} \circ g^{[L]}_{\Theta^{[L]}} \circ \ldots \circ \varrho^{[l+1]} \circ g^{[l+1]}_{\Theta^{[l+1]}}} \hat{\mathcal{R}}(\mathcal{L}, F_\Theta)$ before $\nabla_{\varrho^{[L]} \circ g^{[L]}_{\Theta^{[L]}} \circ \ldots \circ \varrho^{[l]} \circ g^{[l]}_{\Theta^{[l]}}} \hat{\mathcal{R}}(\mathcal{L}, F_\Theta)$ because some terms needed for defining the gradient of $\hat{\mathcal{R}}(\mathcal{L}, F_\Theta)$ at $g^{[l+1]}_{\Theta^{[l+1]}}$ are mandatory to compute $\nabla_{(\varrho^{[L]} \circ g^{[L]}_{\Theta^{[L]}} \circ \ldots \circ \varrho^{[l]} \circ g^{[l]}_{\Theta^{[l]}})} \hat{\mathcal{R}}(\mathcal{L}, F_\Theta)$. The interested reader may refer to [GBC16, Section 6.5.6] for additional information on the backpropagation algorithm.

Remark 4.3.1.1. To accelerate the learning phase, LeCun *et al.* [LBOM12] suggest to preprocessed each dataset such that the leakage traces are standardized (unit variance) and normalized. The rest of this manuscript always considers this preprocessing phase before the beginning of the training phase.

However, given such optimizing algorithm, the Evaluator can question the ability of a learning algorithm to automatically find the optimal trainable parameters Θ^*. The following section underlines the challenges induced by the optimization algorithms.

4.3.2 Optimization Challenges

Optimization can be considered as an arduous task. As defined in Subsection 4.3.1, the application of the gradient descent algorithm requires the consideration of differentiable loss functions. As suggested in Subsection 4.1.2, the $0-1$ loss can be expressed as a suited solution for resolving a classification task. However, while this loss function has a non-negligible issue of null derivative, the use of gradient descent algorithm is useless in such configuration. Consequently, the Evaluator has to optimize alternative loss functions, namely surrogate loss functions, which act as an approximation of the $0-1$ loss function.

Surrogate Loss functions. A classical alternative to the $0-1$ loss function is the *cross-entropy* measure.

Definition 4.3.2.1 (Cross-Entropy)**.** Given $\mathsf{Pr}[Y, \mathbf{T}]$ denoting the joint distribution of a leakage trace $\mathbf{T}$ and the related targeted variable Y, the cross-entropy is defined as:

$$H(Y, \mathbf{T}) = -\mathop{\mathbb{E}}_{Y,\mathbf{T}} \left[\log_2 \left(\mathsf{Pr}\left[Y, \mathbf{T}\right]\right)\right].$$

However, as mentioned in Subsection 3.3.3, Y is assumed to be uniformly distributed. Hence, following Equation 3.4, the cross-entropy can be simplified as follows:

$$H(Y, \mathbf{T}) = -\mathop{\mathbb{E}}_{Y,\mathbf{T}} \left[\log_2 \left(\mathsf{Pr}\left[Y|\mathbf{T}\right]\right)\right].$$

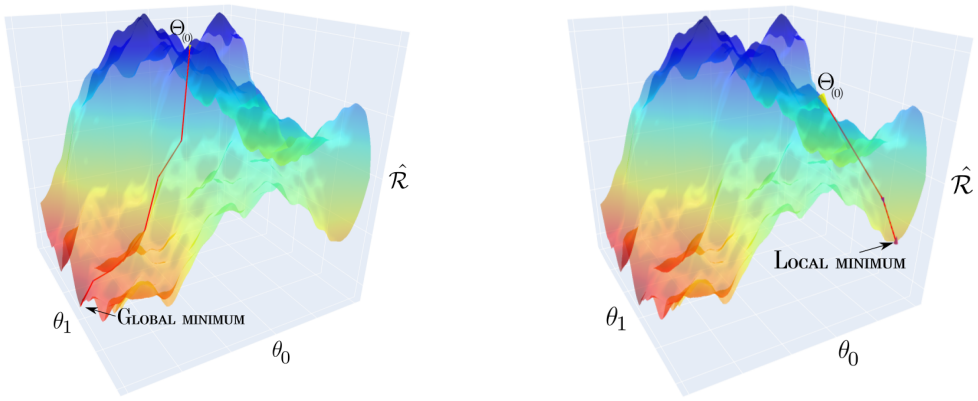
Unfortunately, the computation of the cross-entropy needs to know the joint distribution $\mathsf{Pr}[Y, \mathbf{T}]$ which is impossible for the Evaluator. A solution consists in applying the law of large number in order to approximate the cross entropy. The related loss function, called *negative log-likelihood* (NLL), can be associated with the empirical risk in order to solve this issue.

Definition 4.3.2.2 (Empirical risk combined with negative log-likelihood)**.** Given a set of N_p labeled leakage traces $\mathcal{I}_p = \{(\mathbf{t}_0, y_0), (\mathbf{t}_1, y_1), \ldots, (\mathbf{t}_{N_p-1}, y_{N_p-1})\}$, and a parametric model F_Θ estimating the conditional probability distribution of observing a sensitive cryptographic primitive Y following a leakage $\mathbf{T}$ denoted as $\mathsf{Pr}\left[Y|\mathbf{T}\right]$, the empirical risk combined with the negative log-likelihood loss function can be expressed as:

$$\hat{\mathcal{R}}(\mathcal{L}_{NLL}, F_\Theta) = -\frac{1}{N_p} \sum_{i=0}^{N_p-1} \log_2 F_\Theta(\mathbf{t}_i)[y_i].$$

While the optimal distinguisher (see Definition 3.3.1.4) considers the maximum log-likelihood as the statistical tool, this loss seems a natural choice for generating a suited parametric model F_Θ. Notably, this leads the side channel community to investigate it as the classical loss function to consider when deep learning-based side-channel attacks are performed [MPP16, CDP17a, PHJ+18, CCC+19, MDP19b, BPS+20]. In [MDP19b], Masure *et al.* deeply investigate the benefits of the negative log likelihood to train a specific parametric model for side-channel attacks. However, one limitation of this loss is the lake of concordance with the performance metric that has to be optimized in classical side-channel attacks, *i.e.* success rate (see Definition 3.3.4.2). In Chapter 7, we reduce this issue by defining a new loss function which derives from the success rate and extends the work provided by Masure *et al.*.

However, optimizing a non-convex empirical risk raises questions about the ability of the gradient descent algorithm to converge towards the optimal solution.



(a) Global minimum **(b)** Local minimum

Figure 4.7: Evolution of the gradient descent algorithm (red line) on a two-dimensional loss landscape's projection of a parametric model F_Θ with $\Theta = [\theta_0, \theta_1]$.

Optimizing non-convex functions. The optimization of trainable parameters Θ, through the gradient descent, is beneficial until the derivative produces no information about which direction should be chosen in order to reduce the empirical risk. These points are called *stationary points* (or *critical points*). In other words, for a given parametric model F_Θ, a loss function $\mathcal{L}$ and an empirical risk $\hat{\mathcal{R}}(\mathcal{L}, F_\Theta)$, a stationary point of $\hat{\mathcal{R}}$ is defined if $\nabla_\Theta \hat{\mathcal{R}}(\mathcal{L}, F_\Theta) = 0$. In gradient descent optimization, two stationary points can be underlined: *global minimum* and *local minimum*.

Definition 4.3.2.3 (Global minimum)**.** Given a parametric model F_Θ, a loss function $\mathcal{L}$ and a non-convex empirical risk $\hat{\mathcal{R}}(\mathcal{L}, F_\Theta)$, we define the global minimum as the point of coordinates Θ^*, if, for all Θ, $\hat{\mathcal{R}}(\mathcal{L}, F_{\Theta^*}) \leq \hat{\mathcal{R}}(\mathcal{L}, F_\Theta)$.

Definition 4.3.2.4 (Local minimum)**.** Given a parametric model F_Θ, a loss function $\mathcal{L}$ and a non-convex empirical risk $\hat{\mathcal{R}}(\mathcal{L}, F_\Theta)$, we define the local minimum as the point of coordinates $\Theta^{'}$, if, their exist $\epsilon > 0$ such that, for all Θ respecting $||\Theta^{'} - \Theta|| < \epsilon$, $\hat{\mathcal{R}}(\mathcal{L}, F_{\Theta'}) \leq \hat{\mathcal{R}}(\mathcal{L}, F_\Theta)$.

In the context of gradient descent optimization, the stationary points that are neither local or global minimum are known as *saddle points*. While neural networks, induced in the empirical risk $\hat{\mathcal{R}}$, lead to generate non-convex functions, the gradient descent algorithm can typically converge towards a local minimum. This situation can be problematic if this point has a high empirical risk in comparison with the optimal solution (*i.e.* global minimum). In Figure 4.7, we provide an example of application of the gradient descent algorithm given a parametric model F_Θ, such that $\Theta = [\theta_0, \theta_1]$, a loss function $\mathcal{L}$ and the related empirical risk $\hat{\mathcal{R}}(\mathcal{L}, F_\Theta)$. Depending on the random initialization $\Theta_{(0)}$, we observe that the gradient descent does not converge towards the same minimum. Indeed, in Figure 4.7a, the gradient descent converges towards a global minimum while its application in Figure 4.7b does not. In this example, we observe that $\hat{\mathcal{R}}(\mathcal{L}, F_{\Theta^*})$ and $\hat{\mathcal{R}}(\mathcal{L}, F_{\Theta'})$ do not provide a similar empirical risk value. So, to obtain the most efficient parametric model F_Θ, the Evaluator expects the gradient descent algorithm to converge towards the global minimum. But finding this solution cannot be ensured. However, promising works suggest that even if the local minima with high empirical risk can be considered as an optimization issue, increasing the complexity of the parametric model F_Θ can be useful to exponentially reduce this risk [DPG+14, GVS15, AZLS19]. Another solution consists in considering the stochastic gradient descent instead of the gradient descent itself. As mentioned in Subsection 4.3.1, the SGD can be computed for reducing the computation cost. In addition, the application of this optimization

algorithm on non-convex function converges in the sense of getting to points where the gradient is arbitrarily small [BT00]. However, this result does not mean that the solution converges towards a global or a local minimum. Even if an adequate small learning rate suggests an *almost surely* convergence towards a local minimum [LSJR16] with a random initialization of the trainable parameters Θ, the SGD is not widely used in practice because the convergence can be extremely time-consuming. However, this problem remains an active research area and no general statements can be formally provided.

Depending on the slope of the derivative $\nabla_{\Theta^{[l]}}\mathcal{R}(\mathcal{L}, F_\Theta)$, the convergence of the gradient descent algorithm can be highly impacted and finding a local minimum can be an arduous task. This issue, known as the *vanishing & exploding gradient* problem is detailed below.

Vanishing & exploding gradient. When the Evaluator trains a parametric model F_Θ through the gradient descent algorithm, he updates its trainable parameters (*i.e.* Θ) in order to minimize the empirical risk $\mathcal{R}(\mathcal{L}, F_\Theta)$. In such purpose, he has to compute $\nabla_{\Theta^{[l]}}\mathcal{R}(\mathcal{L}, F_\Theta)$ such that the weights get updated proportionally to the gradient (see Equation 4.10). Hence, a very small the gradient, known as *vanishing gradient problem*, does not effectively update the trainable parameters from a previous iteration. In the worst-case scenario, Θ stays stable such that the optimization process does not improve the related parametric model F_Θ. Due to the chain rule principle used to perform the backpropagation algorithm, this issue grows with the depth of a neural network. This phenomenon leads to converge towards poor local minima or saddle points. While a vanishingly small gradient impacts the optimization of a parametric model, a large gradient does not necessarily improve it and can result in unstable gradients leading to a poor learning. While the goal of the gradient descent algorithm is to converge towards the global minimum, this issue, known as *exploding gradient problem*, can lead to a divergence of the optimization algorithm.

One solution to reduce both impacts is to adequately monitor the learning rate introduced in Equation 4.10. While a vanishing gradient problem induces a high learning rate in order to "get out" from a local minimum or a saddle point, a exploding gradient problem reduces the learning rate value for converging towards a suited point. The learning rate is defined as an *optimizer hyperparameters* that is defined as follows.

Definition 4.3.2.5 (Optimizer hyperparameters)**.** The optimizer hyperparameters refer to parametric variables whose values are defined by the Evaluator in order to configure the training and the optimization processes.

Correlated with the ability of the gradient descent algorithm to converge towards a suited local minimum, the learning rate is considered as one of the most difficult optimizer hyperparameter to configure [GBC16, Section 8.5]. To reduce this issue, other optimizing algorithms with *adaptive* learning rates have been proposed in the literature: Adagrad [DHS11], RMSProp [TH12], Adadelta [Zei12], Adam [KB15], *etc.* In this manuscript, we particularly choose the *adaptive moment estimation* (Adam) as optimizer. The reasons that motivate us can be enumerated as follows. First, to limit the negative impact of unstable gradients at the i^{th} iteration, it computes an exponentially decaying moving average of the previous gradients. This solution demonstrates a high benefit to converge faster towards the solution [Qia99] while reducing the gradient oscillation induced by its instability. Then, to limit the vanishing and exploding gradient problems, the learning rate is monitored by the first and the second moment of the gradients, such that, this process also includes a bias correction induced by the initialization of the optimization algorithm.

SUM UP...

To adequately train a parametric model F_Θ, the Evaluator has to perform an optimization algorithm such as the gradient descent, the stochastic gradient descent or the mini-batch gradient descent in order to find the weights minimizing the empirical risk $\mathcal{R}(\mathcal{L}, F_\Theta)$. In order to perform such algorithm, the backpropagation principle has to be conducted in order to compute $\nabla_\Theta \hat{\mathcal{R}}(\mathcal{L}, F_{\Theta_{(i)}})$. However, the Evaluator has to deal with many challenges as the choice of the loss function, the convergence towards a global minimum and the vanishing & exploding gradient issues. As no universal solution exists to solve these issues, the Evaluator has to find alternatives in order to limit their impact: *e.g.* , using the Adam optimizer, fine tuning the learning rate, finding a specific loss function to the side-channel context.

Considering all the approaches we presented so far, the Evaluator can define a strategy, derived from Chapter 3, in order to construct a model which automatically extracts the secret key manipulated by a targeted cryptographic module.

4.4 Application to the Side-Channel Context

4.4.1 Objective & Strategy

As mentioned at the end of Chapter 3, classical side-channel attacks need multiple steps to retrieve the secret key (*i.e.* leakage assessment, preprocessing, SSCA/DSCA) that require the expertise of the Evaluator. While the Evaluator spends the majority of his time to preprocess the leakage traces (*i.e.* synchronization, PoIs' selection, combining the time samples in order to perform a high-order attack), the goal of deep learning-based side-channel analysis is to reduce this phase by generating learning algorithms which automatically find the configuration optimizing the efficiency of side-channel attacks [MPP16, CDP17a]. While deep learning can be considered to perform non-profiled side-channel attacks [Tim19], this manuscript is mainly focused[c] on the application of deep learning techniques to perform profiled side-channel attacks.

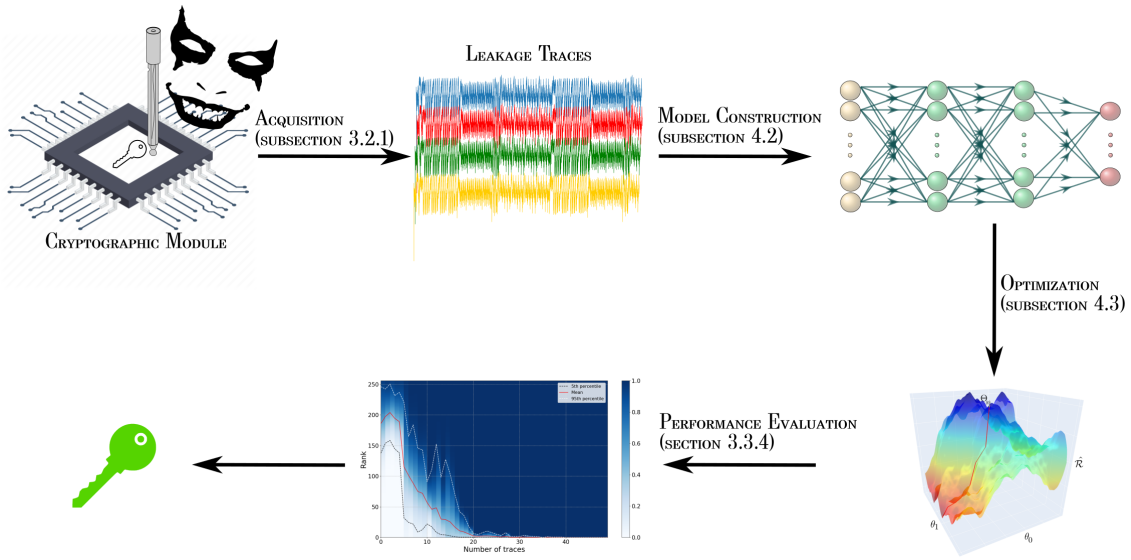


Figure 4.8: Scenario of a deep learning-based side-channel analysis.

[c]Nevertheless, the readers should noticed that all the contributions, provided in this manuscript, can be easily extended to perform non-profiled side-channel attacks if the scenario introduced by Timon [Tim19] is followed.

Similarly to the scenario of classical profiled side-channel analysis (see Figure 3.16), the Evaluator has to capture a set of leakage traces $\mathcal{I}_p$ from a targeted cryptographic module. The main difference between both approaches lies in the statistical tools considered to estimate PDFs $(\widehat{\mathsf{Pr}}[\mathbf{T}, Y = y])_{y \in \mathcal{Y}}$ for a given leakage trace $\mathbf{T}$ and a set of sensitive variable $\mathcal{Y}$ (see Figure 4.8).

Model construction. In deep learning-based side-channel analysis context, the Evaluator has to configure a model F_Θ which maps the leakage traces to the expected targeted sensitive variables. Hence, he has to define suited model hyperparameters in order to increase the ability of F_Θ to retrieve the secret key. While no particular suggestions are provided to successfully construct a neural network in side-channel context, the literature recently investigates the benefits of using tools which automatically tune model hyperparameters [MPP16, BPS+20, WPP20, PRA20, RWPP21, YAGF21]. Even if these suggestions are useful to generate powerful parametric models F_Θ, the model hyperparameters space is often complex and high-dimensional as the Evaluator does not have any clues to construct an efficient neural network. Hence, the *elapsed time*, introduced in Subsection 1.2.2 as an evaluation criterion, can be highly impacted. In order to reduce this issue, Chapter 6 investigates the impact of model hyperparameters on the ability of a CNN to retrieve the points of interest. This proposition is helpful to give some hints to the Evaluator and reduces the model hyperparameters space as well as the *elapsed time* criterion.

Once the parametric model F_Θ is designed, the Evaluator conducts the optimization process as described in Section 4.3.

Optimization. First, the Evaluator has to choose the most adequate loss function to train F_Θ in order to retrieve the secret key manipulated by the targeted cryptographic module. While the negative log likelihood has been demonstrated as suited to conduct side-channel attacks, non-negligible errors can be induced by this loss [MDP19b]. One alternative could consider a loss function optimizing the success rate in order to adhere with the optimal adversary's objective (see Objective 3.3.1.1). A discussion and a proposition, reducing this issue, are detailed in Chapter 7. Then, in order to deal with the optimization challenges, the Evaluator can consider the solutions provided in Subsection 4.3.2, but some optimizer hyperparameters, as the learning rate or the number of iterations required by the descent gradient algorithm, stay problematic to optimize. Even if some solutions are proposed in the literature to optimize the learning rate [Smi17, ST17], no generic rules can be stated. In contrast, the number of iterations of the gradient descent algorithm can be defined during the training process thanks to *early stopping*. This technique is considered to determine how long a gradient descent algorithm has to be iterated in order to obtain a parametric model generalizing well its performance on unseen leakage traces. In such purpose, a set of leakage traces $\mathcal{T}$ can be divided into two sets: a training set $\mathcal{I}_p$ used to train F_Θ, and a validation set $\mathcal{I}_v$ used to evaluate the training process. Once the gradient descent algorithm has been performed on each leakage trace[d] included in $\mathcal{I}_p$, the Evaluator compares the ability of the parametric model to retrieve the secret key from $\mathcal{I}_p$ with the unseen data included in $\mathcal{I}_v$. Typically, the machine learning literature classically considers the accuracy as a suited performance metric. However, as detailed in Subsection 3.3.4, this performance metric cannot be considered for all cryptographic algorithms. While the accuracy is adequate to assess the training process when an asymmetric cryptographic algorithm is targeted [CCC+19, WPB19], the guessing entropy and the success rate are the most suited performance metrics to evaluate the optimization process of a parametric model against symmetric cryptographic implementations [CDP17a, PHJ+18]. In such purpose, the community investigates various solutions to conduct early stopping in deep learning-based side-channel context. These solutions are based on: the success rate [RZC+19], the mutual information [PBP20] or the perceived information [MDP19b]. Even if the early stopping technique is considered beyond the scope of this manuscript, this tool

[d] This process is known as an *epoch*. In other words, it refers to the number of iterations processed by the gradient descent in order to consider the whole training set.

will be considered in Chapter 8 to prevent the overfitting issue and assess the quality of the training process against asymmetric cryptographic implementations.
Once the optimization algorithm is performed, the Evaluator has a parametric model F_Θ which approximates the true unknown PDFs $(\mathsf{Pr}[Y = y|\mathbf{T}])_{y\in\mathcal{Y}}$. Hence, from the log-likelihood distinguisher defined in Definition 3.3.3.4, the Evaluator can guess the most likely key candidate for a given set of N_a attack traces with a fixed key. Hence, the main difference between classical and deep learning-based side-channel approach holds in the estimation of the unknown PDFs. A deeper discussion will be provided in Part II in order to bridge the gap between these two approaches.

Explainability and Interpretability. Once the parametric model F_Θ is obtained, the Evaluator has to explain and interpret its decision-making in order to make sure that the resulted attacks succeed for appropriate reasons (*e.g.* the expected points of interest are successfully exploited). Furthermore, he also has to explain to the Developer, through the Evaluation Technical Report (see Subsection 1.2.2), the security flaws exploited by F_Θ in order to find suitable countermeasures. Chapter 5 and Chapter 6 propose some visualization tools on diverse neural networks in order to ease the explainability and interpretability of the model's results.

EVALUATOR'S GOALS

When the Evaluator conducts an evaluation with neural networks, he wants to:

1. Adequately choose the space dimension of $\mathcal{F}$ in order to limit the underfitting/overfitting issues defined in Subsection 4.1.3.
2. Optimize the elapsed time criterion without altering the attack performance.
3. Explain and Interpret the model's results in order to highlight the security flaws and ease the design of suitable countermeasures by the Developer.

Apart from these problematics, the Evaluator can consider multiple deep learning approaches in order to enhance its evaluation. The following section summarizes the recent research directions proposed by the community.

4.4.2 Related Work

While the deep learning-based side-channel area is a recent field (see Figure 1.6), it is required to get a clear overview of the works provided by the community. This section aims at briefly introducing each research direction.

Preprocessing. While the goal of the deep learning-based side-channel analysis is to reduce the preprocessing phase, several techniques remain beneficial to reduce some risks as the *class imbalance* issue. This phenomenon arises when the classes, defined by $\mathcal{Y}$, are not equally represented[e]. It induces a non-negligible bias such that the trained parametric model emphasizes the majority class. To circumvent this issue, the side-channel community adopts diverse techniques as SMOTE [PHJ+18, WJB20], desynchronization [CDP17a], noise addition [KPH+19], Mixup [LZW+21, Abd21]. Other preprocessing approaches like de-noising [WP20, KKH21] or leakage trace embedding [WPP21] have been proposed to enhance the resulted side-channel attacks. Finally, while the main deep learning-based side-channel attacks are performed in time domain, the community starts to investigate the benefits of other representations (time-frequency domain [YLMZ19], image representation [WHJ+21a, HHGG20]).

[e]A classical example considers the Hamming Weight as leakage model ψ. For example, if $Y \in \mathbb{F}_2^8$, the probability of observing $HW(4)$ equals approximately 0.273 which is more than 2 times higher than uniform probability.

Mitigate the Hiding countermeasures. The CNNs have been introduced in side-channel context to reduce the desynchronization effect due to its invariance translation property [CDP17a]. Indeed, this neural network has shown surprisingly good performance against a wide range of hiding countermeasures as: random process operations [CDP17a], shuffling [ZOB18, Mag19a, Mag19b], code polymorphism [MBC+20], dummy operations [ZOB18, Mag19a, LH20]. However, even if CNNs are suited to limit the effect of desynchronization effect, Zhou and Standaert suggest considering alignment methods before training a neural network as they observed that the training process is faster on synchronized leakage traces [ZS19].

Architectures. While this thesis is mainly focused on the CNNs, other architectures have been studied in the literature: fully-connected neural networks [MZ13, MHM14, Wei20], ResNets [ZS19, JZHY20, GJS20, MS21], Generative Adversarial Networks [WCL+20], RNN [LLY+20], Transformer Neural Network [HSAM21] or Attention mechanisms [LZC+21]. However, the benefits of these neural networks regarding the CNN are not highlighted. This leads us to focus our interest only on the CNN. This suggestion will be confirmed in Chapter 6.

Multi-Task Learning. While classical supervised learning approaches generally train a parametric model to map a leakage trace to a sensitive variable, the multi-task learning consists in targeting simultaneously multiple sensitive variables for a given input. In [ZXF+19], Zhang *et al.* decompose the output space $\mathcal{Y}$ to $\mathbb{F}_2^n$ with $n = \log_2(|\mathcal{Y}|)$. Hence, instead of targeting one word of n bits, they train a neural network to map a leakage trace to n words of 1 bit. By assessing the ability of a neural network to retrieve each bit, the Evaluator can identify the bits where the sensitive information is retrieved. Another solution consists in targeting simultaneously several intermediate operations and/or bytes. This approach has been proposed by Maghrebi [Mag20] and extended in [Mas20, Section 8.2.2]. Hence, instead of considering the "*Divide & Conquer*" strategy (see Subsection 3.3.1) which aims at repeating the training and the inference processes for each target (*e.g.* 16 times for an AES-128 implementation), the Evaluator can target multiple sensitive bytes in parallel, and thus, reduce the *elapsed time* criterion. On the other hand, if the Evaluator simultaneously targets several intermediate operations, the *elapsed time* criterion stays similar to the "*Divide & Conquer*" strategy, but the sensitive information exploited by the parametric model increases with the number of targeted intermediate information. Thus, the number of attack leakage traces that are needed to retrieve the secret key can decrease and the related Evaluator get closer to Objective 3.3.1.1. Even if the multi-task learning is a suited approach from a certification perspective, its study stays out of the scope of this manuscript.

Multi-Channel. Another suited proposition from an evaluation point of view is to combine the source of different channels (*e.g.* power consumption and electromagnetic emanations) [SA08, YZC+17]. The first works considering the deep learning approach to combine different channels are proposed in [GMGH19, HFL+20]. More precisely, they collect electromagnetic emanations captured with multiple probes placed on different areas of the targeted cryptographic module, in order to capture different sources of information. Then, they construct a *multi-channel neural network* [Kim14] that simultaneously operates on the related collected leakage traces. Similar approach is provided in [WHJ+21b]. However, instead of considering different probe position, Won *et al.* combine multiple representations of the same leakage trace such as a reduced leakage trace resulting from the application of the *Principal Component Analysis* (PCA) algorithm or being subject to a moving average technique.

Portability. Classical profiled side-channel attacks imply attacking one cryptographic module with a leakage model generated from a similar copy, but this concept is known as a hard problem to deal with. Indeed, intrinsic differences between the cryptographic modules cause variations in the resulted leakage model and thus, can lead to an unsuccessful attack. While this thesis does not take into consideration this issue, known as *portability*, the following works [CCC+19, WBFD19, BCH+20, RBA20] aim to understand the effect of these variations on the resulted

attack performance and propose solutions to reduce the portability issue. In [CZLG21], Cao *et al.* add a constraint term to the loss function in order to fine-tune a model trained on a cryptographic module similar but not identical to the one used during the attack phase. In [vdVPB21], Van der Valk *et al.* use the *Singular Vector Canonical Correlation Analysis* tool to investigate internal similarities between fully-connected neural networks that targeted diverse implementations.
In addition, the Evaluator can help the neural networks to reach a global minimum by providing further useful information that are beneficial to retrieve the targeted secret key. The approach, known as "*domain knowledge*", refers to the methods inserting additional knowledge to the neural network in order to enhance its performance. This knowledge can be integrated in the form of equations, logic rules, and prior distribution into the neural network. Hence, in addition to the leakage traces, the Evaluator can insert the plaintext as input of the neural network in order to encourage the secret key extraction [HGG19, HHO20]. However, from an evaluation perspective, the domain knowledge raises the following question: *how to insert the new inputs such that the neural network can efficiently exploit its information?*

Transfer Learning & Mimicking. As mentioned in Subsection 4.4.1, finding a suited neural network architecture is not a trivial task. A solution consists in refining parametric models that have been trained to solve different, but still similar, tasks. From these pre-trained parametric models, the Evaluator looks at reducing the time needed for constructing and training a new parametric model from scratch. From an evaluation perspective, this is highly beneficial because the reduction of the *elapsed time* criterion can be non-negligible. While the very first works demonstrate promising results [GMGH20, TAM20, vdVKPB20], transfer learning remains an open topic in the deep learning-based side-channel analysis.

Unsupervised. As the classical profiled and non-profiled side-channel attacks can be included in the supervised learning paradigm, the majority of the research direction is focused on this approach. In [PCBP20], Perin *et al.* propose the first end-to-end unsupervised approach in deep learning-based side-channel attacks. Targeting a secure ECC implementation, they iteratively correct each bit by relabeling them following the confidence of the neural network. However, even if this work is a first insight of how to perform unsupervised learning in side-channel analysis, this approach is still unexplored. In [CLM20], Cristiani *et al.* suggest as a future work to extend the well-known *Mutual Information Neural Estimator* in an unsupervised way in order to conduct side-channel attacks.

To assess the benefits of these approaches, the deep learning side-channel community mainly applies their related propositions on a wide range of implementations described in Section 3.6. To perfectly use these datasets regarding the application of the deep learning approach in the side-channel context, we have to decompose them into three subsets: a profiling set $\mathcal{I}_p$, a validation set $\mathcal{I}_v$ and an attack set $\mathcal{I}_a$. The detailed decomposition is provided in Table 4.1 for each dataset considered in this manuscript.

Table 4.1: Decomposition of the datasets into a training set $\mathcal{I}_p$, a validation set $\mathcal{I}_v$ and an attack set $\mathcal{I}_a$.

	ChipWhisperer	**DPA contest-v4**	**AES_HD**	**AES_RD**	**ASCAD-v1**	**Secure RSA**	**Secure ECC**
$\mathcal{I}_p$	$45,000$	$4,000$	$45,000$	$20,000$	$45,000$	$30,000$	$20,000$
$\mathcal{I}_v$	$5,000$	500	$5,000$	$5,000$	$5,000$	$3,000$	$2,000$
$\mathcal{I}_a$	$50,000$	500	$25,000$	$25,000$	$10,000$	$10,880$	$2,560$

4.5 Conclusion

To reduce the risk enounced at the end of Chapter 3, namely reducing the preprocessing phase, the Evaluator can adopt the deep learning paradigm in order to automatically find a model that

maps a leakage trace to the related sensitive variable. In this chapter, we recall the definition of a learning algorithm and introduce the empirical risk minimization which is beneficial to adequately choose the algorithm that is the most powerful to complete a given task. However, even if the ERM is a suited tool to select a model, the related model space complexity can be problematic regarding the generalization purpose (see Equation 4.4). Indeed, following its complexity, the related model can encounter difficulties to perform well on unseen leakage traces. This issue is known as *overfitting* and can be mitigated through the application of the regularized empirical risk minimization principle which adds a penalty term to the empirical risk $\hat{\mathcal{R}}$.

To construct the learning algorithm, the Evaluator has to design a parametric function F_Θ from neural network structures (*e.g.* multi-layer perceptron, fully-connected neural network, convolutional neural network, residual neural networks). The choice made in this manuscript is to only consider the convolutional neural networks because the literature [CDP17a, Mag19a, MBC+20] demonstrates its robustness against hiding countermeasures. Once the parametric function is characterized, the Evaluator updates the trainable parameters induced in F_Θ in order to optimize the ERM. Even if the deep learning paradigm is highly beneficial to reduce the limitations of classical side-channel attacks, it is unclear how both fields can be connected.

WHAT'S NEXT?

As breafly defined in Subsection 4.1.1, classical side-channel attacks retrieve the secret key by approximating the following conditional probability distribution $\mathsf{Pr}[\mathbf{T}|Y]$. This approach, known as *generative*, can be opposed to the traditional deep learning neural network which estimates the following conditional probability distribution $\mathsf{Pr}[Y|\mathbf{T}]$, known as *discriminative*. While both approaches can be considered as supervised learning, some differentiations can be highlighted. The next part opposes both approaches. While Chapter 5 proposes the first generative model in order to perform deep learning-based side-channel attacks, Chapter 6 introduces a new design to construct discriminative models. Those propositions aim at bridging deep learning and classical profiled side-channel attacks.

Part II

Generative vs. Discriminative Models in Deep Learning Side-Channel Attacks

Chapter 5

Learning Generative Models from Side-Channel Attacks' Foundation

To bridge deep learning and classical profiled side-channel attacks, this chapter firstly describes one of the most generic profiled attacks, namely the *stochastic attack* [SLP05], that approximates the targeted leakage model ψ as a pseudo-Boolean function. After a generic discussion about the generative models, we reduce the gap between the deep learning and the side-channel context by introducing a new neural network architecture, namely *conditional variational autoencoder*, that is based on the stochastic attack. Then, this new approach is faced with typical side-channel issues namely, interpretability, curse of dimensionality, and protection with Boolean masking scheme. Finally, a comparative investigation is proposed with profiled side-channel attacks and some limitations regarding the application of generative models in side-channel analysis are highlighted.

Contents

5.1 Leakage Trace's Characterization

5.1.1 Approximation of a pseudo-Boolean function

Introduced in [SLP05], the stochastic attack is a profiled side-channel attack which takes advantage of the Evaluator's ability to construct a suitable leakage model's estimation. Indeed, as mentioned in Subsection 3.3.3, a profiling attack suggests that the Evaluator has access to an *open* cryptographic module such that he can estimate the PDFs $(\mathsf{Pr}[\mathbf{T}|Y = y])_{y\in\mathcal{Y}}$. Let $\mathbf{T} = [\mathbf{T}[0], \mathbf{T}[1], \ldots, \mathbf{T}[D-1]]$ be a D-dimensional leakage trace depending on a plaintext $X \in \mathbb{F}_2^n$ and a secret key $k^* \in \mathbb{F}_2^n$. Following Equation 3.1, we know that a leakage trace can be decomposed into a deterministic part, defined by a D-dimensional leakage model $\psi(Y = f(X, k^*))$ with f denoting a cryptographic primitive, and a random part $\mathbf{Z}$, characterized by a Gaussian noise $\mathcal{N}_D(0, \Sigma)$, that does not depend on X and k^*. Hence, in such scenario, the Evaluator approximates the D-dimensional leakage model $\psi(Y) = \{\psi_0(Y), \psi_1(Y), \ldots, \psi_{D-1}(Y)\}$ in order to construct the related PDFs.

Theorem 5.1.1.1. *[SLP05, Theorem 1] Let $k^* \in \mathbb{F}_2^n$ be the secret key, then, the following assertions hold:*

1. *Let $i \in [0, D-1]$, then, the minimum*

$$\min_{\hat{\psi}_0,\hat{\psi}_1,\ldots,\hat{\psi}_{D-1}} \mathbb{E}\left[\left|\left|\Big(\boldsymbol{T}[0] - \hat{\psi}_0(f(X,k^*)), \boldsymbol{T}[1] - \hat{\psi}_1(f(X,k^*)), \ldots, \boldsymbol{T}[D-1] - \hat{\psi}_{D-1}(f(X,k^*))\Big)\right|\right|^2\right]$$

   is reached when $\hat{\psi}_0, \hat{\psi}_1, \ldots, \hat{\psi}_{D-1} = \psi_0, \psi_1, \ldots, \psi_{D-1}$ such that $(\psi_i)_{0\leq i<D}$ denotes the unknown deterministic part at the i^{th} time sample. Furthermore, $||.|| : \mathbb{R}^n \to \mathbb{R}$ denotes the Euclidean norm as $||(a_0, a_1, \ldots, a_{D-1})||^2 = \sum_{j=0}^{D-1} a_j^2$. Futhermore, if $\mathsf{Pr}[X = x] > 0$ for all $x \in \mathbb{F}_2^n$, the minimum is exclusively obtained for $\hat{\psi} = \psi$.

2. *For each $x \in \mathbb{F}_2^n$, we have $\psi_i(f(x, k^*)) = \underset{X=x}{\mathbb{E}}[\psi_i(f(x, k^*)) + \boldsymbol{Z}[i]]$.*

IN SHORT...

Given a leakage trace $\mathbf{T}$ such that its i^{th} time sample can be defined as $\mathbf{T}[i] = \psi_i(f(X, k^*)) + \mathbf{Z}[i]$, the goal of the stochastic attack is to find an approximation of the leakage model, denoted $\hat{\psi}_i$, as close as possible to the true unknown $\psi_i(f(X, k^*))$.

Approximation of ψ_i. Let $\mathcal{G}$ be the set of functions mapping $\mathbb{F}_2^n$ to $\mathbb{R}$ which forms a $\mathbb{R}$-vector space of dimension 2^n. More formally, $\mathcal{G}$ describes a set of *pseudo-Boolean functions* that can be defined as follows.

Definition 5.1.1.1 (Pseudo-Boolean function)**.** [Car10, Section 2.1] Given a variable $Y \in \mathbb{F}_2^n$, any function $g \in \mathcal{G}$ can be written as a multilinear polynomial such that there exists a set of real coefficients $(\alpha_u)_{u\in\mathbb{F}_2^m}$ (s.t. $m \leq n$) as:

$$g_\alpha(Y) = \sum_{u=(u[0],\ldots,u[m-1])\in\mathbb{F}_2^m} \alpha_u \cdot \Phi_u(Y),$$

where $(\Phi_u(Y))_{u\in\mathbb{F}_2^m}$ denotes the *monomial basis* of $\mathcal{G}$ and characterizes the conjunction of all bits of Y as $\Phi_u(Y) = Y^u = \prod_{j=0}^{m-1} Y[j]^{u[j]}$ with $Y[j] \in \mathbb{F}_2$ defines the j^{th} bit of Y and the power notation is simply $Y[j]^0 = 1$ and $Y[j]^1 = Y[j]$.

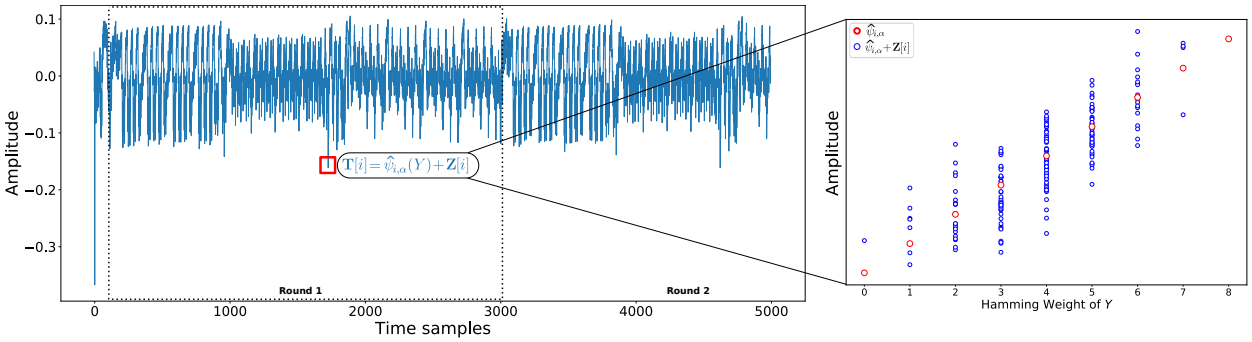


Figure 5.1: Simulation of a linear leakage model $\hat{\psi}_{i,\alpha}$ characterized by Equation 5.2 such that the weighting is similar for all $(\alpha_j[i])_{0\leq j<8}$. The red points define the real unknown leakage model ψ_i while the blue points denote a sampling coming from the distribution of $\mathbf{T}[i]$ for 200 leakage traces.

As the leakage model aims at characterizing the interaction between the bits of Y, Schindler *et al.* approximate ψ as a pseudo-Boolean function [SLP05] such that $m = n$. In the rest of this manuscript, this assumption is followed. Typically, the stochastic attack assumes that a leakage model ψ_i, at a time sample i, can be approximated as a multivariate polynomial in the bit-coordinate $Y[j]$ with coefficients in $\mathbb{R}$.

Definition 5.1.1.2 (Degree of a pseudo-Boolean function)**.** Let $g_\alpha : \mathbb{F}_2^n \to \mathbb{R}$ be a pseudo-Boolean function. We denote d the degree of g_α as the degree of the multilinear polynomial of the monomials $(Y^u)_{u\in\mathbb{F}_2^n}$ (s.t. $d \leq n$).

In other words, the degree of a pseudo-Boolean function g_α is defined by the Hamming Weight of u such that it characterizes the maximal number of bits' interaction induced in the approximation of $\psi_i(Y)$. If we consider a subspace $\mathcal{G}_{d+1}$ that contains all the functions of degree lower than or equal to d, the approximation of the leakage model, denoted $\hat{\psi}_{i,\alpha}$, can be computed as follows:

$$\hat{\psi}_{i,\alpha}(Y) = \alpha_b[i] + \underbrace{\underbrace{\sum_{j<n} \alpha_j[i]Y[j]}_{\text{degree 1 : linear basis}} + \sum_{\substack{j_1,j_2 \\ j_1<n, j_2<n}} \alpha_{j_1,j_2}[i]Y[j_1]\cdot Y[j_2] + \ldots}_{\text{degree 2 : quadratic basis}} + \underbrace{\sum_{\substack{j_1,\ldots,j_d \\ j_1<n,\ldots,j_d<n}} \alpha_{j_1,\ldots,j_d}[i] \prod_{j'=j_1}^{j_d} Y[j']}_{\text{degree d}}, \quad (5.1)$$

where $\alpha_b[i]$ denotes the bias induced at the i^{th} time sample. Following Equation 5.1, we can notice that the degree d represents the maximum number of bits' interaction that can be viewed as logical operator (*e.g.* AND or XOR).

Example 5.1.1.1 (Approximation of the linear basis.). To illustrate the idea of approximating a pseudo-Boolean function, we describe the behavior of a leakage trace such that its deterministic part ψ_i is assumed to be in $\mathcal{G}_2$ (*i.e.* $d = 1$). Hence, the pseudo-Boolean function is defined by a multilinear polynomial of degree 1 such that, given a sensitive variable $Y \in \mathbb{F}_2^8$, we observe:

$$\hat{\psi}_{i,\alpha}(Y) = \alpha_b[i] + \sum_{j=0}^{7} \alpha_j[i]Y[j]. \quad (5.2)$$

If all the bits $(Y[j])_{0\leq j<8}$ have the same positive weight values $(\alpha_j[i])_{0\leq j<8}$, then, the leakage model can be characterized as a linear function defined in Figure 5.1. Hence, from a set of leakage trace $\mathcal{T}$, the Evaluator wants to approximate ψ_i such that Theorem 5.1.1.1 is respected. However, depending on the noise level, this approximation can be erroneous such that the higher the noise the less accurate the estimation $\hat{\psi}_{i,\alpha}$. More precisely, given a degree d, the Evaluator has to find the coefficients $(\alpha_u)_{u\in\mathbb{F}_2^n}$ that satisfy Theorem 5.1.1.1. This process is described in Subsection 5.1.2.

However, considering the monomial basis (see Definition 5.1.1.1) can be problematic from an interpretation perspective. Thus, the Evaluator, who wants to precisely assess the bits' interactions induced in ψ, has to consider another basis that we describe in the following section.

5.1.2 Approximation of the Leakage Model

Orthonormalizing the monomial basis. As defined in [GHMR17], two requirements are mandatory to make an easy interpretation of the leakage structure. First, the basis of $\mathcal{G}$ should be orthonormal so that each vector that constitutes it is uncorrelated with each other. This requirement ensures that the Evaluator who wants to only approximate the leakage structure of degree d, *i.e.* retrieving the real coefficients $(\alpha_u)_{u\in\mathbb{F}_2^n}$ characterizing the interaction between bits, is not impacted by other terms induced in Equation 5.1. On the other hand, the chosen basis should be able to characterize the bit combinations in order to ease the resulted interpretability. While the monomial basis respects the latter requirements (see Definition 5.1.1.2), it is not orthonormal.

Remark 5.1.2.1. As a linear mapping exists between the non-orthogonal basis and its orthogonalized basis, the monomial basis can still be considered to perform a stochastic attack.

Definition 5.1.2.1 (Orthonormal basis)**.** Let $\mathcal{G}$ be the set of pseudo-Boolean function such that $(\Phi_u)_{u\in\mathbb{F}_2^n}$ forms a basis of $\mathcal{G}$. We define $(\Phi_u)_{u\in\mathbb{F}_2^n}$ as orthonormal if the inner product $\langle\Phi_u, \Phi_v\rangle = 0$ if $u \neq v$ and $\langle\Phi_u, \Phi_v\rangle = 1$ if $u = v$.

As defined in [GHMR17, Lemma 3], the monomial basis is not orthonormal.To circumvent this issue, Guilley *et al.* [GHMR17] propose to decompose the monomial basis in order to isolate the leakages and assess the impact of each bit as well as their interactions on ψ. Thus, the authors apply a *Gram-Schmidt Transform* on the ordered[a] monomial basis. Through this application, they introduce a new orthonormal monomial basis that uncorrelates each basis vector and preserves the degree of bits' interaction. This solution is beneficial to evaluate the ability of $\hat{\psi}_\alpha$ to retrieve the unknown leakage model ψ as well as maintaining its interpretability. Surprisingly, it is similar as an orthonormal projection of the leakage traces on the *Fourier basis* denoted $\Phi_u^{(orth.)}(Y)$.

Definition 5.1.2.2 (Orthonormal Monomial Basis [GHMR17])**.** Given a sensitive intermediate value $Y \in \mathbb{F}_2^n$, the orthonormal projection of a monomial basis is defined as:

$$\Phi_u^{(orth.)}(Y) = \frac{1}{2^{n/2}}\left(-1\right)^{Y\cdot u} = \frac{1}{2^{n/2}}\prod_{i=0}^{n-1}\left(1-2\cdot Y[i]\right)^{u[i]},$$

where $u \in \mathbb{F}_2^n$.

Using the orthonormal monomial basis has a major benefit. As shown by Kasper *et al.* [KSS10], when the basis is able to describe the switching activity of the circuit, the estimated basis coefficients highlight specific exploitable security flaws in the studied implementation. Hence, visualizing these coefficients $(\alpha_u)_{u\in\mathbb{F}_2^n}$ is useful to get deeper information on the exploitable security flaws. From Definition 5.1.2.2, we can rewrite Equation 5.1 as follows,

$$\hat{\psi}_{i,\alpha}(Y) = \frac{1}{2^{n/2}}\sum_{u\in\mathbb{F}_2^n}\langle\hat{\psi}_i, \Phi_u^{(orth.)}\rangle\cdot\Phi_u^{(orth.)}(Y) = \frac{1}{2^{n/2}}\sum_{u\in\mathbb{F}_2^n}\alpha_u[i]\cdot\Phi_u^{(orth.)}(Y). \tag{5.3}$$

Once the Evaluator adequately chooses its basis, he can approximate the leakage model ψ through a profiling phase and then, assessing its estimation through an attack phase.

[a]Here, this term refers to an ordering according to the monomial degrees.

Profiling phase. As defined in Theorem 5.1.1.1, the goal of the stochastic attack is to find a leakage model $\hat{\psi_\alpha}$ characterizing a suited approximation of the real unknown leakage model ψ. Considered as a profiled side-channel attack, the stochastic attack can be decomposed into a profiling phase and an attack phase.
For the profiling phase, the stochastic attack mechanism approximates the leakage function using linear regression. Firstly, it consists in choosing the degree d of the pseudo-Boolean function that has to be estimated. Then, given a set of N_p labeled leakage traces $\mathcal{I}_p = \{(\mathbf{t}_0, y_0), (\mathbf{t}_1, y_1), \ldots, (\mathbf{t}_{N_p-1}, y_{N_p-1})\}$, the Evaluator estimates the leakage model $(\hat{\psi}_{i,\alpha}(Y))_{Y\in\mathbb{F}_2^n}$ by finding the best set of coefficients $(\alpha_u[i])_{u\in\mathbb{F}_2^n}$ through the application of the *ordinary least squares* (OLS) method. The *law of large number* theorem implies:

$$\underbrace{\frac{1}{N_p}\sum_{j=0}^{N_p-1}(\mathbf{t}_j[i]-\hat{\psi}_{i,\alpha}(y_j))^2}_{\text{OLS}} \overset{N_p\to\infty}{\longrightarrow} \mathbb{E}\left[\left(\mathbf{T}[i]-\hat{\psi}_{i,\alpha}(Y)\right)\right)^2\Big].$$

The set of coefficients $(\alpha_u[i])_{u\in\mathbb{F}_2^n}$ which minimizes this sum are called the *OLS estimator for* ψ. Once those coefficients are selected for each time sample $i \in [0, D-1]$, the Evaluator obtains an estimation of the unknown leakage model $(\psi_{i,\alpha})_{0\leq i<D}$. Then, he approximates the random part $\mathbf{Z}$ of a leakage trace $\mathbf{T}$. In stochastic attacks, the noise $\mathbf{Z}$ is assumed to follow a multivariate Gaussian distribution $\mathcal{N}_D(0,\Sigma)$. In order to approximate this distribution, the Evaluator has to compute a random vector, based on $(\hat{\psi}_{i,\alpha})_{0\leq i<D}$, that corresponds to the following approximation error vector:

$$\mathbf{Z}_\alpha = \left[\mathbf{T}[0]-\hat{\psi}_{0,\alpha}(Y), \mathbf{T}[1]-\hat{\psi}_{1,\alpha}(Y), \ldots, \mathbf{T}[D-1]-\hat{\psi}_{D-1,\alpha}(Y)\right]. \tag{5.4}$$

Once the profiling phase is performed, the Evaluator has approximated the true unknown leakage model ψ. Then, to retrieve the secret key manipulated by the targeted cryptographic device, he has to combine the knowledge it acquires from this modeling process. Based on $\mathbf{Z}_\alpha$, the Evaluator constructs a $D \times D$ covariance matrix $\Sigma_{\mathbf{Z}_\alpha}$ such that each of its element $\Sigma_{\mathbf{Z}_\alpha}[i,j] = Cov[\mathbf{Z}_\alpha[i], \mathbf{Z}_\alpha[j]]$. To approximate $\mathsf{Pr}[\mathbf{T}|Y]$, the Evaluator aims at exploiting the information acquired during the profiling phase in order to extract the secret key k^* manipulated by the targeted cryptographic module. Thus, given a new set $\mathcal{I}_a$ of N_a unlabeled leakage traces such that $\mathcal{I}_a = \{\mathbf{t}_0, \mathbf{t}_1, \ldots, \mathbf{t}_{N_a-1}\}$, the Evaluator computes a random vector $\mathbf{Z}'_{\alpha,k}$ for each key hypothesis $k \in \mathcal{K}$ following Equation 5.4. Once the N_a random vectors $(\mathbf{z}'_{\alpha,k,i})_{0\leq i<N_a}$ are generated, the Evaluator can compute the score related to k such that:

$$\mathsf{Pr}\left[\mathbf{z}'_{\alpha,k,i}|f(x_i,k)\right] = \frac{1}{\sqrt{(2\pi)^D|\Sigma_{\mathbf{Z}'_{\alpha,k}}|}}\exp\left\{-\frac{1}{2}\mathbf{z}'^T_{\alpha,k,i}\Sigma^{-1}_{\mathbf{Z}'_{\alpha,k}}\mathbf{z}'_{\alpha,k,i}\right\}, \tag{5.5}$$

with $|\Sigma_{\mathbf{Z}'_{\alpha,k}}|$ the determinant of $\Sigma_{\mathbf{Z}'_{\alpha,k}}$ that has been configured during the profiling phase.
Then, this conditional probability is used by the Evaluator to make a decision regarding the most likely hypothetical key $k \in \mathcal{K}$. The related decision rule is characterized by the distinguisher rule such that the log-likelihood distinguisher (see Definition 3.3.3.4) is performed:

$$\hat{k} = \arg\max_{k\in\mathcal{K}} \sum_{i=0}^{N_a-1} \log\left(\mathsf{Pr}\left[\mathbf{z}'_{\alpha,k,i}|f(x_i,k)\right]\right).$$

Thus, he approximates the unknown leakage model ψ which maximizes the unknown conditional probability $\mathsf{Pr}\left[Z|f(X,k^*)\right]$. Estimating such probabilities is typically considered in profiled side-channel attacks [CRR03, SLP05, HRG14].

LIMITATIONS OF THE STOCHASTIC ATTACKS

The limitations of the stochastic attacks approach are twofold: first, as the profiling phase aims at characterizing the interaction between the time samples, the covariance matrix Σ has to be constructed. However, the decision rule needs the computation of its inverse in order to guess which secret key is manipulated by the cryptographic module. Unfortunately, this process can lead to computational issues. One solution suggests to do not capture the interactions between the time samples and thus, reducing Σ to an identity matrix.
Secondly, if the profiling phase is performed on a D-dimensional leakage trace $\mathbf{T}$, the Evaluator has to conduct the related attack based on the computation of Equation 5.5 on the same dimension. However, once the profiling phase is performed, the Evaluator may want to reduce D to a lower dimension D', such that $D' \leq D$, through the visualization of the coefficients $(\alpha_u)_{u \in \mathbb{F}_2^n}$ for each time sample. Hence, due to the lack of flexibility of the attack phase, the Evaluator has to consider the full dimension D except if he conducts a new profiling phase on the reduced leakage traces of dimension D'.

Such problematic (*i.e.* the approximation of $\mathsf{Pr}[\mathbf{T}|Y]$) can be solved by a deep learning approach belonging to the family of *generative approaches*. Indeed, given a leakage trace, the related learning algorithm can automatically approximate the decision boundary that maximizes the decision rule. The next section proposes a taxonomy of generative models to differentiate each solution in order to select the most suitable one from SCA perspective.

5.1.3 Taxonomy of Generative Models

Probabilistic generative approach captures the interactions between all the variables considered by the resulted learning algorithm. To comply with this technical specification, the strategy consists in building a model that estimates the probability distribution of the leakage traces, namely $\mathsf{Pr}[\mathbf{T}]$. Hence, this experience Exp (see Definition 4.1.1.1) can be considered as the application of an unsupervised learning process. However, to fit with the side-channel context (*i.e.* supervised learning), the joint probability distribution, $\mathsf{Pr}[\mathbf{T}, Y]$, has to be estimated such that, afterwards, the Bayes' theorem can be computed in order to retrieve the conditional posterior probabilities $\mathsf{Pr}[Y|\mathbf{T}]$ and pick the most likely label Y. However, in the side-channel context, this problem can be reduced to the estimation of the following conditional probability distribution $\mathsf{Pr}[\mathbf{T}|Y]$ (see Equation 3.4). Hence, to deal with this observation, a generative model should be designed in order to estimate a Θ-parametric conditional distribution $\mathsf{Pr}[\mathbf{T}|Y, \Theta]$ that is as similar as possible to the true unknown conditional distribution $\mathsf{Pr}[\mathbf{T}|Y]$. Through this approach, the Evaluator estimates the unknown leakage function and gets a complete characterization of the targeted cryptographic module. This is beneficial for modeling new sets of leakage traces from the Θ-parametric model.

To conduct such estimation, Goodfellow [Goo17] exposes two solutions (see Figure 5.2):

- *Explicit density estimation* which explicitly computes an approximation of the true unknown conditional probability distribution $\mathsf{Pr}[\mathbf{T}|Y]$.
- *Implicit density estimation* generates a Θ-parametric model F_Θ which draws leakage traces from the underlying true unknown conditional probability distribution $\mathsf{Pr}[\mathbf{T}|Y]$ without explicitly defining it.

Explicit density estimation. This branch of generative models aims at designing a Θ-parametric model F_Θ which captures the complexity of the data structure. Through this characterization, the Evaluator hopes to find a solution which is close to the true unknown $\mathsf{Pr}[\mathbf{T}|Y]$. As defined in [Goo17], if the Θ-parametric model guarantees to find the tractable conditional

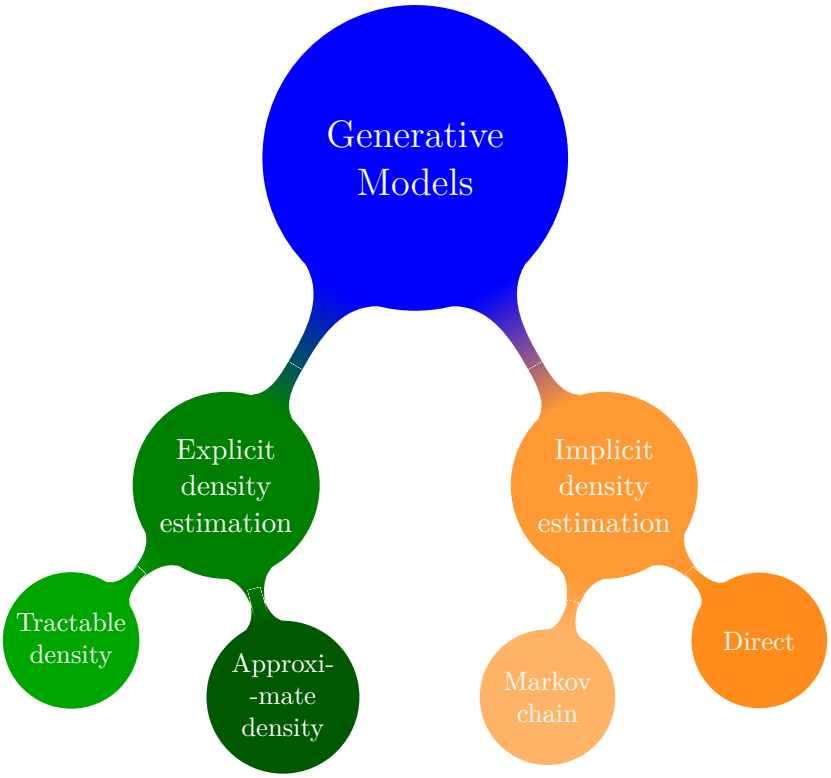


Figure 5.2: Taxonomy of Generative Models.

probability density $\mathsf{Pr}[\mathbf{T}|Y]$, the related F_Θ is therefore defined as a *tractable explicit model*. One of the most common solution to conduct explicit tractable density estimation is to consider the *autoregressive density estimation*. Given a D-dimensional leakage trace $\mathbf{T}$, the Evaluator approximates $\mathsf{Pr}[\mathbf{T}|Y]$ by using the chain rule of probability such that the probability of observing $\mathbf{T}[i]$ given Y is conditioned on $(\mathbf{T}[i-1], \mathbf{T}[i-2], \ldots, \mathbf{T}[0])$ such that:

$$\mathsf{Pr}[\mathbf{T}|Y] = \prod_{i=0}^{D-1} \mathsf{Pr}[\mathbf{T}[i] \mid \mathbf{T}[i-1], \mathbf{T}[i-2], \ldots, \mathbf{T}[0], Y].$$

Through this process, the idea is to use neural networks to model the dependencies between the time samples. This solution is highly effective because it permits to directly optimize a Θ-parametric function from the log-likelihood of a set of leakage traces. However, in the side-channel context, the sensitive variables only leak over a very few PoIs (see Figure 3.4) in comparison to the dimension of a targeted leakage trace. Thus, regarding the dimension of a leakage trace, the time samples are poorly correlated. In addition, the main drawback of this method is that it suffers from slow sequential sampling[b]. Thus, its application during the evaluation can be limited because of the *elapsed time* criterion. However, the readers interested in the autoregressive density estimation method may refer to [vdOKE+16, vdODZ+16, Goo17].
On the other hand, if $\mathsf{Pr}[\mathbf{T}|Y]$ is intractable, the related Θ-parametric model is denoted as an *explicit model requiring approximation*. This estimation suggests designing a surrogate loss function $\mathcal{L}$ such that the resulted empirical risk $\hat{\mathcal{R}}$ optimizes the related Θ-parametric model F_Θ to converge towards $\mathsf{Pr}[\mathbf{T}|Y]$.

$$\hat{\mathcal{R}}(\mathcal{L}, F_\Theta) \geq -\log(\mathsf{Pr}[\mathbf{T}|Y]).$$

[b]For example, a significant drawback of WaveNet [vdODZ+16], considering an *explicit tractable density estimation*, is its sequential (non-parallelizable) generation of samples. As reported in [Goo17], it requires 2 minutes of computation time to generate 1 second of audio.

In such scenario, it is possible to find a surrogate loss function that is computationally tractable while $\mathsf{Pr}[\mathbf{T}|Y]$ is not, and thus, approximating the related unknown conditional probability distribution. Introduced in [KW14], the *variational auto-encoder* is the main solution considered by the literature. In Section 5.2, this solution is investigated in the side-channel context in order to bridge deep learning and classical profiled side-channel attacks.

Implicit density estimation. Alternatively, some models can be optimized without explicitly producing density functions. Thus, the related computation can be eased in comparison to the explicit density estimation approaches. While the previous approach needs to retrieve an approximation of $\mathsf{Pr}[\mathbf{T}|Y]$, this scenario optimizes a model to draw new leakage traces based on a set of pairs $(\mathbf{T}, Y)$ that are independently and identically distributed according to the unknown joint distribution $\mathsf{Pr}[\mathbf{T}, Y]$. Thus, the related models indirectly interact with $\mathsf{Pr}[\mathbf{T}|Y]$. To optimize such algorithms, the Evaluator can follow the *Markov chain* process or conduct a *direct* approach with the application of *Generative Adversarial Networks* (GANs).
Introduced in [GPAM$^+$14], the GANs are characterized by two neural networks: a "*generator*" and a "*discriminator*". The goal of such models is to design a generator which aims at constructing *fake* leakage traces such that, the discriminator cannot differentiate it from *real* leakage traces. Hence, through this process, the Evaluator implicitly characterizes $\mathsf{Pr}[\mathbf{T}|Y]$. In [WCL$^+$20], Wang *et al.* illustrate the benefits of the conditional Generative Adversarial Networks [MO14] to improve the performance of profiled side-channel attacks in a restricted setting where profiling traces are limited. Indeed, the authors consider the conditional Generative Adversarial Networks as a preprocessing phase consisting in implicitly capturing the conditional probability distribution $\mathsf{Pr}[\mathbf{T}|Y]$ in order to increase the number of profiling leakage traces. Once the profiling set is sufficiently large, the Evaluator can consider it to perform a deep learning-based side-channel attack as introduced in Chapter 4. This data augmentation technique has then been extended by Mikhtar *et al.* [MBPK21] to target asymmetric cryptographic implementations. While this approach seems helpful for generating new sets of leakage traces, no relation is made to perform an end-to-end attack.

The following section proposes the first deep learning-based side-channel approach which derives from the stochastic attack previously introduced. It reduces the gap between the deep learning approach and the classical profiled side-channel attacks by bridging the stochastic attacks with the variational auto-encoder introduced in the deep learning literature.

5.2 Conditional Variational AutoEncoder based on Stochastic Attacks

5.2.1 Generative Latent Variable Models

As a remainder, the variational autoencoder [KW14] has been originally introduced as an unsupervised generative approach which explicitly approximates the density $\mathsf{Pr}[\mathbf{T}]$. To extend this neural network to supervised learning, Sohn *et al.* proposed the *conditional variational autoencoder* [SLY15] to approximate the true unknown conditional probability distribution $\mathsf{Pr}[\mathbf{T}]$. Ever since the seminal works have been widely applied in various fields (*e.g.* face generation [KW14, KWKT15], handwritten digits [KW14], objects [KWKT15]), we propose to contextualize conditional variational autoencoder into side-channel analysis in order to give a new perspective for generative models.

Starting from the definition of explicit density estimation approach defined in Subsection 5.1.3, conditional variational autoencoder aims at modeling the Θ-parametric conditional distribution $\mathsf{Pr}[\mathbf{T}|Y, \Theta]$ from two random variables $\mathbf{T} \in \mathbb{R}^D$ and $Y \in \mathbb{F}_2^n$.

Definition 5.2.1.1 (Latent Space)**.** Given a set of leakage traces $\mathbf{T} \in \mathbb{R}^D$, a latent space $\mathcal{V}$ of dimension $D^{'}$ (s.t. $D^{'} \leq D$) is a vector space spanned by the latent variables in which similar leakage traces are closer together in the related $D^{'}$-dimensional space.

Definition 5.2.1.2 (Latent Variable)**.** A latent variable $v \in \mathbb{R}^{D^{'}}$ is a non-observable variable that is sufficient to describe a leakage trace.

Suppose that a leakage trace $\mathbf{T} \in \mathbb{R}^D$ is acquired such that all the time samples are simultaneously generated. To obtain a coherent leakage trace, it is clear that the time samples cannot be sampled independently from each other as it corresponds to the computation of a cryptographic function (*e.g.* encryption) given a plaintext X and a secret key k^*. In particular, these variables suggest that the assignment of a label on a leakage trace $\mathbf{T}$ only depends on a small set of time samples. The conditional variational autoencoder is a *latent variable model* which suggests that the variability in the leakage traces given a label Y can be characterized by a small finite set of time samples. This is highly correlated with the definition of points of interest introduced in Definition 3.2.2.2. With this approach, the interactions between the time samples are captured through the latent space $\mathcal{V}$.

PROBLEM STATEMENT

A Θ-parametric latent variable model F_Θ, providing a Θ-parametric conditional distribution $\mathsf{Pr}\left[\mathbf{T}|Y,\Theta\right]$, is representative of the true unknown conditional distribution $\mathsf{Pr}\left[\mathbf{T}|Y\right]$, for every leakage trace $\mathbf{T}$ and every given sensitive variable Y, if there is a representation of compressed data $\mathbf{V}$, also known as latent space representation, such that the marginal distribution is given by:

$$\mathsf{Pr}\left[\mathbf{T}|Y,\Theta\right] = \int_{\mathbf{v}\in\mathcal{V}} \mathsf{Pr}\left[\mathbf{T}|Y,\mathbf{v},\Theta\right] \mathsf{Pr}\left[\mathbf{v}|Y\right] d\mathbf{v}, \tag{5.6}$$

where $\mathbf{v}$ is the realization of a random variable $\mathbf{V}$ in a $D^{'}$-dimensional space $\mathcal{V}$, with a probability $\mathsf{Pr}\left[\mathbf{V}=\mathbf{v}\right]$ defined over $\mathcal{V}$, and $\mathsf{Pr}\left[\mathbf{V}=\mathbf{v}|Y\right]$ denotes the probability of observing $\mathbf{v}$ over the latent space $\mathcal{V}$ knowing Y. While the latent space characterizes the noise distribution defined in Equation 3.1, we can easily assume that $\mathsf{Pr}\left[\mathbf{V}|Y\right] = \mathsf{Pr}\left[\mathbf{V}\right]$ because $\mathbf{V}$ is independent of the label Y (*deeper details will be provided in the following paragraphs*). In addition, we assume that $\mathsf{Pr}\left[\mathbf{V}\right]$ follows a multivariate Gaussian distribution of parameters $(\boldsymbol{\mu}, \Sigma)$. Through the conditional variational autoencoder, we wish to optimize Θ such that we can sample $\mathbf{V}$ from $\mathsf{Pr}\left[\mathbf{V}\right]$, and obtain, with high probability, a new generated leakage trace $\tilde{\mathbf{T}}$ very similar to the true known $\mathbf{T}$. However, Equation 5.6 is unfortunately intractable as it should be computed for every latent representations induced in the latent space $\mathcal{V}$. Thus, the following part of the section proposes solutions to circumvent this issue.

Hopefully, $\mathsf{Pr}\left[\mathbf{T}|Y,\Theta\right]$ may still be efficiently approximated thanks to the *Monte-Carlo* method [MU49]. Hence, for a large number of latent variables $\{\mathbf{v}_0,\ldots,\mathbf{v}_{N_v}\}$, and a label Y, the Evaluator can compute an estimation of $\mathsf{Pr}\left[\mathbf{T}|Y\right]$. Indeed, for a given label Y and a latent variable $\mathbf{V}$, we can build a neural network that computes $\mathsf{Pr}\left[\mathbf{T}|Y,\mathbf{V},\Theta\right]$. This model, denoted $F_\Theta^{(dec)} : \mathbb{R}^{D^{'}} \times \mathbb{F}_2^n \rightarrow \mathbb{R}^D$, is named *the decoder*. Hence, given a latent variable $\mathbf{V} \in \mathbb{R}^{D^{'}}$ and a sensitive variable $Y \in \mathbb{F}_2^n$, the decoder generates a new leakage trace $\tilde{\mathbf{T}}$ as close as possible to the real observed leakage trace $\mathbf{T}$. However, to perfectly construct a new set of D-dimensional leakage traces from $F_\Theta^{(dec)}$, the Evaluator has to estimate the multivariate Gaussian distribution characterizing the latent space $\mathcal{V}$ by approximating the following probability distribution $\mathsf{Pr}\left[\mathbf{V}|\mathbf{T},Y\right]$. This probability is defined as $\mathsf{Pr}\left[\mathbf{V}|\mathbf{T},Y\right] = \frac{\mathsf{Pr}[\mathbf{T}|\mathbf{V},Y]\cdot\mathsf{Pr}[\mathbf{V}]}{\mathsf{Pr}[\mathbf{T}|Y]}$ and it is also intractable due to Equation 5.6. Consequently, a solution is to find a parametric model that approximates the true unknown posterior $\mathsf{Pr}\left[\mathbf{V}|\mathbf{T},Y\right]$. In statistics, the variational inference techniques can approximate such complex distributions. Given a leakage trace $\mathbf{T} \in \mathbb{R}^D$ and a label $Y \in \mathbb{F}_2^n$, a Θ-parametric model can be constructed

to estimate the parameters $\boldsymbol{\mu}_{\mathbf{V}} \in \mathbb{R}^{D'}$ and $\Sigma_{\mathbf{V}} \in \mathcal{M}_{D',D'}(\mathbb{R})$ of the multivariate Gaussian distribution $\mathcal{N}_{D'}(\boldsymbol{\mu}_{\mathbf{V}}, \Sigma_{\mathbf{V}})$ such that the KL-divergence between the approximation and the targeted probability distribution $\mathsf{Pr}[\mathbf{V}|\mathbf{T}, Y]$ is minimized. Such estimation is made by a Θ-parametric model, denoted $F_{\Theta}^{(enc)} : \mathbb{R}^D \times \mathbb{F}_2^n \to \mathbb{R}^{D'}$, that is called *the encoder*.

Remark 5.2.1.1. As the conditional variational autoencoder is composed by an encoder and a decoder, a distinction has to be made related to the trainable parameters considered in both models. Hence, in the rest of the manuscript, we denote as $F_{\Theta,\phi}$ the resulted generative model such that, for a given leakage trace $\mathbf{T}$, $F_{\Theta,\phi}(\mathbf{T}, Y) = F_{\phi}^{(dec)} \circ F_{\Theta}^{(enc)} \circ (\mathbf{T}, Y)$.

Remark 5.2.1.2. Typically, an encoder is used to process a dimensionality reduction by reducing the number of features (*i.e.* time samples) that describes the input while preserving the main relevant information in the latent space $\mathcal{V}$. Thus, in most cases, the dimensionality of the latent space is chosen to be lower than the dimensionality of the input space from which the leakage traces are drawn. As the aim of this chapter is to bridge deep learning and classical profiled side-channel attack, the following part assumes that $D' = D$.

5.2.2 Latent Space Estimation and Instances' Generation

Through the description of the stochastic attack (see Section 5.1), the Evaluator can construct a conditional variational autoencoder adapted for the side-channel context.

Encoder. As mentioned, it aims at modeling a neural network $F_{\Theta}^{(enc)} : \mathbb{R}^D \times \mathbb{F}_2^n \to \mathbb{R}^D$ that takes a D-dimensional leakage trace and returns an approximation of $\boldsymbol{\mu}_{\mathbf{V}} \in \mathbb{R}^D$ and $\Sigma_{\mathbf{V}} \in \mathcal{M}_{D,D}(\mathbb{R})$ such that it can be used to characterize an element in the latent space $\mathcal{V}$ of dimension D. This element describes the behavior of the targeted cryptographic module (see on the left part of Figure 5.3). In this regard, we design the encoder $F_{\Theta}^{(enc)}$ such that it characterizes the leakage model $\psi(Y)$ and the random part $\mathbf{Z}$ of a leakage trace $\mathbf{T}$ in order to fit with the stochastic attack process. To build a suited encoder, the related neural network should follow the structure defined in Section 5.1 in order to extract the maximum amount of relevant information from leakage traces. First, the Evaluator has to estimate the deterministic part of a leakage trace $\mathbf{T}$ (*i.e.* leakage model ψ) that is defined by Equation 5.3. This modeling can be estimated by a fully-connected layer of D neurons such that each of them is linked with all elements of the orthonormal monomial basis[c] $\Phi_u^{(orth.)}$.
Let $Y \in \mathbb{F}_2^n$ and let $(\Phi_u^{(orth.)}(Y))_{u \in \mathbb{F}_2^n}$ (resp. $\hat{\psi}_{i,\Theta}(Y)$) be the input (resp. output) of the i^{th} neuron such that:

$$\hat{\psi}_{i,\Theta}(Y) = \varrho\left(\frac{1}{2^{n/2}} \sum_{u=(u[0],\ldots,u[n-1]) \in \mathbb{F}_2^n} \Theta_u[i] \cdot \Phi_u^{(orth.)}(Y)\right),$$

where $\varrho(.)$ is a function (linear or non-linear) and $\Theta \in \mathcal{M}_{1+\sum_{i=0}^{d-1}\binom{n}{i},D}(\mathbb{R})$ denotes the set of trainable parameters for a given degree d of $\Phi_u^{(orth.)}$. While our work is to reduce the gap between deep learning and classical profiled side-channel attacks, we decide to define $\varrho(.)$ as the identity function in order to satisfy[d] $\hat{\psi}_{\Theta}[i] = \hat{\psi}_{\alpha}[i]$ and consider that the deterministic part of leakage trace at time sample i can be approximated by a single neuron. In the rest of the manuscript, this layer will be denoted as $\hat{\psi}_{\Theta}$ (see Figure 5.3).
Once the noise-free part $\hat{\psi}_{\Theta}$ is estimated, the next step is to deeply characterize the noise part $\mathbf{Z}$ using leakage traces and the neurons of $\hat{\psi}_{\Theta}$ layer. In the conditional variational autoencoder-based stochastic attack, we choose to deliberately force the subtraction of the leakage traces at time sample i and the i^{th} neuron of $\hat{\psi}_{\Theta}$ layer in order to fit with Equation 5.4. Then, the

[c] The Evaluator can consider the monomial basis (see Definition 5.1.1.1) interchangeably. However, using this basis leads to a loss of interpretability as defined in Subsection 5.1.2.

[d] As a remainder, $\hat{\psi}_{\alpha}$ is defined Equation 5.3.

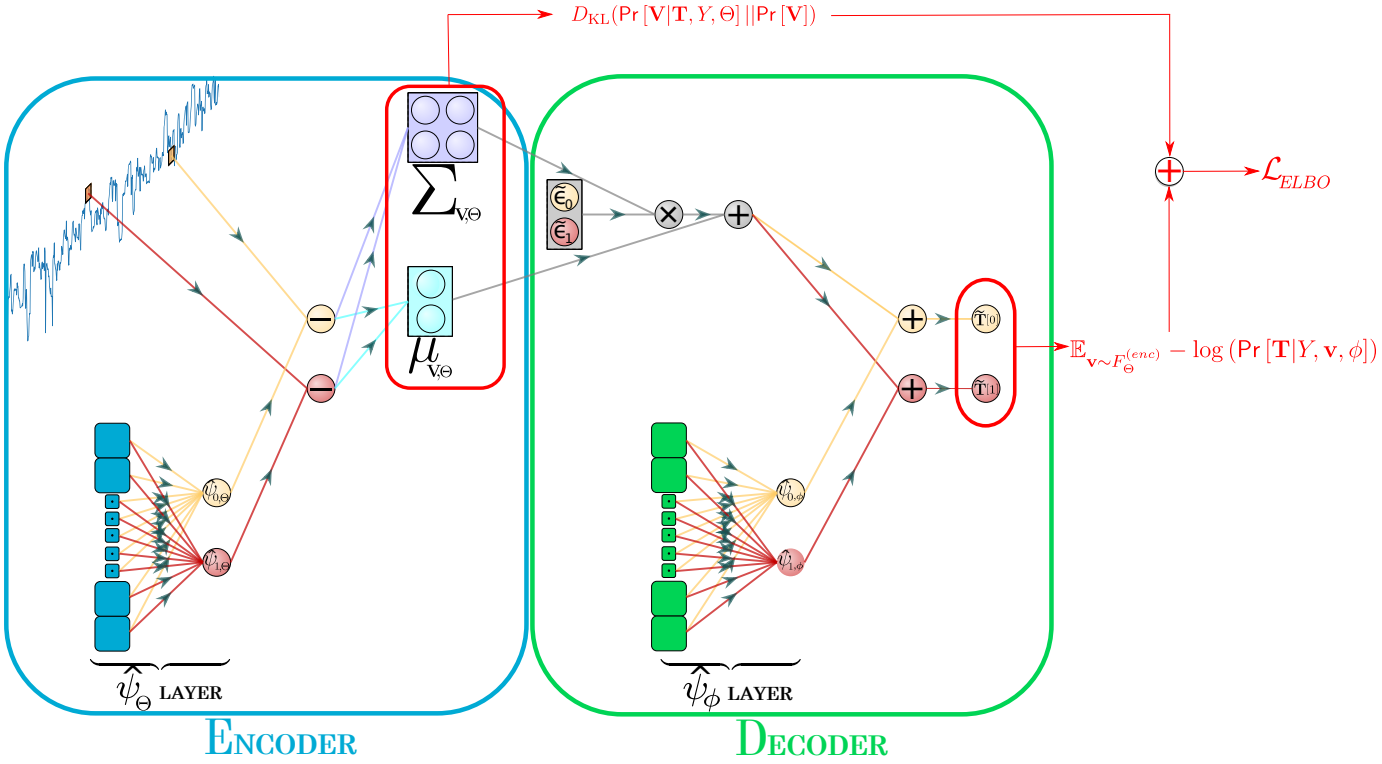


Figure 5.3: cVAE-ST structure, considering $D = D' = 2$, where $D_{\mathrm{KL}}(\mathsf{Pr}[\mathbf{V}|\mathbf{T},Y,\Theta]\,||\mathsf{Pr}[\mathbf{V}])$ (resp. $\mathbb{E}_{\mathbf{v}\sim F_\Theta^{(enc)}} - \log(\mathsf{Pr}[\mathbf{T}|Y,\mathbf{v},\phi])$) denotes the loss introduced in Subsection 5.2.3 namely KL-divergence loss (resp. the reconstruction loss).

encoder $F_\Theta^{(enc)}$ is trained to return a Θ-parametric mean vector $\boldsymbol{\mu}_{\mathbf{V},\Theta} \in \mathbb{R}^D$, and a Θ-parametric covariance matrix $\Sigma_{\mathbf{V},\Theta} \in \mathcal{M}_{D,D}(\mathbb{R})$ that describes the multivariate Gaussian noise for a given trace $\mathbf{T}$. Those approximations respectively estimate $\boldsymbol{\mu}_{\mathbf{V}}$ and $\Sigma_{\mathbf{V}}$. Thus, from these parameters, the Evaluator can compute $\mathsf{Pr}[\mathbf{V}|\mathbf{T},Y,\Theta]$. That is, $F_\Theta^{(enc)}$ extracts all the information induced in a leakage trace such that the latent variables $\mathbf{V} \in \mathcal{V}$ are characterized by its noise part $\mathbf{Z}$.

Decoder. Once the encoder is constructed, the Evaluator can capture the parameters $\boldsymbol{\mu}_{\mathbf{V},\Theta}$ and $\Sigma_{\mathbf{V},\Theta}$ which are needed to design the related Gaussian noise distribution $\mathcal{N}_D(\boldsymbol{\mu}_{\mathbf{V},\Theta}, \Sigma_{\mathbf{V},\Theta})$. From a new sample $\mathbf{V} \sim \mathcal{N}_D(\boldsymbol{\mu}_{\mathbf{V},\Theta}, \Sigma_{\mathbf{V},\Theta})$, the Evaluator designs a decoder $F_\phi^{(dec)} : \mathbb{R}^D \times \mathbb{F}_2^n \to \mathbb{R}^D$ (see on the right part of Figure 5.3) such that given $\mathcal{G}_{d+1}$, containing all the Boolean functions of degree lower or equal to d, he wants to maximize the conditional probability distribution $\mathsf{Pr}[\mathbf{T}|Y,\mathbf{V},\phi]$, *i.e.* building a new leakage trace $\tilde{\mathbf{T}} \in \mathbb{R}^D$ as similar as possible to the related real trace $\mathbf{T}$ and defined as follows:

$$\tilde{\mathbf{T}}[i] = \underbrace{\frac{1}{2^{n/2}} \sum_{u=(u[0],\ldots,u[n-1])\in\mathbb{F}_2^n} \phi_u[i] \cdot \Phi_u^{(orth.)}(Y)}_{\hat{\psi}_{i,\phi}} + \mathbf{V}[i]. \tag{5.7}$$

Note that a latent variable $\mathbf{V} \in \mathbb{R}^D$ is initially sampled from the prior distribution $\mathsf{Pr}[\mathbf{V}]$ such that the dimension of $\mathbf{V}$ should correspond with the dimension of the latent space estimated by the encoder. However, performing the training process in such configuration can be arduous. Indeed, the backpropagation process described in Subsection 4.3.1 cannot be performed because the Evaluator has to compute the gradient of the loss function with respect to samples (*i.e.* latent variable $\mathbf{V} \in \mathcal{V}$), which is inherently non-differentiable.

REPARAMETRIZATION TRICK

To circumvent this issue, the reparametrization trick [KW14] proposes to rewrite $\mathbf{V}$ such that the derivative can be computed with respect to the parametric distributions (*i.e.* $\boldsymbol{\mu}_{\mathbf{V},\Theta}$ and $\Sigma_{\mathbf{V},\Theta}$) that are differentiable. Instead of generating samples from $\mathcal{N}_D(\boldsymbol{\mu}_{\mathbf{V},\Theta}, \Sigma_{\mathbf{V},\Theta})$, sampling is performed from $\epsilon \sim \mathcal{N}_D(\mathbf{0}, \mathbf{I}_D)$, followed by the computation of $\mathbf{V} = \boldsymbol{\mu}_{\mathbf{V},\Theta} + \Sigma^{\frac{1}{2}}_{\mathbf{V},\Theta} \times \epsilon$.

Once $\mathbf{V}$ is constructed, the Evaluator has to approximate the deterministic part of the leakage model, namely $\hat{\psi}_\phi$. As already mentioned for the encoder, its estimation can be made with a fully-connected layer such that the input of size D is characterized by $(\Phi_u^{(orth.)}(Y))_{u\in\mathbb{F}_2^n}$ for a given $Y \in \mathbb{F}_2^n$. Because the Evaluator classically wants to characterize all the input time samples, the number of nodes in the $\hat{\psi}_\phi$ layer depends on the dimensionality of the latent space, *i.e.* dimension of $\mathcal{V}$ (see Figure 5.3). Based on $\hat{\psi}_\phi$ and $\mathbf{V}$, the Evaluator can then build a new trace $\tilde{\mathbf{T}}$ following Equation 5.7.

Then, to adequately find the trainable parameters Θ and ϕ, the Evaluator has to consider some learning metrics that aims at approximating $\mathsf{Pr}[\mathbf{T}|Y]$.

5.2.3 Learning Similarities

As defined in Subsection 5.2.1 and Subsection 5.2.2, our generative model has to optimize a set of parameters ϕ and Θ in order to maximize the marginal log likelihood $\log(\mathsf{Pr}\left[\mathbf{T}|Y,\phi\right])$.

Theorem 5.2.3.1. *For any choice of encoder $F_\Theta^{(enc)}$ and trainable parameters Θ, the conditional marginal log likelihood $\log(\mathsf{Pr}\left[\boldsymbol{T}|Y,\phi\right])$ satisfies:*

$$\log\left(\mathsf{Pr}\left[\boldsymbol{T}|Y,\phi\right]\right) = \mathbb{E}_{\mathbf{v}\sim F_\Theta^{(enc)}}\left[\log\left(\mathsf{Pr}\left[\boldsymbol{T},\mathbf{v}|Y,\phi\right]\right) - \log\left(\mathsf{Pr}\left[\mathbf{v}|\boldsymbol{T},Y,\Theta\right]\right)\right] + \mathcal{D}_{KL}\left(\mathsf{Pr}\left[\boldsymbol{V}|\boldsymbol{T},Y,\Theta\right]||\mathsf{Pr}\left[\boldsymbol{V}|\boldsymbol{T},Y,\phi\right]\right). \quad (5.8)$$

Proof. From Equation 5.6 and the work provided by Kingma and Welling [KW19, Section 2.2], we can extend their result to the conditional marginal log likelihood $\log\ \mathsf{Pr}[\mathbf{T}|Y,\phi]$ as follows:

$$\begin{aligned}
\log\left(\mathsf{Pr}\left[\mathbf{T}|Y,\phi\right]\right) &= \mathbb{E}_{\mathbf{v}\sim F_\Theta^{(enc)}}\left[\log\left(\mathsf{Pr}\left[\mathbf{T}|Y,\mathbf{v},\phi\right]\right)\right] \\
&= \mathbb{E}_{\mathbf{v}\sim F_\Theta^{(enc)}}\left[\log\left(\frac{\mathsf{Pr}\left[\mathbf{T},\mathbf{v}|Y,\phi\right]}{\mathsf{Pr}\left[\mathbf{v}|\mathbf{T},Y,\phi\right]}\right)\right] \\
&= \mathbb{E}_{\mathbf{v}\sim F_\Theta^{(enc)}}\left[\log\left(\frac{\mathsf{Pr}\left[\mathbf{T},\mathbf{v}|Y,\phi\right]}{\mathsf{Pr}\left[\mathbf{v}|\mathbf{T},Y,\Theta\right]}\cdot\frac{\mathsf{Pr}\left[\mathbf{v}|\mathbf{T},Y,\Theta\right]}{\mathsf{Pr}\left[\mathbf{v}|\mathbf{T},Y,\phi\right]}\right)\right] \\
&= \mathbb{E}_{\mathbf{v}\sim F_\Theta^{(enc)}}\left[\log\left(\frac{\mathsf{Pr}\left[\mathbf{T},\mathbf{v}|Y,\phi\right]}{\mathsf{Pr}\left[\mathbf{v}|\mathbf{T},Y,\Theta\right]}\right)\right] + \mathbb{E}_{\mathbf{v}\sim F_\Theta^{(enc)}}\left[\log\left(\frac{\mathsf{Pr}\left[\mathbf{v}|\mathbf{T},Y,\Theta\right]}{\mathsf{Pr}\left[\mathbf{v}|\mathbf{T},Y,\phi\right]}\right)\right] \\
&= \mathbb{E}_{\mathbf{v}\sim F_\Theta^{(enc)}}\left[\log\left(\mathsf{Pr}\left[\mathbf{T},\mathbf{v}|Y,\phi\right]\right) - \log\left(\mathsf{Pr}\left[\mathbf{v}|\mathbf{T},Y,\Theta\right]\right)\right] \\
&\quad + \mathcal{D}_{\mathrm{KL}}\left(\mathsf{Pr}\left[\mathbf{V}|\mathbf{T},Y,\Theta\right]||\mathsf{Pr}\left[\mathbf{V}|\mathbf{T},Y,\phi\right]\right).
\end{aligned}$$

□

Unfortunately, due to the intractability of $\mathsf{Pr}[\mathbf{V}|\mathbf{T},Y,\phi]$ (see Subsection 5.2.1), Equation 5.8 cannot be solved in practice. Hence, we have to define a function such that $\log(\mathsf{Pr}[\mathbf{T}|Y,\phi])$ can be approximated through an optimization algorithm. In [KW14], Kingma and Welling propose a variational lower bound on the marginal likelihood which was generalized by Sohn *et al.* on conditional marginal likelihood [SLY15].

Theorem 5.2.3.2. *[SLY15] For any choice of encoder $F_\Theta^{(enc)}$ and trainable parameters Θ, the variational lower bound of the conditional log-likelihood $\log(\mathsf{Pr}[\boldsymbol{T}|Y,\phi])$ is defined as:*

$$\log\left(\mathsf{Pr}\left[\boldsymbol{T}|Y,\phi\right]\right) \geq -\mathcal{D}_{KL}\left(\mathsf{Pr}\left[\boldsymbol{V}|\boldsymbol{T},Y,\Theta\right]||\mathsf{Pr}\left[\boldsymbol{V}\right]\right) + \mathbb{E}_{\mathbf{v}\sim F_\Theta^{(enc)}}\left[\log\left(\mathsf{Pr}\left[\boldsymbol{T}|Y,\mathbf{v},\phi\right]\right)\right]. \quad (5.9)$$

Proof.

$$\begin{aligned}
\log\left(\mathsf{Pr}\left[\mathbf{T}|Y,\phi\right]\right) &= \mathbb{E}_{\mathbf{v}\sim F_{\Theta}^{(enc)}}\left[\log\left(\mathsf{Pr}\left[\mathbf{T},\mathbf{v}|Y,\phi\right]\right) - \log\left(\mathsf{Pr}\left[\mathbf{v}|\mathbf{T},Y,\Theta\right]\right)\right] \\
&\quad + \mathcal{D}_{\mathrm{KL}}\left(\mathsf{Pr}\left[\mathbf{V}|\mathbf{T},Y,\Theta\right]||\mathsf{Pr}\left[\mathbf{V}|\mathbf{T},Y,\phi\right]\right) \\
&\geq \mathbb{E}_{\mathbf{v}\sim F_{\Theta}^{(enc)}}\left[\log\left(\mathsf{Pr}\left[\mathbf{T},\mathbf{v}|Y,\phi\right]\right) - \log\left(\mathsf{Pr}\left[\mathbf{v}|\mathbf{T},Y,\Theta\right]\right)\right] \\
&= \mathbb{E}_{\mathbf{v}\sim F_{\Theta}^{(enc)}}\left[\log\left(\mathsf{Pr}\left[\mathbf{v}|Y,\Theta\right]\right) - \log\left(\mathsf{Pr}\left[\mathbf{v}|\mathbf{T},Y,\Theta\right]\right)\right] + \mathbb{E}_{\mathbf{v}\sim F_{\Theta}^{(enc)}}\left[\log\left(\mathsf{Pr}\left[\mathbf{T}|Y,\mathbf{v},\phi\right]\right)\right] \\
&= -\mathcal{D}_{\mathrm{KL}}\left(\mathsf{Pr}\left[\mathbf{V}|\mathbf{T},Y,\Theta\right]||\mathsf{Pr}\left[\mathbf{V}|Y,\Theta\right]\right) + \mathbb{E}_{\mathbf{v}\sim F_{\Theta}^{(enc)}}\left[\log\left(\mathsf{Pr}\left[\mathbf{T}|Y,\mathbf{v},\phi\right]\right)\right].
\end{aligned}$$

As mentioned in Subsection 5.2.1, the prior distribution $\mathsf{Pr}\left[\mathbf{V}|Y,\Theta\right]$ can be reduced to $\mathsf{Pr}\left[\mathbf{V}\right]$ because $\mathbf{V}$ is independent from the label Y and Θ. □

The equality between Equation 5.8 and Equation 5.9 holds if and only if the encoder $F_{\Theta}^{(enc)}$, which approximates the parameters $\boldsymbol{\mu}_{\mathbf{V},\Theta}$ and $\Sigma_{\mathbf{V},\Theta}$ that are needed to compute $\mathsf{Pr}[\mathbf{V}|\mathbf{T},Y,\Theta]$, is able to perfectly predict $\mathsf{Pr}[\mathbf{V}|\mathbf{T},Y,\phi]$ (*i.e.* $\mathcal{D}_{\mathrm{KL}}\left(\mathsf{Pr}[\mathbf{V}|\mathbf{T},Y,\Theta]||\mathsf{Pr}[\mathbf{V}|\mathbf{T},Y,\phi]\right) = 0$). In such configuration, the Evaluator exactly captures the random part induced in a leakage trace $\mathbf{T}$. Based on Equation 5.9, we define the empirical risk that we minimize to train the conditional variational autoencoder based on stochastic attacks.

Definition 5.2.3.1 (Empirical risk combined with Evidence Lower BOund (ELBO) Loss)**.** Given a latent space $\mathcal{V}$, a set of N_p labeled leakage traces $\mathcal{I}_p = \{(\mathbf{t}_0, y_0), \ldots, (\mathbf{t}_{N_p-1}, y_{N_p-1})\}$, we define the empirical risk optimizing $F_{\Theta,\phi}$, that approximates the generative distribution $\mathsf{Pr}\left[\mathbf{T}|Y\right]$, as follows:

$$\hat{\mathcal{R}}\left(\mathcal{L}_{ELBO}, F_{\Theta,\phi}\right) = \frac{1}{N_p}\sum_{i=0}^{N_p-1} \underbrace{\mathcal{D}_{\mathrm{KL}}\left(\mathsf{Pr}\left[\mathbf{V}|\mathbf{t}_i,y_i,\Theta\right]||\mathsf{Pr}\left[\mathbf{V}\right]\right)}_{\text{KL-Divergence Loss}} - \underbrace{\mathbb{E}_{\mathbf{v}\sim F_{\Theta}^{(enc)}}\log\left(\mathsf{Pr}[\mathbf{t}_i|y_i,\mathbf{v},\phi]\right)}_{\text{Reconstruction Loss}}, \quad (5.10)$$

such that $\left(\mathsf{Pr}\left[\mathbf{V}|\mathbf{t}_i,y_i,\Theta\right]\right)_{0\leq i<N_p}$ is computed from $\boldsymbol{\mu}_{\mathbf{V},\Theta}$ and $\Sigma_{\mathbf{V},\Theta}$ provided by the encoder $(F_{\Theta}^{(enc)}(\mathbf{t}_i))_{0\leq i<N_p}$ and $(\mathsf{Pr}[\mathbf{t}_i|y_i,\mathbf{v},\phi])_{0\leq i<N_p}$ is obtained from $F_{\phi}^{(dec)}(\mathbf{v})$.

Sampling $\mathbf{v}$ from the learned posterior $\mathsf{Pr}[\mathbf{V}|\mathbf{T},Y,\Theta]$ knowing the leakage trace $\mathbf{T}$, the related label Y and the multivariate Gaussian distribution $\mathcal{N}_D(\boldsymbol{\mu}_{\mathbf{V},\Theta},\Sigma_{\mathbf{V},\Theta})$, can be seen as encoding $\mathbf{T}$ into $\mathbf{v}$, while $F_{\phi}^{(dec)}$ seeks to reconstruct $\mathbf{T}$ from $\mathbf{v}$. Classically used for training a variational autoencoder, the loss function defined in Equation 5.10 can be decomposed into two terms: the reconstruction and the KL-divergence terms. To get a better reconstruction loss, the embedding means $\mu_{\mathbf{V},\Theta}$ are pushed far away from each other and embedding standard deviations $\Sigma_{\mathbf{V},\Theta}$ are pulled toward zero. To get smaller $\mathcal{D}_{\mathrm{KL}}\left(\mathsf{Pr}\left[\mathbf{V}|\mathbf{T},Y,\Theta\right]||\mathsf{Pr}\left[\mathbf{V}\right]\right)$, the embedding means are pulled toward zero and the embedding standard deviations are pulled toward one. While the KL-divergence term is opposed to the reconstruction loss, it can be seen as a regularization term. Indeed, putting a lot of information about $\mathbf{T}$ in $\mathbf{V}$ makes reconstruction trivial, but the penalization induced by the regularization term is non-negligible. Therefore, the regularization term acts as an information bottleneck, so a balance between both terms must be found. If necessary, the KL-divergence loss can be monitored by a hyperparameter β. In the state-of-the-art, these models are called β-*Variational AutoEncoders* [HMP+17]. However, the impact of the β-parameter on the resulted learning algorithm is considered as out of the scope of this manuscript.

Reconstruction Loss. This term, denoted by $\mathbb{E}_{\mathbf{v}\sim F_{\Theta}^{(enc)}}\log\left(\mathsf{Pr}[\mathbf{T}|Y,\mathbf{v},\phi]\right)$, is linked with the decoder $F_{\phi}^{(dec)}$ introduced in Subsection 5.2.2. It defines the probability of constructing $\mathbf{T}$ given the label Y and a sample $\mathbf{v}$ of the latent space $\mathcal{V}$ of size D. Hence, the reconstruction loss tends to maximize the log likelihood in order to construct leakage traces that are correlated with the true unknown leakage model ψ and the noise $\mathbf{Z}$ related to $\mathbf{T}$. Thus, it encourages the decoder to learn how a leakage trace can be reconstructed from a given noise representation defined by a latent variable $\mathbf{V}$. The reconstruction loss optimizes the parameters ϕ to retrieve the correct

coefficients associated with each vector of the orthonormal monomial basis $\Phi_u^{(orth.)}$ such that the probability distribution in latent space $\mathcal{V}$ is defined by a multivariate Gaussian distribution $\mathcal{N}_D(\boldsymbol{\mu}_{\mathbf{V},\Theta}, \Sigma_{\mathbf{V},\Theta})$. Typically, if we only consider the case where no interaction between the time samples of $\mathbf{T}$ occurs, then, the covariance matrix $\Sigma_{\mathbf{V},\Theta}$ can be simplified to a diagonal matrix such that its vector representation can be described as $\sigma^2_{\mathbf{V},\Theta} = [\Sigma_{\mathbf{V},\Theta}[0,0], \Sigma_{\mathbf{V},\Theta}[1,1], \ldots, \Sigma_{\mathbf{V},\Theta}[D,D]]$. In such configuration, we do not expect to capture the time samples' interaction related to the constructed leakage trace $\tilde{\mathbf{T}} \in \mathbb{R}^D$. Thus, the reconstruction loss can be computed as follows:

$$\mathbb{E}_{\mathbf{v}\sim F_\Theta^{(enc)}} - \log\left(\mathsf{Pr}\left[\mathbf{T}|Y,\mathbf{v},\phi\right]\right) = \mathbb{E}_{\mathbf{v}\sim F_\Theta^{(enc)}}\left[\sum_{i=0}^{D-1}\frac{1}{2}\log\left(2\pi\sigma^2_{\tilde{\mathbf{T}}}[i]\right) + \frac{(\mathbf{T}[i]-\boldsymbol{\mu}_{\tilde{\mathbf{T}}}[i])^2}{2\sigma^2_{\tilde{\mathbf{T}}}[i]}\right], \tag{5.11}$$

with $\mu_{\tilde{\mathbf{T}}}[i]$ (resp. $\sigma^2_{\tilde{\mathbf{T}}}[i]$) indicates the i^{th} element of the mean (resp. variance) vector of generated leakage traces $\tilde{\mathbf{T}}$ given a latent representation $\mathbf{v}$ and a deterministic part $\hat{\psi}_\phi$ which depends on $Y = f(X, k^*)$ (see Equation 5.7). Thus, assuming that $\Sigma_{\mathbf{V},\Theta}$ can be simplified to a diagonal matrix affects the ability of the generated leakage trace $\tilde{\mathbf{T}}$ to capture the interaction between the time samples of $\mathbf{T}$. While this choice can be problematic from a performance perspective[e], the computation gain is non-negligible as the matrix inversion does not have to be computed in order to process the reconstruction loss.
Then, we assume that the output distribution of the conditional variational autoencoder is an isotropic Gaussian (*i.e.* for all $\mathbf{v} \sim \mathcal{N}_D(\boldsymbol{\mu}_{\mathbf{V},\Theta}, \mathrm{diag}(\Sigma_{\mathbf{V},\Theta}))$, we can define $\Sigma_{\tilde{\mathbf{T}}} = \sigma^2 \cdot \mathbf{I}_D$ where σ^2 is a scalar). While the *Mean Squared Error* (MSE) loss function[f] can be written as $\mathbb{E}_{\mathbf{v}\sim F_\Theta^{(enc)}}||\mathbf{T}-\boldsymbol{\mu}_{\tilde{\mathbf{T}}}||_2$, Equation 5.11 can be simplified as follows:

$$\mathbb{E}_{\mathbf{v}\sim F_\Theta^{(enc)}} - \log\left(\mathsf{Pr}\left[\mathbf{T}|Y,\mathbf{v},\phi\right]\right) = \frac{1}{2}\log\left(2\pi\sigma^2\right) + \frac{\mathbb{E}_{\mathbf{v}\sim F_\Theta^{(enc)}}||\mathbf{T}-\boldsymbol{\mu}_{\tilde{\mathbf{T}}}||_2}{2\sigma^2}. \tag{5.12}$$

Note that this solution is minimized if the scalar $\sigma^2 = \mathbb{E}_{\mathbf{v}\sim F_\Theta^{(enc)}}||\mathbf{T}-\boldsymbol{\mu}_{\tilde{\mathbf{T}}}||_2 = \text{MSE}$ [Yu20].
This loss is approximated via *Monte-Carlo* sampling, however, due to computation constraints, we consider only one sample $\mathbf{v}$ for computing Equation 5.12 during the training process. Consequently, for an estimated leakage trace $\tilde{\mathbf{T}}$, we minimize its L^2-norm from the related true leakage trace $\mathbf{T}$ in order to find the best parameters ϕ. This result is in line with Theorem 5.1.1.1 from which the stochastic attack is designed. In other words, through this solution, we attempt to find an estimated leakage trace $\tilde{\mathbf{T}}$ as similar as the real one $\mathbf{T}$. Thus, the decoder $F_\phi^{(dec)}$ is only affected by the reconstruction loss and seeks to suitably reconstruct $\tilde{\mathbf{T}}$ based on a latent representation $\mathbf{V}$ and a deterministic part $\hat{\psi}_\phi$.

KL-divergence loss. However, to reduce the overfitting issue, a *regularization* term is added (see Subsection 4.1.4). In addition to the optimization of ϕ, the conditional variational autoencoder concurrently optimizes Θ to minimize the KL-divergence of the approximation $\mathsf{Pr}\left[\mathbf{V}|\mathbf{T},Y,\Theta\right]$ from $\mathsf{Pr}\left[\mathbf{V}\right]$. In addition, the better $\mathsf{Pr}\left[\mathbf{V}|\mathbf{T},Y,\Theta\right]$ approximates the true posterior distribution $\mathsf{Pr}\left[\mathbf{V}|\mathbf{T},Y,\phi\right]$, in terms of the KL divergence, the smaller the gap between Equation 5.10 and the marginal log-likelihood $\log(\mathsf{Pr}\left[\mathbf{T}|Y,\phi\right])$. As remainder, both $\mathsf{Pr}\left[\mathbf{V}|\mathbf{T},Y,\Theta\right]$ and $\mathsf{Pr}\left[\mathbf{V}\right]$ are assumed to be Gaussian, specifically, $\mathsf{Pr}\left[\mathbf{V}|\mathbf{T},Y,\Theta\right]$ follows $\mathcal{N}_D(\boldsymbol{\mu}_{\mathbf{V},\Theta}, \Sigma_{\mathbf{V},\Theta})$ and $\mathsf{Pr}\left[\mathbf{V}\right]$ follows $\mathcal{N}_D(0, \mathbf{I}_D)$. The latter distribution is assumed as the leakage traces are standardized, *i.e.* zero mean and unit variance, and such that no interactions are captured between the time samples. Indeed, as $\mathsf{Pr}[\mathbf{V}]$ characterizes the random part of $\tilde{\mathbf{T}}$ (see Equation 5.7), it has to follow the same distribution as the random part of the real trace $\mathbf{T}$ which is $\mathcal{N}(0,1)$ for each non-informative time sample.

[e] To nuance this issue, Bruneau *et al.* [BGH+15, Figure 3] illustrate that the information induced by the covariance matrix is mainly brought by its diagonal.

[f] This loss function has already been considered in the side-channel context [Tim19, vdVP19, MWM21].

Through this configuration, the KL-divergence can be computed as follows:

$$\begin{aligned}
D_{\mathrm{KL}}(\mathsf{Pr}\left[\mathbf{V}|\mathbf{T},Y,\Theta\right]||\mathsf{Pr}\left[\mathbf{V}\right]) &= \mathbb{E}_{\mathbf{v}\sim F_{\Theta}^{(enc)}}\left[\log(\mathsf{Pr}\left[\mathbf{v}|\mathbf{T},Y,\Theta\right]) - \log(\mathsf{Pr}\left[\mathbf{v}\right])\right] \\
&= \mathbb{E}_{\mathbf{v}\sim F_{\Theta}^{(enc)}}\left[\frac{1}{2}\log\left(\frac{|\mathbf{I}_D|}{|\Sigma_{\mathbf{V},\Theta}|}\right) - \frac{1}{2}(\mathbf{v}-\boldsymbol{\mu}_{\mathbf{V},\Theta})^T\Sigma_{\mathbf{V},\Theta}^{-1}(\mathbf{v}-\boldsymbol{\mu}_{\mathbf{V},\Theta})\right. \\
&\quad \left. +\frac{1}{2}(\mathbf{v}-0)^T\mathbf{I}_D^{-1}(\mathbf{v}-0)\right] \\
&= -\frac{1}{2}\log\left(|\Sigma_{\mathbf{V},\Theta}|\right) - \frac{1}{2}\mathbb{E}_{\mathbf{v}\sim F_{\Theta}^{(enc)}}\left[(\mathbf{v}-\boldsymbol{\mu}_{\mathbf{V},\Theta})^T\Sigma_{\mathbf{V},\Theta}^{-1}(\mathbf{v}-\boldsymbol{\mu}_{\mathbf{V},\Theta})\right] \\
&\quad + \frac{1}{2}\mathbb{E}_{\mathbf{v}\sim F_{\Theta}^{(enc)}}\left[\mathbf{v}^T\mathbf{I}_D^{-1}\mathbf{v}\right] \\
&= -\frac{1}{2}\log\left(|\Sigma_{\mathbf{V},\Theta}|\right) - \frac{1}{2}\mathrm{tr}(\Sigma_{\mathbf{V},\Theta}^{-1}\Sigma_{\mathbf{V},\Theta}) + \frac{1}{2}\left(\boldsymbol{\mu}_{\mathbf{V},\Theta}^T\mathbf{I}_D^{-1}\boldsymbol{\mu}_{\mathbf{V},\Theta} + \mathrm{tr}(\mathbf{I}_D^{-1}\Sigma_{\mathbf{V},\Theta})\right) \\
&= \frac{1}{2}\left(-\log\left(|\Sigma_{\mathbf{V},\Theta}|\right) - tr(\mathbf{I}_D) + \boldsymbol{\mu}_{\mathbf{V},\Theta}^T\cdot\boldsymbol{\mu}_{\mathbf{V},\Theta} + \mathrm{tr}(\Sigma_{\mathbf{V},\Theta})\right). \qquad (5.13)
\end{aligned}$$

As $\Sigma_{\mathbf{V},\Theta}$ can be rewritten as a vector $\sigma^2_{\mathbf{V},\Theta}$ such that each element of $(\sigma^2_{\mathbf{V},\Theta}[i])_{0\leq i<D}$ defines the i^{th} diagonal of $\Sigma_{\mathbf{V},\Theta}$, then, Equation 5.13 can be expressed as follows:

$$\begin{aligned}
\mathcal{D}_{\mathrm{KL}}\left(\mathsf{Pr}\left[\mathbf{V}|\mathbf{T},Y,\Theta\right]||\mathsf{Pr}\left[\mathbf{V}\right]\right) &= -\frac{1}{2}\left(\log\prod_{i=0}^{D-1}\sigma^2_{\mathbf{V},\Theta}[i] + \sum_{i=0}^{D-1}1 - \sum_{i=0}^{D-1}\boldsymbol{\mu}^2_{\mathbf{V},\Theta}[i] - \sum_{i=0}^{D-1}\sigma^2_{\mathbf{V},\Theta}[i]\right) \\
&= -\frac{1}{2}\sum_{i=0}^{D-1}\left(1+\log\left(\sigma^2_{\mathbf{V},\Theta}[i]\right) - \boldsymbol{\mu}^2_{\mathbf{V},\Theta}[i] - \sigma^2_{\mathbf{V},\Theta}[i]\right).
\end{aligned}$$

While the KL-divergence measures how the $\mathsf{Pr}[\mathbf{V}|\mathbf{T},Y,\Theta]$ distribution is different from $\mathsf{Pr}[\mathbf{V}]$, standardizing input traces is helpful to reduce the impact of irrelevant time samples.

To clearly explain the impact of the KL-divergence loss on the learning process, let us denote $\mathbf{T}$ a D-dimensional leakage trace that has been standardized at each sample (zero mean, unit variance). Let $\{l_0,\ldots,l_{u-1}\}$ defines a set of indices where the sensitive information leaks (*i.e.* PoIs) such that,

$$\mathbf{T}[i] = \begin{cases} \psi(Y)[i] + \mathbf{Z}[i] \sim \mathcal{N}(0,1) & \text{if} \quad i\in\{l_0,\ldots,l_{u-1}\}, \\ \mathbf{Z}[i] \sim \mathcal{N}(0,1) & \text{otherwise.} \end{cases}$$

In this setting, we assume that the interactions between trace samples are negligible. Thus, if the leakage trace $\mathbf{T}$ is standardized, each time sample has a zero mean and unit variance. Hence, if the i^{th} time sample of $\mathbf{T}$ has no deterministic part (*i.e.* $\psi(Y)=0$), the related element of the latent variable $\mathbf{V}[i]=\mathbf{Z}[i]-\hat{\psi}_{\Theta}(Y)[i]$ follows $\mathcal{N}(0,1)$ which induces that $\hat{\psi}_{\Theta}[i]$ is negligible. Consequently, if $i\notin\{l_0,\ldots,l_{u-1}\}$, the related sample of the latent variable follows the same distribution as $\mathbf{Z}[i]$. Thus, in such scenario, the KL-divergence loss is negligible while the latent representation does not contain any information about the secret key k^*. Thus, considering these non-informative time samples does not affect the regularization term and are unsuitable for the decision process. This result encourages the Evaluator to only consider the time samples with a non-negligible deterministic part.
On the other hand, if $i\in\{l_0,\ldots,l_{u-1}\}$, then $\mathbf{V}[i]=\psi(Y)[i]+\mathbf{Z}[i]-\hat{\psi}_{\Theta}(Y)[i]$ follows $\mathcal{N}(0,1)$. Thus, it suggests that $\mathbf{Z}[i]\sim\mathcal{N}(\mathbb{E}[\hat{\psi}_{\Theta}(Y)[i]-\psi(Y)[i]],\mathbb{V}[\psi(Y)[i]]+\mathbb{V}[\hat{\psi}_{\Theta}(Y)[i]]+\mathbb{V}[\mathbf{V}[i]]-2\cdot Cov[\psi[i],\hat{\psi}_{\Theta}[i]]-2\cdot Cov[\psi[i],\mathbf{V}[i]]+2\cdot Cov[\hat{\psi}_{\Theta}[i],\mathbf{V}[i]])$. However, due to the KL-divergence loss function involving during the training process, we force the latent variable $(\mathbf{V}[i])_{0\leq i<D}$ to follow $\mathcal{N}_D(0,\mathbf{I}_D)$. As defined in Subsection 5.2.2, this latent variable characterizes an estimation of the noise $\mathbf{Z}$ induced in the leakage trace $\mathbf{T}$. Thus, during the training process of the cVAE-ST, we penalize the model to tend $\mathbb{E}[\hat{\psi}_{\Theta}(Y)[i]-\psi(Y)[i]]$ towards 0 and $\mathbb{V}[\psi(Y)[i]]+\mathbb{V}[\hat{\psi}_{\Theta}(Y)[i]]+$

$\mathbb{V}[\mathbf{V}[i]] - 2 \cdot Cov[\psi[i], \hat{\psi}_\Theta[i]] - 2 \cdot Cov[\psi[i], \mathbf{V}[i]] + 2 \cdot Cov[\hat{\psi}_\Theta[i], \mathbf{V}[i]]$ towards 1 such that this solution is reached if and only if $\hat{\psi}_\Theta = \psi$. Consequently, when KL-divergence loss is computed, the conditional variational autoencoder aims at optimizing the trainable parameters Θ of the encoder $F_\Theta^{(enc)}$ such that the regularization term equals 0 if and only if Θ is optimal. This justification suggests that the latent space should be only composed by points of interest. One solution is to consider dimensionality reduction techniques. In addition, when the Gaussian noise increases, the dependence between $\mathbf{T}[i]$ and $\psi[i]$ decreases. In this configuration, differentiating the sensitive information from the noise can be difficult as $\mathbf{Z}[i]$ approximately follows $\mathcal{N}(0, 1)$ regardless the information included in the time sample i. This observation demonstrates the benefits of the noise to reduce the efficiency of side-channel attacks.

Once the generative model $F_{\Theta,\phi}$ is trained, the Evaluator has to make a decision following the approximation of $\mathsf{Pr}[\mathbf{T}|Y]$ in order to fit with the stochastic attack approach (see Subsection 5.1.2). The following section describes this strategy.

5.2.4 Decision Rule & Network Complexity

Typically, the inference phase of conditional variational autoencoder consists in generating new set of data based on an input and a conditional known label. In the side-channel context, our goal is different and tends to find the conditional unknown label Y that fits the best for a given trace $\mathbf{T}$. The following part describes the proposed solution to retrieve the secret key from the model previously defined.

Key Extraction Phase. During the training phase, we defined a function $F_{\Theta,\phi}$ that approximates $\log(\mathsf{Pr}[\mathbf{T}|Y, \phi])$ through an optimization algorithm (*i.e.* gradient descent-based algorithms) such that the generated leakage trace $\tilde{\mathbf{T}}$, defined by the output of the decoder $F_\phi^{(dec)}$, are close to the real one $\mathbf{T}$ captured for a given label. Based on these new generated traces $\tilde{\mathbf{T}}$, the Evaluator can find the key hypothesis that approximates the marginal likelihood. Let $\mathcal{I}_a$ be a set of N_a attack traces such that $\mathcal{I}_a = \{\mathbf{t}_0, \ldots, \mathbf{t}_{N_a-1}\}$. For each of these attack traces, the Evaluator can compute the related label $Y = f(X, k)$ mixing the known plaintexts $X \in \mathcal{X}$ and a key hypothesis $k \in \mathcal{K}$. Then, the *Monte Carlo* method can be performed to get an estimation of $\mathsf{Pr}[\mathbf{T}|Y, \phi]$. Hence, for a set of N_v latent samples $\{\mathbf{v}_0, \ldots, \mathbf{v}_{N_v-1}\}$, we can compute an approximation of the marginal log-likelihood as follows:

$$\log\left(\mathsf{Pr}\left[\mathbf{t}_i|y_i, \phi\right]\right) \approx -\mathcal{D}_{\mathrm{KL}}\left(\mathsf{Pr}\left[\mathbf{V}|\mathbf{t}_i, y_i, \Theta\right] || \mathsf{Pr}\left[\mathbf{V}\right]\right) - \frac{1}{2}\log\left(2\pi \cdot \sum_{j=0}^{D-1}\left(\mathbf{t}_i[j] - \frac{1}{N_v}\sum_{h=0}^{N_v-1}\tilde{\mathbf{t}}_h[j]\right)^2\right) - \frac{1}{2}, \tag{5.14}$$

where $\tilde{\mathbf{t}}_h$ is the h^{th} generated leakage trace constructed from the decoder $\mathsf{Pr}[\mathbf{t}_i|y_i, \mathbf{v}_h, \phi]$.
When the inferred posterior $\mathsf{Pr}[\mathbf{V}|\mathbf{T}, Y, \Theta]$ deviates from the true unknown posterior $\mathsf{Pr}[\mathbf{V}|\mathbf{T}, Y, \phi]$, the number of samples N_v increases in order to obtain an accurate approximation of $\mathsf{Pr}\left[\mathbf{T}|Y, \phi\right]$. If the profiling phase has been performed successfully, then $\left(\mathbf{t}_i - \frac{1}{N_v}\sum_{j=0}^{N_v-1}\tilde{\mathbf{t}}_j\right)^2$ should be minimized when $k = k^*$ (see Theorem 5.1.1.1). Hence, the most likely candidate is defined through the maximum likelihood rule:

$$\hat{k} = \underset{k \in \mathcal{K}}{\arg\max}\left(\sum_{i=0}^{N_a-1}\log\left(\mathsf{Pr}\left[\mathbf{t}_i|f(x_i, k), \phi\right]\right)\right).$$

To enhance the key extraction phase, the Evaluator can precisely define the PoIs' indexes *via* a leakage assessment once the profiling phase is performed. Indeed, if Θ and ϕ are correctly learned, the Evaluator can visualize them in order to properly select the points of interest (see Subsection 5.3.1). Thus, during the attack phase, the Evaluator can only compute Equation 5.14 on the samples that are considered as relevant by the conditional variational autoencoder based on stochastic attacks.

Theoretical Network Complexity Bounds. Based on the previous sections, we can efficiently find an architecture for a given implementation. Consequently, some theoretical network complexity bounds can be expressed following the Evaluator's knowledge.

Indeed, our generative neural network can be easily built for a given $Y \in \mathbb{F}_2^n$, a degree d (s.t $d \leq n$) of bits' interaction and a D-dimensional leakage trace $\mathbf{T} \in \mathbb{R}^D$. First, for estimating $(\hat{\psi}_\Theta[i])_{0\leq i<D}$ (resp. $(\hat{\psi}_\phi[i])_{0\leq i<D}$), the encoder (resp. decoder) needs to optimize Θ (resp. ϕ) in order to retrieve the correct leakage model. Hence, for a given $Y \in \mathbb{F}_2^n$, the number of weights that has to be optimized is $((1 + \sum_{i=0}^{d-1} \binom{n}{i}) \cdot D)$ in both cases. Then, for estimating $(\mathbf{V}[i])_{0\leq i<D}$ (resp. $(\tilde{\mathbf{T}}[i])_{0\leq i<D}$), we have to link the i^{th} sample of the leakage trace $\mathbf{T}$ (resp. the noise $\mathbf{V}$) with the related $\hat{\psi}_\Theta[i]$ (resp. $\hat{\psi}_\phi[i]$). Here, we decide to follow the classical stochastic models in order to easily extract the related noise. Hence, no weights are needed for this operation. Finally, to approximate $\boldsymbol{\mu}_{\mathbf{V},\Theta}$ (resp. $\Sigma_{\mathbf{V},\Theta}$), we need $(D \cdot (D+1))$ (resp. $D^2 \cdot (D+1)$) neurons. For the simplified diagonal case, $\Sigma_{\mathbf{V},\Theta}$ can be reduced to $\sigma^2_{\mathbf{V},\Theta}$, thus, only $D \cdot (D+1)$ neurons are needed in this configuration. To sum up this complexity, the Evaluator needs to construct a generative model with $(D \cdot ((D+1)^2 + 2 \cdot (1 + \sum_{i=0}^{d-1} \binom{n}{i})))$ weights (resp. $(2D \cdot ((D+1) + 1 + \sum_{i=0}^{d-1} \binom{n}{i}))$ weights if $\Sigma_{\mathbf{V},\Theta}$ is reduced to $\sigma^2_{\mathbf{V},\Theta}$).

If a leakage assessment is performed before the construction of the generative model, then, the Evaluator can extract a subset of indices defining the time samples where the sensitive information leaks (*i.e.* PoIs). For example, if the Evaluator detects u PoIs, he can construct a vector $\{l_0, \ldots, l_{u-1}\}$ of u indices such that l_i denotes the index related to the i$^{\text{th}}$ point of interest. Based on this knowledge, he can build a conditional variational autoencoder with lower complexity such that most of the relevant information, chosen to be related to the u PoIs, can be extracted from a leakage trace. Instead of considering all the samples of the D-dimensional trace $(D \geq u)$, he can construct a neural network with $(u \cdot ((u+1)^2 + 2 \cdot (1 + \sum_{i=0}^{d-1} \binom{n}{i})))$ weights (resp. $(2u \cdot ((u+1) + 1 + \sum_{i=0}^{d-1} \binom{n}{i}))$ weights if $\Sigma_{\mathbf{V},\Theta}$ is reduced to $\sigma^2_{\mathbf{V},\Theta}$). As a consequence, we drastically reduce the network complexity without altering the ability of the generative model to retrieve the secret key as suggested in Subsection 5.2.3. For example, the network complexity of Figure 5.3 is about $1,040$ weights if all bits' interactions are considered (*i.e.* $d = 8$, $u = 2$, $n = 8$).

SUM UP...

The goal of the conditional variational autoencoder based on stochastic attacks is to approximate the true unknown conditional probability $\mathsf{Pr}[\mathbf{T}|Y]$ from a neural network $F_{\Theta,\phi} = F_\phi^{(dec)} \circ F_\Theta^{(enc)}$. To perform such estimation, the Evaluator has to respect the following steps:

1. Construct the encoder $F_\Theta^{(enc)}$ and the decoder $F_\phi^{(dec)}$ in order to respect the stochastic attacks structure defined in Section 5.1. The Evaluator can refer to Subsection 5.2.2 in order to design suited model.

2. Train the conditional variational autoencoder with the empirical risk combined with evidence lower bound loss (see Definition 5.2.3.1) in order to maximize the similarities between new generated leakage traces $\tilde{\mathbf{T}}$ and the real one $\mathbf{T}$.

3. Extract the secret key manipulated by the cryptographic module through the application of the decision rule described in Subsection 5.2.4.

One of the main benefits of the proposed variational autoencoder is its explainability and its interpretability regarding the side-channel context. In addition, our theoretical results suggest that its width does not have to be large no matter the dimension of the leakage traces. This result is faithful with the Universal Approximation Theorem (see Theorem 4.2.1.1). Through the following section, we validate these properties and broaden the attacks' spectrum on protected implementations considering Boolean making scheme.

5.3 Investigations on the Constructed Generative Model

Through this section, we validate all the theoretical observations provided in Section 5.2 without optimizing the hyperparameters selection. While classical discriminative models need to tune a lot of hyperparameters introduced in Subsection 4.2.2 and Subsection 4.3.1 (*i.e.* type of neural network structure, number of layers, number of nodes per layer, activation function, optimizer algorithms, learning rate, number of epochs, batch size), the configuration of the proposed conditional variational autoencoder based on stochastic attacks only considers the optimizer algorithm[g], the batch size, the learning rate and the number of epochs. In this section, optimization is done using the *Adam* algorithm on batch size $\{8, 16, 32, 64, 128\}$ and the learning rate is set to $\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$. We construct each model with the following number of epochs $\{10, 20, 30, 40\}$ and select the parameters that provide the best ranking value.
Finally, in this section, Nt_{rank} denotes the number of attack traces that are needed to reach a constant rank of 1. The attack leakage traces are randomly shuffled and picked up from a set of attack leakage traces $\mathcal{I}_a$ which is characterized by simulations that are described in the following. For a good estimation of Nt_{rank}, an average over 10 simulations, denoted $\bar{Nt}_{\text{rank}}$, is computed. Finally, as the goal of this basis is to ease the explainability and the interpretability of the results provided by our generative model, the basis that is considered for estimating the leakage model ψ is the orthonormal monomial basis $\Phi_u^{(orth.)}$ defined in Definition 5.1.2.2.

The next section proposes to visualizes the trainable parameters Θ and ϕ in order to assess the suitability of the conditional variational autoencoder to extract the expected leakage model ψ.

5.3.1 Leakage Model Visualization

To evaluate the benefits of the generative model to retrieve the correct basis, we simulate a set of $10,000$ traces ($9,000$ for the profiling phase and $1,000$ for the validation phase) through multiple scenarios:

- **Scenario 1** – We assume that each leakage trace is configured by 3 time samples such that only 1 point of interest is considered. The leakage model induces the maximum amount of interactions between bits (*i.e.* $\mathcal{G}_9$, see Definition 5.1.1.2) such that all bits influencing the leakage model have the same weighting. Hence, the i^{th} time sample of the simulated trace $\mathbf{T}$ is defined as follows:

$$\mathbf{T}[i] = \begin{cases} 1 \cdot Y[1] + 1 \cdot Y[3] + 1 \cdot Y[6] \\ + 1 \cdot \oplus_{b=0}^{1} Y[b] + 1 \cdot \oplus_{b=0}^{2} Y[b] + 1 \cdot \oplus_{b=0}^{3} Y[b] \\ + 1 \cdot \oplus_{b=0}^{4} Y[b] + 1 \cdot \oplus_{b=0}^{5} Y[b] + 1 \cdot \oplus_{b=0}^{6} Y[b] \\ + 1 \cdot \oplus_{b=0}^{7} Y[b] + \mathbf{Z}[i] & \text{if} \quad i = 1, \\ \mathbf{Z}[i] & \text{otherwise,} \end{cases} \tag{5.15}$$

  where $\oplus_{b=0}^{n} Y[b] = Y[0] \oplus \ldots \oplus Y[n]$, $Y[b] = Sbox[X \oplus k^*][b]$ denotes the b^{th} bit of the output of the Sbox, and $\mathbf{Z}[i]$ is a Gaussian noise following $\mathcal{N}(0, \sigma^2)$ such that $\sigma^2 \in \{0.1, 1, 10\}$.

- **Scenario 2** – We assume that each leakage trace is configured by 4 time samples such that only 2 points of interest are considered. The leakage model does not induce interactions between bits (*i.e.* $d = 1$ and $\mathcal{G}_2$) but differs by the location of the points of interest. Hence, the i^{th} time sample of the leakage trace $\mathbf{T}$ is defined as follows:

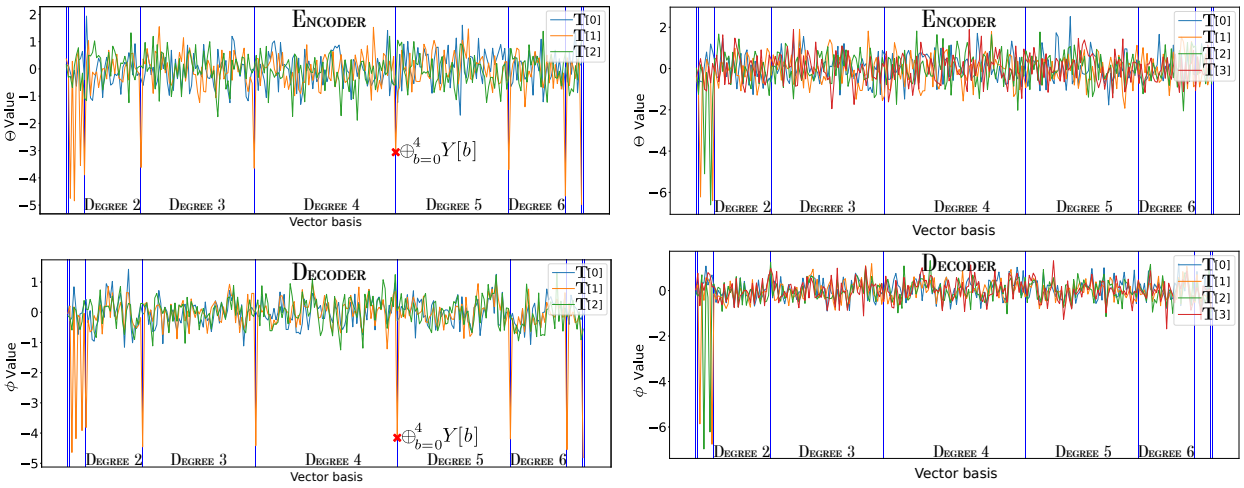
$$\mathbf{T}[i] = \begin{cases} 1 \cdot Y[3] + 1 \cdot Y[6] + \mathbf{Z}[i] & \text{if} \quad i = 1, \\ 1 \cdot Y[1] + 1 \cdot Y[7] + \mathbf{Z}[i] & \text{if} \quad i = 2, \\ \mathbf{Z}[i] & \text{otherwise,} \end{cases} \tag{5.16}$$

[g]This observation is an intrinsic property of the proposed neural network architecture. For example, if the Evaluator wants to construct an encoder in order to reduce the desynchronization effect as well as the trace dimension, he has to deal with model hyperparameters.

where $Y[b] = Sbox[X \oplus k^*][b]$ and $\mathbf{Z}[i]$ is a Gaussian noise following $\mathcal{N}(0, \sigma^2)$ such that $\sigma^2 \in \{0.1, 1, 10\}$.

Different levels of noise are considered for all scenarios in order to get an overview into how $F_{\Theta,\phi}$ performs depending on the SNR result but their results are provided in Appendix A.

As mentioned in Section 5.2, the encoder (resp. decoder) is trained to retrieve the parameters Θ (resp. ϕ) in order to maximize their correlation with the targeted leakage model. Once the generative model is correctly trained, the Evaluator can visualize these trainable parameters (*i.e.* Θ and ϕ) in order to find the security flaws induced in the studied implementation. In all scenarios (see Figure 5.4), the weight visualization can be used to assess the ability of the encoder (resp. decoder) to retrieve the leakage function defined in each scenario (*i.e.* Equation 5.15, Equation 5.16). Indeed, these figures illustrate the coefficient associated to each vector of the orthonormal monomial basis. The first coefficients of each figure define the lowest bits' interaction induced in the leakage model. For example, the first element, included in the interaction of degree 5 area, is characterized by $\oplus_{b=0}^{4} Y[b]$. While the related weight is non-negligible, the conditional variational autoencoder based on stochastic attacks identifies the following interactions $\oplus_{b=0}^{4} Y[b]$ in the leakage model. This observation can be confirmed with Equation 5.15. Proceeding this analysis for the entire set of non-negligible weights can be helpful to evaluate the ability of the generative model to retrieve the leakage model. Indeed, if we compare the real simulated leakage model defined in Equation 5.15 with the non-negligible weights depicted in Figure 5.4a, we can see that all the peaks are associated with the correct basis vector. This observation can be extended when the leakage model varies following the targeted time sample (see Figure 5.4b).



(a) Scenario 1 (see Equation 5.15). **(b)** Scenario 2 (see Equation 5.16).

Figure 5.4: Weight visualization assessing the suitability of our generative model to retrieve the leakage model.

In addition, each coefficient associated with the sensitive interactions seems to get approximately the same impact which corresponds to the real leakage function defined in Equation 5.15. Consequently, if the conditional variational autoencoders based on stochastic attacks are correctly trained, they sound helpful to retrieve complex leakage models and the related security flaws. Through the additional simulations provided in Appendix A, our generative model can be considered to evaluate the security flaws when large bits' interactions are observed, when the deterministic part is different for all PoIs and when different weighting occurs. As a consequence, large use-cases can be considered when the cVAE-ST is applied. Moreover, through the visualizations provided in Figure 5.4, the Evaluator can also identify the time samples where the sensitive information leaks. Indeed, Figure 5.4a (resp. Figure 5.4b) highlights that only $\mathbf{t}[1]$ (resp. $\mathbf{t}[1]$ and $\mathbf{t}[2]$) is useful to extract the leakage model. Hence, once the generative model is correctly

trained, the Evaluator can easily retrieve the points of interest. Then, during the attack phase, the Evaluator can decide to focus its attack by computing Equation 5.14 only on a small set of relevant samples instead of the entire trace dimension as mentioned in Subsection 5.2.4. One advantage of the stochastic model is to approximate the data that depends on the secret key. Through this process, the Evaluator directly obtains a score related to the key manipulated by the cryptographic module. Hence, the Evaluator can adapt the orthonormal monomial basis to target simultaneously multiple cryptographic primitives (*e.g.* input and output of the Sbox)[h].

The following section explores the ability of the conditional variational autoencoder to perform such simultaneous attacks.

5.3.2 Multi-Sensitive Variable Attacks

To evaluate the benefits of our generative model to retrieve the multiple sensitive variables, we simulate a set of $10,000$ leakage traces ($9,000$ for the profiling phase and $1,000$ for the validation phase). In the following scenario, $\mathbf{t}$ is a 4-dimensional leakage trace such that 2 points of interest are configured. The i^{th} time sample of the leakage trace $\mathbf{T}$ is defined as follows:

$$\mathbf{T}[i] = \begin{cases} 1 \cdot (X \oplus k^*)[5] + 1 \cdot (X \oplus k^*)[3] & \\ \oplus\, (X \oplus k^*)[7] + \mathbf{Z}[i] & \text{if} \quad i = 1, \\ 1 \cdot Sbox\left[X \oplus k^*\right][3] + 1 \cdot Sbox\left[X \oplus k^*\right][6] + \mathbf{Z}[i] & \text{if} \quad i = 2, \\ \mathbf{Z}[i] & \text{otherwise}, \end{cases} \tag{5.17}$$

where $Sbox\left[X \oplus k^*\right][b]$ denotes the b^{th} bit of the output of the Sbox considering a plaintext X and the secret key k^*, $\mathbf{Z}[i]$ is a Gaussian noise following $\mathcal{N}(0, \sigma^2)$ such that $\sigma^2 = 1$.
To exploit all the bits' interaction for each sensitive variable, we set $d = 8$ and compute two orthonormal monomial basis, *i.e.* $(\Phi_u^{(orth.)}(X \oplus k^*))_{u \in \mathbb{F}_2^n}$ et $(\Phi_u^{(orth.)}(Sbox\left[X \oplus k^*\right]))_{u \in \mathbb{F}_2^n}$, such that Figure 5.5 illustrates the impact of each vector of each basis in $\mathcal{G}_9$. When the time sample $\mathbf{t}[1]$ is considered, we can see an interaction of degree 1 and 2 that corresponds to the bit 5 and the interaction between the bits 3 and 7 of the input of the Sbox. Then, through Figure 5.5, we can see that $\mathbf{t}[2]$ extracts a leakage model with two interactions of degree 1. When we refer to Equation 5.17, we observe that the true leakage model depends on 3^{th} and the 6^{th} bit of the Sbox output. Hence, through this simulation, we can validate the ability of the conditional variational autoencoder to correctly retrieve the leakage model of multiple sensitive variables simultaneously. However, as mentioned in Subsection 5.2.4, the degree d of the orthonormal monomial basis $\mathcal{G}_{d+1}$ directly affects the complexity of the conditional variational autoencoder based on stochastic attacks. Hence, considering the attacks of multi-sensitive variable increase by $2D \cdot (1 + \sum_{i=0}^{d-1} \binom{n}{i})$ the number of trainable parameters (*i.e.* weights) for each new targeted sensitive variable[i]. Thus, depending on his computational capacity, the Evaluator has to define the most suitable structure to employ for defeating the targeted cryptographic module.

However, the Evaluator can wonder how does the conditional variational autoencoder retrieves the points of interest when widely uninformative samples are considered. The following section answers this question.

5.3.3 Curse of Dimensionality

When the Evaluator performs a side-channel attack, he wants to precisely find the relevant key-dependent time sample even if a large part of the leakage trace contains uninformative time samples. Usually, the number of points of interest u is far lower than the trace dimension D (*i.e.*

[h]Similar proposition was introduced by Maghrebi for the discriminative models [Mag20]. But its investigation on discriminative models is not considered in this thesis.

[i]This number can be reduced to $2u \cdot (1 + \sum_{i=0}^{d-1} \binom{n}{i})$ if the Evaluator only targets the points of interest.

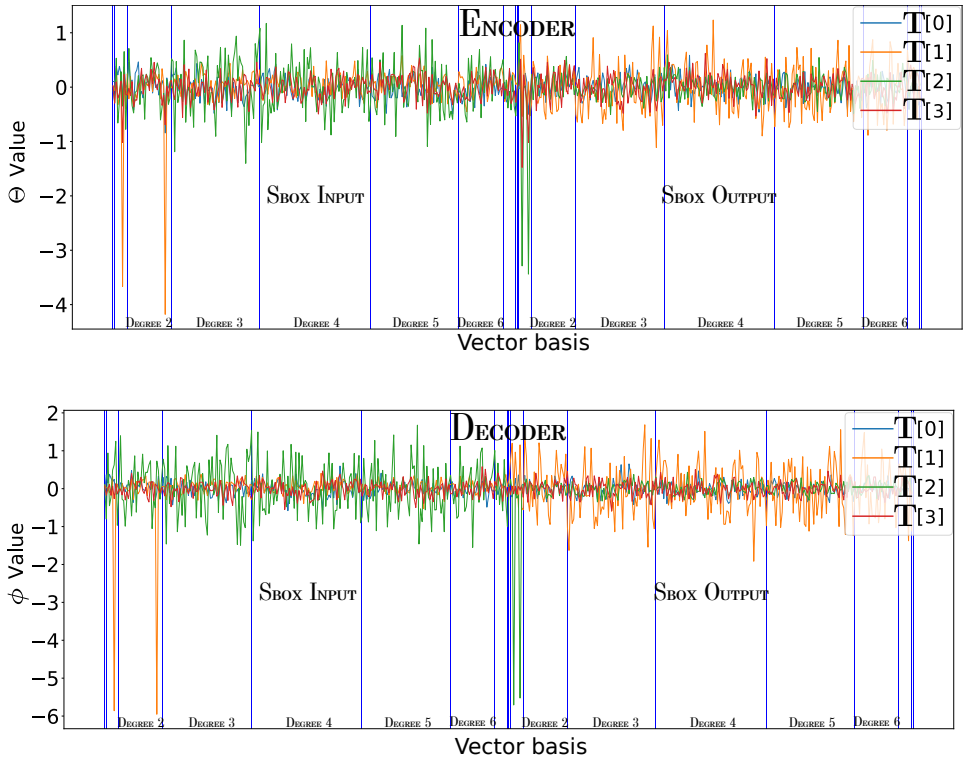


Figure 5.5: Weight visualization for assessing the suitability our generative model to simultaneously target multiple intermediate values (see Equation 5.17).

$u \ll D$). Thus, to assess the benefits of the conditional variational autoencoder in side-channel attacks, we have to understand the ability of this new generative model to retrieve the points of interest when a lot of samples are irrelevant. In order to evaluate it under this restriction, a wide range of datasets are simulated (see Section 3.6) with different levels of D. We construct 6 scenarios where $D \in \{3, 10, 50, 100, 500, 1000\}$ and $u = 1$ such that the noisy samples follows a Gaussian distribution with parameters $\mu = 0$ and $\sigma^2 = 1$. The related SNR equals to 0.549. Hence, for each case study, only a single PoI is configured while the dimension of the simulated leakage traces increases. In Table 5.1, we denote $P_{RI} = \frac{u}{D}$, the percentage of relevant information in each scenario and evaluate the impact of this variable on other parameters, namely the batch-size and the learning rate. Finally, N_v denotes the number of samples $\mathbf{V}$ used to perform Equation 5.14.

Table 5.1: Impact of the trace dimension on the conditional variational autoencoder performance (with $u = 1$, $N_v = 10$, batch-size $= 10$).

P_{RI}%	D	Learning rate	$\bar{N}t_{\text{rank}}$	Network complexity	Training time (seconds)
33%	3	10^{-2}	47	$1,566$	$7s$
10%	10	10^{-2}	51	$5,360$	$8s$
2%	50	10^{-2}	49	$30,800$	$35s$
1%	100	10^{-2}	46	$71,600$	$47s$
0.2%	500	10^{-3}	52	$758,000$	$401s$
0.1%	1000	10^{-3}	65	$2,516,000$	$933s$

As suggested in Subsection 5.2.4, the attack process is performed only on the time sample defined as relevant[j] by the generative model. Hence, the weight visualization applied on ϕ and Θ is very helpful to define which samples can be considered as PoIs. Through Table 5.1, we can see that

[j]Here, the relevance of a time sample is characterized by its coefficients ϕ and Θ such that the most relevant have the highest coefficients.

increasing D does not impact the resulted performance of the generative model (*i.e.* $\bar{N}t_{\text{rank}}$). Indeed, if the evaluator adequately finds the correct hyperparameters, namely batch-size and learning rate, he can expect to get similar results for high values of D. However, as detailed in Subsection 5.2.4, increasing the input dimension highly impacts the complexity of the conditional variational autoencoder. Finding a way to focus the interest of the model only on the relevant time samples can drastically reduce the network complexity (see Definition 4.2.2.3) without altering its resulted performance. Such investigation could be part of a future work.

Once the Evaluator validates the ability of the generative model to deal with a low percentage of relevant information, he can question the benefits of the conditional variational autoencoder to defeat Boolean masking implementations. The next section deeper investigates this protection against this type of generative model.

5.3.4 Generalization on Boolean Masking Implementation

Typically, the generative approach approximates the leakage model without altering its representation in order to get a global characterization of $\mathsf{Pr}[\mathbf{T}|Y]$ with $\mathbf{T}$ the leakage trace and Y the related label. Consequently, our generative approach does not automatically recombine the PoIs but preserve the network's explainability that is mandatory for the evaluation process. Thus, if the Evaluator has to face with a masking implementation, the preprocessing phase, introduced in Section 3.5, has to be performed.
As a remainder, a masking scheme of order $d-1$ consists in a d-sharing of the sensitive variables $\{Y^0, Y^1, \ldots, Y^{d-1}\}$. In our setting, this corresponds to a situation where the leakages of the d shares are hidden among values that have no relation with the target. As defined in Section 3.5, the Evaluator has to *combine* the leakage traces in order to reveal the dependence of the leakage trace $\mathbf{T}$ and its related label Y. To evaluate the suitability of the conditional variational autoencoder in such scenarios, we decide to simulate 5-dimensional leakage traces with different levels of masking order $d \in \{0, 1, 2, 3\}$. For each case study, we apply the product (see Equation 3.8), the absolute difference (see Equation 3.9) and the optimal product (see Equation 3.10) combining functions and list the best result obtained in Table 5.2.

Table 5.2: Impact of Boolean masking implementations on the conditional variational autoencoder performance (with batch-size = 64, $N_v = 10$).

Order	**Learning rate**	$\bar{\boldsymbol{N}}\boldsymbol{t}_{\text{rank}}$	**Combining function**	**Network complexity**
0	10^{-2}	9	Optimal product	$2,630$
1	10^{-2}	32	Optimal product	$14,150$
2	10^{-3}	100	Optimal product	$95,750$
3	10^{-3}	247	Absolute difference	$1,103,750$

Through this table, we demonstrate the ability of the proposed generative model to defeat a high-order Boolean masking implementation. Surprisingly, the number of shares does not highly impact the hyperparameters' value[k], namely the learning rate and the batch-size, unlike the network complexity. Indeed, for a given set of D-dimensional leakage traces, the combining methods multiplied by D the number of time samples for each mask reduction. Hence, for performing a d order attack, the Evaluator has to deal with leakage traces of D^{d+1} samples. As the dimension of the leakage traces impacts the network complexity (see Subsection 5.2.4), the Evaluator has to exponentially increase his computational ability with the order of the side-channel attack.

[k]The readers must be aware that this observation cannot be generalized on all implementations and would benefit from further investigations.

Once all these simulations validate the theoretical observations provided in Section 5.2, we compare the benefits of considering the new conditional variational autoencoder with the classical profiled side-channel attacks on real unprotected and protected implementations.

5.4 Practical Experiments

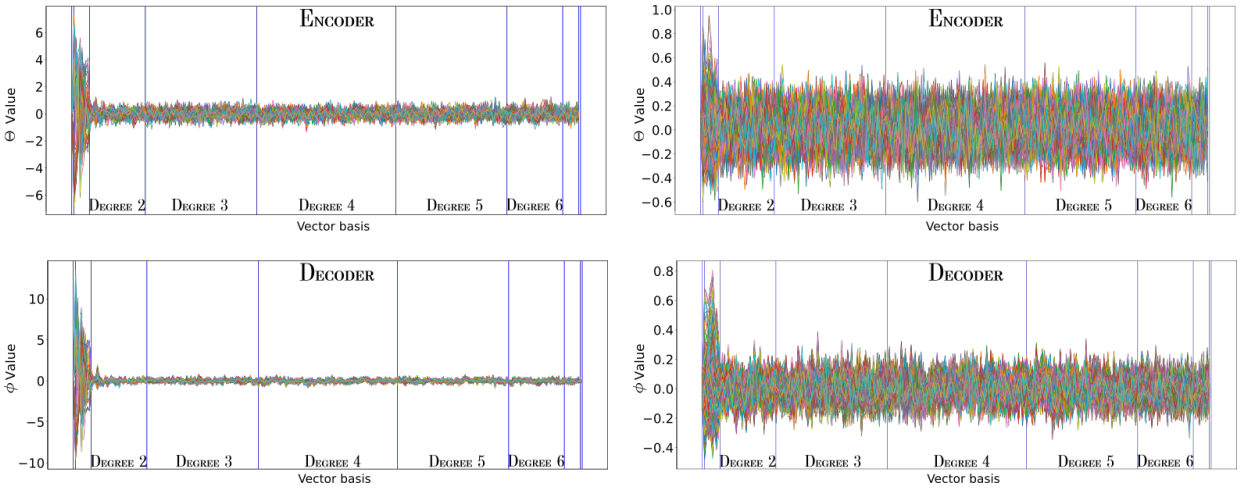
The experiments are implemented in Python using the *Keras* library [Cho15] and are run on a workstation equipped with 128GB RAM and a NVIDIA GTX1080Ti with 11GB memory. The configurable hyperparameters, namely the batch-size and the learning rate, are respectively set to $\{8, 16, 32, 64\}$ and $\{10^{-3}, 10^{-2}, 10^{-1}\}$. We construct each model with the following number of epochs $\{10, 20, 30, 40, 50, 75, 100\}$ and select the value that provides the best rank. As mentioned in Section 5.3, we denote $\bar{N}t_{\text{rank}}$ the average value of Nt_{rank} over 10 shuffled experiments. In the following, we always capture the maximum amount of interactions (*i.e.* $\mathcal{G}_9$). This choice was made because we assume that the Evaluator does not have *a priori* knowledge on the bits' interactions. This section considers three datasets that have been presented in Section 3.6, namely DPA contest-v4, AES_HD and ASCAD-v1. All these experiments can be reproduced through the following reference [ZBC+21a].

5.4.1 A Comparison with Classical Generative Side-Channel Attacks

In this section, we evaluate the benefits of the conditional variational autoencoder against the classical side-channel attacks (*i.e.* template attacks, stochastic attacks). We respect the same conditions as the state-of-the-art result. Hence, we construct a *pooled template attack* [CK14] and a *pooled stochastic attack* [CK15] that consist in manipulating different key hypotheses $k \in \mathcal{K}$ with different means but the same covariance. Thus, we *pool* the covariance matrices into a single solution in order to cope with statistical difficulties. For the DPA contest-v4 and AES_HD datasets, Kim *et al.* [KPH+19] propose to select 50 features with the highest SNR in order to reduce the needs of computation when classical side-channel attacks are considered. For the ASCAD-v1 dataset, [BPS+20] applies a dimensionality reduction algorithm, namely *Principal Components Analysis* (see [BPS+20, Figure 12]). While no clear description of high-order attack is defined in [BPS+20], we select the 8 most relevant samples related to the mask and the masked values and then, apply the three combining functions introduced in Section 3.5.

DPA contest-v4. Once the conditional variational autoencoder is trained, the Evaluator can observe the coefficients related to each time sample as illustrated in Figure 5.6a. Then, he can select those with the highest trainable parameters (*i.e.* Θ and ψ) and perform his attack on this subset. For this dataset, we compute Equation 5.14 on the 50 time samples previously extracted. When a high-SNR unprotected implementation is considered, we observe that our generative model has the same performance as classical profiled side-channel attacks (see Table 5.3). Hence, for this implementation, similar results can be obtained whatever the attack performed. Consequently, in this configuration, considering the conditional variational autoencoder based on stochastic attacks is equivalent to classical profiled side-channel attacks.

AES_HD. Following the state-of-the-art results [KPH+19], a pooled template attack needs approximately $25,000$ attack traces to retrieve the secret key. Surprisingly, performing the stochastic attack on the same dataset highly improves the related performance. Indeed, when this approach is considered, the Evaluator can recover the secret key with $4,500$ attack traces which is 5.5 times better. Finally, when our generative model is applied, an even better attack can be performed. As mentioned in Subsection 5.2.4, the attack phase is based on sample similarity measures. Hence, the Evaluator can only compute Equation 5.14 on the relevant samples detected during the profiling phase. Thus, we drastically reduce the impact of the uninformative samples



(a) DPA contest-v4 dataset. **(b)** AES_HD dataset.

Figure 5.6: Weight visualization of 50 time samples assessing the suitability of our generative model to retrieve the leakage model.

Table 5.3: Comparison of $\bar{Nt}_{\text{rank}}$ value depending on datasets

	Pooled Stochastic Attacks	**Pooled Template Attacks**	**cVAE-ST**
DPA-contest v4	4	4 [KPH+19]	4
AES_HD	4,500	25,000 [KPH+19]	250
ASCAD-v1	290	351	190

during the attack phase. In this configuration, we only consider the samples where the related ϕ coefficients are greater than 0.5 (see Figure 5.6b). Accordingly, while the training process was performed on traces with 50 samples, the computation of Equation 5.14 was made on the 14 time samples complying with the restriction configured. This processing tremendously increases the performance of the resulted attack. Indeed, the cVAE-ST model divides by 100 (resp. 18) the number of attack leakage traces that are needed to perform a pooled template attack (resp. pooled stochastic attack).

ASCAD-v1. As mentioned in Section 3.5, we perform high-order attacks with the help of combining functions as preprocessing (*i.e. product combining*, *optimal product combining*, *absolute difference combining*). Then, we profile the generative models on the unmasked value in order to extract the relevant information. In Table 5.3, the optimal product combination provides the best performance on the ASCAD-v1 dataset. Through this experiment, we observe that the conditional variational autoencoder performs better than template and stochastic attacks. While 351 (resp. 290) attack traces are needed to reach a constant rank of 1 when the pooled template attacks (resp. pooled stochastic attack) are considered, our generative model retrieves the secret key within 190 attack traces. As previously mentioned for the AES_HD dataset, this result can be explained by the ability of the conditional variational autoencoder to target a specific range of relevant combined time samples during the attack phase. On the contrary, the classical profiled side-channel attacks have to consider the 64 time samples (*i.e.* 8 time samples related to the masks and the masked value) used to perform the related attacks. Hence, resulted noisy time samples can highly influence the performance of the resulted attacks.

In conclusion, the conditional variational autoencoder based on stochastic attacks provides similar performance as the pooled template attack and the pooled stochastic attacks. However, when a classical profiled side-channel attack is trained on D-dimensional traces, the Evaluator has to perform the exploitation phase on the same leakage trace dimension. Unfortunately, the Evaluator

does not know *a priori* which time samples are considered as relevant once the profiling phase is applied. Hence, performing the exploitation phase on the D-dimensional leakage traces could be impacted by the uninformative time samples. On the other hand, once the profiling phase is performed on the D-dimensional traces, our generative model is beneficial to select a subset of u time samples, such that $u \ll D$, in order to compute Equation 5.14 only on the informative time samples (*i.e.* points of interest). Hence, this new proposition is more flexible than classical profiled side-channel attacks.

However, the generative approach can be limited by multiple factors, the following section highlights these limitations.

5.4.2 Benefits & Limitations of cVAE-ST

As previously illustrated, the conditional variational autoencoders can be considered in side-channel context in order to perform physical attacks and extract the secret key from a cryptographic module. From an evaluation perspective, this new neural network architecture is suitable as it respects the following requirements:

1. **Theoretical similarities with classical profiled side-channel attacks** – As illustrated in Section 5.2, the conditional variational autoencoder can be monitored to fit with the stochastic attacks paradigm introduced by Schindler *et al.* [SLP05] and recalled in Section 5.1. From the Evaluator point of view, this approach is useful to ease the configuration of the neural network and get a clear overview of the decision-making process. Indeed, as the conditional variational autoencoder is designed on well-known theoretical attack strategy, the Evaluator can be confident on the employed neural network structure and thus, expect to get a resulted predictive model as efficient as classical profiled side-channel attacks, namely template attacks and stochastic attacks. From this new bridge, the Evaluator can deeply understand the future improvements that can be provided in order to fully exploit the automation process in side-channel context.

2. **Explainability & Interpretability** – One major benefit of the proposed neural network architecture is to preserve the interpretability and the explainability on the results provided by the learning algorithm. As the conditional variational autoencoders are constructed from the classical profiled side-channel attacks, the Evaluator can adapt its interpretation tools (*e.g.* visualization) in order to deeply explain the results provided by the model. As suggested in Subsection 5.3.1 and Subsection 5.4.1, the Evaluator can visualize the trainable parameters of the conditional variational autoencoder in order to assess the ability of the encoder and the decoder to retrieve a hypothetical leakage model as similar as possible to the true unknown ψ. Once the Evaluator retrieves an approximation of ψ, he highlights the security flaws induced by the targeted implementation and thus, can alert the Developer on potential vulnerabilities and ease the development of countermeasures.

3. **Hiding countermeasures** – Even if this manuscript does not assess the robustness of the conditional variational autoencoder based on stochastic attacks against desynchronization effect[1], an intuitive solution suggests to add convolutional layers to the encoder. However, it should be validated in practice. While this intuition could be a suitable solution to mitigate the desynchronization effect, it also helps the network to automatically select the points of interest and prevent the effect of uninformative time samples. Indeed, as defined in Subsection 5.2.3 and Subsection 5.2.4, the empirical risk as well as the decision process are only affected by the points of interest. Hence, this dimensionality reduction technique can also be useful to quadratically reduce the network complexity.

[1]This choice is motivated by our willingness to fit with the stochastic attacks and thus, bridging the deep learning with classical profiled side-channel attacks.

However, the conditional variational autoencoder based on stochastic attacks also has some limitations that are listed below.

1. **Combining function** – As the generative approach, as well as classical profiled side-channel attacks, captures the conditional distribution $\mathsf{Pr}[\mathbf{T}|Y]$, it cannot handle masking implementations as the targeted unmasked sensitive variable Y is not directly observable through the leakage trace $\mathbf{T}$. Thus, the Evaluator has to consider combining functions in order to reveal the dependence between $\mathbf{T}$ and Y. Unfortunately, this suggests the need of preprocessing phase which is not necessarily optimal from an attack perspective. Indeed, as this combining function is not automatically learned by the generative model, the Evaluator may not converge towards the optimal solution and reaches Objective 3.3.1.1.

2. **Performance** – While the goal of a side-channel attack is to optimize a learning algorithm which approximates $\mathsf{Pr}[Y|\mathbf{T}]$ in order to discriminate a sensitive variable Y from a set $\mathcal{Y}$, the application of generative models can be considered as suboptimal [NJ02]. However, as computing $\mathsf{Pr}[\mathbf{T}|Y]$ is approximately similar to $\mathsf{Pr}[Y|\mathbf{T}]$ up to a constant independent of the secret key (see Equation 3.4), the Evaluator expects to obtain similar result regardless the estimated conditional probability. This verification is performed in Chapter 6.

One solution to solve the latter issues is to substitute the generative approach by the discriminative approach. Probabilistic discriminative approach captures the decision boundaries between the classes in $\mathcal{Y}$ such that no particular modeling of the leakage trace distribution is performed. It models the conditional posterior probabilities $\mathsf{Pr}\left[Y|\mathbf{T}\right]$ directly in order to discriminate and pick the most likely hypothetical candidate (*i.e.* sensitive information) given a leakage trace. A discriminative model estimates a Θ-parametric probability conditional distribution $\mathsf{Pr}\left[Y|\mathbf{T},\Theta\right]$ that is as similar as possible to the true unknown conditional probability distribution $\mathsf{Pr}\left[Y|\mathbf{T}\right]$. In Subsection 4.2.2, this model is characterized by F_Θ such that the last layer of the neural network is defined by a softmax (resp. sigmoid) activation function if a multiclass (resp. binary) classification problem has to be solved. In deep learning-based side-channel attacks, this approach is beneficial for directly retrieving the secret key manipulated by the cryptographic module without modeling unnecessary information or performing preprocessing phase [MPP16, CDP17a, PHJ+18, CCC+19]. This approach is notably beneficial to automatically find a suited combining function and thus, enhancing the high-order attacks. Furthermore, as the discriminative model directly computes $\mathsf{Pr}\left[Y|\mathbf{T},\Theta\right]$, it generally outperforms generative models at conditional prediction tasks [NJ02]. However, discriminative models need to be configured more carefully than generative approach. Hence, if a discriminative model is not well constructed, a generative solution can be more suitable to optimize the conditional Θ-parametric probability distribution $\mathsf{Pr}\left[Y|\mathbf{T},\Theta\right]$ after the application of the Bayes' Theorem. In the following chapter, we propose to deeply investigate the notion of discriminative approach in order to ease the construction of neural networks in deep learning-based side-channel attacks.

5.5 Conclusion

In this chapter, we bridge the deep learning paradigm with a classical profiled side-channel attack, namely stochastic attack. Hence, we design a new conditional variational autoencoder from the attack proposed by Schindler *et al.* [SLP05] and highlight its similarities with the theoretical foundations of stochastic attacks (see Section 5.2). This development leads us to investigate the ability of the Evaluator to explain and interpret the results provided by the generative model. Indeed, by visualizing the trainable parameters (*i.e.* Θ and ϕ), the Evaluator retrieves the leakage model approximated by the conditional variational autoencoder as well as the points of interest induced in the targeted implementation. Through this process, he can assess the suitability of the resulted model. Then, we validate the benefits of our approach under traditional restrictions, namely low percentage of relevant information and Boolean masking implementations. Finally,

the conditional variational autoencoder based on stochastic attacks is more flexible than classical profiled side-channel attacks. Indeed, while template attacks and stochastic attacks consider the same leakage traces' dimension during the profiling phase and the attack phase, our generative model can conduct the key extraction phase on a smaller dimension (*i.e.* points of interest) and therefore, reduce the impact of uninformative time samples on the attack performance.

WHAT'S NEXT?

However, the conditional variational autoencoder based on stochastic attacks does not automatically combine the leakage function in order to deal with Boolean masking implementations. Thus, a preprocessing phase is still needed for the Evaluator. A solution to this issue is to consider the discriminative approach. As suggested in [NJ02], a discriminative model, that is well designed, converges towards a better solution and classically outperforms a generative learning algorithm. Nevertheless, as no particular correlation has been made with the side-channel context, its configuration is considered as an arduous task. To mitigate this issue, Chapter 6 exploits some visualization tools introduced by the deep learning community, in order to understand how the model hyperparameters affect the ability of the convolutional neural network to retrieve the points of interest. We illustrate their impacts and propose a new convolutional structure for a discriminate perspective.

Chapter 6

Designing Discriminative Models in Profiled Side-Channel Analysis

In this chapter, we develop an approach for evaluating the impact of model hyperparameters induced in convolutional neural networks. First, we recall the restrictions the Evaluateur has to deal with and define some issues that have to be reduced in order to optimize the evaluation process. Thus, we adapt visualization tools introduced in the deep learning literature in order to enhance the understanding of the discriminative models and to reduce the black box issue that characterized it. Then, a focus is made on the hyperparameters which composed the convolutional part of a CNN in order to investigate the impact of the length of the filters, the pooling operation and the number of the convolutional blocks on the ability of the convolutional part to extract the points of interest from a leakage trace. Finally, we design a methodology which reduces the impact of uninformative time samples by focusing the interest of the neural network on the points of interest only. This results in drastically reducing the network complexity as well as the training time without altering the neural network performance. The solutions proposed in chapter have been presented at CHES and published in the journal IACR TCHES [ZBHV19b].

Contents

6.1 Discriminative Models in Certification process

6.1.1 In The Reality of Evaluator's World

As mentioned in Subsection 1.2.2, the robustness of a cryptographic module is assessed by a cotation table which is based on some criteria, namely elapsed time, adversary's expertise, knowledge of the target of evaluation (TOE), access to TOE, equipment and tools. The combination of these criteria provides a score to the Evaluator that reflects the difficulty of conducting an attack. Thus, during the certification process, the Evaluator aims at optimizing these criteria in order to perform the most efficient attack within the shortest time. Indeed, as the elapsed time is the most flexible criterion that the Evaluator has to deal with, he aims at designing discriminative models which approximate the optimal adversary (see Objective 3.3.1.1) with a minimum amount of time.

To design discriminative models, the Evaluator typically considers the fully-connected neural networks or the convolutional neural networks as they have been proven to be effective against Boolean masking implementations [MPP16, MDP19b] and hiding countermeasures [CDP17a, ZOB18, Mag19a, Mag19b, MBC+20, ZOB18, Mag19a, LH20]. While both countermeasures can be mitigated by convolutional neural networks, the rest of this chapter will be focused on its construction. However, designing such neural networks is an arduous task because of the plethora of model hyperparameters[a] that are involved in the construction of the CNNs. Indeed, as mentioned in Subsection 4.2.2, a CNN can be decomposed into two parts, namely feature selection part and classification part, such that the model hyperparameters that characterized them differ:

- Feature selection part – as a remainder, this part extracts the sensitive information (*i.e.* points of interest) from a leakage trace $\mathbf{T}$ in order to help the decision-making. Typically, the Evaluator has to deal with the following model hyperparameters: the number of convolutional blocks, the number of convolutional layers per convolutional block, the number and the length of filters, the activation function applied on each layer, the stride value, the padding value, the pooling operation and the pooling stride.

- Classification part – as a remainder, this part is defined by numerous fully-connected layers characterized by the following model hyperparameters: the number of fully-connected layers, the number of neurons per fully-connected layers and the activation function that has to be applied on each layer.

To deal with the model hyperparameters space, the Evaluator uses tools which automatically tune the model hyperparameters values in order to find the most efficient solution from an adversary's perspective [MPP16, BPS+20, WPP20, PRA20, RWPP21, YAGF21]. However, if the Evaluator has no intuition about how the model hyperparameters values should be bounded, the time needed to find a good predictive model can be tremendously impacted.

Black-box issue. Because a discriminative model can be seen as a black-box tool, the lack of interpretability is a major drawback for constructing a neural network as well as explaining the decision-making of the Θ-parametric model. Hence, in deep learning-based side-channel attack, the configuration of the profiling phase is an arduous task for the Evaluator. One solution to reduce these bounds is to get a better intuition on the decision-making of the predictive models in order to understand how it retrieves the secret key manipulated by a targeted cryptographic module. A common technique to perform such investigations is to visualize the model's behavior that is selected by the learning algorithm.

The following section defines the *gradient visualization* tool [MDP19a] as a solution for assessing the ability of a model to exploit the points of interest, and denotes its limitation regarding the grasp of the model hyperparameters' impact.

[a] We recall that this term has been introduced in Definition 4.2.2.4.

6.1.2 Evaluation of Discriminative Models

As mentioned in Subsection 3.2.2, the signal-to-noise Ratio is a useful tool to identify the time samples that depend on the targeted sensitive variable. However, this solution cannot be considered to evaluate the intrinsic ability of a parametric function F_Θ to retrieve the related points of interest as the SNR tool does not depend on a distinguisher[b]. Alternatively, some visualization tools have been introduced by the deep learning community in order to interpret the results provided by F_Θ.

Gradient visualization. Introduced in [SVZ14], the gradient visualization has been investigated by Masure *et al.* in the side-channel context [MDP19a]. To use this tool, two assumptions are needed. The first assumption[c] states that the points of interest are non-uniformly distributed over a D-dimensional leakage trace $\mathbf{T}$ such that the set $\{l_0, \ldots, l_{u-1}\}$, which defines the coordinates related to each point of interest, has a smaller dimension than D (*i.e.* $u \ll D$). Hence there are only a few time samples that depend on the sensitive variable Y. The second assumption[d] considers that a Θ-parametric model $F_\Theta : \mathcal{T} \to \mathcal{P}(\mathcal{Y})$ is differentiable over $\mathbf{T}$.
To evaluate F_Θ with the gradient visualization tool, the Evaluator computes the derivatives of the model with respect to a leakage trace $\mathbf{T}$ such that the magnitude of the derivatives indicate which time samples need to be modified the least to affect the prediction the most. In other words, following the value of the gradient of F_Θ at the i^{th} time sample, the Evaluator can assess if i is included in the set of points of interest $\{l_0, \ldots, l_{u-1}\}$ such that, for a given sensitive variable Y, he expects that,

$$\frac{\partial}{\mathbf{T}[i]} F_\Theta(\mathbf{T})[Y] \begin{cases} \neq 0 & \text{if} \quad i \in \{l_0, \ldots, l_{u-1}\}, \\ \approx 0 & \text{otherwise.} \end{cases} \tag{6.1}$$

Thus, the gradient visualization is useful for identifying the time samples that influence the most the prediction of F_Θ. Comparing these time samples to the points of interest, the Evaluator is able to assess the suitability of the selected model F_Θ to solve the side-channel classification task which consists in mapping a leakage trace to the true sensitive variable Y. More concretely, the gradient visualization can be considered to:

1. Detect the time samples that the model considers as correlated with the targeted sensitive information Y (see Equation 6.1).

2. Assess the impact of each time sample to solve the classification task. Indeed, for a given time sample i, a gradient with a large magnitude ensures that the information included at i has a huge impact on the PMF and consequently, on the classification task.

Thus, comparing the gradient visualization with the signal-to-noise ratio is a relevant solution to the Evaluator in order to interpret the suitability of the model F_Θ for retrieving the time samples which affect the most its predictions. However, this tool highlights the impact of time sample's variation on the final decision, but not the decision-making itself. Consequently, the *layer-wise relevance propagation* [BBM+15] have been introduced in order to mitigate this issue by directly bridging the visualization tool with the decision of F_Θ. In [HGG20], Hettwer *et al.* propose a comparison between the gradient visualization, the layer-wise relevance propagation and the occlusion approach[e] such that no clear benefit has emerged for one of those methods. As the gradient visualization tool provides some theoretical coherence with the side-channel context, this manuscript is only focused on the gradient visualization tool for the evaluation of the overall model's behavior F_Θ.

[b] Following Definition 3.3.1.3, a parametric function F_Θ can be defined as a distinguisher.
[c] This assumption is called *Sparsity* in [MDP19a].
[d] This assumption is called *Regularity* in [MDP19a].
[e] The occlusion technique consists in removing some time samples from the leakage trace in order to observe how it impacts the model's predictions.

Limitations. However, these techniques are not ones required to understand how the feature selection part or the convolutional part affects the decision process as they only give a general interpretation of how a neural network performs. One solution to reduce this issue is to find local visualization tools that evaluate the impact of model hyperparameters to retrieve the points of interest.

The following section adapts two visualization tools widely used in deep learning, namely *weight visualization* and *heatmaps*. Through the application of these approaches, we want to get a better understanding of the model hyperparameters which characterized the feature selection part of a convolutional neural network.

6.1.3 Evaluation of the Feature Selection

One way to interpret the suitability of a neural network to solve a classification task is to visualize how the configuration of its model hyperparameters affects the points of interest's selection.

VISUALIZATION'S GOAL

As mentioned in Subsection 4.2.2, we assume that a convolutional neural network can be decomposed into a feature selection part, which extracts the points of interest from a leakage trace, and a classification part, which combines those relevant time samples in order to make a decision regarding the classification task to solve. In this chapter, we assume that, once the neural network adequately retrieves the points of interest, the classification part can be easily designed. In addition, compared to the classification part, the configuration of the feature selection part is the most arduous task due to the plethora of model hyperparameters (see Subsection 6.1.1). Thus, adapting some visualization tools to the side-channel context can be beneficial to highlight the impact of those hyperparameters to retrieve the points of interest and consequently, ease the design of convolutional neural networks for the Evaluator as well as reducing the *elapsed time* criterion.

Thus, the Evaluator has to consider visualization tools which allow an independent interpretation of the feature selection part, *i.e.* leave apart the classification component of the convolutional neural network, such that he can evaluate the *Network Confidence* to retrieve the points of interest.

Definition 6.1.3.1. [Network Confidence] Given a leakage trace $\mathbf{T}$ at a time sample i, the Network Confidence defines the capacity of the neural network to retrieve a point of interest at $\mathbf{T}[i]$.

Hence, the higher the network confidence, the higher the probability that the related point of interest will be exploited in the classification part. One common strategy is to visualize the trainable weights to interpret the decision-making of the convolutional neural network.

Weight Visualization as Leakage Detection. This method was introduced in [BPK92] as a useful framework when dealing with spatial information. During the training process, the neural network evaluates influential neurons which can be helpful to address the classification task by attributing large weight values. In deep learning, this technique is usually used to evaluate the patterns extracted by the first layers of a deep architecture [HOWT06, LBLL09, OH08, HOT06]. Therefore, we propose to adapt the weight visualization tool in order to evaluate the feature selection part of a convolutional neural network. As defined in Subsection 4.2.2, the flatten operation characterizes the border between the feature selection part and the classification part of a convolutional neural network. Thus, visualizing the weights at the output of this operation is beneficial to assess the ability of the feature selection part to capture the expected points of interest. As remainder, the flatten operation concatenates each intermediate leakage trace following an axis in order to reduce the space dimension of the feature selection part to a 1-D space that fits with the classification part. By concatenating the output of the flatten layer

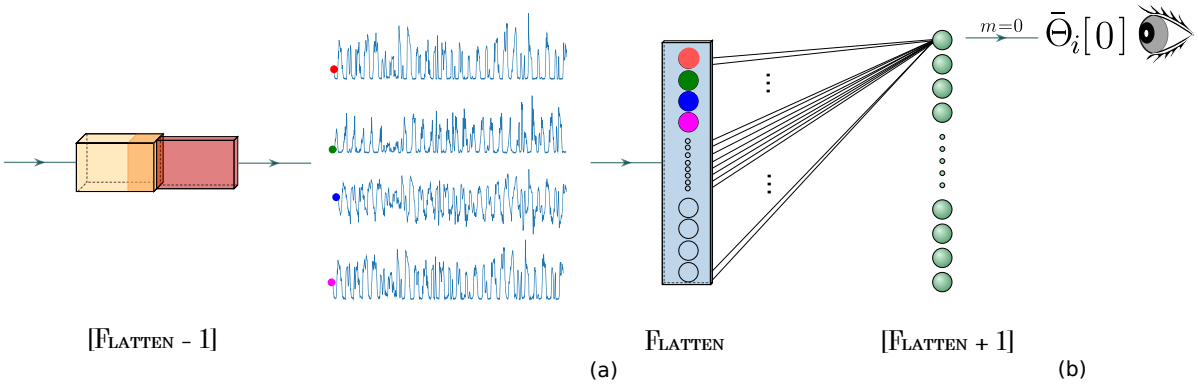


Figure 6.1: Methodology for weight visualization ((a) - concatenation of each intermediate trace following their temporal axis ; (b) - computation of $\bar{\Theta}[0]$ to get a temporal correspondence).

following the columns (see Equation 4.9), we retain timing information to be able to interpret the neurons considered as relevant for the decision-making (see Figure 6.1a). Once the training process is performed, the neurons where the sensitive variable leaks will be evaluated with high weights if the feature selection part effectively extracts the points of interest. Thus, by visualizing the weight values of the flatten layer, the Evaluator can understand which neurons have a positive impact on the classification and hence, thanks to the *feedforward propagation*, we can interpret which time samples influence the most the model F_Θ.
Let $N^{[f+1]}$ be the number of neurons in the layer following the flatten operation, $N_{\text{filt.}}^{[f-1]}$ be the number of filters in the last convolutional blocks and $dim(\mathbf{T}^{[f-1]})$ be the dimension associated with each intermediate leakage trace after the last pooling layer. Let $\Theta^{[f+1]}$ be the weights corresponding to the layer indexed after the flatten operation such that $\Theta^{[f+1]} \in \mathcal{M}_{dim(\mathbf{T}^{[f-1]})\times N_{\text{filt.}}^{[f-1]}, N^{[f+1]}}(\mathbb{R})$.
Let $\bar{\Theta} \in \mathcal{M}_{dim(\mathbf{T}^{[f-1]}), N^{[f+1]}}(\mathbb{R})$ be a matrix that enables visualization of the weights such that $\bar{\Theta}[m]$, which is associated with the m^{th} neuron of the layer following the flatten operation, can be computed as follows:

$$\bar{\Theta}_i[m] = \frac{1}{N_{\text{filt.}}^{[f-1]}} \sum_{j=i\times N_{\text{filt.}}^{[f-1]}}^{(i+1)\times N_{\text{filt.}}^{[f-1]}} \left|\Theta_j^{[f+1]}[m]\right|,$$

where $i \in [0, dim(\mathbf{T}^{[f-1]})]$.
Then, let $\bar{\Theta} \in \mathbb{R}^{dim(\mathbf{T}^{[f-1]})}$ be a vector that enables visualization of the mean weight related to each neuron of the layer $[f+1]$ such that:

$$\bar{\Theta}_i = \frac{1}{N^{[f+1]}} \sum_{m=0}^{N^{[f+1]}} \bar{\Theta}_i[m], \tag{6.2}$$

where $i \in [0, dim(\mathbf{T}^{[f-1]})]$.
In other words, we first reduce the dimension of $\Theta^{[f+1]}$ so that it corresponds to the temporal space defined by $dim(\mathbf{T}^{[f-1]})$. This reduction is obtained by averaging the weights related to the same time sample (see Figure 6.1b). Finally, to obtain a global overview of feature selection part, the Evaluator has to compute the average of the weights associated with each neuron in order to evaluate the confidence of the network in revealing the points of interest.
If the feature selection part retrieves a point of interest on a time sample i, the related weight will be higher than 0. Depending on its magnitude, the network will be more or less confident in its detection. If the network is extremely confident in the information included at i, the related weight will be high and the exploitation of the sensitive information will be easy. On the other

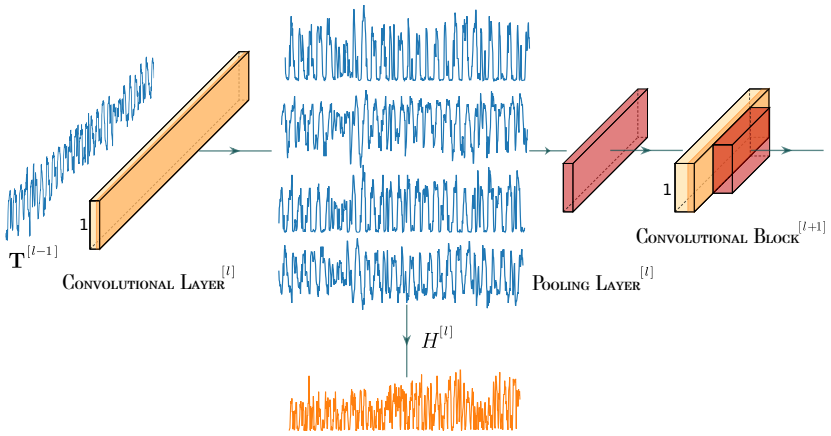


Figure 6.2: Heatmap

hand, if the weight related to i is low, *i.e.* similar to the weight related to uninformative time samples, the neural network will be distrustful on the leakage detection. Hence, the resulted exploitation will be harder or inexistent and the training time could increase. If necessary, this visualization can be performed during the training process in order to evaluate the suitability of the selected model F_Θ.
However, even if the weight visualization is a useful tool to get a global overview of the feature selection part, the Evaluator still cannot investigate the sample extracted by each filter. One solution is to analyze the activation generated by the convolution operation between an input layer and each filter.

HeatMaps. Introduced in [ZF14], heatmaps (or feature maps) help to understand and interpret the role of each filter in a convolutional neural network. We apply this technique in the side-channel context as a new tool to interpret the impact of filters so that we can adapt the neural network according to how the time samples are selected.
Let us denote $N_{\text{filt.}}^{[l]}$ the number of filters in the l^{th} layer of the convolutional part, $\mathbf{T}^{[l]}$ the input(s) associated with the l^{th} convolutional block such that $\mathbf{T}^{[0]}$ corresponds to the leakage trace $\mathbf{T}$. Then, the convolution operation between $\mathbf{T}^{[l]}$ and the $N_{\text{filt.}}^{[l]}$ filters returns $N_{\text{filt.}}^{[l]}$ intermediate traces. We denote $H^{[l]}$ the heatmap (or feature map) associated with the l^{th} layer such that:

$$H^{[l]} = \frac{1}{N_{\text{filt.}}^{[l]}} \sum_{i=0}^{N_{\text{filt.}}^{[l]}} (\mathbf{T}^{[l]} \circledast W_{\Theta,i}^{[l]}),$$

where $W_{\Theta,i}^{[l]}$ is the i^{th} filter of the l^{th} layer of the feature selection part and it is composed by the Θ trainable parameters.
Thanks to this visualization technique, we can assess which neurons are *activated*[f] by the filters of each convolutional layer and understand how the time samples are selected (see Figure 6.2). Indeed, the resulted convolution operation reflects the time samples that are considered as relevant during the classification task. The comparison of this heatmap with the SNR peaks is helpful to investigate the ability of the network to find the points of interest. This result will be verified in the experimental section (see Section 6.2).

The weight visualization and the heatmaps are two tools that help the Evaluator to explain and interpret the feature selection part more clearly. By using these techniques to understand the internal layers of a CNN, it is possible to define a method to build suitable neural network architectures in the presence of synchronization and desynchronization effect.

[f]In this manuscript, the term *activation* refers to the output of a convolutional operation.

The following section analyzes the effect of the most influencing model hyperparameters (*i.e.* the length of the filters, the pooling operation and the number of convolutional blocks) on the extraction of the points of interest.

6.2 Impact of Hyperparameters on the Leakage Detection

To illustrate and explain the effect of each hyperparameters as far as possible, we consider the Chipwhisperer dataset introduced in Section 3.6. To train the learning algorithm, we use the negative log-likelihood as loss function (see Definition 4.3.2.2) as it is generally considered in deep learning based side-channel analysis [MPP16, CDP17a, PHJ+18, CCC+19, MDP19b, BPS+20]. Further discussion on this loss function will be provided in Chapter 7. As mentioned in Subsection 4.3.2, the optimizing algorithm used in this manuscript is the *Adam* optimizer [GBC16]. A batch-size of 50 is configured. As explained in Subsection 4.2.1, we use the SELU activation function to mitigate the vanishing and exploding gradient issues. The loss function, the optimizer and the activation function have been selected following a *grid search optimization* [GBC16] such that these optimizer hyperparameters[g] have been selected from a finite set of values (see Table 6.1). Our choices are motivated by the fact that they provide good results, in terms of classification, on our datasets. Indeed, using these hyperparameters lead to a faster convergence towards a constant guessing entropy of 1.

Table 6.1: Grid search optimization on hyperparameters

	Values
Optimizer	{SGD, RMSprop, Adam}
Activation function	{tanh, ReLU, ELU [CUH16], SELU}
Learning Rate	One-Cycle policy (see Section 6.3)
Batch size	$\{50, 64, 128, 256\}$
Epochs	$\{10, 20, 25, 50, 75, 100, 125, 150\}$
$N_{\textbf{FC layers}}$	$\{2, 5, 10, 15, 20, 25, 100\}$
$N_{\textbf{nodes per FC layers}}$	$\{1, 2, 3, 4, 5\}$

Finally, to accelerate the learning phase, we pre-process the data such that all trace samples are *standardized* (unit variance, zero mean) and *normalized* between 0 and 1 [GBC16]. In this section, we only focus on the points of interest extraction made by the feature selection part and not on the performance of the selected model F_Θ.

Remark 6.2.1. As defined in Subsection 4.3.2, the learning rate is one of the most challenging hyperparameters to tune because it influences the time needed to train an algorithm and monitors the risk of vanishing and exploding gradient. In this section, we configure the learning rate by using a technique called *One Cycle Policy* [ST17, Smi17, Smi18] that helps choose a suitable hyperparameter value. Surprisingly, using this policy means we can choose much higher learning rates and significantly reduce training time while preventing overfitting.

6.2.1 Length of Filters

In this section, we demonstrate that increasing the length of the filter causes *entanglement* and reduces the network confidence in the detection of the points of interest. Let $W_\Theta \in \mathbb{R}^n$ be a filter of length n such that $W_\Theta = \{\theta_0, \theta_1, ..., \theta_n\}$ is optimized for maximizing the detection of a set

[g]This term has been defined in Definition 4.3.2.5.

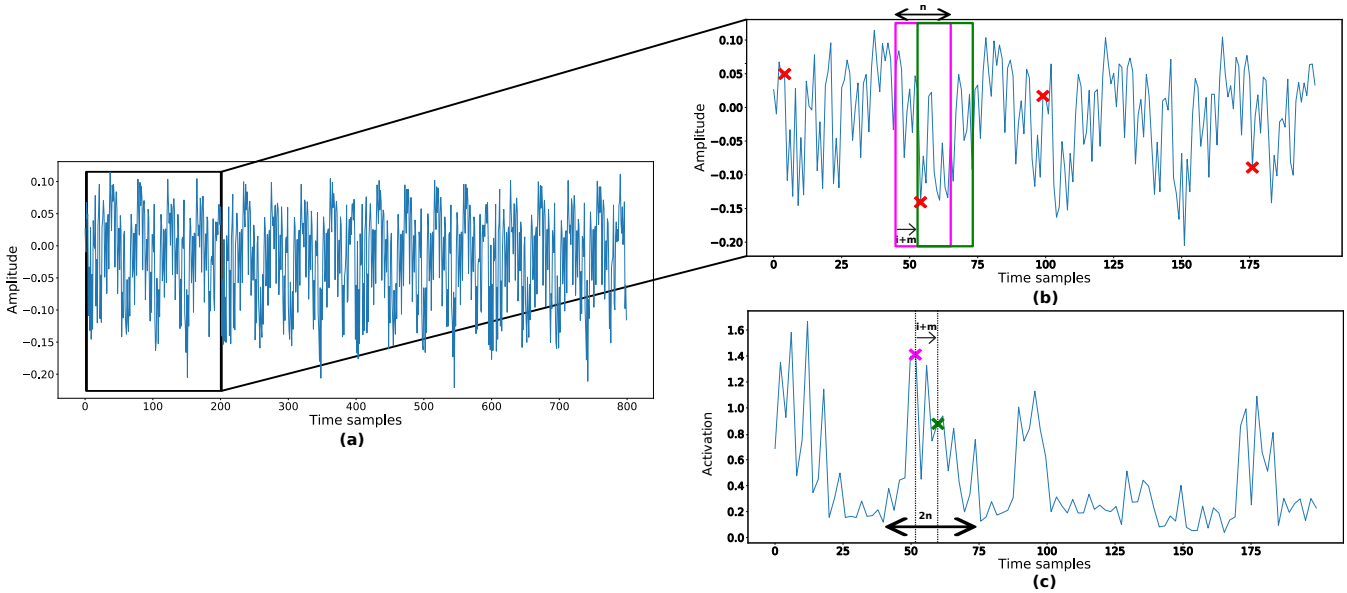


Figure 6.3: Convolution operation between a ChipWhisperer leakage trace $\mathbf{T}$ and a filter W_Θ of size n ((a) - a leakage trace; (b) - zoom on a leakage trace's portion which includes points of interest that are characterized by red crosses; (c) - Heatmap between $\mathbf{T}$ and W_Θ)

$\{l_0, \ldots, l_{u-1}\}$ of u points of interest. Convolution operation between a trace $\mathbf{T}$ and a filter W_Θ follows the equation:

$$(\mathbf{T} \circledast W_\Theta)[i] = \sum_{j=0}^{n} \left(\mathbf{T}\left[j + i - \frac{n}{2} \right] \times W_\Theta[j] \right),$$

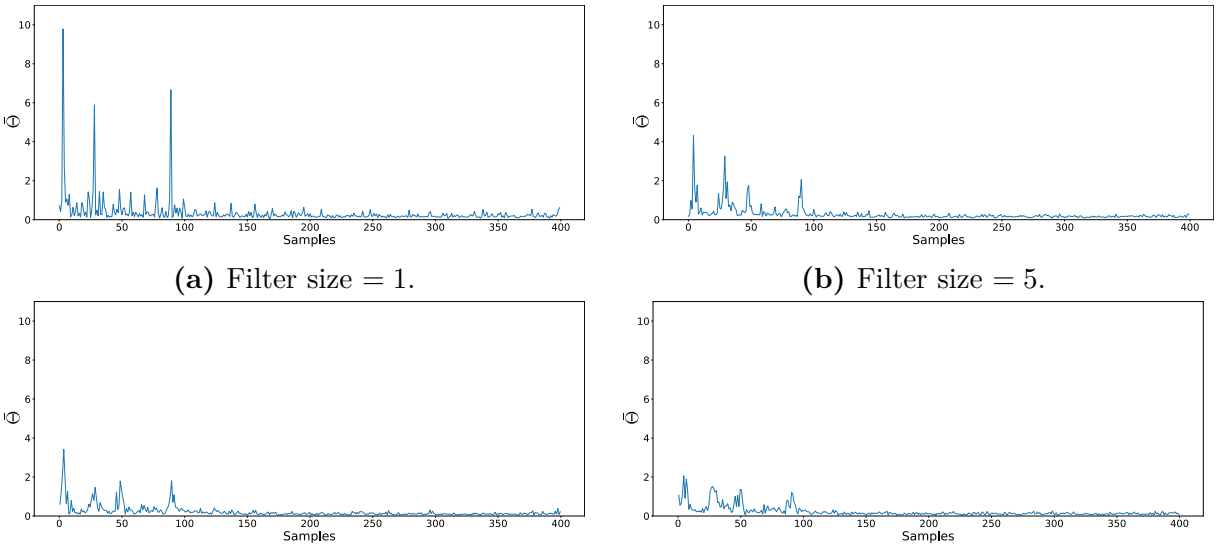
such that,

$$\mathbf{T}\left[j + i - \frac{n}{2} \right] \times W_\Theta[j] = \begin{cases} \theta_j \times \mathbf{T}\left[j + i - \frac{n}{2} \right] & \text{if} \quad (j + i - \frac{n}{2}) \in \{l_0, \ldots, l_{u-1}\}, \\ \epsilon & \text{otherwise.} \end{cases} \tag{6.3}$$

where $\epsilon \approx 0$ and θ_j denotes the weight related to the index j (*i.e.* $W_\Theta[j]$).
Denote $\mathbf{T}$ a leakage trace such that $(\mathbf{T} \circledast W_\Theta)[i]$ covers the u relevant time samples $\{l_0, \ldots, l_{u-1}\}$ and $(\mathbf{T} \circledast W_\Theta)[i+m]$ covers less than u of those time samples. Because W_Θ maximize the detection of the u PoIs, then,

$$(\mathbf{T} \circledast W_\Theta)[i] > (\mathbf{T} \circledast W_\Theta)[i + m].$$

Figure 6.3 provides an example of this operation such that four points of interest can be extracted from the leakage trace $\mathbf{T}$ (see red crosses in Figure 6.3b). In this figure, we show the result of two convolution operations $(\mathbf{T} \circledast W_\Theta)[i]$ (see purple cross) and $(\mathbf{T} \circledast W_\Theta)[i + m]$ (see green cross) that share the same information related to a point of interest (see Figure 3.9b). Consequently, when entanglement occurs the convolved samples share the same relevant information. Therefore, in our example, the PoI is spread over the two convolved samples. By increasing the size n of the filters, we increase the entanglement, consequently, more PoIs are shared between these convolved points. Thus, for a given filter $W_\Theta \in \mathbb{R}^n$, the point of interest is spread over $2n$ convolved samples. If the Evaluator wants to precisely define the temporal space where the points of interest occur, we recommend minimizing the length of the filters in order to reduce the risk of entanglement and ease the training phase.



(a) Filter size = 1. **(b)** Filter size = 5.

(c) Filter size = 11. **(d)** Filter size = 15.

Figure 6.4: Impact of the filter length on the weights $\bar{\Theta}$ introduced in Equation 6.2.

We illustrate this property by considering different scenarios on the ChipWhisperer dataset with a fixed neural network architecture[h]. To efficiently evaluate the impact of the length of the filter, we select a wide range of lengths (*i.e.* in $\{1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 50, 75, 100\}$) and visualize the weights $\bar{\Theta}$ to experimentally verify the previous claim. The shorter the length, the less the informative time samples are spread over the convolved traces. In addition, comparing the results provided by the SNR (see Figure 3.9b) with Figure 6.4 shows that the weight visualization succeeds in recovering the location of the points of interest. The number of samples is divided by 2 because of the pooling stride value (*i.e.* 2) that we used in the first convolutional block. Thus, the weight visualization is not directly linked with the temporal instants of a leakage trace $\mathbf{T}$. However, as the weight visualization is introduced to complement the gradient visualization, it is mainly useful to evaluate the impact of the model hyperparameters.
Through Figure 6.4, small filters locate information better because the related points of interest are not shared with a lot of convolved samples, while using larger filters causes entanglement and provide the detection of "*global features*". The latter exploitation could be difficult during the classification part because the same information is spread over the leakage trace and the informative weights are almost similar to the noisy convolved samples. Hence, the network can be less confident on the detection of the resulted PoI and the exploitation phase can be affected. Indeed, if the weight related to the PoI is 0, the classification part cannot extract the relevant information provided by this sample. Increasing the entanglement can cause a dramatic impact on the leakage retrieval. Thus, the resulted training time needed to reach the final performance could dramatically increase. Another approach consists in linking the length of the filters with the number of trainable parameters such that the more the number of weights, the longer the training time. By reducing the entanglement, we can accelerate the training process without significantly degrading the final performance as the points of interest are extracted (see Figure 6.4a). If the feature selection part can capture the dependence between the leakage trace and the sensitive information with small filters, increasing its length does not bring further useful information for resolving the classification task.

Limitations of small filter size. Questioned by Wouters *et al.* in [WAGP20], using small filters have some limitations related to the *robustness*[i] of the training phase. They suggest this

[h] We recall that this term has been introduced in Definition 4.2.2.2.

[i] Here, the robustness refers to the ability of the neural network to provide the same performance while being trained on datasets which partially differ.

observation based on the *Receptive Field* [LLUZ16, IFLF+20] which is similar to the entanglement [ZBHV20]. Therefore, the concept of entanglement helps us to better understand in which context larger filters can provide more stability. Indeed, increasing the length of the filters spread the relevant information over multiple convoluted samples. Hence, multiple weights are linked to the same PoI. During the training phase, the resulted network could be less disrupted to retrieve the sensitive value because there are multiple ways to extract the point of interest. However, through Subsection 6.3.1, we will experimentally show that small filters seem not widely impacted by this issue.

Once the Evaluator sets his convolutional layer, he has to configure the pooling operations to consider in order to complete the construction on a convolutional block. The following section is focused on the impact of the common pooling operations on the points of interest detection.

6.2.2 Pooling Operations

Introduced in Subsection 4.2.2, the pooling layer aims at reducing the dimension of the output of the convolutional layer (*i.e.* convolved leakage trace) while preserving the relevant information extracted by the filters. However, depending on the pooling operation the Evaluator considers, some distinctions can be highlighted. In this section, we compare the two widely used pooling operations, namely average pooling and maxpooling, that we briefly introduced in Subsection 4.2.2.

Average pooling operation. Let $\mathbf{T}^{[l]}$ be the input of the l^{th} convolutional block and $W_{\Theta}^{[l]} \in \mathbb{R}^n$ a filter related to the same convolutional block. Let $ps^{[l]}$ be the pooling stride related to the l^{th} convolutional block. Assume that the j^{th} convolved sample $(\mathbf{T}^{[l]} \circledast W_{\Theta}^{[l]})[j]$ covers $u_1^{[l]}$ relevant information $\{l_0, \ldots, l_{u_1^{[l]}-1}\}$ and includes the sample with the most of relevant information $l^*_{\text{opt.}}$ on the targeted sensitive variable. The resulted $ps^{[l]}$ convolved samples are covered by the i^{th} pooling sample $AvgPool(\mathbf{T}^{[l]} \circledast W_{\Theta}^{[l]})[i]$ such that,

$$
\begin{aligned}
AvgPool^{[l]}(\varrho(\mathbf{T}^{[l]} \circledast W_{\Theta}^{[l]})))[i] &= \sum_{j=i}^{i+ps^{[l]}} \frac{\varrho(\mathbf{T}^{[l]} \circledast W_{\Theta}^{[l]})[j]}{ps^{[l]}} \\
&= \frac{1}{ps^{[l]}} \sum_{j=i}^{i+ps^{[l]}} \varrho\left(\sum_{k=0}^{n} \mathbf{T}^{[l]}\left[k+j-\frac{n}{2}\right] \times W_{\Theta}^{[l]}[k]\right) \\
&\approx \frac{1}{ps^{[l]}} \sum_{j=i}^{i+ps^{[l]}} \varrho\left(\sum_{\substack{k=0 \\ (k+j-\frac{n}{2}) \in \{l_0, \ldots, l_{u_1^{[l]}-1}\}}}^{n} \left(\theta_k^{[l]} \times \mathbf{T}^{[l]}\left[k+j-\frac{n}{2}\right]\right)\right).
\end{aligned}
$$

Because $(k+j-\frac{n}{2}) \in \{l_0, \ldots, l_{u_1^{[l]}-1}\}$, $\theta_k^{[l]} \times \mathbf{T}^{[l]}[k+j-\frac{n}{2}] > 0$ is true only for the $u_1^{[l]}$ points of interest it covers (see Equation 6.3). Following the SELU function (see Equation 4.8), we can simplify our solution such that $\varrho(x) = \lambda \cdot x$.

On the other hand, assume that the $(i+m)^{\text{th}}$ pooling sample $AvgPool(\mathbf{T}^{[l]} \circledast W_{\Theta}^{[l]})[i+m]$ covers $(u_1^{[l]} + u_2^{[l]})$ points of interest $\{l_0, \ldots, l_{u_1^{[l]}} + l_{u_2^{[l]}} - 1\}$ such that it also includes $l^*_{\text{opt.}}$. It this setting, we assume that its $u_1^{[l]}$ points of interest are shared with $AvgPool(\mathbf{T}^{[l]} \circledast W_{\Theta}^{[l]})[i]$. Thus, from similar approach, we can observe that,

$$
AvgPool^{[l]}(\varrho(\mathbf{T}^{[l]} \circledast W_{\Theta}^{[l]})))[i+m] \approx \frac{\lambda}{ps^{[h]}} \sum_{j=i+m}^{i+m+ps^{[l]}} \sum_{\substack{k=0 \\ (k+j-\frac{n}{2}) \in \{l_0, \ldots, l_{u_1^{[l]}} + l_{u_2^{[l]}} - 1\}}}^{n} \left(\theta_k^{[l]} \times \mathbf{T}^{[l]}\left[k+j-\frac{n}{2}\right]\right).
$$

As previously mentioned, $\theta_k^{[l]} \times \mathbf{T}^{[l]}[k+j-\frac{n}{2}] > 0$ is true only for the $(u_1^{[l]} + u_2^{[l]})$ points of interest it covers (see Equation 6.3), *i.e.* when $(k+j-\frac{n}{2}) \in \{l_0, \ldots, l_{u-1}\}$. Therefore,

$$AvgPool^{[l]}(\varrho(\mathbf{T}^{[l]} \circledast W_\Theta^{[l]})))[i+m] - AvgPool^{[l]}(\varrho(\mathbf{T}^{[l]} \circledast W_\Theta^{[l]})))[i] = \frac{\lambda}{ps^{[h]}} \sum_{j=i+m}^{i+m+ps^{[l]}} \sum_{\substack{k=0 \\ (k+j-\frac{n}{2}) \in \{l_{u_1^{[l]}}, \ldots, l_{u_1^{[l]}+l_{u_2^{[l]}}-1}\}}}^{n} \left(\theta_k^{[l]} \times \mathbf{T}^{[l]}\left[k+j-\frac{n}{2}\right]\right).$$

Therefore, using the average pooling operation preserves the unshared information when two consecutive overlapped pooling operation are performed which is beneficial to keep the maximum amount of information related to the targeted sensitive variable.

MaxPooling operation In the case where the MaxPooling is used, then,

$$\begin{aligned} MaxPool^{[l]}(\varrho(\mathbf{T}^{[l]} \circledast W_\Theta^{[l]})))[i] &= \max_{j \in \{i, \ldots, i+ps^{[h]}\}} \left(\varrho(\mathbf{T}^{[l]} \circledast W_\Theta^{[l]})[j]\right) \\ &= l^*_{\text{opt.}}, \end{aligned}$$

and,

$$\begin{aligned} MaxPool^{[l]}(\varrho(\mathbf{T}^{[l]} \circledast W_\Theta^{[l]})))[i+m] &= \max_{j \in \{i+m, \ldots, i+m+ps^{[h]}\}} \left(\varrho(\mathbf{T}^{[l]} \circledast W_\Theta^{[l]})[j]\right) \\ &= l^*_{\text{opt.}}. \end{aligned}$$

Therefore,

$$MaxPool^{[l]}(\varrho(\mathbf{T}^{[l]} \circledast W_\Theta^{[l]})))[i+m] - MaxPool^{[l]}(\varrho(\mathbf{T}^{[l]} \circledast W_\Theta^{[l]})))[i] = 0.$$

When MaxPooling is used, if two consecutive pooling computation share the same optimal leakage $l^*_{\text{opt.}}$, then this information is spread over the pooling samples. Furthermore, because $l^*_{\text{opt.}}$ is the only sample selected, all other leakages are discarded. Thus, if the Evaluator considers the maxpooling operation, he can lose information that seems less important while it is essential for solving the classification task. For this reason, we recommend using average pooling as much as possible because no relevant points are discarded. This observation is in accordance with [WP21] that experimentally evaluate the impact of the pooling hyperparameters in a wide range of scenarios. Therefore, we will only consider the average pooling operation in the rest of the manuscript.

Once the Evaluator selects the adequate pooling operation to design the feature selection part, he has to define the number of convolutional blocks[j] that constitutes the chosen convolutional neural network. The following section concludes our investigation by evaluating the impact of the number of convolutional blocks on the extractions of the points of interest.

6.2.3 Number of Convolutional Blocks

In this section, we provide an interpretation of the number of convolutional blocks. As shown in Subsection 6.2.2, using average pooling has the advantage that unshared information is preserved. However, when a large pooling stride is configured, the network confidence on the relevant information is reduced because the information is divided by pooling stride value after each convolutional block. Consequently, the related weights become similar to those of uninformative samples.

[j] As remainder, in this manuscript, a convolutional block is characterized by one convolutional layer followed by a pooling layer.

Furthermore, increasing the number of convolutional blocks reduces the trace dimension while preserving relevant information associated with the leakages. Indeed, in most cases, $dim(\mathbf{T}^{[l]}) = \frac{dim(\mathbf{T}^{[l-1]})}{ps^{[l-1]}}$. Thus, for each additional convolutional block, the distance between the relevant points is reduced by a factor which depends on the pooling stride value. Consequently, if the Evaluator has to deal with a cryptographic module, in which hiding countermeasures are implemented, adding convolutional blocks is useful to reduce the desynchronization effect, and thus, retrieving the points of interest is eased. By adding more convolutional blocks, we can drastically reduce the impact of desynchronization by choosing an appropriate pooling stride $ps^{[l]}$.

To illustrate this phenomenon, we visualize the weights $\bar{\Theta}$ associated with four parametric models F_Θ configured with different numbers of convolutional blocks. Indeed, to efficiently evaluate its impact, we select a wide range of numbers of the convolutional blocks (*i.e.* in $\{1, 2, 3, 4, 5\}$) to experimentally verify the previous claim. The shorter the number of convolutional blocks, the less the informative time samples are blurred. Each convolutional block is configured with one convolutional layer composed by two filters of size 1 and one average pooling layer with a pooling stride of 2 for each convolutional block. The parametric models are trained on the Chipwhisperer dataset and the obtained result are illustrated in Figure 6.5. As expected, the model with 1 convolutional block seems to be more confident in its detection as large weight values are assigned on the related points of interest. However, as we can see, the deeper the network, the less confident it is in its feature detection. In the presence of desynchronization, the Evaluator needs to find a trade-off between the detection of the desynchronization effect and the preservation of the maximum amount of information related to the points of interest.

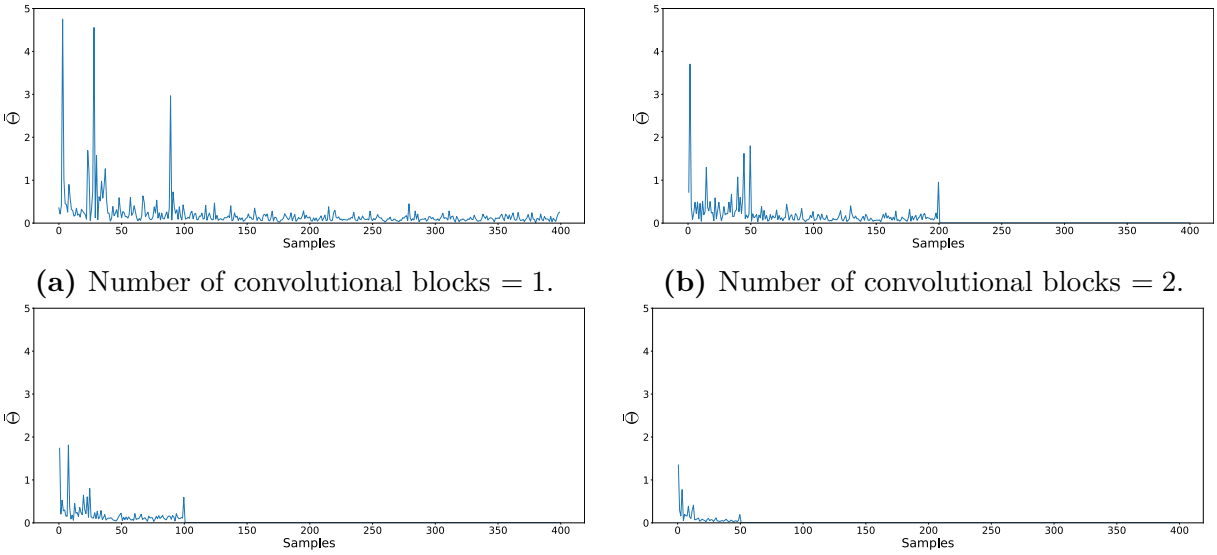
SUM UP...

Through this section, we validate our assumption which suggests that the feature selection part aims at extracting the points of interest from the leakage traces. Introduced as a complement of the gradient visualization, the weight visualization tool and the heatmaps have been defined to explain and interpret the impact of model hyperparameters that composed the feature selection part. Their application allows us to define some guidelines that the Evaluator can follow in order to design a convolutional neural network. Our observations can be summarized as follows:

- **Length of the filters** – As illustrated in Figure 6.4, smaller filters tend to identify local features, which is expected in side-channel context, while larger filters focus on global features. In addition, increasing the length of the filters spreads the relevant information over convolved samples. This phenomenon causes a larger training time as the weights related to informative samples converge towards those of uninformative samples. Thus, using fitlers with small length seem favourable from an attack perspective.

- **Pooling operations** – While the average pooling captures all the points of interest induce in a leakage trace, the maxpooling operation only keeps the time sample which provides the most of information regarding the targeted sensitive variable. Thus, considering the latter operation can be problematic from a performance perspective because some points of interest can be discarded which is an undesired situation. Hence, even if the average pooling is impacted by uninformative samples, we suggest the Evaluator to consider the it instead of the maxpooling in order to mitigate this issue.

- **Number of convolutional blocks** – Because of the pooling operations, increasing the number of convolutional block reduces the confidence of network in its ability to extract the sensitive information. However, thanks to the pooling stride value, adding convolutional block is beneficial to reduce the desynchronization effect. Thus, the Evaluator has to find a trade-off between the network confidence and the reduction of the desynchronization effect.



(a) Number of convolutional blocks = 1.

(b) Number of convolutional blocks = 2.

(c) Number of convolutional blocks = 3.

(d) Number of convolutional blocks = 4.

Figure 6.5: Impact of the number of convolutional blocks on the weights $\bar{\Theta}$ introduced in Equation 6.2.

From the observations provided in this section, we propose a methodology for designing suitable feature selection part on a set of desynchronized leakage traces which implement a random delay effect. But first, we propose a solution when the Evaluator has to deal with a set of synchronized leakage traces.

6.3 Methodology for CNN Architectures

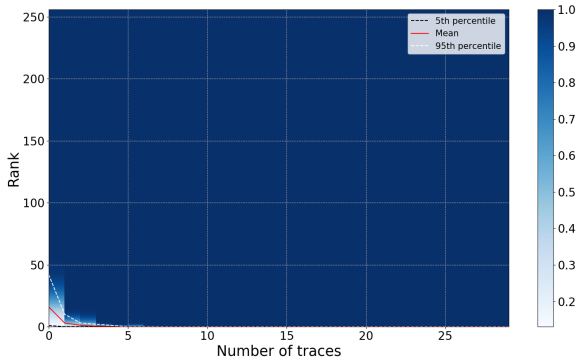
6.3.1 Application on Synchronized Traces

Methodology. According to the observations made in Subsection 6.2.3, adding convolutional blocks reduces the distance between the samples in order to ease the detection of the desynchronization effect. When the Evaluator wants to build a convolutional neural network such that leakage traces in the training, validation and test sets are synchronized, he does not need to configure more than one convolutional block. Indeed, using more than one convolutional block has two major drawbacks. First, in the case when the points of interest are temporally close to each other, adding convolutional blocks increases the risk of entanglement. As we showed in Subsection 6.2.1, entanglement can generate a spreading of relevant information through the convolved time samples and thus, reducing the network confidence (see Definition 6.1.3.1). Secondly, because of the pooling operation (see Subsection 6.2.2), some information can be lost: the same most relevant feature can be spread over the convolved samples (*i.e.* maxpooling) or each relevant information can be reduced following the pooling stride value (*i.e.* average pooling). Thus, when no desynchronization occurs, we recommend setting the number of convolutional blocks to 1 in order to ease the detection of the points of interest.
In [MPP16] and [BPS+20], the authors show that the fully-connected neural networks can be a good alternative when the Evaluator wants to build a suitable architecture for classifying a set of synchronized leakage traces. However, fully-connected neural networks are only composed of fully-connected layers. Thus, considering a convolutional neural network, which is viewed as a FCNN where each neuron of the layer l is linked with a reduced set of neurons of the layer $l-1$ [Kle17], is beneficial in side-channel context as only a few samples are needed for decision-making. Using a CNN with short filter helps to focus its interest on local perturbations and dramatically reduces the complexity of the neural network. It is thus recommended to use CNNs

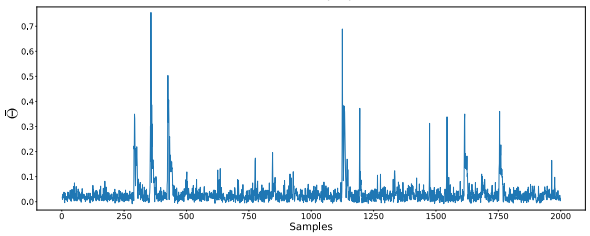
with the shortest filters possible (*i.e.* , 1). In [WAGP20], Wouters *et al.* suggest that considering filters of size 1 perform an additional preprocessing to the leakage traces in order to enhance the points of interest extraction. Finally, the number of filters and the dense layers that compose the classification part should be managed depending on the device under test.

DPA contest-v4. DPA contest-v4 is the easiest dataset because we consider it without countermeasures, consequently the relevant information leaks a lot. Thus, it is straightforward for the network to extract sensitive variables. To generate the convolutional neural network, we consider only two filters of size 1 and compose the classification part with one dense layer of 2 nodes. Finally, we set our learning rate to 10^{-3}. The network is trained for 50 epochs with a batch size of 50.

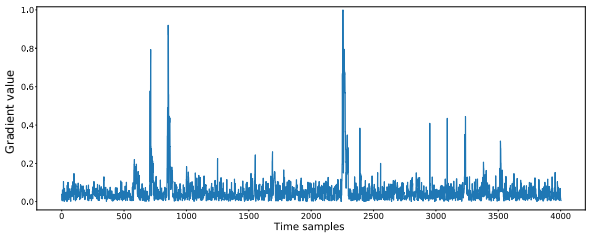
Visualization tools help evaluate the training process (see Figure 6.6). As explained in Subsection 6.1.3, visualization of the weights shows that the feature selection part is accurately set because the points of interest are correctly extracted. Indeed, comparing Figure 6.6b with the SNR computation (see Figure 3.10b) is helpful to verify this observation. Furthermore, by visualizing the gradient (see Figure 6.6c), we can consider that the classification part is also working accurately because the points of interest exploited at the end of the network are the same as those recognized by the feature selection part. Thus, no information is lost between the convolutional part and the end of the network.



(a) Partial security graph over 100 attacks.



(b) Weight visualization.



(c) Gradient visualization.

Figure 6.6: Evaluation of the DPA contest-v4 dataset.

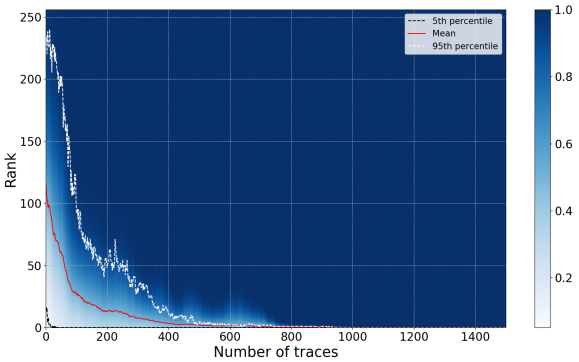
Then, we compare the resulted performance with [PHJ+18] in which the authors published the best performance on DPA contest-v4 dataset using deep learning techniques. The results related to $\bar{N}t_{\text{rank}}$ seem to be similar to those we obtain. However, when we compare the related network complexity, we observe a non-negligible improvement when our methodology is considered. Indeed, the complexity associated with our network is 6 times lower than the previous most powerful model. By reducing the complexity of the network, we achieve a remarkable reduction in training time (see Table 6.2).

Table 6.2: Comparison of performance on DPA contest-v4

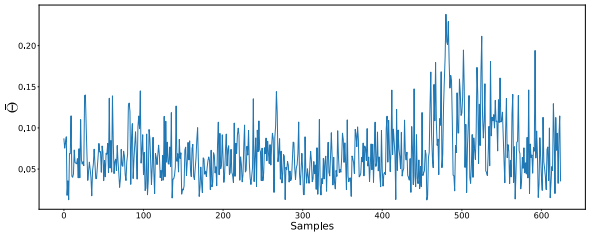
	State-of-the-art ([PHJ+18])	**Our methodology**
Complexity (trainable parameters)	52,112	8,782
$\bar{N}t_{\text{rank}}$	4	3
Training time (seconds)	1,000	23

AES_HD. For the evaluation of this dataset, we design a convolutional neural network such that the number of filters is set to 2. Then, the classification part is configured with 1 dense layer of 2 neurons. Finally, we set the learning rate to 10^{-3}. The training runs for 20 epochs with a batch size of 256.

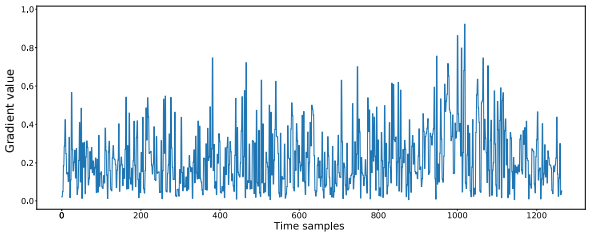
Looking at the Figure 6.7, we can conclude that our model is not optimized. Indeed, the weight visualization shows us that the selection of features is effective because our network can detect the relevant leakage points. When we look at the SNR computation (see Figure 3.11b), the sensitive information leaks between the time samples 950 and 1,050 which corresponds to the features detected by the feature selection part. However, by visualizing the gradient, we observe that the overall neural network is not enabled to significantly recognize the points of interest (see Figure 6.7c). Thanks to these visualization techniques, we can identify which part of the network needs optimizing. Currently, no tool is available that allows interpretation of the classification part to improve its efficiency. However, even if the classification part is not powerful, our network still performs much better than the state-of-the-art.



(a) Partial security graph over 100 attacks.



(b) Weight visualization.



(c) Gradient visualization.

Figure 6.7: Evaluation of the AES_HD dataset.

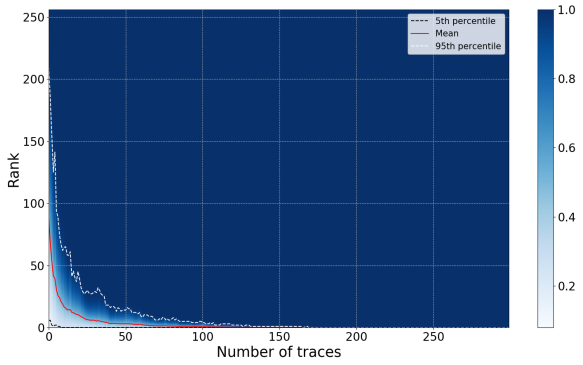
We compare our results with the architecture proposed in [KPH+19] where they achieved the best performance on this dataset. On average, $\bar{N}t_{\text{rank}}$ reaches 25,000 traces. By applying our methodology, we dramatically improve this result. First, the new architecture has 31,810 times fewer parameters. Now, only 31 seconds are necessary to train the network. Finally, our network outperforms the previous state-of-the art network by getting $\bar{N}t_{\text{rank}}$ around 1,050 (see Table 6.3)

while, from Figure 6.7a, we observe that the best side-channel attack retrieves the secret key in less than 100 leakage traces.

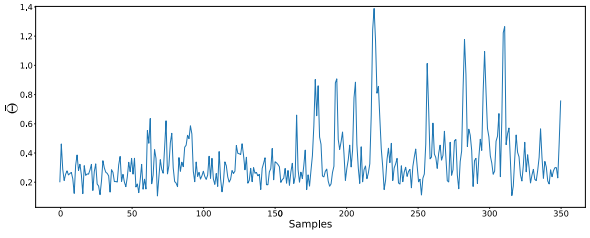
Table 6.3: Comparison of performance on AES_HD

	State-of-the-art ([KPH+19])	**Our methodology**
Complexity (trainable parameters)	104,401,280	3,282
$\bar{Nt}_{\text{rank}}$	25,000	1,050
Training time (seconds)	6,075	31

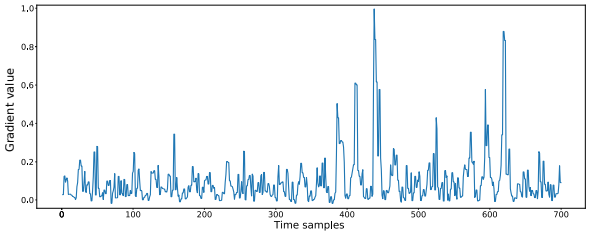
ASCAD-v1 (synchronized traces). To generate our convolutional neural network, we initialize the number of filters to 4 with a length of 1. Then, two dense layers composed of 10 neurons complete the neural network. Thanks to the one cycle policy, we are able to configure our learning rate in the range $[5 \times 10^{-4}; 5 \times 10^{-3}]$. The network is trained for 50 epochs with a batch size of 50.
Next, we compare our result with the original paper [BPS+20]. Surprisingly, we can notice that the network detects more points of interest than the mask and the masked value. By visualizing the weights, we can evaluate that the neural network recognizes relevant information related to the mask and the masked values, although a sensitive area (between time samples 380 and 450, see Figure 6.8c) is detected by the neural network whereas nothing appears on the SNR (see Figure 3.13b). Like the DPA contest-v4 experiment, the weight visualization and the gradient visualization are similar. The classification part appears to be optimized because no information is lost between the end of the feature selection part and the output of the neural network.



(a) Partial security graph over 100 attacks.



(b) Weight visualization.



(c) Gradient visualization.

Figure 6.8: Evaluation of the ASCAD dataset with no desynchronization.

When we compare the new performances, we notice that our new network is 3,930 times less complex than the original paper. In terms of performance, our simpler neural network achieves $\bar{Nt}_{\text{rank}}$ in 191 leakage traces in average while the best attack retrieves the secret key in less than 10 leakage traces. In comparison, the original neural network reaches the same performance only

after $1,146$ traces (see Table 6.4).

Table 6.4: Comparison of performance on ASCAD-v1 with $N_0 = 0$

	State-of-the-art ([BPS+20])	**Our methodology**
Complexity (trainable parameters)	$66,652,444$	$16,960$
$\bar{\boldsymbol{Nt}}_{\text{rank}}$	$1,146$	191
Training time (seconds)	$5,475$	253

Through this section, we validate the benefits of our methodology on a wide range of scenarios when no hiding countermeasure is implemented. From the obtained results, we can suggest that the convolutional neural networks do not have to be complex in order to extract the secret key manipulated by the cryptographic module. However, this observation cannot be extended to large leakage traces as our work is applied only on very small leakage traces (*i.e.* less than $10,000$ time samples) where few sensitive variables leak (*i.e.* less than 10). But this scenario corresponds to a classical evaluation process where the Evaluator selects a small portion of a leakage trace where the targeted sensitive variable leaks.

In the following section, we propose another methodology that the Evaluator can consider in order to mitigate the random delay issue.

6.3.2 Random Delay Effect

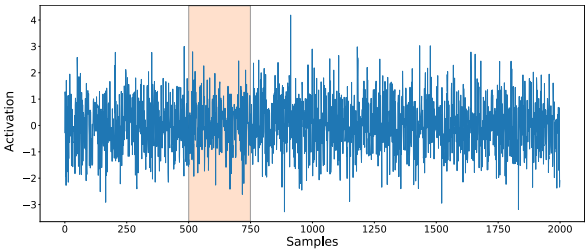
As mentioned in Subsection 3.4.2, the random delay countermeasure significantly reduces the correlation between the targeted sensitive variable and the time samples where it leaks. The higher the amplitude of the random delay, the lesser the probability of observing the execution of a given cryptographic operation for a certain period of time. Thus, adding desynchronization effect does not reduce the information contained in a set of leakage traces, but, retrieving the secret is more challenging. In this section, we aim to find a methodology such that we design a Θ-parametric model F_Θ to deal with random delay effect while the performance to solve the classification task is preserved.

Methodology. To build a suitable neural network architecture, we suggest using a new methodology that helps the detection of desynchronization effect while reducing the dimension of the leakage traces so as to focus the neural network on the points of interest (see Figure 6.10). For a better understanding, we propose to apply the following methodology on a toy example. We simulate a set of $50,000$ leakage traces ($45,000$ for the profiling phase, $5,000$ for the validation phase) such that we assume that each leakage trace is configured by $4,000$ time samples with 5 points of interest equally distributed the leakage trace. In addition, a random delay effect with a maximum amplitude of 100 is considered in order to assess our proposition. The simulated leakage model does not induce interactions between bits (*i.e.* $\mathcal{G}_1$) such that all bits have the same weighting. Hence, the i^{th} time sample of the simulated leakage trace $\mathbf{T}$ is defined as follows:

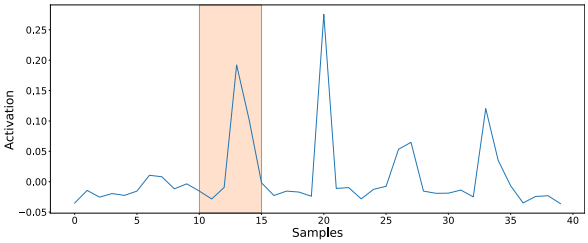
$$\mathbf{T}[i] = \begin{cases} 1 \cdot Y[3] + 1 \cdot Y[6] + \mathbf{Z}[i] & \text{if} \quad i = 1, \\ \mathbf{Z}[i] & \text{otherwise,} \end{cases} \tag{6.4}$$

where $Y[b] = Sbox[X \oplus k^*][b]$ denotes the b^{th} bit of the output of the Sbox and $\mathbf{Z}[i]$ is a Gaussian noise following $\mathcal{N}(0, \sigma^2)$ such that $\sigma^2 = 0.1$.
The related SNR computation is provided in Figure 6.9d. The methodology can be described as follows, such that the feature selection part can be divided into three blocks:

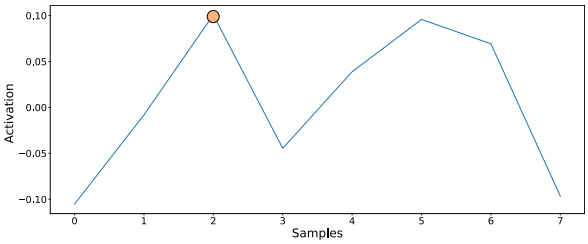
- As shown in Subsection 6.2.1, the first layer aims at minimizing the length of the filters in order to mimic the entanglement between each point of interest and extract the relevant



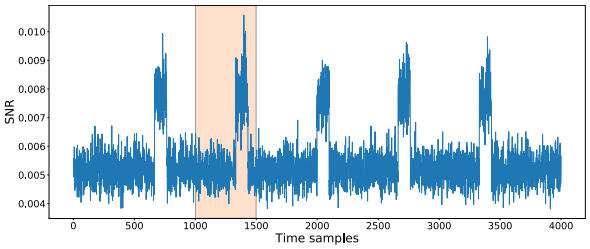
(a) Heatmap of the first convolutional block.



(b) Heatmap of the second convolutional block.



(c) Heatmap of the third convolutional block.



(d) Signal-to-noise ratio.

Figure 6.9: Evaluation of the methodology on a simulation characterized in Equation 6.4.

information. Reducing the length of the filters helps the network to focus its interest of local perturbation and consequently, can get a precise characterization of the points of interest. We suggest setting the length of the filters in the first convolutional block, to 1 in order to optimize the entanglement minimization. This can also be explained as an automatic preprocessing phase where the neural network adjusts the leakage traces in order to enhance the classification task [WAGP20]. Then, the first pooling layer is set to $ps^{[1]} = 2$ in order to reduce the dimension of the trace and help detect desynchronization. In order to confirm this claim, we compute the heatmap of the first convolutional block (see Figure 6.9a). Therefore, we are able to evaluate which neurons are activated through this convolution.

- The second block tries to detect the point of interest impacted by the random delay effect such that A_{RD} be its maximum amplitude. By applying filter length of size $\frac{A_{\text{RD}}}{ps^{[1]}}$, we focus the interest of the network on the detection of the desynchronization of each leakage trace. With this proposition, we are assured that the relevant pattern is included in the filter. Hence, the leakage will be necessarily extracted. The network obtains a global evaluation of the leakages by concentrating its detection on the leakage desynchronization and not on the leakages themselves. Then, to focus the information of a single sample, we set the pooling stride $ps^{[2]}$ to $\frac{A_{\text{RD}}}{ps^{[1]}}$. This maximizes the reduction of the intermediate trace dimension while preserving the information related to the targeted sensitive variable. This process is illustrated in Figure 6.9b. Thanks to the heatmap, we evaluate that the SNR peaks are perfectly identified by the convolutional block. Indeed, by comparing the activity detected by the convolution operation with the SNRs peaks (see Figure 6.9d), we can argue that the features are correctly selected by the filters of the 2^{nd} convolutional block.

- The third block aims at reducing the dimensionality of each intermediate trace in order to focus the network on the relevant points and to remove any irrelevant ones. In the case where our leakage traces have u points of interest, then, only the u time samples help the network to make a decision. By dividing the output of the second convolutional block into u different parts, we force the neural network to only focus its interest on the PoIs. Indeed, each part contains information related to a single spread leakage. This process reduces the dimensionality of each intermediate trace by u such that each sample of the output of the 3^{rd} convolutional block characterizes a point of interest. Furthermore, applying this technique limits the desynchronization effect because we force the network to concentrate

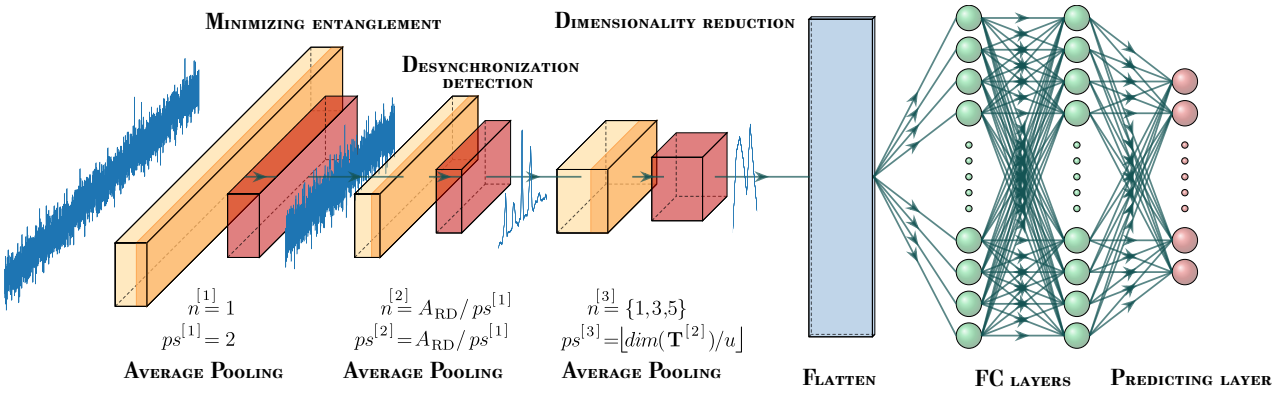


Figure 6.10: Convolutional neural network architecture mitigating random delay effect.

the initial desynchronized PoIs into a single sample. This property can be visualized in Figure 6.9c. In our simulation, we can identify $u = 5$ SNR peaks. In order to differentiate informative from uninformative convolved samples, we reduce the leakage trace dimension into 8 samples such that each of them contains the information related to the targeted sensitive variable.

Through Figure 6.9, the Evaluator can assess how a point of interest is spread through the feature selection part. Indeed, when he visualizes the heatmap after each convolutional block, he is able to evaluate the evolution of the dimension of the leakage trace as well as the related activation. Through the visualization of the heatmap of the 3$^{\mathrm{rd}}$ convolutional block (see Figure 6.9c), the Evaluator can define the amount of relevant information included in each portion of the leakage trace such that the sample at index 2 contains the information related to the portion of the leakage trace in $[1,000, 1,500]$ (see Figure 6.9d). The other samples define the information related to the portion of the leakage trace. Thus, through the proposed feature selection part, we reduce the dimension of the leakage traces such that only the informative time samples are kept. This reduces the complexity of the neural network as well as the resulted training time which is beneficial from an evaluation perspective (see Subsection 6.1.1). An overview of this methodology is provided in Figure 6.10.

AES_RD. Our new architecture can be set as shown in Figure 6.10. In the second convolutional block, the length of the filters and the pooling stride are configured following the value of the desynchronization effect (*i.e.* $\frac{A_{\mathrm{RD}}}{ps^{[1]}}$). As A_{RD} is nearly equal to $1,000$ [WAGP20], the length of the filters $n^{[2]}$ as well as the pooling stride $ps^{[2]}$ are set to 500. More details about these hyper-parameters are provided in Table 6.8. Then, two fully-connected layers composed of 10 neurons define the classification part. Thanks to the one cycle policy, we are able to set our learning rate in the range $[10^{-4}; 10^{-3}]$. The network is trained for 50 epochs with a batch size of 50.

Table 6.5: Comparison of performance on AES_RD

	State-of-the-art ([KPH+19])	**Our methodology**
Complexity (trainable parameters)	$512,711$	$69,080$
$\bar{\boldsymbol{N}}\boldsymbol{t}_{\mathrm{rank}}$	10	5
Training time (seconds)	$4,500$	$1,965$

When our methodology is applied, the impact of the desynchronization is dramatically reduced and the performance of the neural network is also less affected by this countermeasure. Through Figure 6.11, we observe the most efficient attack retrieves the secret key in 1 leakage trace while the worst case, with 10 leakage traces. Compared with the state-of-the-art [KPH+19], the performance

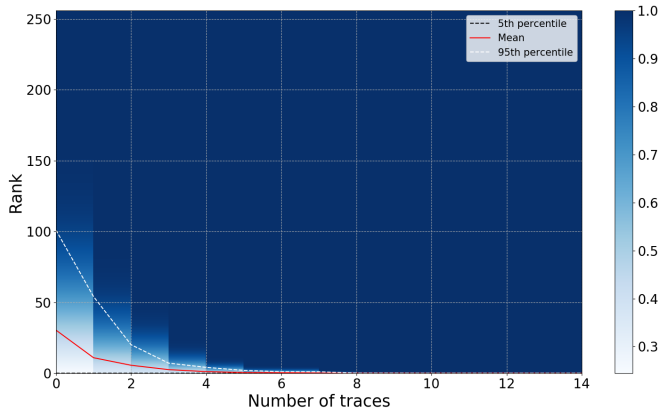


Figure 6.11: Partial security graph over 100 attacks on the AES_RD dataset.

$\bar{N}t_{\text{rank}}$ is similar but the complexity related to our network proposal is greatly reduced (7.35 times smaller). This reduction improves the training time by a non-negligible factor and consequently, ease the application of deep learning discriminative models in the evaluation process.

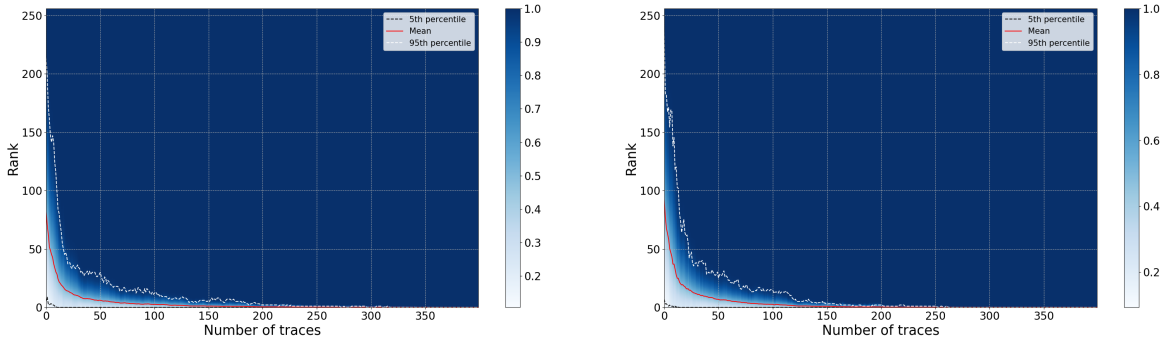
ASCAD-v1 with $A_{\text{RD}} = 50$. Thanks to Figure 6.8, we set $u = 3$ because three global leakage areas appear. As the previous paragraph, the hyperparameters are configured following Figure 6.10. In this dataset, the maximum amplitude of random delay equals 50. Thus, the length $n^{[2]}$ of the filters and the pooling stride $ps^{[2]}$ defined in the second convolutional block are configured to $\frac{A_{\text{RD}}}{ps^{[1]}} = 2$. Thus, the neural network focuses its interest on the detection of the desynchronization effect. In order to preserve the information related to the points of interest, we set $ps^{[3]}$ to 4. This generates an output with 3 samples at the end of the feature selection part. More details about these hyperparameters are provided in Table 6.8. The best performance is obtained when we configure three fully-connected layers with 15 neurons. We apply a range of learning rate between $[5 \times 10^{-4}; 5 \times 10^{-3}]$ and we set the number of epochs to 50 with a batch size 256.

Table 6.6: Comparison of performance on ASCAD-v1 with $A_{\text{RD}} = 50$

	State-of-the-art ([BPS+20])	**Our methodology**
Complexity (trainable parameters)	$66,652,444$	$87,279$
$\bar{\boldsymbol{N}}\boldsymbol{t}_{\text{rank}}$	$> 5,000$	244
Training time (seconds)	$5,475$	380

Applied on this dataset, our methodology outperforms the state of the art remarkably. In the original paper, Prouff et al. did not reach a constant guessing entropy of 1 with $5,000$ leakage traces [BPS+20]. Applying our methodology, we converge, in average, towards a constant guessing entropy of 1 with 244 leakage traces (see Figure 6.12a) while the complexity of our networks is divided by 763. As a reminder, the performance related to the neural network trained with synchronized traces converges towards a guessing entropy of 1 with around 200 traces. Thus, we succeed in dramatically mitigating the random delay effect without impacting the network performance.

ASCAD-v1 with a Random delay: $A_{\text{RD}} = 100$. Finally, we apply our methodology on an even more complex system implementing a random delay with $A_{\text{RD}} = 100$. As the previous paragraphs, we set our network with the hyperparameters detailed in Table 6.8. In the second convolutional block, we set the length of the filters and the pooling stride to 50 to focus the interest



(a) $A_{\text{RD}} = 50$. **(b)** $A_{\text{RD}} = 100$.

Figure 6.12: Partial security graph over 100 attacks on the ASCAD dataset implementing a random delay effect.

Table 6.7: Comparison of performance on ASCAD-v1 with $A_{\text{RD}} = 100$

	State-of-the-art ([BPS+20])	**Our methodology**
Complexity (trainable parameters)	$66,652,444$	$142,044$
$\bar{\boldsymbol{N}}\boldsymbol{t}_{\text{rank}}$	$> 5,000$	270
Training time (seconds)	$5,475$	512

of the network on the desynchronization effect. In order to preserve the information related to the leakages, we set $ps^{[3]}$ to 2. This results in an output with 3 samples which is similar to ASCAD-v1 with a random delay of 50. Thanks to the one cycle policy, we define our learning rate in the range $[10^{-3}; 10^{-2}]$ to allow a robust training and to reach a good local minimum. As before, our network is trained during 50 epochs with a batch size 256.

In comparison with [BPS+20], our methodology generates a neural network that is far less complex and $\bar{N}t_{\text{rank}}$ is outperformed. Indeed, we converge towards a constant guessing entropy of 1 with 270 leakage traces while the original paper couldn't converge with $5,000$ leakage traces (see Table 6.7). In addition, over 100 consecutive attacks, the Evaluator retrieves the secret key with less than 15 leakage traces in the best-case scenario (see Figure 6.12b). Furthermore, by reducing the network complexity, the training process can be performed much faster. Comparing this performance with Subsection 6.3.1, we note that our methodology dramatically reduce the impact of the desynchronization. Indeed, the performance of the two models is similar, thus, our methodology eliminates the effect of the random delay countermeasure.

Table 6.8: Convolutional hyperparameters for each dataset

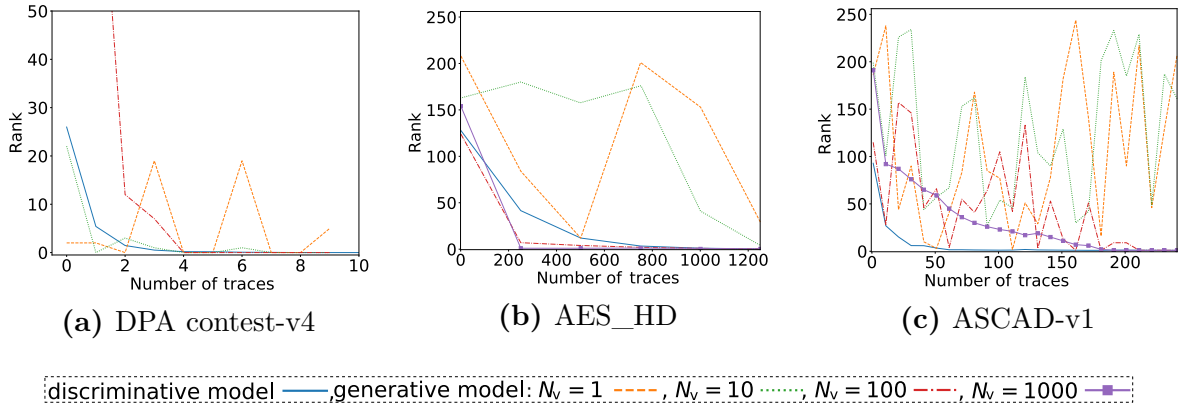
	Layer	**Length of the filters**	**Pooling stride**	**Number of filters**
AES_RD	1^{st} convolutional block	1	2	8
	2^{nd} convolutional block	500	500	16
	3^{rd} convolutional block	3	$\lfloor \frac{T^{[2]}}{u} \rfloor = 1$	32
ASCAD-v1 ($N^{[0]} = 50$)	1^{st} convolutional block	1	2	32
	2^{nd} convolutional block	25	25	64
	3^{rd} convolutional block	3	4	128
ASCAD-v1 ($N^{[0]} = 100$)	1^{st} convolutional block	1	2	32
	2^{nd} convolutional block	50	50	64
	3^{rd} convolutional block	3	2	128

6.3.3 Discussion on Discriminative Neural Networks in Side-Channel Analysis

Methodology's improvement. This methodology can be applied to implementations where random delay as well as Boolean masking are considered by the Developer. However, to construct the most efficient convolutional neural network, the fully-connected layers and the optimizer hyperparameters have to be adapted following the targeted implementation. Indeed, as observed through Subsection 6.3.1 and Subsection 6.3.2, the Evaluator has to monitor those hyperparameters in order to adequately solve the classification task. This aspect should be a part of a future work. Regarding the proposed methodology, some improvements can be brought. Indeed, as experimentally shown by Wouters *et al.* [WAGP20], the length of the filters can be reduced without altering the related model performance. However, no theoretical demonstration or guidelines have been provided to ease the construction of convolutional neural networks. Finally, the proposed neural network can be optimized with the related work proposed in deep learning based side-channel context (see Subsection 4.4.2) or by finding the most suitable hyperparameters like the weight initialization technique [LKP20] and the activation function [KFJP21].

Network Complexity. Through Subsection 6.3.1 and Subsection 6.3.2, we demonstrate that the discriminative models applied in deep learning based side-channel context do not have to be complex to perform well on low leakage trace dimension. This observation has been then validated by the side-channel community on small [LKP20, WPP20, WAGP20, PP21, WWJ+21, PWP21, RWPP21] and large leakage traces (*i.e.* $D > 100,000$ time samples) [MBC+20]. However, to defeat the random delay effect, we observe that the neural network complexity has to be increased. Thus, from the Evaluator point of view, performing a preprocessing phase that reduces the desynchronization effect can be beneficial from a network complexity perspective. In addition, if the Evaluator is confronted to a cryptographic module implementing hiding countermeasures, Zhou and Standaert suggest performing a preprocessing phase as it can be beneficial from an attack perspective [ZS19]. Finally, to assess the perfect suitability of discriminative model to perform side-channel attacks, the Evaluator can question its relevance regarding the performance of the generative approach introduced in Chapter 5.

Comparison with Generative models. When the discriminative approaches are considered, a major drawback can be highlighted regarding the architecture configuration (see Subsection 6.1.1). The more effort the Evaluator spends on the hyperparameter tuning of the network architecture, the more efficient the resulted attack is expected. Even if the proposed methodology tends to reduce this gap, it remains a major issue for the construction of a suitable neural network. In addition, due to their black-box property, the discriminative models are difficult to interpret. However, the main benefit of this approach is about automatically combining the point of interests in order to mitigate the data randomization countermeasure (*e.g.* the masking effect). To compare both approaches, the following generative models are designed with respect to the conditional variational autoencoder proposed in Chapter 5. For being compliant with Subsection 5.4.1, we focus the interest of the generative models on the most relevant samples only, while the discriminative model considers all the samples of the leakage traces. Indeed, as highlighted in Subsection 5.3.3, increasing the number of irrelevant time samples highly impact the network complexity of the cVAE-ST and the training time without altering the related performance. While our work does not investigate the dimensionality reduction techniques for improving the generative model, we want to evaluate this new proposition in side-channel field.
First, on Figure 6.13a, we can visualize the rank evolution of the generative and the discriminative models on the DPA contest-v4 dataset. While a discriminative model can retrieve the secret key with 4 leakage traces, a generative model reaches the same performance depending on the number N_v of latent samples. As mentioned in Subsection 5.2.4, the value of N_v depends on the ability of the conditional variational autoencoder to correctly approximate $\mathsf{Pr}[\mathbf{T}|Y, \phi]$. The higher the N_v, the more confident the resulted attack. This observation can be made in Figure 6.13a. Indeed, if



(a) DPA contest-v4 **(b)** AES_HD **(c)** ASCAD-v1

Figure 6.13: Mean rank evolution for generative and discriminative models.

$N_v = 1$, two leakage traces are needed to retrieve the secret key. However, a poor rank stabilization is observed. To rectify this point, increasing the N_v's value preserves a constant rank convergence towards 1.

The same observation can be made when the AES_HD dataset is considered. Indeed, when the N_v value increases, a rank stabilization is observed when the number of attack traces grows. In addition, Figure 6.13b highlights a better model when the generative approach is considered. Indeed, for $N_v = \{100, 1000\}$, the resulted model converges towards a constant rank of 1 with 250 attack traces. Even if the discriminative approach directly estimates $\mathsf{Pr}[Y|\mathbf{T}]$, this figure indicates a lower performance when classical deep learning based side-channel models are considered. Indeed, on average, the related amount of leakage traces that are needed for retrieving the secret key is about $1,000$. As illustrated by Ng and Jordan [NJ02], this result suggests that a better discriminative model can be found on this dataset. This is in accordance with the claim provided in Subsection 6.3.1. Indeed, an optimal discriminative model should be, at least, as efficient as a generative approach[k] because it optimizes an approximation of the true unknown PDFs $\mathsf{Pr}[Y|\mathbf{T}]$. However, finding the best discriminative model can be difficult due to the broad hyperparameter selection. This result highlights the benefits of the cVAE-ST in comparison with the classical deep learning-based side-channel models from a practical perspective.

One benefit of the discriminative approach is to automatically recombine the point of interests. In opposition, the generative approach does not take advantage of this property. Through Figure 6.13c, we can visualize the benefits of automatically combining the point of interest. Indeed, the discriminative approach converges faster towards a constant rank of 1. This result could be explained by the ability of the discriminative model to find a custom combining function that maximizes the posterior probabilities $\mathsf{Pr}[Y|\mathbf{T}]$. Hence, this custom unknown function can be more adapted for the targeted dataset. On the other hand, the cVAE-ST model is trained on combined leakage traces that are constructed from classical approaches (*i.e.* optimal product combining). Unfortunately, those combining functions are not fully adapted for all the targeted cryptographic modules. Consequently, when masking implementations are considered, a discriminative approach can provide better result than a generative approach.

These results highlight the benefits and the limitations of classical discriminative models against the cVAE-ST. While the configuration of cVAE-ST is simple in comparison with the discriminative models, the latter approach can perform, at least similarly, in all attack scenarios.

[k]While this statement can be justified from a Machine Learning perspective [NJ02], it can be qualified in Side-Channel context as some simplification can be made (see Equation 3.4).

6.4 Conclusion

This chapter proposes a new methodology for designing convolutional neural networks that fits with the Evaluator's restrictions. To construct such proposition, we analyze the interpretability of convolutional neural networks by evaluating the impact of different model hyperparameters that compose the feature selection part. As the latter extracts the points of interest from the leakage traces, we ease its construction in order to enhance the model's performance. Thus, we introduce two visualization tools that help analyze which patterns are the most influential during the training process. These patterns are similar to points of interest which are well known in the side-channel context. We demonstrate the theoretical effect of the length of filters, the pooling operations and the number of convolutional blocks and we use these visualization techniques in order to check the validity of our demonstrations. These visualization tools help us find a method of generating an appropriate methodology with respect to the Evaluator's constraints, *i.e.* finding a good trade-off between leakage detection, network complexity and training time. Through the application of the methodology, we dramatically reduce the network complexity without altering the attack performance. This illustrates that the neural networks needed to perform side-channel attacks do not have to be complex.

Which solution should the Evaluator consider? To conclude about Part II, the Evaluator can question the relevance of using the discriminative or the generative approach in side-channel context. Both approaches (*i.e.* generative vs. discriminative) should be considered independently. However, depending on the Evaluator's objective, a solution can be preferred. Typically, if the Evaluator wants to build a neural network in order to get a first insight into how well a task can be solved, the construction of cVAE-ST seems natural as it provides a fully interpretable generative model. This approach can be helpful to evaluate the feasibility of an attack and get a security bound of a device. In addition, if the objective of the Evaluator is to deeply characterize the component behavior (*i.e.* leakage model), using such approach is beneficial from interpretability and explainability perspective. Furthermore, in [NJ02], Ng and Jordan illustrate that a generative model can converge towards its minimal asymptotic error faster than a discriminative model. However, after a certain period of time, a discriminative approach converge towards a better solution and outperforms a generative classifier.

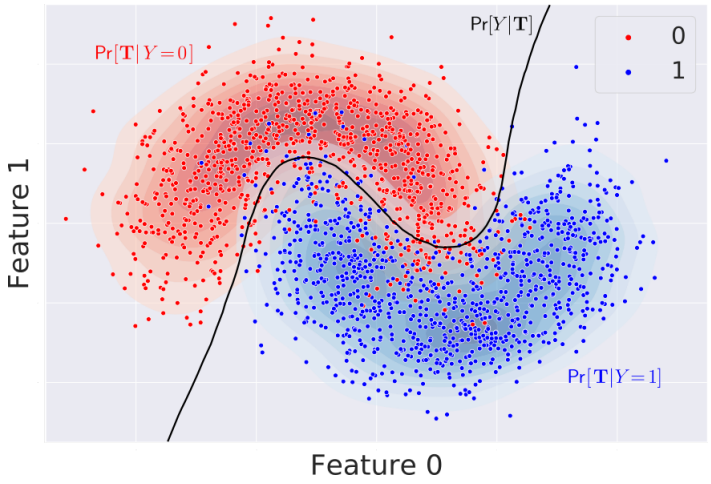


Figure 6.14: Difference between the probabilistic discriminative $\mathsf{Pr}[Y|\mathbf{T}]$ and the probabilistic generative $\mathsf{Pr}[\mathbf{T}|Y]$ approaches given a binary classification problem.

Thus, if the Evaluator wants to perform the most powerful side-channel attack, considering the discriminative approach is the most suitable solution. Indeed, a discriminative model is specifically trained to optimize the conditional Θ-parametric probability distribution $\mathsf{Pr}[Y|\mathbf{T}, \Theta]$. Hence, it is built to capture the decision boundary of the underlying problem (see Figure 6.14). In opposition,

the generative models tackle a more difficult task than analogous discriminative models as it requires to capture the behavior of the cryptographic module given each class $Y \in \mathcal{Y}$. An example for a binary classification task is provided in Figure 6.14. As the discriminative model directly computes $\mathsf{Pr}[Y|\mathbf{T}, \Theta]$, it generally outperforms generative models at conditional prediction tasks [NJ02].

WHAT'S NEXT?

While the Evaluator's goal is to converge towards the optimal Adversary (see Objective 3.3.1.1), the following part of this manuscript will be focused on the discriminative approach only as it provides the best performance result [NJ02]. More precisely, Part III tackles the problem of optimal discriminative model by proposing new loss functions adapted from the optimal distinguisher detailed in Definition 3.3.1.4.

Part III

Towards a Better Optimization of the Discriminative Models in Deep Learning Side-Channel Attacks

Chapter 7

Ranking Loss: Maximizing the Success Rate

As defined in Definition 3.3.1.4, the optimal distinguisher rule retrieves the secret key, manipulated by the cryptographic module, from the true unknown conditional probability $\mathsf{Pr}[\mathbf{T}|Y]$ or $\mathsf{Pr}[Y|\mathbf{T}]$. Thus, the Evaluator has to construct a Θ-parametric model that approximates such solutions through the use of an adequate loss function. While finding this learning metric is considered as a non-trivial task [PHJ+18], this chapter proposes to reduce this gap by deriving a new loss function from the *Learning to Rank* field. After a short introduction of this field, we detail its link with the success rate metric (see Equation 3.6). By adapting this new loss function, called *Ranking Loss* (RkL), for the side-channel context, we force the learning algorithm to explicitly select the model that optimizes the attack performance by extracting the leakage traces' information. Then, a theoretical comparison with the negative log-likelihood loss function is proposed. Finally, experimental results validate our theoretical observations and suggest that the model trained with the ranking loss performed, at least, similarly to a model trained with the negative log-likelihood loss function. The solutions proposed in this chapter have been presented at CHES and published in the journal IACR TCHES [ZBD+20b].

Contents

7.1 A Learning Metric Adapted for Side-Channel Analysis

7.1.1 Learning To Rank Approach

"Learning to Rank" refers to supervised machine learning techniques for training a model in solving a ranking task. While classical supervised learning tasks can be solved by classification or regression, the learning to rank approach requires other properties than regular classes or scores.

Definition 7.1.1.1 (Ranking task)**.** Given a set of inputs $\mathcal{I}$ and a set of queries $\mathcal{Q}$, the ranking task is characterized by the ability of the learning algorithm to automatically select $F \in \mathcal{F}$ that defines the relevance degree of each input in $\mathcal{I}$ given a query in $\mathcal{Q}$.

In other words, a model that follows the learning to rank paradigm aims at sorting a list of inputs with respect to a given query such that inputs with a high degree of relevancy are ranked higher than inputs with a low degree of relevancy. Thus, it does not care about the exact classes or scores of each item and favors the relative order between each input.
This task is useful for many applications in *Natural Language Processing*, *Data Mining* and *Information Retrieval* [Liu09, Bur10, Li11a, Li11b]. A typical example of inputs is characterized by a set of documents such that the ranking Θ-parametric model F_Θ learns to sort this set with respect to $q \in \mathcal{Q}$. The relevance of these documents is represented by several grades in a set $\mathcal{S}$. The higher the grade, the more relevant is the document. To perform such approach, Tie-Yan Liu [Liu09] defines three strategies: the pointwise approach, the pairwise approach and the listwise approach.

Pointwise approach. Given a query $q \in \mathcal{Q}$, a degree of relevancy has been specified for each input. Thus, the pointwise approach constructs a Θ-parametric *scoring* (or ranking) model $F_\Theta : \mathcal{I} \times \mathcal{Q} \to \mathcal{S}$ that predicts the relevance of an input for a given query. Once this prediction is made for each element of $\mathcal{I}$, one can sort all the inputs in order to produce a final rank list. To solve this problem, the classification or the regression task can be considered. In the classification task, it is expected to assign similar inputs in the same class. Typically, this degree may be binary (*e.g.* $\mathcal{S} = \{$"irrelevant", "relevant"$\}$ or $\mathcal{S} = \{0, 1\}$) or may contain multiple classes (*e.g.* $\mathcal{S} = \{$"unsatisfactory", "satisfactory", "good", "excellent", "perfect"$\}$ or $\mathcal{S} = \{0, 1, 2, 3, 4\}$). On the other hand, the regression means giving similar continuous value to similar inputs, so that we can assign them identical preferences during the ranking procedure.
In side-channel context, the Evaluator's goal is to rank a set of hypothetical key $\mathcal{K}$ given a leakage trace $\mathbf{T}$. Thus, the set of queries $\mathcal{Q}$ is defined by the set of leakage traces $\mathcal{T}$ while the set of inputs $\mathcal{I}$, that has to be ranked, is characterized by $\mathcal{K}$. While, in our context, the set $\mathcal{S}$ denotes the score associated with each hypothetical sensitive variable $(Y_k)_{k\in\mathcal{K}}$ in $\mathcal{Y}$, we can rewrite it as $\mathcal{S} \subseteq \mathbb{R}^{|\mathcal{Y}|}$. Consequently, the Θ-parametric model can be rewritten as $F_\Theta : \mathcal{T} \to \mathcal{S}$ such that it predicts the relevance of observing each hypothetical sensitive variable in $\mathcal{Y}$ given a leakage trace $\mathbf{T}$. Indeed, the Evaluator can query F_Θ to retrieve the score related to each hypothetical sensitive variable $(Y_k = f(X, k))_{k\in\mathcal{K}}$, with $f : \mathcal{X} \times \mathcal{K} \to \mathcal{Y}$ a cryptographic primitive (*e.g.* output of the Sbox, a XOR between a plaintext and a hypothetical key) which depends on a known plaintext X and a key hypothesis $k \in \mathcal{K}$. As he knows the plaintext X and the related cryptographic primitive, the Evaluator is able to assign this set of scores to each key hypothesis in $\mathcal{K}$, and therefore, sorting the list of key candidates from the most likely element to the less likely one. This entire process is similar to what the Evaluator does in the attack phase.

Pairwise approach. For a given pair of inputs, the model identifies the most relevant with respect to $q \in \mathcal{Q}$. Thus, the pairwise approach constructs a Θ-parametric scoring model $F_\Theta : \mathcal{I} \times \mathcal{I} \times \mathcal{Q} \to \mathcal{S}$ that takes a pair of inputs and predicts which is the most relevant for a given query [BSR+05, BRL07, WBSG10]. Once the model predicts the relative order for each pair of inputs $(\mathcal{I} \times \mathcal{I})$, all the results can be embedded into a classical sorting algorithm (*e.g.* bubble

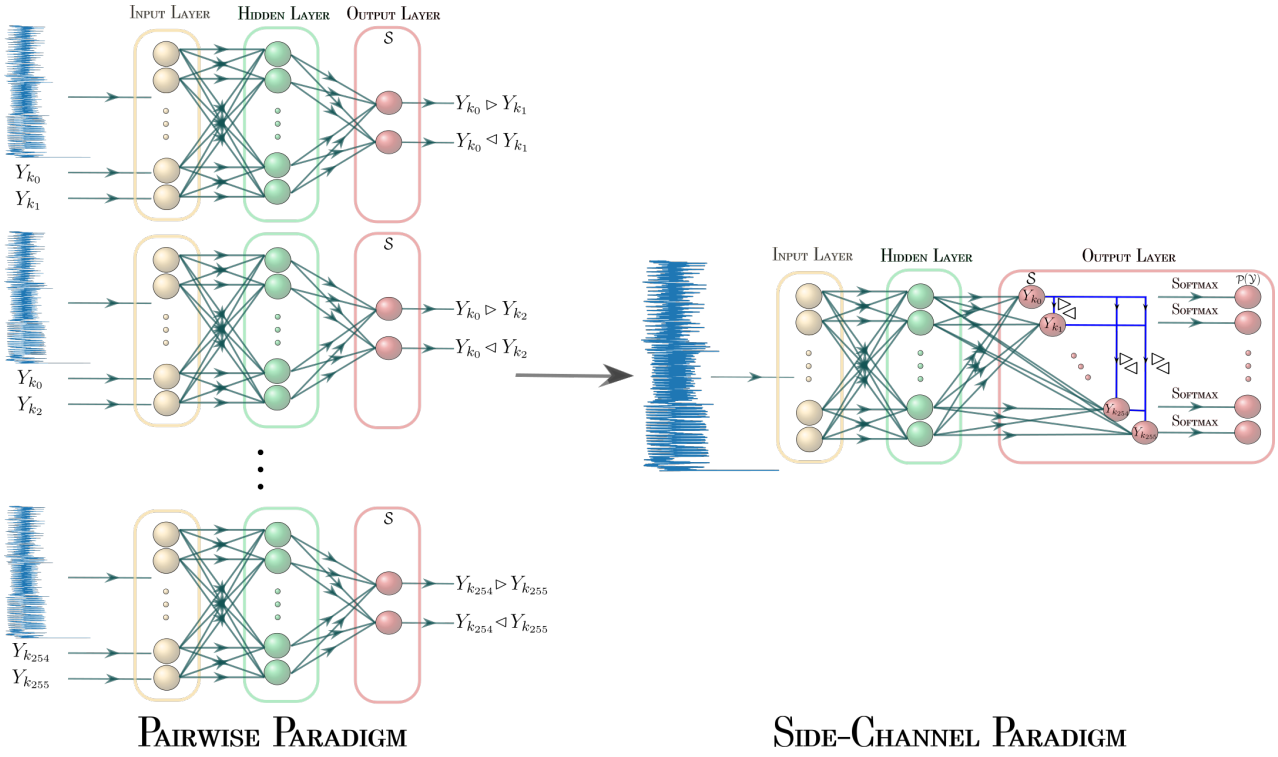


Figure 7.1: Transposition between the Pairwise and the Side-Channel paradigms.

sort, insertion sort) in order to provide a global ranking list. In this scenario, a classical example is to consider a binary classification problem such that given a pair of inputs $(u, v) \in (\mathcal{I} \times \mathcal{I})$ the model assigns 1 (resp. -1) if u (resp. v) is more relevant than v (resp. u). We denote $u \triangleright v$ the event that u should be ranked higher than v. Thus, the set $\mathcal{S} = \{-1, 1\}$.
In side-channel context, this approach suggests that the Evaluator considers a pair of sensitive variable in $\mathcal{Y} \times \mathcal{Y}$ as input[a] such that, given a leakage trace $\mathbf{T}$, the Θ-parametric model is able to define the most likely solution. The comparison between each pair of hypothetical sensitive variable builds the ranking of the whole model. As previously mentioned, the set of scores $\mathcal{S}$ can be described as $\mathcal{S} \subseteq \mathbb{R}^{|\mathcal{Y}|}$. Thus, instead of evaluating a pair of inputs, the Evaluator can directly compare the score associated with each pair of classes in order to obtain a ranking list. This transposition is illustrated in Figure 7.1. This approach consists in predicting the score related to each hypothetical sensitive variable included in $\mathcal{Y}$, and then, assigns it to the expected key hypothesis. The pairwise approach can be linked with the success rate metric considered in side-channel context. This result will be detailed in Subsection 7.1.2.

Listwise approach. It addresses the learning-to-rank problem in the most natural way, by taking a list of inputs and returning the ranked list for a given $q \in \mathcal{Q}$ [XL07, XLW+08, PGH18, CHX+19]. In comparison with the pointwise and the pairwise approaches, the listwise approach is quite complex to optimize.
From a side-channel context, it is assumed that the Evaluator can predetermine the relevance of each hypothetical sensitive variable for a given leakage trace $\mathbf{T}$. In this scenario, the Evaluator should be able to define a relative order between each irrelevant hypothetical sensitive variable. While his goal is *only* to differentiate the correct hypothetical sensitive variable from the irrelevant ones, this approach seems difficult to consider in practice.

Thus, contrary to the classical learning to rank approach that consists in the comparison between

[a]The Evaluator can also consider a pair of key hypotheses in $\mathcal{K} \times \mathcal{K}$ as he knows the targeted cryptographic primitive $f : \mathcal{X} \times \mathcal{K} \to \mathcal{Y}$ and the manipulated plaintext X.

inputs' relevance, we propose to adapt the "Learning to Rank" approach for the side-channel context through the comparison of the score, related to each hypothetical sensitive information $(Y_k)_{k\in\mathcal{K}}$, with the other classes. Consequently, for a given input, this approach aims at penalizing the training process when the score related to the correct hypothetical sensitive variable Y_{k^*} is not considered as the most relevant. In the following sections, we demonstrate how the pairwise approach can be useful to optimize the rank of the secret key in comparison with the other hypothetical ones. The other learning to rank approaches are beyond the scope of this thesis.

7.1.2 Ranking loss Maximizes the Success Rate

Problem statement. During the certification process, the Evaluator wants to determine the optimal distinguisher (see Definition 3.3.1.4) so as to minimize the probability of errors or equivalently, maximize the probability of success of an attack [HRG14]. As mentioned in Subsection 3.3.4, this metric is characterized by the success rate of retrieving the secret key k^* such that, given a set of N_a attack leakage traces $\mathcal{T}_a = \{\mathbf{t}_0, \ldots, \mathbf{t}_{N_a-1}\}$, it can be defined as follows:

$$\mathsf{SR}^o(F_\Theta, N_a) = \mathsf{Pr}\left[\sum_{k\in\mathcal{K}} \mathbb{1}_{s_{N_a}(F_\Theta,k)\geq s_{N_a}(F_\Theta,k^*)} \leq o\right], \tag{7.1}$$

where $s_{N_a}(F_\Theta, k)$ denotes the score related to a key hypothesis $k \in \mathcal{K}$. Thus, designing a loss function which automatically finds a Θ-parametric model that maximizes $\mathsf{SR}^1(F_\Theta, N_a)$ can be beneficial from an evaluation perspective.

Pairwise approach and success rate approximation. As mentioned in Subsection 7.1.1, the pairwise approach aims at defining a relative order between a pair of inputs for a given query. In the side-channel context, the pair of inputs is characterized by the score related to two key hypotheses $(k, k') \in \mathcal{K} \times \mathcal{K}$ such that the score $s_{N_a}(F_\Theta, k)$ should be higher than $s_{N_a}(F_\Theta, k')$ (*i.e.* $s_{N_a}(F_\Theta, k) \triangleright s_{N_a}(F_\Theta, k')$). The corresponding loss function maps the scores associated with k and k' and penalizes the training process once the relation $s_{N_a}(F_\Theta, k) \triangleright s_{N_a}(F_\Theta, k')$ is not respected. This penalization is exactly what the Evaluator wants to optimize with the 1st order success rate. Indeed, from Equation 7.1, the 1st order success rate can be approximated through the computation of the probability that the score related to the secret key k^* is higher than all key hypotheses. This means defining the probability that the secret key k^* is ranked higher than all $k \in \mathcal{K} \setminus \{k^*\}$. Let a key hypothesis $k \in \mathcal{K}$ and a secret key k^*, the probability that $s_{N_a}(F_\Theta, k^*) \triangleright s_{N_a}(F_\Theta, k)$ can be estimated via a sigmoid function [QLL10, BZBN19] such that:

$$\mathsf{Pr}\left[s_{N_a}(F_\Theta, k^*) \approx s_{N_a}(F_\Theta, k)\right] \equiv \frac{1}{1 + e^{-\alpha(s_{N_a}(F_\Theta,k^*) - s_{N_a}(F_\Theta,k))}}, \tag{7.2}$$

where α denotes the parameter of the sigmoid function. The value of α greatly impacts the training process. We evaluate its impact in Appendix B. In the following, we assume that α is well configured.

Definition of the ranking loss. We apply the binary cross-entropy in order to penalize the deviation of the model probabilities from the desired prediction. In other words, we want to penalize the loss function when the expected relation $s_{N_a}(F_\Theta, k^*) \triangleright s_{N_a}(F_\Theta, k)$ is not observed. Thus, we define a partial loss function $l(s_{N_a}(F_\Theta, k^*), s_{N_a}(F_\Theta, k))$, for a given hypothesis $k \in \mathcal{K}$, as:

$$l(s_{N_a}(F_\Theta, k^*), s_{N_a}(F_\Theta, k)) = -P_{k^*,k} \cdot \log_2\left(\bar{P}_{k^*,k}\right) - (1 - P_{k^*,k}) \cdot \log_2\left(1 - \bar{P}_{k^*,k}\right), \tag{7.3}$$

where $\bar{P}_{k^*,k} = \mathsf{Pr}\left[s_{N_a}(F_\Theta, k^*) \triangleright s_{N_a}(F_\Theta, k)\right]$ and $P_{k^*,k}$ defines the true unknown probability that k^* is ranked higher than k.

In the rest of the manuscript, we assume that the ranking value is deterministically known during the learning process, such that, $P_{k^*,k} = \frac{1}{2}(1 + \mathsf{rel}_{k^*,k})$ [BSR+05] where $\mathsf{rel}_{k^*,k} \in \{-1, 0, 1\}$ defines the relation between the secret key k^* and k such that $\mathsf{rel}_{k^*,k} = -1$ if k^* is less relevant than k; $\mathsf{rel}_{k^*,k} = 0$ if k^* is as relevant as k; $\mathsf{rel}_{k^*,k} = 1$ if k^* is more relevant than k. We assume that $\mathsf{rel}_{k^*,k}$ is always equal to 1. In the side-channel context, this approximation is reliable because the Evaluator wants to maximize the score related to k^* compared with the other hypotheses. From Equation 7.2 and Equation 7.3, we can deduce the following partial loss function:

$$l(s_{N_a}(F_\Theta, k^*), s_{N_a}(F_\Theta, k)) = \log_2\left(1 + e^{-\alpha(s_{N_a}(F_\Theta,k^*) - s_{N_a}(F_\Theta,k))}\right), \tag{7.4}$$

with $F_\Theta : \mathcal{T} \to \mathcal{S}$ such that the score[b] $\mathcal{S} \subseteq \mathbb{R}^{|\mathcal{Y}|}$ and $s_{N_a}(F_\Theta, k^*) = \sum_{i=0}^{N_a} F_\Theta(\mathbf{t}_i)[f(x_i, k)]$ denotes the score captured by the parametric model and $f : \mathcal{X} \times \mathcal{K} \to \mathcal{Y}$ is the targeted cryptographic primitive.
Thus, Equation 7.4 gives us an insight into how the cost function penalizes the training process when the relation $s_{N_a}(F_\Theta, k^*) \triangleright s_{N_a}(F_\Theta, k)$ is not the expected result. Therefore, maximizing the success rate tends to minimize the ranking error between the secret key k^* and a hypothesis $k \in \mathcal{K}$. As a remainder, this cost function, presented in Equation 7.4, is only applied on a single key hypothesis. In order to efficiently find the model that minimizes the ranking error, we have to apply this cost function on each key hypothesis in order to maximize the rank of the secret key.

Definition 7.1.2.1 (Empirical risk combined with ranking loss)**.** Given a profiling set $\mathcal{I}_p$ of N_p pairs $(\mathbf{t}_i, y_i)_{0 \le i \le N_p}$, a Θ-parametric model F_Θ and a number of attack traces N_a such that $N_a | N_p$, we define the empirical risk combined with the ranking loss (RkL) function as:

$$\hat{\mathcal{R}}(\mathcal{L}_{RkL}, F_\Theta) = \frac{N_a}{N_p} \sum_{i=1}^{N_p/N_a} \sum_{\substack{k \in \mathcal{K} \\ k \neq k^*}} \left(\log_2\left(1 + e^{-\alpha(s_{N_a,i}(F_\Theta,k^*) - s_{N_a,i}(F_\Theta,k))}\right)\right), \tag{7.5}$$

where $s_{N_a,i}(F_\Theta, k) = \sum_{j=1}^{N_a} F_\Theta\left(\mathbf{t}_{j+N_a\cdot(i-1)}\right)[f(x_j, k)]$ defines the score[c] of the hypothesis $k \in |\mathcal{K}|$ for given plaintexts $(x_j)_{1 \le j \le N_a}$ while α denotes one hyperparameter[d] related to the sigmoid that is needed for estimating the success rate.

Remark 7.1.2.1. To discriminate the right sensitive variable and normalize the score vector to a probability distribution over $\mathcal{Y}$, the softmax function is computed during the attack phase. This is essential to perform a side-channel attack (see Subsection 3.3.4).

With the ranking loss, we are more concerned with the relative order of the relevance of the key hypothesis than its absolute value (*e.g.* negative log-likelihood). Consequently, maximizing the success rate is equivalent to minimizing the ranking error for each pair $(k^*, k)_{k \in \mathcal{K}}$. Furthermore, contrary to the classical negative log-likelihood, Definition 7.1.2.1 takes into account the number of attack traces that is needed to perform a successful side-channel attack. Thus, the ranking loss penalizes the learning process depending on the number N_a of scores that the Evaluator aggregates before iterating the training process. Intuitively, this loss function is beneficial to select the model in $\mathcal{F}$ which minimizes the error provided on the success rate. This statement will be verified in the following section. As the ranking loss is linked with the success rate metric, visualizing the

[b] Here, the score $\mathcal{S}$ denotes the value before the softmax function (see Figure 7.1). This choice is made to impact the training process accordingly to the relative order of the key hypotheses' relevance instead of the normalized probability distribution. However, the classical side-channel score (see Equation 3.5) can also be applied.

[c] If the normalized probability distribution is considered as the score $\mathcal{S}$, therefore, $s_{N_a,i}(F_\Theta, k) = \sum_{j=1}^{N_a} \log(F_\Theta\left(\mathbf{t}_{j+N_a\cdot(i-1)}\right)[f(x_j, k)])$.

[d] The coefficient α can be specific for each $k \in \mathcal{K}$. The resulted $(\alpha_k)_{k \in \mathcal{K}}$ can be useful to mitigate the imbalanced data representation by assigning higher weights to classes with smaller occurrences.

evolution of the related empirical risk during the training process gives an overview of how well the related attack on N_a leakage traces should perform.

Remark 7.1.2.2. In classic information retrieval tasks, (i_0, i_1) and (i'_0, i'_1) characterize two different pairs of inputs such that the following relation holds $i_0 \triangleright i_1 \triangleright i'_0 \triangleright i'_1$. For each pair of inputs, if the difference of their scores is equal, thus, the empirical risk is the same regardless of their relative order. Indeed, in this configuration, swapping the rank of i_0 (resp. i_1) and i'_0 (resp. i'_1) does not impact the empirical risk $\hat{\mathcal{R}}(\mathcal{L}_{RkL}, F_\Theta)$ because it only cares about the total number of pairwise-rankings it gets wrong. This can be particularly problematic if we are interested in the top-ranking items. To solve this issue, the pairwise approach's literature applies some *information retrieval* measures (*e.g.* Discounted Cumulative Gain [JK02], Normalized Discounted Cumulative Gain [JK00], Expected Reciprocal Rank [CMZG09], Mean Average Precision, *etc*) to compute the empirical risk. However, in our context, $i_0 = i'_0$, therefore the difference between the scores of each pair (i_0, i_1) and (i_0, i'_1) gives us enough information on the position of i_0 related to i_1 and i'_1. Consequently, the addition of IR metrics is not relevant. Moreover, information retrieval metrics can be either discontinuous or flat, so gradient descent appears to be problematic (*i.e.* gradient equals to 0 or not defined) unless some appropriate approximations are used.

To emphasize the link with the side-channel context, the following section shows that the ranking loss is an upper bound of the measure-based ranking error.

7.1.3 Theoretical Bounds of the Ranking Loss

In the learning to rank research area, the information retrieval measures are used to evaluate the network performance. In most cases, two categories of metrics can be used: those designed for binary relevance levels (*e.g. mean average precision*, *mean reciprocal rank* [Cra09]) and those designed for multiple levels of relevance (*e.g. discounted cumulative gain*, *normalized discounted cumulative gain* [JK00]). In the side-channel context, there are only two levels of relevance such that 1 is associated with the sensitive information Y related to the secret key k^* and 0 otherwise. Thus, the metrics for binary relevance levels have to be considered.

Mean average precision and success rate. The *mean average precision* (MAP) defines the average precision of the secret key k^* over the $|\mathcal{K}|$ hypotheses. Let $g_{N_a} = \left[g^1_{N_a}, g^2_{N_a}, \ldots, g^{|\mathcal{K}|}_{N_a}\right]$ be a vector that defines the rank for each key hypothesis in $\mathcal{K}$ as introduced in Subsection 3.3.4. We consider $g^1_{N_a}$ as the most likely candidate and $g^{|\mathcal{K}|}_{N_a}$ as the least likely one.
Let d be a threshold and $MAP@d$ be the average precision of the secret key k^* in the top d relevant position of g_{N_a} such that:

$$MAP@d = \frac{1}{N_q} \sum_{q=1}^{N_q} AP_q@d,$$

where N_q denotes the number of queries and $AP_q@d$ is the average precision of the query q over the top i relevant position of g_{N_a}. As the only query considered in the side-channel context concerns the position of the secret key k^* over g_{N_a}, $MAP@d$ can be reduced to the computation of $AP_{k^*}@d$ such that:

$$AP_{k^*}@d = \frac{1}{\text{Number of True Positives at d}} \sum_{i=1}^{d} \frac{\text{Number of True Positive seen} \times \mathsf{rel}(i)}{i},$$

where $\mathsf{rel}(i)$ is an indicator that equals 1 if the element at the i^{th} rank is the secret key k^* and 0 otherwise. As k^* is assigned to a single position in g_{N_a}, the total number of True Positives equals 1 and the previous equation can be simplified as below:

$$MAP@d = AP_{k^*}@d = \sum_{j=1}^{d} \frac{\mathbf{1}_{g_{N_a}(F_\Theta,k^*)=j}}{j},$$

where $g_{N_a}(F_\Theta, k^*)$ denotes the position of k^* in g_{N_a} such that the sorted vector has been constructed from the Θ-parametric model F_Θ (see Equation 3.6). Therefore,

$$MAP@1 = AP_{k^*}@1 = \mathbf{1}_{g_{N_a}(F_\Theta,k^*)=1} = \mathsf{SR}^1(F_\Theta, N_a).$$

Thus, $MAP@1$ can be seen as a 1^{st} order success rate.

Lower bound of the ranking loss. As mentioned in [CLL+09a], the standard pairwise loss is considered as the upper bound of the measure-based ranking error that is defined by $1 - MAP@|\mathcal{K}|$ (justifications are provided in Appendix C).

Theorem 7.1.3.1 ([CLL+09a])**.** *Given a 2-level rating data with n_1 elements having grade 1 and $n_1 > 0$, then, the following inequality holds,*

$$1 - MAP@|\mathcal{K}| \leq \frac{1}{n_1} \sum_{i=0}^{|\mathcal{K}|-1} \sum_{\substack{j=0 \\ gr(j)<gr(i)}}^{|\mathcal{K}|-1} \log_2\left(1 + e^{-\alpha(s_{N_a}(F_\Theta,i) - s_{N_a}(F_\Theta,j))}\right), \tag{7.6}$$

*where $gr(i)$ defines the grade associated to the i^{th} key hypothesis (*i.e. 0 *or* 1*).*

In our context, a 2-level rating data means that the grade related to k is in $\{0, 1\}$ such that it equals 1 if and only if $k = k^*$. Thus, Theorem 7.1.3.1 can be easily written following the ranking loss.

Proposition 7.1.3.1. *Given a 2-level rating data with n_1 elements having grade 1 and $n_1 > 0$, a set of N_a leakage traces $(\boldsymbol{t}_i)_{0\leq i\leq N_a}$, and a Θ-parametric model F_Θ, the following inequality holds:*

$$1 - \mathsf{SR}^1(F_\Theta, N_a) \leq \frac{1}{n_1} \sum_{\substack{k\in\mathcal{K} \\ k\neq k^*}} \log_2\left(1 + e^{-\alpha(s_{N_a}(F_\Theta,k^*) - s_{N_a}(F_\Theta,k))}\right), \tag{7.7}$$

where $s_{N_a}(F_\Theta, k) = \sum_{i=1}^{N_a} F_\Theta(\boldsymbol{t}_i)[f(x_j, k)]$ defines the output score of the hypothesis $k \in |\mathcal{K}|$ for given plaintexts $(x_j)_{1\leq j\leq N_a}$

Proof. Following [CLL+09a, Theorem 2] and [CLL+09b, Lemma 1], given a 2-level rating data, it can be proved that:

$$n_1 - i_0 + 1 \leq \frac{1}{n_1} \sum_{i=0}^{|\mathcal{K}|-1} \sum_{\substack{j=0 \\ gr(j)<gr(i)}}^{|\mathcal{K}|-1} \log_2\left(1 + e^{-\alpha(s_{N_a}(F_\Theta,i) - s_{N_a}(F_\Theta,j))}\right),$$

where n_1 denotes the number of elements having grade 1 and i_0 defines the position of the first element with grade 0 in a ranking list.
If $i_0 > n_1$, the first element with grade 0 is ranked after position n_1. In side-channel analysis, there is only one candidate with a grade 1 (*i.e.* k^*). Hence, the correct candidate is ranked at the first position and $1 - \mathsf{SR}^1(F_\Theta, N_a) = 0$. Similarly, the first element with a label 0 is ranked at the second position and $n_1 - i_0 + 1 = 0$.
If $i_0 \leq n_1$, the first element with grade 0 is ranked before the secret key k^*. Consequently, the correct candidate is ranked at the second position and $1 - \mathsf{SR}^1(F_\Theta, N_a) = 1$. Similarly, the first element with a grade 0 is ranked at the first position and $n_1 - i_0 + 1 = 1$. Thus,

$$n_1 - i_0 + 1 = 1 - \mathsf{SR}^1(F_\Theta, N_a) \leq \frac{1}{n_1} \sum_{i=0}^{|\mathcal{K}|-1} \sum_{\substack{j=0 \\ gr(j)<gr(i)}}^{|\mathcal{K}|-1} \log_2\left(1 + e^{-\alpha(s_{N_a}(F_\Theta,i) - s_{N_a}(F_\Theta,j))}\right).$$

Finally, we can easily rewrite the right part of the inequality as:

$$\sum_{i=0}^{|\mathcal{K}|-1} \sum_{\substack{j=0 \\ gr(j)<gr(i)}}^{|\mathcal{K}|-1} \log_2\left(1+e^{-\alpha(s_{N_a}(F_\Theta,i)-s_{N_a}(F_\Theta,j))}\right) = \sum_{\substack{k\in\mathcal{K} \\ k\neq k^*}} \log_2\left(1+e^{-\alpha(s_{N_a}(F_\Theta,k^*)-s_{N_a}(F_\Theta,k))}\right).$$

Indeed, the condition $gr(j) < gr(i)$ holds for all $j \in \mathcal{K} \setminus \{k^*\}$ if and only if i corresponds to k^*. This result implies:

$$1 - \mathsf{SR}^1(F_\Theta, N_a) \leq \frac{1}{n_1} \sum_{\substack{k\in\mathcal{K} \\ k\neq k^*}} \log_2\left(1+e^{-\alpha(s_{N_a}(F_\Theta,k^*)-s_{N_a}(F_\Theta,k))}\right).$$

□

From Proposition 7.1.3.1, we can deduce that minimizing the ranking loss is useful to find a model that maximizes the 1st order success rate. Therefore, the value of the related empirical risk gives to the Evaluator an insight into how well F_Θ performs related to the success rate. Thus, the ranking loss fits with the performance measure classically considered during the evaluation process.

Finally, to fully understand the role of the ranking loss, we investigate the impact of the empirical risk minimization on the trainable parameters.

7.1.4 Impact of the Ranking Loss during the Training Process

This subsection theoretically explains how the training process can be useful in order to precisely order the secret key k^* amongst all the hypotheses.

The training process aims at minimizing the empirical risk $\hat{\mathcal{R}}$ in order to reduce the error induced by F_Θ (see Equation 4.3). As explained in Chapter 4, this process can be decomposed into two phases: the *forward propagation* and the *backward propagation*. As remainder, the goal of the forward propagation is to feed training examples to the Θ-parametric model F_Θ in the forward direction by processing successive linear and non-linear transformations in order to predict a distribution over $\mathcal{Y}$ associated with a given leakage trace $\mathbf{T}$. Once this process is done, the backward propagation measures the gap between the predictions and the expected output and reduces this error by updating the trainable parameters Θ that compose the neural network. To optimize the parameters Θ, the gradient descent algorithm, or its derivative (*e.g.* stochastic gradient descent), is employed (see Subsection 4.3.1). Let $\Theta_{(i)}^{[l]} \in \Theta$ be the trainable parameters indexed at the l^{th} layer such that, the $(i+1)^{\text{th}}$ iteration of the stochastic gradient descent algorithm can be defined as below:

$$\begin{aligned}
\Theta_{(i+1)}^{[l]} &= \Theta_{(i)}^{[l]} - \eta \cdot \nabla_{\Theta_{(i)}^{[l]}} \hat{\mathcal{R}}(\mathcal{L}_{RkL}, F_{\Theta_{(i)}}(\mathbf{T})) \\
&= \Theta_{(i)}^{[l]} - \eta \cdot \sum_{\substack{k\in\mathcal{K} \\ k\neq k^*}} \left(\nabla_{\Theta_{(i)}^{[l]}} s_{N_a}(F_{\Theta_{(i)}}, k^*) \frac{\partial \hat{\mathcal{R}}(\mathcal{L}_{RkL}, F_{\Theta_{(i)}}(\mathbf{T}))}{\partial s_{N_a}(F_{\Theta_{(i)}}, k^*)} \right. \\
&\quad \left. + \nabla_{\Theta_{(i)}^{[l]}} s_{N_a}(F_{\Theta_{(i)}}, k) \frac{\partial \hat{\mathcal{R}}(\mathcal{L}_{RkL}, F_{\Theta_{(i)}}(\mathbf{T}))}{\partial s_{N_a}(F_{\Theta_{(i)}}, k)} \right),
\end{aligned} \tag{7.8}$$

where η denotes the learning rate.
From Equation 7.4 and Equation 7.8, we can deduce the following equation,

$$\begin{aligned}
\nabla_{\Theta_{(i)}^{[l]}} \hat{\mathcal{R}}(\mathcal{L}_{RkL}, F_{\Theta_{(i)}}(\mathbf{T})) = -\alpha \sum_{\substack{k\in\mathcal{K} \\ k\neq k^*}} \left(\frac{1}{1+e^{\alpha\left(s_{N_a}(F_{\Theta_{(i)}},k^*)-s_{N_a}(F_{\Theta_{(i)}},k)\right)}} \right) \left(\nabla_{\Theta_{(i)}^{[l]}} s_{N_a}(F_{\Theta_{(i)}}, k^*) \right. \\
\left. - \nabla_{\Theta_{(i)}^{[l]}} s_{N_a}(F_{\Theta_{(i)}}, k) \right).
\end{aligned} \tag{7.9}$$

This derivative can be decomposed in two parts. First, computing the gradient of the empirical risk combined with the ranking loss is equivalent to computing an ascent gradient of the score $s_{N_a}(F_\Theta, k^*)$ and a gradient descent of $s_{N_a}(F_\Theta, k)$. As mentioned in Subsection 7.1.2, the score value is defined by the prediction before the softmax function. The training process updates the weights to increase the score related to the secret key and reduces the score related to all the hypotheses $k \in \mathcal{K} \setminus \{k^*\}$. Secondly, the norm of the gradient vectors is scaled by $\gamma_\alpha(s_{N_a}(F_\Theta, k^*), s_{N_a}(F_\Theta, k)) = \frac{\alpha}{1+e^{\alpha(s_{N_a}(F_\Theta,k^*)-s_{N_a}(F_\Theta,k))}}$. Depending on the difference between $s_{N_a}(F_\Theta, k^*)$ and $s_{N_a}(F_\Theta, k)$, the resulted norm varies as below:

- If $s_{N_a}(F_\Theta, k) \gg s_{N_a}(F_\Theta, k^*)$, $\gamma_\alpha(s_{N_a}(F_\Theta, k^*), s_{N_a}(F_\Theta, k))$ tends to converge towards α, thus the norm of the gradient vector related to each score is maximized.
- If $s_{N_a}(F_\Theta, k) = s_{N_a}(F_\Theta, k^*)$, $\gamma_\alpha(s_{N_a}(F_\Theta, k^*), s_{N_a}(F_\Theta, k))$ tends to converge towards $\frac{\alpha}{2}$, thus the norm of the gradient vector related to each score is divided by 2.
- If $s_{N_a}(F_\Theta, k) \ll s_{N_a}(F_\Theta, k^*)$, $\gamma_\alpha(s_{N_a}(F_\Theta, k^*), s_{N_a}(F_\Theta, k))$ tends to converge towards 0, thus the norm of the gradient vector related to each score is minimized.

Therefore, the ranking loss proposed in Equation 7.5 pushes the score of the secret key up and pushes the score of the key hypotheses down via gradient ascent/descent on a pair of items. This is equivalent to maximizing the success rate. For each pair $(k^*, k)_{k\in\mathcal{K}}$, there are two "forces" at play. The force that each pair exerts is proportionate to the difference of their scores multiplied with α. Consequently, α should be carefully configured during the training process. The force applied on the secret key k^* is equal to the sum of the forces exerted on each pair. Consequently, using the ranking loss tends to order the secret key as the highest position which is equivalent to maximizing the success rate.

SUM UP...

Adapted from the "Learning to Rank" approach, the ranking loss penalizes the training process depending on the distance between the scores related to the secret key and the other key hypotheses included in $\mathcal{K} \setminus \{k^*\}$. To the minimization of the empirical risk combined with the ranking loss, the Evaluator wants to select a Θ-parametric function F_Θ that maximizes the 1^{st} order success rate given N_a attack leakage traces. Thus, the ranking loss is directly correlated with the performance metric classically considered by the Evaluator to assess the robustness of a cryptographic module against side-channel attacks. It notably allows to reduce the black-box issue of the discriminative approach in deep learning based side-channel context.

However, the Evaluator can question the suitability of the ranking loss with respect to the negative log-likelihood from a performance perspective. The next section proposes to highlight the benefits of the ranking loss from an information-theoretic purpose.

7.2 Analysis of Optimal Model

7.2.1 An Approximation of the Optimal Distinguisher

In [dCGRP19], de Chérisey *et al.* link the probability of success of an attack, denoted P_s, given a distinguisher F and the mutual information between a leakage trace $\mathbf{T}$ and a sensitive targeted variable $Y \in \mathbb{F}_2^n$ given a plaintext $X \in \mathbb{F}_2^n$, denoted $MI(\mathbf{T}; Y|X)$.

Theorem 7.2.1.1. *[dCGRP19, Theorem 1] The following inequality is always true for any distinguishing rule:*

$$H(K) - (1 - P_s)\log_2(2^n - 1) - H_2(P_s) \leq N_a \cdot MI(\boldsymbol{T}; Y|X),$$

where $P_s = \mathsf{Pr}[\sum_{k \in \mathcal{K}} \mathbb{1}_{s_{N_a}(F,k) > s_{N_a}(F,k^*)} = 0]$ *and* $H_2(P_s)$ *denotes the binary entropy of the probability of success of an attack.*

This inequality suggests that no attack can succeed to retrieve the secret key within $\left\lceil \frac{g(P_s)}{MI(\mathbf{T};Y|X)} \right\rceil$ attack leakage traces, with $g(P_s) = H(K) - (1 - P_s)\log_2(2^n - 1) - H_2(P_s)$. Following the work provided by de Chérisey [Che18, Section 3.2.2], we can notice that $(1-P_s)\log_2(2^n-1)+H_2(P_s) \geq H(K|\hat{K})$. Consequently, minimizing $(1 - P_s)\log_2(2^n - 1) + H_2(P_s)$ (or equivalently, maximizing P_s) is beneficial to converge $H(K|\hat{K})$ towards 0. Therefore, finding a distinguisher that maximizes the success rate can be beneficial to reduce the uncertainty on K. By reducing this uncertainty for a given number N_a of attack traces, we intuitively ease the ability of the Evaluator to recover the secret key. In other words, finding a distinguisher that maximizes the success rate can be beneficial to reduce the related number of attack traces. Through the consideration of the ranking loss, the Evaluator wants to generate such a model. More importantly, a model that maximizes the success rate can be linked with the optimal distinguisher. Indeed, following Objective 3.3.1.1 and Definition 3.3.1.4, a distinguisher $F : \mathcal{T} \rightarrow P(\mathcal{Y})$ is defined as optimal if, for any dataset set of N_a leakage traces $\mathcal{T} = \{\mathbf{t}_0, \ldots, \mathbf{t}_{N_a-1}\}$, the attack, that considers $(F(\mathbf{t}_i))_{0 \leq i < N_a}$, maximizes the success rate.
As, in this chapter, the Θ-parametric ranking model $F_\Theta : \mathcal{T} \rightarrow \mathcal{S}$, such that $\mathcal{S} \subseteq \mathbb{R}^{|\mathcal{Y}|}$, does not output a normalized probability distribution[e], the Evaluator has to apply a softmax function on the output scores in order to get an approximation of the true unknown $\mathsf{Pr}[Y|\mathbf{T}]$. This approximation is denoted by $\mathsf{Pr}[Y|\mathbf{T}, \Theta]$. Once the related model finds the trainable parameters Θ, the Evaluator can apply the distinguisher rule with respect to the log-likelihood distinguisher (see Definition 3.3.3.4) in order to extract the secret key that has been manipulated by the cryptographic module.

Definition 7.2.1.1 (Approximation of the Optimal Distinguisher)**.** Given a set $\mathcal{I}_a$ of N_a leakage traces and a Θ-parametric conditional probability distribution of observing a sensitive cryptographic primitive Y following a leakage $\mathbf{T}$ denoted as $\mathsf{Pr}[Y|\mathbf{T}, \Theta]$, we define the approximation of the optimal distinguisher as:

$$\begin{aligned} \mathcal{D}_\Theta(\mathcal{I}_a, k) &= \max_\Theta \sum_{i=0}^{N_a-1} \log\left(\mathsf{Pr}\left[Y = f(x_i, k) \,|\mathbf{T} = \mathbf{t}_i, \Theta\right]\right) \\ &= \min_\Theta \sum_{i=0}^{N_a-1} \log\left(1 - \mathsf{Pr}\left[Y = f(x_i, k) \,|\mathbf{T} = \mathbf{t}_i, \Theta\right]\right), \end{aligned} \tag{7.10}$$

where $f(X, k) = Y$ is the sensitive information computed from a cryptographic primitive $f : \mathcal{X} \times \mathcal{K} \rightarrow \mathcal{Y}$, a plaintext X, a key $k \in \mathcal{K}$.

Once the approximation optimal distinguisher is performed, the Evaluator has to design the related decision rule such that:

$$\hat{k} = \arg\min_{k \in \mathcal{K}} \left(\min_\Theta \sum_{i=0}^{N_a-1} \log\left(1 - \mathsf{Pr}\left[Y = f(x_i, k) \,|\mathbf{T} = \mathbf{t}_i, \Theta\right]\right) \right),$$

such that the attack process is considered as a success if $\hat{k} = k^*$.
When the empirical risk combined with the ranking loss is considered, the training process optimizes the Θ trainable parameters in order to maximize the score of the secret key $s_{N_a}(F_\Theta, k^*)$ amongst each key hypothesis $k \in \mathcal{K} \setminus k^*$ for a given number N_a of attack leakage. Thus, a ranking model $F_\Theta : \mathcal{T} \rightarrow \mathcal{S}$, such that $\mathcal{S} \subseteq \mathbb{R}^{|\mathcal{Y}|}$, considering the ranking loss and the softmax function, that normalizes the output score vector to a probability distribution over $\mathcal{Y}$, during the attack

[e] As mentioned in Definition 7.1.2.1, the output score can also be characterized by $\mathcal{P}(\mathcal{Y})$ in order to penalize the network with respect to the normalized probability distribution.

phase is beneficial to fit with Definition 7.2.1.1. Furthermore, through Equation 7.10, we can observe that minimizing the error on the success rate is helpful to estimate the optimal distinguisher defined in Definition 3.3.1.4. To converge towards the true distinguisher, some optimization algorithms (*e.g.* the gradient-based algorithms, Subsection 4.3.1) shall be run in order to minimize the error on $\mathsf{Pr}\left[Y|\mathbf{T},\Theta\right]$ and find a local or a global minimum. When Θ is optimal, finding a model F_Θ that minimizes the error on the success rate is equivalent to generating a distinguisher $\left(\mathcal{D}_\Theta\left(\mathcal{I}_a,k\right)\right)_{k\in\mathcal{K}}$ that converges towards the true $\left(\mathcal{D}\left(\mathcal{I}_a,k\right)\right)_{k\in\mathcal{K}}$. Through Definition 2.5.2, we can assume that minimizing the error on the success rate is equivalent to optimizing an estimation of a mutual information between the sensitive information and the leakage trace. Indeed, given a profiling set $\mathcal{I}_p$ of N_p pairs $(\mathbf{t}_i,y_i)_{0\leq i\leq N_p}$, a Θ-parametric ranking model F_Θ and a number of attack traces N_a such that $N_a|N_p$, the following inequality holds:

$$\begin{aligned} H\left(Y\right)+\sum_{y\in\mathcal{Y}}\mathsf{Pr}\left[Y=y\right]\frac{N_a}{N_p}\sum_{i=1}^{N_p/N_a}\max_{\Theta}\left(\sum_{j=0}^{N_a-1}\log_2\left(\mathsf{Pr}\left[Y=y|\mathbf{t}_{j+N_a\cdot(i-1)},\Theta\right]\right)\right) \\ \leq H\left(Y\right)+\sum_{y\in\mathcal{Y}}\mathsf{Pr}\left[Y=y\right]\sum_{\mathbf{t}\in\mathcal{T}}\mathsf{Pr}\left[\mathbf{T}=\mathbf{t}|y\right]\ \log_2\left(\mathsf{Pr}\left[Y=y|\mathbf{t}\right]\right)=MI\left(Y;\mathbf{T}\right). \end{aligned} \tag{7.11}$$

Hence, following Theorem 7.2.1.1, finding a ranking model F_Θ that maximizes the success rate aims at simultaneously minimizing an approximation of $g(P_s)$ while maximizing an approximation of $MI(\mathbf{T};Y|X)$. As mentioned in [DFS15, BHM+19], a higher mutual information implies a more powerful maximum likelihood attack where the secret key k^* can be extracted more efficiently. In other words, the ranking model F_Θ optimizes the number of attack leakage traces that are needed to retrieve the secret key manipulated by the cryptographic module. Thus, from an evaluation perspective, the ranking loss can be highly beneficial.

In contrast, the Evaluator can question the interest of this approach with respect to the negative log-likelihood loss function combined with the softmax function.

7.2.2 The Negative Log-Likelihood as a Lower Bound of the Ranking Loss

Recall on the negative log-likelihood minimization. In deep learning based side-channel context, a classical model $F_{\Theta,NLL}:\mathcal{T}\rightarrow\mathcal{P}(\mathcal{Y})$ is selected from the minimization of the empirical risk combined with the negative log-likelihood loss function. To solve the related classification task, the Θ-parametric model $F_{\Theta,NLL}$ uses the softmax activation function to learn a probability distribution over $|\mathcal{Y}|$ labels conditioned on a leakage trace $\mathbf{T}$. Indeed, given a set of N_p labeled leakage traces $\mathcal{I}_p=\{(\mathbf{t}_0,y_0),(\mathbf{t}_1,y_1),\ldots,(\mathbf{t}_{N_p-1},y_{N_p-1})\}$, and a parametric model F_Θ estimating the true unknown $\mathsf{Pr}\left[Y|\mathbf{T}\right]$, the empirical risk combined with the negative log-likelihood loss function can be expressed as:

$$\hat{\mathcal{R}}(\mathcal{L}_{NLL},F_{\Theta,NLL})=-\frac{1}{N_p}\sum_{i=0}^{N_p-1}\log_2\left(\frac{e^{s_{1,i}(F_{\Theta,NLL},k^*)}}{\sum_{k\in\mathcal{K}}e^{s_{1,i}(F_{\Theta,NLL},k)}}\right), \tag{7.12}$$

where $s_{1,i}(F_{\Theta,NLL},k)$ denotes the score related to the key hypothesis $k\in\mathcal{K}$ such that a leakage trace $(\mathbf{t}_i)_{0\leq i<N_p}$ is considered as input. Here, the score defines the value before the softmax activation function.
Since the loss function is minimized by gradient descent based algorithms (*e.g.* stochastic gradient descent, mini-batch gradient descent), the related derivative can be easily calculated as follows:

$$\frac{\partial\mathcal{L}_{NLL}(F_{\Theta,NLL},k^*)}{\partial s_1(F_\Theta,j)}=\begin{cases}1-\dfrac{e^{s_1(F_{\Theta,NLL},j)}}{\sum_{k\in\mathcal{K}}e^{s_1(F_{\Theta,NLL},k)}} & \text{if}\quad j=k^*,\\[2ex] -\dfrac{e^{s_1(F_{\Theta,NLL},j)}}{\sum_{k\in\mathcal{K}}e^{s_1(F_{\Theta,NLL},k)}} & \text{if}\quad j\neq k^*.\end{cases} \tag{7.13}$$

Thus, applying the negative log-likelihood loss function on a Θ-parametric model that considers the softmax to approximate a probability distribution is helpful to penalize the trainable parameters linked to each output class. However, contrary to the ranking loss that penalizes the training process with respect to the relative order of each pair $(k, k^*)_{k\in\mathcal{K}}$, the negative log-likelihood with the softmax function considers the absolute value of each score. Thus, in side-channel context, the latter loss function can provide different sources of errors.

Difference on the penalization process. Comparing Equation 7.9 to Equation 7.13 gives to the Evaluator an insight into how each approach impacts the trainable parameters Θ. Typically, two scenarios can be observed:

- If $j \neq k^*$, $\frac{\partial \mathcal{L}_{NLL}(F_{\Theta,NLL},k^*)}{\partial s_1(F_{\Theta,NLL},j)}$ penalizes the training process with respect to the probability related to the j^{th} class. Thus, it does not reflect its relative order with the targeted output. In opposition, for $\alpha = 1$, the derivative of the ranking loss defined in Equation 7.9 can be rewritten as follows:

$$-\frac{\partial \mathcal{L}_{RkL}(F_\Theta, k^*)}{\partial s_{N_a}(F_\Theta, j)} = -\frac{e^{s_{N_a}(F_\Theta,j)}}{\sum_{k\in\{j,k^*\}} e^{s_{N_a}(F_\Theta,k)}}. \tag{7.14}$$

  This penalization term is very similar to Equation 7.13. However a non-negligible distinction can be made. While the ranking loss penalizes the trainable parameters of the j^{th} class with respect to the distance between its score and the one related to the correct output, the negative log-likelihood loss updates its trainable parameters with respect to all key hypotheses in $\mathcal{K}$. Thus, the penalization term applied on the j^{th} class can be overlooked if the score $s_1(F_{\Theta,NLL}, j) \ll s_1(F_{\Theta,NLL}, i)$ such that $i \in \mathcal{K} \setminus \{j, k^*\}$. Thus, even if the relative order between $s_1(F_{\Theta,NLL}, j)$ and $s_1(F_{\Theta,NLL}, k^*)$ is non-negligible, it will not be taken into account during the training process. Hence, for a given iteration, the gradient descent-based algorithms can only penalize a limited number of output classes if a large difference is observed between their related scores. This result does not reflect what the Evaluator wants to optimize and a blurred parametric model can be obtained. In opposition, the ranking loss is not impacted by this scenario as it does not consider irrelevant classes for updating the trainable parameters of a given class.

- If $j = k^*$, $\frac{\partial \mathcal{L}_{NLL}(F_{\Theta,NLL},k^*)}{\partial s_1(F_{\Theta,NLL},j)}$ penalizes the training process with respect to the probability error induced in the class k^*. Thus, it reflects the impact of each score $(s_1(F_{\Theta,NLL}, k))_{k\in\mathcal{K}\setminus\{k^*\}}$ on $s_1(F_{\Theta,NLL}, k^*)$ which is beneficial from a success rate approximation perspective. On the other hand, the derivative of the ranking loss is defined by the sum over $\mathcal{K} \setminus \{k^*\}$ of the inverse of Equation 7.14. As mentioned in Subsection 7.1.4, the latter solution opposes two "forces" for each pair $(k^*, k)_{k\in\mathcal{K}}$ such that it penalizes the training process in order to optimize the score related to k^* with respect to each key hypothesis $k \in \mathcal{K}$. Thus, both solutions take into account $(s(F_\Theta, k))_{k\in\mathcal{K}\setminus\{k^*\}}$ to compute the gradient with respect to k^*. They are intuitively useful to correctly optimize the score k^* amongst all key hypotheses which is necessary to perform a successful side-channel attack.

While the penalization process related to the scenario $j = k^*$ is useful for the maximization of the score $s(F_\Theta, k^*)$, some limitations can be highlighted when the negative log-likelihood with the softmax activation function is considered. Indeed, when $j \neq k^*$, the derivative $\frac{\partial \mathcal{L}_{NLL}(F_{\Theta,NLL},k^*)}{\partial s_1(F_{\Theta,NLL},j)}$ does not adequately penalize the trainable parameters related to the j^{th} class. Consequently, the probability term induced in Equation 7.12 can be impacted by a wrong penalization conducted on each $(s_1(F_{\Theta,NLL}, j))_{j\in\mathcal{K}\setminus k^*}$. Consequently, the loss function may emphasize irrelevant sensitive information. Thus, contrary to the ranking model obtained from the minimization of the empirical risk combined with the ranking loss, the model selected from the minimization of the empirical risk combined with the negative log-likelihood can be seen as an approximation of $\mathsf{Pr}[Y|\mathbf{T}, \Theta]$. In the

following, the distance between the probability distribution provided by $F_{\Theta,NLL}$ and $\mathsf{Pr}[Y|\mathbf{T},\Theta]$ is called *Approximation Error* and is negligible if and only if the probability distribution associated with $F_{\Theta,NLL}$ is similar to the one related to the ranking model.

The link with the Mutual Information. As the approximation error suggests that a ranking model is at least, as efficient as a model $F_{\Theta,NLL}$ using the negative log-likelihood loss function, the latter solution can be considered as a lower bound of the inequality introduced in Equation 7.11:

$$\begin{aligned}
&H(Y) + \sum_{y\in\mathcal{Y}} \mathsf{Pr}[Y=y] \frac{1}{N_p} \sum_{i=0}^{N_p} \log_2 \left(F_{\Theta,NLL}(\mathbf{t}_i)[y]\right) \\
&\leq H(Y) + \sum_{y\in\mathcal{Y}} \mathsf{Pr}[Y=y] \frac{N_a}{N_p} \sum_{i=1}^{N_p/N_a} \max_{\Theta} \left(\sum_{j=0}^{N_a-1} \log_2 \left(\mathsf{Pr}\left[Y=y|\mathbf{t}_{j+N_a\cdot(i-1)},\Theta\right]\right)\right) \\
&\leq H(Y) + \sum_{y\in\mathcal{Y}} \mathsf{Pr}[Y=y] \sum_{\mathbf{t}\in\mathcal{T}} \mathsf{Pr}[\mathbf{T}=\mathbf{t}|y] \log_2 \left(\mathsf{Pr}[Y=y|\mathbf{t}]\right) = MI(Y;\mathbf{T}). \qquad (7.15)
\end{aligned}$$

Consequently, when the negative log-likelihood is used as a loss function, the number of leakage traces needed to reach a 1st order success rate is defined as an upper bound of the optimal solution (see Theorem 7.2.1.1). Moreover, due to the approximation error, the number of attack leakage traces needed to perform a successful attack on $F_{\Theta,NLL}$ is defined as an upper bound of the number of attack leakage traces needed to perform a successful attack on a model selected from the minimization of the empirical risk combined with the ranking loss. Thus, it illustrates that the latter solution is more efficient than the current usual negative log-likelihood loss function used in the side-channel context.

In the next section, we deeply analyze the errors made by the negative log-likelihood and we compare them with the ranking loss.

7.2.3 Error Analysis

The negative log-likelihood in side-channel context. In [MDP19b], Masure *et al.* study the theoretical soundness of the minimization of the empirical risk combined with the negative log-likelihood in order to conduct side-channel attacks. They demonstrate that minimizing this empirical risk is equivalent to maximizing an estimation of the *Perceived Information* [RSVC+11] that is defined as a lower bound of the *Mutual Information* between a leakage trace and the targeted secret [BHM+19]. Specifically, the perceived information is the amount of information that can be extracted from data with the help of an estimated model. Because the perceived information is substituted to the mutual information, some source of imprecision could affect the quality of the model F_Θ. The gap between the estimation of the perceived information, defined by the minimization of the empirical risk combined with the negative log-likelihood, and the mutual information can be decomposed into three errors[f]:

- **Approximation error** – it defines the deviation between the perceived information obtained from a parametric model F_Θ and the mutual information. In theory, the Kullback-Leibler divergence [KL51] can be computed in order to evaluate this deviation. However, the Evaluator faces the problem that the leakage *Probability Density Function* (PDF) is unknown.

- **Estimation error** – the minimization of the empirical risk combined with the negative log-likelihood maximizes the empirical estimation of the perceived information rather than the real value of the perceived information. Therefore, the finite set of profiling leakage

[f]The readers should notice that these errors differ from those introduced by Masure *et al.* in [MDP19b].

traces may be too low to estimate the perceived information properly. Consequently, this error can be quantified for a given number of profiling leakage traces.

- **Optimization error** – it characterizes the error induced by the learning process, such as the choice of a restricted model space $\mathcal{F}$ that does not necessarily contain the optimal model F^*, and the ability of the gradient-based algorithm (*e.g.* stochastic gradient descent, mini-batch gradient descent, Adam optimizer) to converge towards the optimal solution.

Section 7.3 emphasizes the impact of these error terms on different datasets. First, this section recalls the gap between the negative log-likelihood loss function and the mutual information. Then, we explain the theoretical benefits of the ranking loss through an error analysis between the ranking loss, the negative log-likelihood and the mutual information. To assess the quality of the ranking loss, we have to evaluate the tightness of the inequalities defined in Equation 7.15. To ease the following analysis, let $\widehat{PI}(Y, \mathbf{T}, \hat{\Theta})$ and $\widehat{MI}(Y, \mathbf{T}, \hat{\Theta})$ be defined as follows[g]:

$$\widehat{PI}(Y, \mathbf{T}, \hat{\Theta}) = H(Y) + \sum_{y \in \mathcal{Y}} \mathsf{Pr}[Y = y] \frac{1}{N_p} \sum_{i=0}^{N_p} \log_2 \left(F_{\hat{\Theta}, NLL}(\mathbf{t}_i)[y] \right).$$

$$\widehat{MI}(Y, \mathbf{T}, \hat{\Theta}) = H(Y) + \sum_{y \in \mathcal{Y}} \mathsf{Pr}[Y = y] \frac{N_a}{N_p} \sum_{i=1}^{N_p/N_a} \max_{\hat{\Theta}} \left(\sum_{j=0}^{N_a - 1} \log_2 \left(\mathsf{Pr}\left[Y = y | \mathbf{t}_{j+N_a \cdot (i-1)}, \hat{\Theta} \right] \right) \right).$$

Error Analysis of the negative log-likelihood. Proposed in [MDP19b], this error decomposition establishes the gap between the mutual information and the perceived information that we are maximizing with the negative log-likelihood. To facilitate the comparison of our results with [MDP19b], we use the same notations. Let $\hat{\Theta}$ denotes an estimation of the trainable parameters when a gradient descent-based algorithm is used (*e.g.* stochastic gradient descent, mini-batch gradient descent) and Θ^* the optimal trainable parameters obtained when a global minimum of the loss function is reached and such that the optimal model F^* is selected. According to Equation 7.15, this error gap can be decomposed into the three errors previously introduced:

$$\widehat{PI}\left(Y; \mathbf{T}; \hat{\Theta}\right) - MI(Y; \mathbf{T}) = \left(\widehat{PI}\left(Y; \mathbf{T}; \hat{\Theta}\right) - \widehat{PI}(Y; \mathbf{T}; \Theta^*) \right) \tag{7.16}$$

$$+ \left(\widehat{PI}(Y; \mathbf{T}; \Theta^*) - PI(Y; \mathbf{T}; \Theta^*) \right) \tag{7.17}$$

$$+ \left(PI(Y; \mathbf{T}; \Theta^*) - MI(Y; \mathbf{T}) \right). \tag{7.18}$$

such that Equation 7.16 defines the optimization error, Equation 7.17 defines the estimation error and Equation 7.18 denotes the approximation error.
The approximation error is negligible if and only if the Evaluator obtains the true distribution $\mathsf{Pr}[Y|\mathbf{T}]$ through the empirical risk minimization. Through Section 7.3, we will show that, even in the simplest case, the approximation error can have a huge impact on the training process such that some irrelevant features could be defined as points of interest. Consequently, these errors could highly impact the performance of the network.

Error analysis of the ranking loss. In order to complete the latter analysis, we estimate the errors generated when $\widehat{MI}$ and MI are taken into consideration. Through Equation 7.15, it can be easily observed that the gap between the estimated and the true mutual information can be decomposed into an estimation error and an optimization error as below:

$$\widehat{MI}\left(Y; \mathbf{T}; \hat{\Theta}\right) - MI(Y; \mathbf{T}) = \left(\widehat{MI}\left(Y; \mathbf{T}; \hat{\Theta}\right) - \widehat{MI}(Y; \mathbf{T}; \Theta^*) \right) \tag{7.19}$$

$$+ \left(\widehat{MI}(Y; \mathbf{T}; \Theta^*) - MI(Y; \mathbf{T}) \right). \tag{7.20}$$

[g]The notation $\widehat{MI}(Y, \mathbf{T}, \hat{\Theta})$ differs from [BHM+19]. In this manuscript, this notation is used to differentiate the probability distribution obtained from $F_{\Theta, NLL}$ and a ranking model. It notably useful to denote the inequality introduced in Equation 7.15.

Indeed, Equation 7.19 defines the error related to the optimization of the model such that the ranking loss is considered. Finally, Equation 7.20 characterizes the estimation error that can be reduced when the number of profiling traces converges towards infinity [BHM+19]. In Subsection 7.3.2, we will experimentally see that this error is reduced by up to 23% when the ranking loss is used compared to the classical categorical cross-entropy.

Error gap between the negative log-likelihood and the ranking loss. Finally, the gap between the negative log-likelihood and the ranking loss can be divided into different error terms. Here, we assume that the optimization error generated by both models is similar. This strong assumption is useful to simplify our analysis and focus on the benefits of using the ranking loss compared to the negative log-likelihood with softmax activation function. Consequently,

$$\begin{aligned}\widehat{PI}\left(Y;\mathbf{T};\hat{\Theta}\right) - \widehat{MI}\left(Y;\mathbf{T};\hat{\Theta}\right) &= \left(\widehat{PI}\left(Y;\mathbf{T};\hat{\Theta}\right) - \widehat{PI}\left(Y;\mathbf{T};\Theta^*\right)\right) \\ &- \left(\widehat{MI}\left(Y;\mathbf{T};\hat{\Theta}\right) - \widehat{MI}\left(Y;\mathbf{T};\Theta^*\right)\right) \\ &+ \left(\widehat{PI}\left(Y;\mathbf{T};\Theta^*\right) - \widehat{MI}\left(Y;\mathbf{T};\Theta^*\right)\right) \\ &= \left(\widehat{PI}\left(Y;\mathbf{T};\Theta^*\right) - \widehat{MI}\left(Y;\mathbf{T};\Theta^*\right)\right). \end{aligned} \tag{7.21}$$

Equation 7.21 defines that the difference of the models lies in the approximation error generated between $\widehat{PI}\left(Y;\mathbf{T};\Theta^*\right)$ and $\widehat{MI}\left(Y;\mathbf{T};\Theta^*\right)$. Indeed, the approximation error that defines the distance between the perceived information and the mutual information is removed when the success rate is maximized. One of the most challenging issues, induced by this approximation error, is then prevented when the ranking loss is considered.

SUM UP...

Through this section, we demonstrate that a ranking model selected from the minimization of the empirical risk combined with the ranking loss can be considered as an approximation of the optimal distinguisher (see Definition 3.3.1.4). From this observation, we evaluate the distinctions between the penalization process induced by the negative log-likelihood and the ranking loss. While both approaches are useful to maximize the correct output with respect to each key hypothesis $k \in \mathcal{K} \setminus k^*$, the application of the negative log-likelihood does not properly penalize the irrelevant classes. Consequently, the probability distribution provided by $F_{\Theta,NLL}$ induces an approximation error as its optimal model does not necessarily reflect the true unknown probability distribution $\mathsf{Pr}[Y|\mathbf{T}]$. This error is prevented by the ranking loss function. Thus, according to Theorem 7.1.3.1, a model obtained from the minimization of the empirical risk combined with the ranking loss is at least as efficient as a model obtained from a classical deep learning based side-channel approach.

In the next section, we validate all the theoretical observations on unprotected and protected implementations.

7.3 Exploitation of the Ranking Loss

Settings. The experiments are implemented in Python using the *Keras* library [Cho15] and are run on a workstation equipped with 16GB RAM and a NVIDIA GTX1080Ti with 11GB memory. All of the following architectures and hyperparameters are based on the results provided in Section 6.3 and can be reproduced [ZBD+20a]. The attack leakage traces are randomly shuffled and picked up from a set of attack leakage traces $\mathcal{I}_a$. For a good estimation of Nt_{rank}, 100 realizations are performed to give the average value denoted $\bar{Nt}_{\text{rank}}$. In the next sections, the number N_a of attack leakage traces considered to compute the ranking loss (see Definition 7.1.2.1), is set to 1. Hence, the ranking loss tends to maximize the success rate when only 1 attack leakage trace is considered.

Comparison with the Cross-Entropy Ratio [ZZN+20, ISUH21]. In the following, we compare the negative log-likelihood and the cross-entropy ratio (CER) loss functions with the ranking loss on different publicly available datasets. Given a set $\mathcal{I}_p$ of N_p elements such that $\mathcal{I}_p = \{(\mathbf{t}_0, y_0), \ldots, (\mathbf{t}_{N_p-1}, y_{N_p-1})\}$ with $(\mathbf{t}_i, y_i)_{0 \leq i < N_p} \in (\mathcal{T} \times \mathcal{Y})$, the empirical risk combined with the cross-entropy ratio has been introduced to deal with imbalanced data issues such that it is defined as:

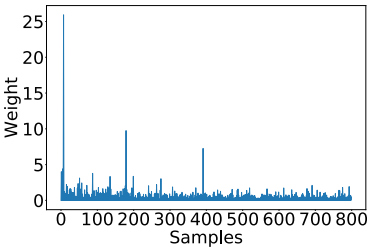
$$\hat{\mathcal{R}}(\mathcal{L}_{CER}, F_\Theta) = \frac{\hat{\mathcal{R}}(\mathcal{L}_{NLL}, F_\Theta)}{-\frac{1}{N}\sum_{r=0}^{N-1} \hat{\mathcal{R}}(\mathcal{L}_{NLL}^{(r)}, F_\Theta)},$$

where $\hat{\mathcal{R}}(\mathcal{L}_{NLL}^{(r)}, F_\Theta)$ identifies the empirical risk combined with the negative log-likelihood such that the labels are shuffled while keeping the leakage traces unchanged.

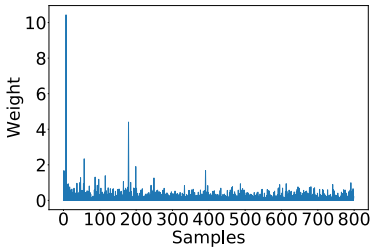
From a practical perspective, the generation of suitable architectures is known as a difficult task. Hence, two scenarios can be considered. In Subsection 7.3.1, it is assumed that the Evaluator selects a model that exploits a partial set of PoIs. In Subsection 7.3.2, models that exploit all relevant information in the leakage traces are considered. This section evaluates the efficiency of the ranking loss compared to the negative log-likelihood and the cross-entropy ratio in both cases through various scenarios, notably in presence of high noise, masking and desynchronization.

7.3.1 A Partial Exploitation of the Leakages

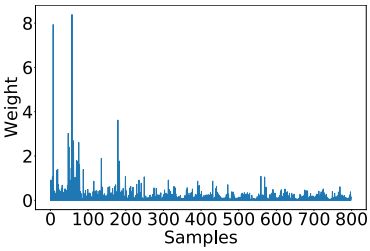
For simplicity, we evaluate this case study with the ChipWhisperer dataset which is an unprotected emulation of AES-128 implemented in software on a Chipwhisperer board [OC14] (see Section 3.6 for deeper information). To construct the model, we follow the recommendations provided in Subsection 6.3.1. The implemented model is a CNN architecture with one convolutional block of 2 filters of size 1 and one fully-connected layer with 2 nodes. When considering unprotected implementation with low noise, all the models trained with different losses provide the same $\bar{N}t_{\text{rank}}$ value (see Table 7.1 for small σ values). In Subsection 6.1.3, we propose to visualize the weights corresponding to the flatten layer in order to evaluate the capacity of the network to extract the relevant features. Through this visualization, the Evaluator is able to retrieve the points of interest selected by a neural network. However, due to the effect of the convolutional block, the number of weighted samples is divided by the value of the pooling stride (see Subsection 6.2.3). Thus, the comparison of these visualizations with the SNR computation can be difficult. For ease of visualization, we add padding on the weight representation in order to get the same x-axis on each figures. In Figure 7.2, we compare the features retained by the negative log-likelihood (see Figure 7.2a), the cross-entropy ratio (see Figure 7.2b) and the ranking loss (see Figure 7.2c) with the classical SNR (see Figure 7.2d). In this section, the comparison with an univariate leakage metric, such as SNR, is sufficient to assess the benefits of each loss as no interactions between the time samples are induced in the convolutional part of the CNN (*i.e.* the length of the filters is set to 1).
Interestingly, depending on the loss, the model does not select the same relevant features. The Figure 7.2a, Figure 7.2b and Figure 7.2c do not show the same points of interest. While the SNR computation reveals 4 high peaks between 0 and 200 samples, the models trained with the negative log-likelihood and the cross-entropy ratio losses detect only 2 high peaks in the same area. Hence, only a partial set of leakages is exploited by these cross-entropy losses. In comparison, the ranking loss extracts most of the sensitive information. Moreover, the negative log-likelihood loss identifies a false-positive leakage while no irrelevant peak occurs when the ranking loss is applied. This error underlines an important issue when the negative log-likelihood loss is used in side-channel: the approximation error is non-negligible and some false-positive leakages can occur. If the Evaluator cannot find a more suitable neural network architecture, these noisy points (*i.e.* irrelevant PoIs) could dramatically impact the performance of the network. As mentioned in [DSVC14], the approximation (or *assumption*) error can be dramatic if the model, characterizing the perceived information, does not converge towards the right distribution $\mathsf{Pr}[Y|\mathbf{T}]$ defined by



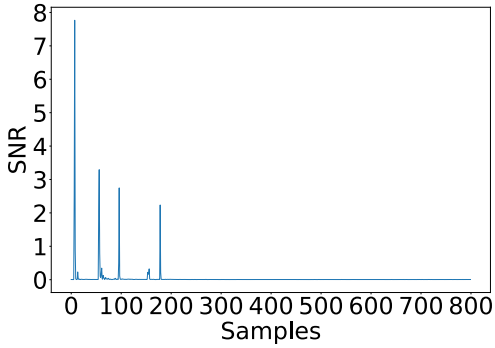
(a) Weight visualization (**Negative Log-Likelihood** (NLL))



(b) Weight visualization (**Cross-Entropy Ratio** (CER))



(c) Weight visualization (**Ranking Loss** (RkL) ($\alpha = 5.0$))



(d) SNR of the **Chipwhisperer** dataset

Figure 7.2: Visualization of the approximation error (**Chipwhisperer**)

the mutual information $MI(Y, \mathbf{T})$. In Subsection 7.2.3, we have shown that the ranking loss prevents the approximation error compared to the negative log-likelihood. Hence, when the ranking loss is used, the related performance should be, at least, as good as a model trained with the negative log-likelihood.

If the PoIs amplitude is low compared to the noise, the performance gap between a model trained with the negative log-likelihood, the cross-entropy ratio and the ranking loss could increase. To illustrate this phenomenon, we add Gaussian noise $\mathcal{N}_D(0, \sigma \cdot \mathbf{I}_D)$ such that σ defines the standard deviation of the noise and D denotes the dimension of the leakage traces. Table 7.1 shows the evolution of the $\bar{N}t_{\text{rank}}$ value depending on the added noise on the Chipwhisperer dataset. When the additional noise level is low (*i.e.* $\sigma \leq 10^{-2}$), the feature detection is effective regardless of the loss function and the performance gap is low (*i.e.* less than 9). However, for high noise level (*i.e.* $\sigma \geq 10^{-1}$), the performance gap increases dramatically and reaches $1,031$ when we compare the negative log-likelihood and the ranking loss and 885 when we compare the cross-entropy ratio and the ranking loss. The ranking loss is clearly the most efficient loss function, even in the presence of high noise levels.

Table 7.1: Evolution of $\bar{N}t_{\text{rank}}$ depending on σ (average over 10 converging models)

$\bar{N}t_{\text{rank}}$ \ σ	0	10^{-6}	10^{-5}	10^{-4}	10^{-3}	10^{-2}	10^{-1}	**Training time (seconds)**
Negative Log-Likelihood	4	4	6	6	7	21	3,958	81 s
Cross-Entropy Ratio	4	5	6	8	17	20	3,812	143 s
Ranking Loss	**3**	**3**	**3**	**3**	**3**	**13**	**2,927**	**294 s**

In conclusion, if the Evaluator generates a model that does not exploit the entire set of leakages, he shall use the ranking loss in order to obtain a model mitigating the approximation error. Indeed,

depending on the level of the SNR peaks, this error can dramatically impact the performance of a network. However, from a practical perspective, a model trained with the ranking loss can also extract false-positive leakages due to the optimization error. But, if the hyperparameter α is correctly configured, its overall error rate stays a lower bound of the error rate generated by a model trained with the negative-log likelihood loss function (see Subsection 7.2.3) with the assumption that the optimization errors are similar. The evaluation of the approximation error was also made on the AES_HD and the ASCAD datasets but the architectures proposed in Subsection 6.3.1 already give the same best solution for all the losses. For these datasets, we can assume that the approximation error is negligible. Hence, we assume that all losses exploit the entire set of relevant information. The next section evaluates the benefits of the ranking loss against the negative log-likelihood and the cross-entropy ratio when all leakages are detected.

Remark 7.3.1.1. In this experiment, we noticed that when the noise level is high, the best value of α used by the ranking loss decreases. Consequently, α is configured to obtain the most powerful model when the noise level is high. Even if the resulted performance is similar for many values of α, this observation illustrates that α should be correctly configured depending on the characteristic of the traces (*i.e.* level of noise, number of profiling traces ...).

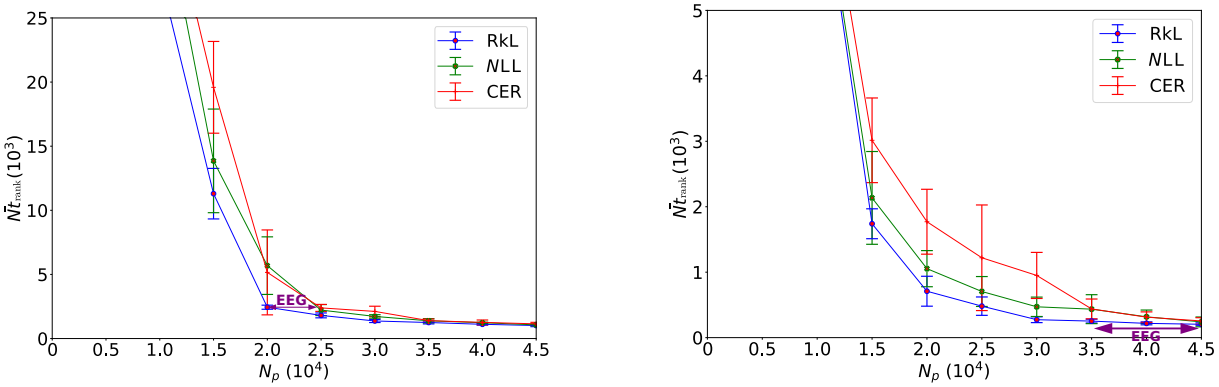
7.3.2 A Total Exploitation of the Leakages

As mentioned, given the architecture provided in Section 6.3, the entire set of PoIs is detected on the AES_HD and the ASCAD datasets (see Figure 6.7 and Figure 6.8). Hence, we can assume that the approximation error does not impact the overall performance of the model regardless of the loss function used. When all the losses converge towards the same best solution, a comparison method consists in the evaluation of the number of profiling leakage traces that are needed to reach this performance. From the Evaluator point of view, it is more interesting to converge faster towards the best solution because it is difficult to estimate *a priori* the number of profiling leakage traces needed to reach the best performance. To highlight the benefits of each loss, we decompose this experimental study into an *Estimation Error Gap* (EEG) and a performance gap evaluation. When various losses are used, the EEG characterizes the difference between the number of profiling leakage traces N_p that are needed to reach a given $\bar{N}t_{\text{rank}}$ value. We note $\mathsf{EEG}(\mathcal{L}_i, \mathcal{L}_j)$ the EEG value between models obtained from the loss functions $\mathcal{L}_i$ and $\mathcal{L}_j$. For each dataset, we average the performance results given by 10 models converging towards a constant guessing entropy of 1. Then, we display the evolution of the average $\bar{N}t_{\text{rank}}$ value for different levels of N_p. When the number of profiling traces is low (*i.e.* $N_p \leq 30,000$), some models do not retrieve the sensitive information and the resulted $\bar{N}t_{\text{rank}}$ value cannot be estimated. In order to fairly compare the losses, we only consider the models for which the $\bar{N}t_{\text{rank}}$ value can be computed for all the learning metrics.

AES_HD. In Figure 7.3a, we compare the convergence capacity of each model depending on the loss used. When the model is obtained from the ranking loss, only $20,000$ profiling leakage traces are needed to perform a successful attack such that $\bar{N}t_{\text{rank}} = 2,500$. To reach the same performance, a model trained with the negative log-likelihood needs $24,700$ profiling traces. Thus, when $\bar{N}t_{\text{rank}} = 2,500$, $\mathsf{EEG}(\mathcal{L}_{RkL}, \mathcal{L}_{NLL}) = 4,700$. Similarly, if the Evaluator chooses the cross-entropy ratio as loss function, he needs to increase its training set by $4,800$ traces to perform similar attacks. When the ranking loss is used, the number of profiling traces needed to reach a constant $\bar{N}t_{\text{rank}}$ solution is, in the worse case, similar to the cross-entropy propositions (*i.e.* negative log-likelihood, cross-entropy ratio). Through Table 7.2, we compare the performance of each loss for a given number of profiling traces. When N_p is low (*i.e.* $\leq 30,000$), the performance gap is relatively high (up to $8,293$ traces) between the ranking loss and the cross-entropy losses. Hence, when the number of profiling leakage traces is limited (as often in practice), the ranking loss is the most efficient loss function. However, as defined in [MDP19b], if the number of profiling traces is large enough and no approximation error occurs, the performance gap is reduced and the negative log-likelihood loss function generates a model that converges towards the same best

solution (see Table 7.2). The same observation can be made if we consider the cross-entropy ratio loss function. These experimental results confirm the theoretical propositions of Subsection 7.2.3 such that the ranking loss is, at least, as efficient as a model trained with a cross-entropy loss function (*e.g.* negative log-likelihood, cross-entropy ratio).

Remark 7.3.2.1. In comparison with the negative log-likelihood function, the training time can be impacted when the ranking loss is considered. Following Definition 7.1.2.1, the partial ranking loss function has to be summed over $\mathcal{K}$. Hence, the training time needed to reach a given number of epochs could be increased depending on the leakage model (*e.g.* identity, Hamming Weight). The lesser the number of classes (or key hypotheses), the lesser the training time is impacted. For the AES_HD dataset, the worst-case scenario is considered when 8 bits are targeted (*i.e.* $|K| = 256$). The best-case scenario will be considered in Chapter 8 (*i.e.* $|K| = 2$ and $|K| = 3$). In comparison with the cross-entropy losses, the training time increased by $10s$, when $N_p = 15,000$, and by up to $145s$ for $45,000$ profiling leakage traces. In the worst-case scenario (*i.e.* $N_p = 45,000$), the training time is multiplied by 4.



(a) AES_HD **(b) ASCAD**

Figure 7.3: Evolution of $\bar{N}t_{\textbf{rank}}$ depending on N_p (average over 10 converging models - synchronized datasets)

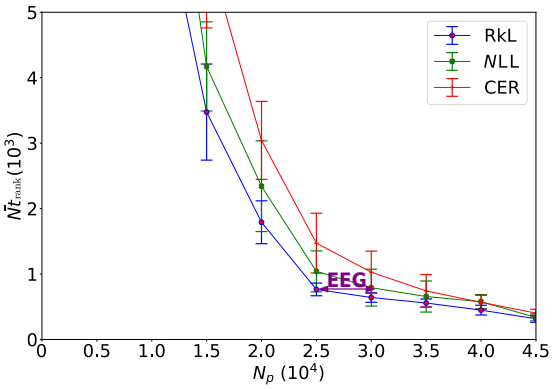
Table 7.2: Evolution of $\bar{N}t_{\text{rank}}$ depending on the number of profiling traces N_p (AES_HD - average over 10 converging models)

$\bar{N}t_{\text{rank}}$ \ N_p	10,000	15,000	20,000	25,000	30,000	35,000	40,000	45,000
Negative Log-Likelihood	>25,000	13,855	5,685	2,220	1,725	1,385	1,235	1,165
Cross-Entropy Ratio	>25,000	19,591	5,158	2,390	2,115	1,397	1,259	1,206
Ranking Loss	**>25,000**	**11,298**	**2,443**	**1,805**	**1,370**	**1,280**	**1,210**	**1,115**

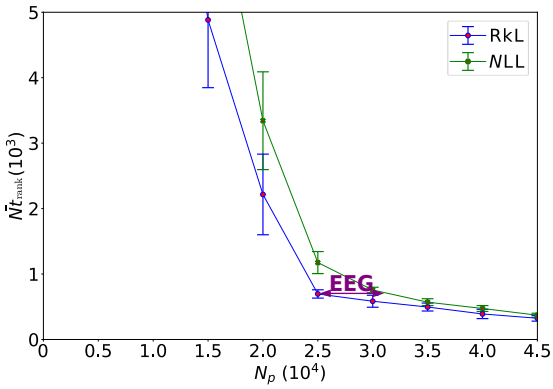
Remark 7.3.2.2. The value α of the ranking loss needs to be adapted depending on the number of profiling traces. For example, when N_p is low, the risk of overfitting is a major issue. One solution, to limit the overfitting effect, is to fix a higher learning rate [HHS17, ST17]. Hence, following Equation 7.8 and Equation 7.9, α can be monitored as the learning rate, in order to optimize the training process. For the AES_HD dataset, increasing α to 10 generates a more powerful model than α equal to 1 (see Appendix B) when the number of profiling traces equals $20,000$.

ASCAD. In contrast with the previous datasets, ASCAD is a protected implementation with 1^{st}-order masking and random-delay countermeasures. Figure 7.3b, Figure 7.4a and Figure 7.4b provide a comparison between models trained with the different losses for synchronized and desynchronized leakage traces. In Figure 7.3b, when the model is trained with the ranking loss, only

15,000 profiling traces are needed to perform a successful attack, such that $\bar{N}t_{\text{rank}} = 1,700$, while 18,500 (resp. 20,000) are needed to reach the same performance when the negative log-likelihood (resp. cross-entropy ratio) loss is used for the training process. Consequently, if the Evaluator chooses the negative log-likelihood (resp. cross-entropy ratio) as loss function, he needs to increase its training set by 3,500 (resp. 5,000) profiling leakage traces on average. Thus, when $\bar{N}t_{\text{rank}} = 1,700$, $\mathsf{EEG}(\mathcal{L}_{RkL}, \mathcal{L}_{NLL}) = 3,500$. Furthermore, when no desynchronization occurs, the model converges faster towards the average best solution (*i.e.* $\bar{N}t_{\text{rank}} \approx 260$) when the ranking loss is used (*i.e.* $N_p = 35,000$) compared to the negative log-likelihood or the cross-entropy ratio losses (*i.e.* about 45,000 profiling traces). The resulting EEG value equals 10,000.
This estimation error gap is up to 6,000 profiling traces when desynchronization occurs (see Figure 7.4a and Figure 7.4b)[h]. Hence for the ASCAD dataset, EEG is not increased with the desynchronization effect. Indeed, in comparison with synchronized traces, this countermeasure only impacts the exploitation of the relevant information. Finding suitable CNN architectures reduce the desynchronization effect (see Chapter 6) while preserving the same performance as a model trained with synchronized traces.



(a) ASCAD (desynchronization 50)



(b) ASCAD (desynchronization 100)

Figure 7.4: Evolution of $\bar{N}t_{\text{rank}}$ depending on N_p (average over 10 converging models - desynchronized datasets)

Through Table 7.3, we confirm the observations on the AES_HD dataset. In our experiment, for a small number of profiling traces N_p (*i.e.* $\leq 25,000$), a model trained with the ranking loss is, on average, more efficient than one trained with the negative log-likelihood or the cross-entropy ratio. For synchronized and desynchronized traces, the Evaluator with a limited number of profiling traces shall use the ranking loss.
When the entire set of PoIs is detected by the network, a model trained with the ranking loss converges faster towards the best solution compared to the negative log-likelihood and the cross-entropy ratio losses. From a theoretical perspective, we can assume that the estimation error is reduced when the ranking loss is considered. However, when the number of profiling traces N_p is large enough, we validate that the impact of the estimation error can be negligible. However, in practice, the number of profiling traces is limited. For that purpose, the ranking loss function seems to be more appropriate with the assumption that the Evaluator does not have an infinite number of traces in the profiling phase [PHG19]. Hence, the ranking loss is a solid alternative to the negative log-likelihood for performing side-channel attacks.

Remark 7.3.2.3. As previously mentioned, depending on the number of classes, the training time can be impacted when the ranking loss is considered. In comparison with the cross-entropy losses, the training time is increased by $60s$, when $N_p = 15,000$, and by up to $253s$ when 45,000 profiling traces are used for generating a network on the ASCAD dataset when no desynchronization occurs.

[h]Note that models trained with the cross-entropy ratio did not converge when desynchronization 100 is considered.

Table 7.3: Evolution of $\bar{N}t_{\text{rank}}$ depending on the number of profiling traces N_p (ASCAD - average over 10 converging models)

	$\bar{N}t_{\text{rank}}$ \ N_p	10,000	15,000	20,000	25,000	30,000	35,000	40,000	45,000
Desync 0	**Negative Log-Likelihood**	>8,000	2,135	1,054	705	471	435	318	270
	Cross-Entropy Ratio	>8,000	3,015	1,771	1,220	950	437	313	295
	Ranking Loss	**>8,000**	**1,740**	**709**	**481**	**334**	**272**	**258**	**244**
Desync 50	**Negative Log-Likelihood**	>8,000	4,172	2,342	1,041	792	656	577	337
	Cross-Entropy Ratio	>8,000	6,640	3,042	1,473	1,028	742	567	403
	Ranking Loss	**>8,000**	**3,475**	**1,792**	**765**	**641**	**558**	**449**	**345**
Desync 100	**Negative Log-Likelihood**	>8,000	7,593	3,343	1,175	756	570	473	371
	Cross-Entropy Ratio	>8,000	>8,000	>8,000	>8,000	>8,000	>8,000	>8,000	>8,000
	Ranking Loss	**>8,000**	**4,883**	**2,216**	**696**	**584**	**496**	**390**	**325**

In the worst-case scenario (*i.e.* $N_p = 45,000$), the training time is multiplied by 1.9. In addition, when a random delay effect with a maximum amplitude of 100 samples is implemented (*i.e.* "Desync 100"), the neural network architecture proposed in Section 6.3 is more complex than the neural network architecture trained on the synchronized traces (16,960 against 142,044 trainable parameters). In comparison with the cross-entropy losses, the model obtained from desynchronized leakage traces increases the resulted training time by $97s$, when $N_p = 15,000$, and by up to $290s$ when 45,000 profiling traces are considered. Hence, the training time is multiplied by 2 in the worst-case scenario (*i.e.* $N_p = 45,000$). Hence, through this example, a more complex neural network does not impact a lot the training time induced by the ranking loss. However, a further study should be made to validate this observation on deeper networks.

Remark 7.3.2.4. As mentioned earlier, the value α of the ranking loss needs to be adapted depending on the number of profiling traces. For example, when $N_p = 15,000$ traces, increasing α to 5 generates a more powerful model than $\alpha = 0.5$ (see Figure B.1b) if the desynchronization effect equals 100. Finally, as we can see in Figure 7.4b and Table 7.3, a model obtained from the cross-entropy ratio does not converge towards a constant GE of 1 when the random-delay effect equals 100. However, the cross-entropy ratio aims at reducing the imbalanced effect [ZZN+20, ISUH21] which is not considered in this manuscript.

7.4 Conclusion

This chapter proposes a new loss function which is derived from the success rate metric. Indeed, we extend the work done by Masure *et al.* [MDP19b] that consists in the interpretation and the explainability of the loss functions in the side-channel context. We use the learning to rank approach in order to propose a new loss, called *Ranking Loss*. We theoretically demonstrate that maximizing the success rate is equivalent to minimizing the ranking error of the secret key compared to all other hypotheses. Hence, the ranking loss tends to maximize the success rate for a given N_a leakage traces and converges towards the optimal distinguisher introduced in Definition 3.3.1.4. Through this new proposition, we are more concerned with the relative order of the relevance of the key hypothesis than their absolute value. This result notably allows preventing the approximation error induced by the softmax function considered by the negative log-likelihood loss function. While the minimization of the empirical risk combined with the ranking loss generates a model that converges towards the mutual information between the sensitive information Y and the leakage trace $\mathbf{T}$, its application is helpful to optimize the side-channel performance metrics that are considered to assess the robustness of a cryptographic module.

All these observations are experimentally validated through two scenarios. Firstly, if the Evaluator does not generate a model exploiting all the sensitive information from a leakage trace, using the ranking loss prevents the approximation error and provides a good alternative to the negative log-likelihood loss function. Otherwise, if the Evaluator generates a model that exploits the entire set of leakages, the model trained with the ranking loss converges faster towards the best solution compared to the cross-entropy losses (*i.e.* negative log-likelihood and the cross-entropy ratio). Hence, if the Evaluator deals with a limited number of traces, using the ranking loss should provide the most efficient model. Consequently, in all situations, the Evaluator shall consider the ranking loss as a clear alternative to the negative log-likelihood.

WHAT'S NEXT?

As the ranking loss approximates the mutual information between a sensitive information Y and a leakage trace $\mathbf{T}$, its application in side-channel context sounds natural. However, as mentioned in Section 7.3, the related training time can be impacted by the number of output classes. Indeed, following Definition 7.1.2.1, a sum over $\mathcal{K}$ has to be performed in order to compute the empirical risk combined with the ranking loss. To assess the suitability of the ranking loss on a wide-range of side-channe use-cases, the following chapter proposes to target cryptographic module implementing asymmetric algorithms (*i.e.* RSA and ECC). In such scenario, side-channel attacks aim to recover a secret key using the least number of leakage traces. Thus, increasing an attack's accuracy is particularly important when the Evaluator targets public-key cryptographic implementations where the recovery of each secret key bits is directly related to the model's accuracy. Commonly used in the deep learning field, ensemble models are a well suited method that combine the predictions of multiple models to increase the ensemble accuracy by reducing the correlation between their errors. One common solution to optimize ensemble models is to maximize an approximation of the mutual information between the ensemble model and the targeted sensitive information. From the knowledge acquired in this chapter, we extend the notion of ranking loss to the ensemble model and enhance the related attack performance against public-key cryptographic implementations.

Chapter 8

Efficiency through Diversity in Ensemble Models

A common solution to enhance the performance of the deep learning approach consists in combining individual predictions from several parametric models via a consensus method (*e.g.* majority vote, average vote) in order to reduce the global error. We study this technique, called *Ensembling*, in order to defeat cryptographic module implementing asymmetric algorithms. First, we recall the Evaluator' restrictions occurring during the evaluation of such implementations. From these observations, we motivate the need of accuracy gains and translate it to a drastic reduction of the remaining time complexity of a side-channel attack. This scenario fits with the Ensembling approach such that constructing ensemble models is beneficial to diversify the predictions induced by each of its member and thus, enhancing the attack performance. Then, we propose a new loss, namely *Ensembling Loss*, that generates an ensemble model which increases the diversity among its members. Based on the mutual information between the ensemble model and its related label, we theoretically demonstrate how this loss generates interaction between the ensemble members during the training process. It is further beneficial from the Evaluator perspective because he can reconsider the robustness of cryptographic module against side-channel attacks. We conclude this chapter by validating our theoretical observations on diverse datasets and side-channel attack scenarios. The solutions proposed in this chapter have been presented at CHES and published in the journal IACR TCHES [ZBHV21b].

Contents

8.1 Evaluation of Public-Key Cryptographic Implementations

8.1.1 Evaluator's Restrictions

Targeting symmetric vs. asymmetric implementations. As explained in Subsection 3.3.4, the attack process differs depending of the targeted cryptographic algorithms. While side-channel attacks against symmetric algorithm implementations usually target 8 bits, a key recovery attack against asymmetric algorithm implementations is considered as successful if, for one or few attack leakage traces, the Evaluator retrieves the whole private key $s_{k^*} \in \mathbb{F}_2^n$. Indeed, due to a careful combination of countermeasures (*e.g.* message blinding, modulus randomization, exponent/scalar blinding, point blinding), the Evaluator must be able to recover most of the secret bits from one or few leakage traces. Typically, two scenarios can be considered:

- **N leakage traces exploitation** – The Evaluator wants to recover the secret from a set of N leakage traces that share the same secret exponent/scalar s_{k^*}. This use case corresponds to ECDH (Elliptic Curve Diffie-Hellman) or RSA signature computations when exponent/scalar blinding countermeasure is applied. In this scenario, the Evaluator wants to recover the blinding factor related to each of those N leakage traces in order to recover s_{k^*} (deeper details are provided in Subsection 8.1.2).
- **1 leakage trace exploitation** – The Evaluator has access to only 1 leakage trace in order to recover the secret exponent/scalar s_{k^*}. This use case corresponds to ECDSA (Elliptic Curve Digital Signature Algorithm) targeting the scalar multiplication with a random nonce[a].

Consequently, instead of aggregating the output probability of a Θ-parametric model F_Θ over multiple leakage traces, the Evaluator's strategy consists in making use of the accuracy of F_Θ to retrieve each private key bit $(s_{k^*}[i])_{0 \leq i < n}$. If F_Θ does not correctly find each bit with 100% accuracy, the Evaluator has to perform additional operations, called *Remaining Operations* (see Definition 3.3.4.6), to retrieve the last bits of the private key. Hence, the accuracy of the attack is crucial in order to lower the remaining operations required to find the entire secret key s_{k^*}.

Practicability and remaining brute-force complexity. To define the practicability of an attack, the European SOG-IS scheme[b] introduces three complexity measures [SI20, Section 1.3]:

- **Time complexity** – This metric corresponds to the number of offline computations the Evaluator can perform in order to extract the targeted data from the cryptographic module. This notion does not consider the computation power available to the Evaluator (*e.g.* parallelization). This term is characterized by the *Remaining Operations* introduced in Definition 3.3.4.6.
- **Memory complexity** – This metric defines the amount of memory the Evaluator needs to store the acquired leakage traces and perform the related attack.
- **Data complexity** – The metric quantifies the amount of interactions the Evaluator needs with the targeted cryptographic module in order to perform his attack.

[a] *Nonce* is the abbreviation of "*number only used once*". It refers to any arbitrary random number and it is useful to mitigate the impact of some attacks (*e.g.* replay attacks)

[b] As remainder, the Senior Officials Group Information Systems Security (SOG-IS) agreement defines a set of requirements and evaluation procedures related to cryptographic aspects of CC security evaluations of IT products and mutually agreed by SOG-IS participants. The interested readers may find useful information in `https://www.sogis.eu/index_en.html`.

While these three metrics can be combined following the Evaluator's capability (*e.g.* the time complexity can be reduced by increasing the memory or the data complexity), this manuscript only considers the time complexity measure for assessing the practicability of an attack. Thus, we put this chapter in a general context where no assumption is provided related to the Evaluator's capability. To measure the time complexity, the Evaluator considers the number $N_{op,\alpha}$ of remaining operations depending on the accuracy α obtained from a Θ-parametric model F_Θ. We define the *Attack Complexity* as $\log_2(N_{op,\alpha})$. In [SI20], the SOG-IS agreement considers that a maximum attack complexity of around 100 (*i.e.* $N_{op,\alpha} = 2^{100}$ operations) is practical.

Hence, to fit with the Evaluator's restrictions, the following sections consider this threshold to evaluate if an attack becomes feasible. Furthermore, while no theoretical result links the accuracy and the remaining operations, we experimentally evaluate how the accuracy impacts the final attack complexity.

8.1.2 Complexity Measures

In this section, we consider three attack complexity measures, namely *Naive complexity*, 2^b-*complexity* and *Alternate attack complexity*, such that, depending on the targeted asymmetric cryptographic implementations, we can assess how the gain of accuracy influences the resulted side-channel attack complexity. The goal of the adversary is to recover a blinded secret key composed by a secret exponent/scalar s_{k^*} (resp. blinded exponent/scalar) of bit-length n (resp. r).

Naive complexity. In this scenario, the Evaluator guesses the bits related to s_{k^*} based on a Θ-parametric model F_Θ and a leakage trace $\mathbf{T}$. While some of the guessed bits may be wrong, the Evaluator cannot locate them among the n bits of s_{k^*} as he cannot perfectly distinguish the process $s_{k^*}[j] = 0$ from $s_{k^*}[j] = 1$ for $0 \leq j < n$. The *Naive complexity* metric measures the number of combinations the Evaluator has to perform in order to correctly recover s_{k^*}. Thus, given a secret exponent/scalar s_{k^*} of n bits, a blinding exponent/scalar of bit-length r and an error rate $\epsilon_{bit} \in [0, 1]$, the *Naive Complexity*, denoted $\mathcal{C}_{NC}$, is defined as:

$$\mathcal{C}_{NC}\left(n, r, \epsilon_{bit}\right) = \log_2\left(\sum_{i=0}^{\lceil (n+r)\times\epsilon_{bit}\rceil} \binom{n+r}{i}\right),$$

where $(n + r) \times \epsilon_{bit}$ denotes the number of erroneous bits related to the secret blinded exponent/scalar of $n + r$ bits.
Indeed, as the Evaluator cannot identify the correct from the wrong predictions, he has to compute all the possible combinations. This attack complexity metric considers the worst-case scenario.

2^b-complexity. This scenario is similar to the naive complexity metric. However, while the previous case suggests that the Evaluator cannot locate the wrongly guessed bits of s_{k^*}, this scenario assumes that the Evaluator can locate the uncertain predictions. For example, given a leakage trace $\mathbf{T}$, the Evaluator defines the related predictions as uncertain if the probability assigned to one hypothetical bit value does not exceed a given threshold. Thus, given a secret exponent/scalar of n bits, a blinding exponent/scalar of bit-length r and a percentage of bits ϵ_{bit} under a given threshold, the 2^b-*complexity*, denoted $\mathcal{C}_{2^b}$, is defined as the best-case scenario such that:

$$\mathcal{C}_{2^b}\left(n, r, \epsilon_{bit}\right) = \lceil (n + r) \times \epsilon_{bit} \rceil.$$

Indeed, as each uncertain prediction has 2 possible values (*i.e.* $\{0, 1\}$), the resulted number of remaining operations is $2^{\lceil (n+r)\times\epsilon_{bit}\rceil}$. In the following, an attack that can be performed with 2^b-*complexity* is called a 2^b-*attack*.

Alternate attack complexity. Introduced in [SW14], the *Alternate Attack* targets RSA modular exponentiation protected with exponent blinding. Based on the *Basic Attack* and the *Enhanced Attack* [SI11], the alternate attack retrieves the secret exponent bits from multiple traces. This attack can be extended to Elliptic Curves [SW14] and RSA with CRT[c] [SW17]. However, some tricks are specific to each case study. In this manuscript, we only focus on the application of the alternate attack on RSA without CRT. In particular, we formulate a complexity equation for this alternate attack that is missing from the original paper.
In [SW14], Schindler and Wiemers define the blinded exponent s'_{k^*} with a blinding scalar R' as:

$$s'_{k^*} = s_{k^*} + R' \cdot \phi(N),$$

where s_{k^*} is the secret exponent and $\phi(N)$ defines the Euler totient function of the modulus N. In the alternative attack scenario against RSA without CRT, it is assumed that the attacker knows the upper halves of the binary representation of $\phi(N)$ because it is similar to N. Let n be the bit-length of the secret exponent and $s_{k^*} \gg j = \lfloor \frac{s_{k^*}}{2^j} \rfloor$ defines the bits of s_{k^*} shifted to the right by j places. If $j \geq \frac{n}{2} + r + 6$, then $\lfloor \frac{s_{k^*}}{2^j} \rfloor$ depends on the upper half of the bits of $\phi(N)$. Given a secret blinding exponent s'_{k^*}, Schindler and Wiemers introduce $\alpha = \left\lfloor \frac{s'_{k^*}}{2^{n-1}} \right\rfloor$ and β such that $0 \leq \beta < 2^{n-1} < \phi(N)$ to rewrite s'_{k^*} in such a way that the $(r+1)$ most significant bits influence α while the $(n-1)$ least significant bits influence β. Then, the authors define $(s'_{k^*} \gg j)$ as:

$$(s'_{k^*} \gg j) = \left\lfloor \frac{s_{k^*} + R' \cdot \phi(N)}{2^j} \right\rfloor = \left\lfloor \frac{\alpha 2^{n-1} + \beta}{2^j} \right\rfloor,$$

and,

$$(s_{k^*} \gg j) = (s'_{k^*} \bmod \phi(N) \gg j) = \left\lfloor \frac{\alpha 2^{n-1} (\bmod \cdot \phi(N)) + \beta - \omega\phi(N)}{2^j} \right\rfloor = \left\lfloor \frac{\alpha 2^{n-1} (\bmod N) + \beta - \omega N}{2^j} \right\rfloor,$$

with high probability for an unknown $\omega \in \{0, 1\}$ and $j \geq \frac{n}{2} + r + 6$.
When the Evaluator captures the leakage traces, he guesses the randomized exponent to obtain an estimation $\hat{s}'_{k^*}$ of the true blinded exponent s'_{k^*}:

$$\left\lfloor \frac{\hat{s}'_{k^*}}{2^j} \right\rfloor = \left\lfloor \frac{s'_{k^*} \oplus e}{2^j} \right\rfloor = \left\lfloor \frac{\hat{\alpha} 2^{n-1} + \hat{\beta}}{2^j} \right\rfloor,$$

where e expresses the guessing error induced by exponent $\hat{s}'_{k^*}$, '$\oplus$' denotes the bitwise XOR operation, $\hat{\alpha}$ (resp. $\hat{\beta}$, $\hat{\omega}$) is an estimation of α (resp. β, ω).
Given an error rate ϵ_{bit}, the Evaluator can estimate the number of erroneous bits in $\hat{\alpha}$. The idea of the alternative attack against RSA without CRT is to generate all candidates for α (denoted $\hat{\alpha}_c$) and compute the resulted blinding factor candidates as $\hat{R}'_c = (\hat{s}'_{k^*} - \hat{s}_{k^*})/N = \lfloor \hat{\alpha}_c 2^{n-1}/N \rfloor + \omega$ with $\omega \in \{0, 1\}$. Then, for each candidate $\hat{\alpha}_c$ and $\hat{R}'_c$, the Evaluator can compute an estimation of the resulted error $\hat{e}$ based on a guess on the secret exponent s_{k^*} such that:

$$\hat{e} = \left(\hat{R}'_c N + \left\lfloor \frac{\hat{s}_{k^*}}{2^j} \right\rfloor 2^j \right) \oplus \left(\hat{\alpha}_c 2^{n-1} + \hat{\beta} \right).$$

If $\left\lfloor \frac{\hat{s}_{k^*}}{2^j} \right\rfloor = \lfloor \frac{s_{k^*}}{2^j} \rfloor$, a blinding factor estimation $\hat{R}'_c$ is defined as a candidate for R' if $HW\left(\lfloor \hat{e}/2^j \rfloor\right) \leq t_0$ with t_0 a threshold configured by the Evaluator. A smaller t_0 value induces a more restrictive

[c]The *Chinese Remainder Theorem* (CRT) mode optimizes the modular exponentiation process which is rather slow. Thus, the application of the CRT mode is beneficial from computational perspective as it deals with larger key size in about 4 times faster than a classical RSA without CRT mode.

candidate selection. The threshold t_0 should be selected such that no false candidates for R' are kept. More details on the alternative attack algorithm are provided in [SW14, Algorithm 4]. However, it is acceptable that some of the $\lfloor \hat{s}_{k^*}/2^j \rfloor$ candidates are wrongly guessed. Then, to retrieve the remaining bits of $\phi(N)$, the adversary has to perform the **Step 3** of the *Enhanced Attack* introduced by Schindler and Itoh [SI11]. Of course, for a number N_a of attack traces, we expect $q_{x_0,t_0}N$ candidates for $\left\lfloor s'_{k^*}/2^s \right\rfloor$ where,

$$q_{x_0,t_0} = \left(\sum_{i \leq x_0} \binom{r+1}{i} \epsilon_{bit}^i \left(1-\epsilon_{bit}\right)^{r+1-i} \right) \cdot \left(\sum_{i \leq t_0} \binom{n-1-j}{i} \epsilon_{bit}^i \left(1-\epsilon_{bit}\right)^{(n-1-j)-i} \right),$$

such that, the two brackets quantify the probabilities that $\hat{\alpha}$ and the most significant bits of $\hat{\beta}$ contain at most x_0 or t_0 guessing errors, respectively [SW17]. Through all these components, we can estimate the complexity of the resulted alternate attack for a given j and t_0 values.
First, the Evaluator has to configure the j, t_0 and N_a values to perform successful attacks. Then, for a given $\hat{s}'_{k^*} = \hat{\alpha}2^{n-1} + \hat{\beta}$, the Evaluator has to generate all $\hat{\alpha}_c$ candidates that differ by x_0 bits from $\hat{\alpha}$ at most. Hence, there are $M_0 = \sum_{i \leq x_0} \binom{r+1}{i}$ candidates for α.
The computation of each candidate $\hat{R}'_c$ and $\hat{e}_c$ depends on the number of $\hat{\alpha}_c$ elements. Therefore, there are $2 \cdot M_0$ candidates for R', $2 \cdot M_0 \cdot 2^{n-j}$ candidates for e in the worst case (*i.e.* if $\lfloor \hat{s}_{k^*}/2^j \rfloor \neq \lfloor s_{k^*}/2^j \rfloor$) or $2 \cdot M_0$ candidates for e otherwise (*i.e.* if $\lfloor \hat{s}_{k^*}/2^j \rfloor = \lfloor s_{k^*}/2^j \rfloor$). In the following, we only consider the worst-case scenario for the attack complexity estimation.
Given an n-bit secret exponent s_{k^*}, an r-bit blinding exponent, an error rate ϵ_{bit} and a number of attack traces N_a, the alternate attack complexity $\mathcal{C}_{AA}$ is defined as:

$$\mathcal{C}_{AA}\left(n, r, \epsilon_{bit}, N_a\right) = \log_2 \left(N_a \cdot 2^{n-j+1} \cdot \sum_{i=0}^{\lceil (r+1) \times \epsilon_{bit} \rceil} \binom{r+1}{i} \right),$$

with t_0 configured such that no false candidates for R' are selected.
To consider the alternate attack has a success, the Evaluator has to define the number of attack traces N_a that are needed to recover the full bits of $\phi(N)$. Hence, to correctly estimate $\mathcal{C}_{AA}$, the Evaluator has to perform the **Step 3** of the enhanced attack [SI11] in order to find a correct assumption about N_a. We consider an alternate attack as ineffective if the success rate related to $\phi(N)$ is less than 100% when 300 successive alternate attacks are performed.

Remark 8.1.2.1. In [SW17], Schindler and Wiemers set $j = n - r + 2$ and $t_0 = 2$ for $r = 32$. Even if using the same parameters is restrictive when $r = 64$, these conditions respect the above result. Thus, the alternate attack complexity considered in Subsection 8.4.1 employs those parameters.

All these complexity measures are helpful to evaluate the efficiency of an attack. These tools are suited to highlight the impact of the accuracy on the resulted attack complexity. As $\epsilon_{bit} = 1 - \alpha_{bit}$, with α_{bit} denoting the accuracy of retrieving one bit of s_{k^*}, it can be observed that a slight improvement in α_{bit} can drastically reduce all the attack complexity metrics introduced above. Thus, a slight improvement in terms of accuracy is non-negligible from a side-channel perspective.

This result leads us to investigate the benefits of using Ensembling approach in side-channel context in order to defeat asymmetric cryptographic implementations.

8.2 The Principle of Ensembling

In machine learning, ensemble methods combine individual predictions from parametric models of a pool via a consensus method (*e.g.* majority vote, average) [HS90, Kun04, Zho12]. These

approaches are useful when the models of the ensemble, known as *committee members*, learn and predict uncorrelated errors. Hence, a simple consensus method can efficiently reduce the global error of the system. However, in practice, the errors induced by the committee members are correlated and the overall ensemble error reduction is hard. One solution to reduce this correlation is to conduct a diversity investigation on the members in order to reduce the global error and increase to some extent, the ensemble performance [Die00a]. The following sections deeply characterize these observations.

8.2.1 A Source of Diversity

Reduction of the Global Error. In [TG96a, TG96b], Tumer and Ghosh provide theoretical observations for analyzing the interest of ensembling to solve a classification problem. They analyze the classification errors that are added to the *Bayes error* (*i.e.* the lowest possible error rate for any classifier of a random outcome) for an ensemble model. Let $\mathcal{E} = \{F_{\Theta_0}, F_{\Theta_1}, \cdots, F_{\Theta_{N_c-1}}\}$ be a set (or committee) of N_c models (or members) with trainable parameter $(\Theta_n)_{0 \leq n < N_c}$. In the following, F_{Θ_n} will be denoted as F_n. The parametric models are assumed to have the same[d] error rate such that E_{add} denotes the expected added error of the individual members included in $\mathcal{E}$. In [TG96a, TG96b], Tumer and Ghosh show that the expected added error of the ensemble committee, denoted $E_{add,ens}$, can be expressed as:

$$E_{add,ens} = E_{add}\left(\frac{1 + \delta(N_c - 1)}{N_c}\right), \tag{8.1}$$

where δ is a correlation factor that quantifies the error dependence among the parametric models. From Equation 8.1, we can easily evaluate the benefits of using ensemble methods to reduce the global error. If δ is 0, then the errors induced by the parametric models are independent and the ensemble expected added error is divided by N_c. Therefore, the global error will be N_c times smaller than the individual error provided by each parametric model included in $\mathcal{E}$. On the other hand, if δ is 1, the errors induced by the parametric models are correlated and $E_{add,ens}$ characterizes the average error of each classifier. To ensure uncorrelated errors, the classifiers included in the ensemble model must be diverse [Die00a].

Ensemble Diversity Definitions. Diversity has been recognized as a very important concept in parametric models combination [CC00, Lam00]. Indeed, it characterizes the difference among individual members of a committee $\mathcal{E}$. However, in the machine learning literature, there are multiple techniques that serve the ensemble diversity. For example, bagging [Bre96] and boosting [FS96] manipulate input data to promote diversity by choosing different subsets of input during the training process while other approaches consist in diversifying the neural network architectures considered by each parametric model in $\mathcal{E}$. In this manuscript, the diversity is defined as follows:

Definition 8.2.1.1 (Diversity)**.** Given an ensemble model $\mathcal{E}$ composed by N_c committee members $(F_n)_{0 \leq n < N_c}$, the diversity is defined as a quantity measuring the difference among the elements of $\mathcal{E}$ based of their predictions.

This definition is not new and was already considered by the machine learning community (*e.g.* majority vote [MHA14], PAC-Bayesian theory [GMGA17], ...). From Definition 8.2.1.1, diversity is greater when the parametric models that make wrong decisions for a given example spread their decisions more evenly over the possible incorrect decisions. Indeed, the more uniformly distributed the errors are, the greater the diversity. In [FR05], Fumera and Roli found that the performance of ensembles depends on the performance of individual classifiers and their correlation. To efficiently promote the ensemble diversity, the output of the ensemble model $\mathcal{E}$ can be decomposed into three

[d] This strong hypothesis is proposed by Tumer and Ghosh in [TG96a, TG96b] to illustrate the benefits of the ensembling approach.

categories [XKS92, Kun04]. Let $\mathcal{E} = \{F_0, F_1, \cdots, F_{N_c-1}\}$ be a set of N_c parametric models and $\mathcal{C} = \{c_0, c_1, \cdots, c_{|\mathcal{K}|-1}\}$ be a set of $|\mathcal{K}|$ labels (or classes). For a given input $\mathbf{T}$, we can define these categories as follows:

- **Abstract level** – Each parametric model $(F_n(\mathbf{T}))_{0\leq n<N_c}$ outputs unique value included in $\mathcal{C}$. Thus, the N_c predictions define a vector in $\mathcal{C}^{N_c}$ that characterizes the output of $\mathcal{E}$. Based on this vector, the Evaluator can define the diversity induced in $\mathcal{E}$.

- **Oracle level** – The output of $F_n(\mathbf{T})$ is 1 if $\mathbf{T}$ is correctly classified by F_n, and $F_n(\mathbf{T}) = 0$ otherwise. This representation is called *oracle* because the Evaluator has to know the label for each input in order to configure the output of $\mathcal{E}$.

- **Measurement level** – The output of $F_n(\mathbf{T})$ is defined by a vector of posterior probabilities $\Big[\mathsf{Pr}[c_0|\mathbf{T}], \mathsf{Pr}[c_1|\mathbf{T}], \cdots, \mathsf{Pr}[c_{|\mathcal{K}|-1}|\mathbf{T}]\Big]$ that quantifies the confidence in the prediction of each class $(c_i)_{0\leq i<|\mathcal{K}|-1}$. Hence, the output of the ensemble model $\mathcal{E}$ is characterized by N_c confidence vectors of size $|\mathcal{K}|$.

The measurement level contains the highest amount of information while the abstract level contains the lowest [XKS92]. In this manuscript, we want to precisely measure the diversity between each parametric model of the committee $\mathcal{E}$. For that purpose, we focus only on the *posterior probability representation* to evaluate the performance and the diversity of an ensemble model $\mathcal{E}$. These probabilities will be combined following the *Average Method* [XKS92] to define the overall performance of $\mathcal{E}$ but a comparison will also be provided with *Voting* in Subsection 8.4.3. The average method consists in predicting the most likely class based on the average conditional probability of observing each element of $\mathcal{C}$ over $\mathcal{E}$.
Thus, based on the measurement level, the Evaluator has to design an ensemble model $\mathcal{E}$ such that the diversity between the committee members is optimized. However, the diversity methods are legion and it could be hard to categorize them. In [LWC+19], Liu *et al.* decompose the diversity into three categories:

- **Type I diversity** characterizes the variety of committee members' structure such as types of neural network (*e.g.* multi-layer perceptrons, fully-connected neural networks, convolutional neural networks, recurrent neural network, residual neural network), weight initialization, training dataset, optimizer hyperparameters (*e.g.* optimizer algorithm, learning rate, number of epochs).

- **Type II diversity** selects a subset of parametric models included in $\mathcal{E}$ that minimize their errors correlation. Hence, the resulted ensemble model $\mathcal{E}'$ promotes independence between its members and tends to reduce the related global error.

- **Type III diversity** forces the ensemble model $\mathcal{E}$ to decorrelate the errors generated by each of its parametric model $(F_n)_{0\leq n<N_c}$ during the training process. Hence, an error decorrelation penalty term is incorporated in the loss function to create complementary members and consequently, reduces the overall error.

The type II and the type III diversities are both defined and quantified based on the disagreement among the parametric model included in $\mathcal{E}$. While the type II diversity captures the disagreement measure of each committee member after the training process for selecting a subset of parametric models $\mathcal{E}'$, the type III diversity considers the *posterior probability representation* to create and promote interactions during the profiling phase. Hence, even if an ensemble model $\mathcal{E}$ is composed by committee members with a high disagreement measure, applying the type III diversity is useful to penalize the remaining error correlation and providing a slight improvement of the ensemble model's performance as recommended in Section 8.1. Consequently, we design a solution to satisfy the type III diversity by proposing a new loss that promotes the penalization of correlated errors during the training process. This metric is based on the mutual information between an ensemble

model and its related labels.

The next section introduces the concept of mutual information ensemble diversity as a foundation of our proposition. In addition, to efficiently evaluate the overall benefits of using ensemble methods in side-channel context, we combine all types of diversity in Section 8.4.

8.2.2 Mutual Information Ensemble Diversity

Type III diversity can be characterized by the application of a specific loss function promoting the diversity between committee members. Unlike the correlation that is classically employed to measure the similarity between two entities, the mutual information captures non-linear statistical dependencies between variables. Hence, this measurement can be used as a real source of dependence information [KA14]. In [Bro09], Brown evaluates the benefits of using mutual information to improve ensemble models. In [ZL10], Zhou rewrites the ensembling as a communication channel problem such that it can be summarized as follows:

ENSEMBLING AND INFORMATION THEORY

From an information theoretic point of view, let Y be a message sent through a communication channel and X be the received value such that X should be decoded to recover the input message Y. For that purpose, a decoding function $g(.)$ is defined such that an estimation of the message can be written as $\hat{Y} = g(X)$. From a machine learning perspective, X is the set of features characterizing the input of a learner $g(.)$ and Y is the true unknown label. From ensembling paradigm, the goal is to recover Y from a set of N_c models $X_{0:N_c-1} = \{X_0, X_1, \ldots, X_{N_c-1}\}$ by a combination function $g(.)$ such that during the training process, we want to minimize $\mathsf{Pr}\left[g(\{X_0, X_1, \ldots, X_{N_c-1}\}) \neq Y\right]$. For any model g, [Fan61, HR70, Bro09] provide theoretical bounds for $\mathsf{Pr}\left[g(X_{0:N_c-1}) \neq Y\right]$ such that:

$$\frac{H(Y) - MI(X_{0:N_c-1}; Y) - 1}{\log(|Y|)} \leq \mathsf{Pr}\left[g(X_{0:N_c-1}) \neq Y\right] \leq \frac{H(Y) - MI(X_{0:N_c-1}; Y)}{2}.$$

Hence, to minimize $\mathsf{Pr}\left[g(X_{0:N_c-1}) \neq Y\right]$, we have to maximize the mutual information between $X_{0:N_c-1}$ and Y.

In [Bro09], Brown proposes a solution to compute the mutual information between an ensemble model $\mathcal{E} = \{X_0, X_1, \ldots, X_{N_c-1}\}$ and a set of true unknown labels Y.

Definition 8.2.2.1 (Mutual Information Ensemble Diversity [Bro09, ZL10])**.** Given an ensemble model $\mathcal{E}$ composed by N_c committee members $(F_n)_{0 \leq n < N_c}$ and a targeted sensitive variable Y, the mutual information ensemble diversity is defined as:

$$MI(\mathcal{E}; Y) = \sum_{n=0}^{N_c-1} MI(F_n; Y) - \sum_{n=1}^{N_c-1} \sum_{\mathcal{E}_n \subseteq \mathcal{E}} MI(\{\mathcal{E}_n\}) + \sum_{n=1}^{N_c-1} \sum_{\mathcal{E}_n \subseteq \mathcal{E}} MI(\{\mathcal{E}_n\}|Y), \tag{8.2}$$

where $MI(F_n; Y)$ is called **relevancy**, $MI(\{\mathcal{E}_i\})$ defines the **redundancy** and $MI(\{\mathcal{E}_n\}|Y)$ characterizes the **conditional redundancy**. Note that $\sum_{\mathcal{E}_n \subseteq \mathcal{E}}$ should be read as "sum over all possible subsets $\mathcal{E}_n$ drawn from $\mathcal{E}$" where $\mathcal{E}_n$ is of size $n + 1$.

Example 8.2.2.1 (Case-study with $N_c = 3$.). If $N_c = 3$, the multivariate mutual information, also known as *Interaction Information*, between the joint variable $\mathcal{E} = F_{0:N_c-1}$ and the targeted

sensitive variable Y can be rewritten as follows:

$$\begin{aligned}
MI(\mathcal{E};Y) &= MI(F_0;Y) + MI(F_1;Y) + MI(F_2;Y) + MI(\{F_0,F_1,Y\}) + MI(\{F_0,F_2,Y\}) \\
&\quad + MI(\{F_1,F_2,Y\}) + MI(\{F_0,F_1,F_2,Y\}) \\
&= \sum_{n=0}^{2} MI(F_n;Y) + \sum_{\mathcal{E}_1\subseteq\mathcal{E}} MI(\{\mathcal{E}_1 \cup Y\}) + MI(\{F_0,F_1,F_2,Y\}) \\
&= \sum_{n=0}^{2} MI(F_n;Y) - \sum_{\mathcal{E}_1\subseteq\mathcal{E}} MI(\{\mathcal{E}_1\}) + \sum_{\mathcal{E}_1\subseteq\mathcal{E}} MI(\{\mathcal{E}_1\}|Y) - MI(\{F_0,F_1,F_2\}) \\
&\quad + MI(\{F_0,F_1,F_2\}|Y) \\
&= \sum_{n=0}^{2} MI(F_n;Y) - \sum_{n=1}^{2}\sum_{\mathcal{E}_n\subseteq\mathcal{E}} MI(\{\mathcal{E}_n\}) + \sum_{n=1}^{2}\sum_{\mathcal{E}_n\subseteq\mathcal{E}} MI(\{\mathcal{E}_n\}|Y).
\end{aligned}$$

This result is consistent with Equation 8.2.

The relevancy computes the mutual information between the n^{th} parametric model of $\mathcal{E}$ and the target Y. The redundancy is independent of the class label Y and measures the interactions between all the models. Hence a large $\sum_{n=1}^{N_c-1}\sum_{\mathcal{E}_n\subseteq\mathcal{E}} MI(\{\mathcal{E}_n\})$ indicates strong correlations between the parametric model induced in $\mathcal{E}_n$. Finally, $\sum_{n=1}^{N_c-1}\sum_{\mathcal{E}_n\subseteq\mathcal{E}} MI(\{\mathcal{E}_n\}|Y)$ indicates that a strong class-conditional correlation is needed to perform an efficient ensemble model. However, from a practical perspective, it is quite difficult to estimate higher-order interaction information. Currently, there is no effective computational approach in the literature. Hence, Brown proposes to simplify Equation 8.2 by considering only pairwise components as follows [Bro09]:

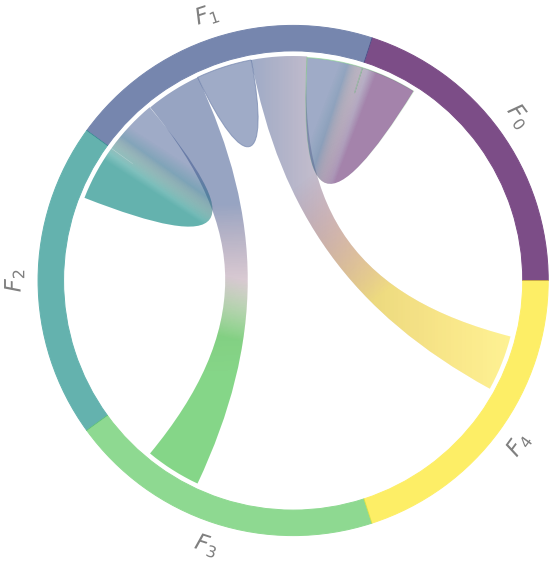
$$MI(\mathcal{E};Y) \approx \sum_{n=0}^{N_c-1} MI(F_n;Y) - \sum_{n=0}^{N_c-2}\sum_{m=n+1}^{N_c-1} MI(F_n;F_m) + \sum_{n=0}^{N_c-2}\sum_{m=n+1}^{N_c-1} MI(F_n;F_m|Y), \tag{8.3}$$

where $MI(F_n;Y)$ computes the mutual information between the n^{th} model of $\mathcal{E}$ and the target Y, $MI(F_n;F_m)$ measures the mutual information between two models F_n and F_m and $MI(F_n;F_m|Y)$ measures the conditional redundancy between two models F_n and F_m knowing Y. Based on the pairwise approach, Equation 8.3 omits higher-order components such that the interactions it introduced are illustrated in Figure 8.1. Assuming that the Evaluator designs an ensemble model such that $\mathcal{E} = \{F_0, F_1, F_2, F_3, F_4\}$, this figure illustrates the interactions induced by the parametric model F_1 (resp. each parametric model) with each other (see Figure 8.1a) (resp. see Figure 8.1b). Indeed, through the computation of Equation 8.3, the Evaluator induces interactions between the committee members such that each parametric model interacts with itself, in order to optimize $MI(F_n;Y)$, and with each other, in order to maximize conditional redundancy while minimize the redundancy.

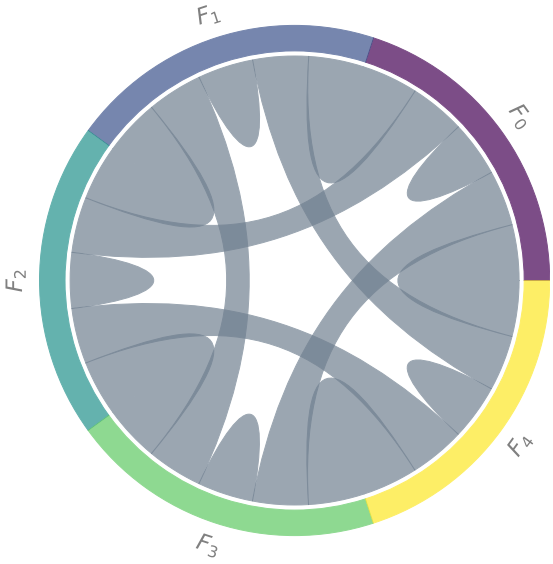
However, from a practical perspective, the Evaluator cannot directly compute $MI(\mathcal{E};Y)$ because it is intractable. Thus, we have to find a loss function that penalizes a set of parametric models included in $\mathcal{E}$ such that an approximation of this mutual information $MI(\mathcal{E};Y)$ is maximized. The following section proposes a solution to this issue.

8.3 Ensembling Loss: A Pairwise Ensemble Diversity Metric

This section presents our solution: the *Empirical Risk combined with the Ensembling Loss*. In Subsection 8.3.1, we first define three sub-empirical risks that are combined with three new losses, namely *Relevance Loss*, *Conditional Redundancy Loss* and *Redundancy Loss*, derived from the mutual information ensemble diversity. This decomposition allows us to define the *Ensembling Loss* as a diversity learning metric. Then, Subsection 8.3.2 validates the theoretical aspects of the ensembling loss through visualization techniques.



(a) Interactions induced by the parametric model F_1 with each other.



(b) Interactions induced by the parametric models included in $\mathcal{E}$.

Figure 8.1: Mutual information ensemble diversity principle considering pairwise components and an ensemble model $\mathcal{E} = \{F_0, F_1, F_2, F_3, F_4\}$.

8.3.1 Mutual Information Ensemble Diversity Estimation

This section proposes a new empirical risk derived from Equation 8.3 in order to maximize an approximation of the pairwise mutual information $MI(\mathcal{E}; Y)$ and the diversity between the committee members. To this end, we propose three empirical risks combined with three losses namely **Relevance loss**, **Conditional Redundancy loss** and **Redundancy loss**. In order to achieve a general-purpose estimator, we base our propositions on the characterization of the mutual information as the Kullback-Leibler (KL-) divergence [KL51] between the joint distribution and the product of the marginals. In Appendix D, we deeply explain the equations mentioned in this section.

Relevance Loss. In Equation 8.3, the relevance $MI(F_n; Y)$ highlights the dependence of a learner $F_n \in \mathcal{E}$ and a sensitive variable Y. Following [ZL10, Zho12], this term gives a bound on the accuracy of the individual parametric models. In Chapter 7, we extend this result by demonstrating that this term, also referred as the **Ranking Loss**, is defined as an upper bound on the success rate related to each individual parametric model. Hence, a large relevance is preferred to maximize the performance of the ensemble model. As mentioned in Chapter 7, minimizing the ranking loss is equivalent to maximizing an approximation of the mutual information $MI(F_n; Y)$. The minimization of this loss function is exactly what the relevance quantifies in [Bro09]. Thus, given a profiling set $\mathcal{I}_p$ of N_p pairs $(\mathbf{t}_i, y_i)_{0 \leq i \leq N_p}$, a parametric model F_n and a number of attack traces N_a such that $N_a | N_p$, we define the empirical risk combined with the *Relevance Loss* function as:

$$\hat{\mathcal{R}}(\mathcal{L}_{ReL}, F_n) = \frac{N_a}{N_p} \sum_{i=1}^{N_p/N_a} \sum_{\substack{k \in \mathcal{K} \\ k \neq k^*}} \left(\log_2 \left(1 + e^{-\alpha(s_{N_a,i}(F_n,k^*) - s_{N_a,i}(F_n,k))} \right) \right), \tag{8.4}$$

where $s_{N_a,i}(F_n, k) = \sum_{j=1}^{N_a} F_n \left(\mathbf{t}_{j+N_a \cdot (i-1)} \right) [f(x_j, k)]$ defines the output score of the hypothesis $k \in |\mathcal{K}|$ for a given plaintext $(x_j)_{1 \leq j \leq N_a}$ while α approximates the sigmoid function needed for estimating the success rate.
Minimizing Equation 8.4 tends to maximize the mutual information $MI(F_n; Y)$ through the minimization of the error induced by $\mathsf{Pr}[Y|F_n]$. In other words, we want to penalize a model F_n when

the correct label Y is not ranked as the highest hypothetical class. This penalization term is monitored by the distance between the score associated with the correct label Y and the other hypotheses. From a machine learning perspective, the maximization of $MI(F_n; Y)$ tends to generate compact clusters, one for each class. If False-Positives (FP) or False-Negatives (FN) appear during the training process, the ensemble model will be overconfident on its predictions and the resulted errors could be persistent. To reduce this effect, a solution is to provide diversity in order to limit the impact of these FP, FN examples. Hence, other empirical risks defined below bring more diversity during the training process.

Remark 8.3.1.1. The empirical risk combined with the relevant loss is actually the same as the ranking loss introduced in Definition 7.1.2.1. We reformulate it to facilitate the comprehension of the ensembling loss and the comparison with the mutual information ensemble diversity (see Definition 8.2.2.1) introduced by Brown [Bro09].

Conditional Redundancy Loss. The conditional redundancy $MI(F_n; F_m|Y)$ quantifies the dependence between F_n and F_m given a set of labels Y. This mutual information helps the committee members to converge towards the correct label hypothesis with the same confidence. Maximizing $MI(F_n; F_m|Y)$ is asymptotically equivalent to minimizing the error on $\mathsf{Pr}[F_n, F_m|Y]$ which defines the probability of observing F_n and F_m given Y (see Appendix D). In other words, we want to minimize the distance between the scores of F_n and F_m given the correct class. Thus, given a profiling set $\mathcal{I}_p$ of N_p pairs $(\mathbf{t}_i, y_i)_{0\leq i\leq N_p}$, a pair of parametric models (F_n, F_m) and a number of attack traces N_a such that $N_a|N_p$, we define the empirical risk combined with the *Conditional Redundancy Loss* function as:

$$\hat{\mathcal{R}}(\mathcal{L}_{CRL}, F_n, F_m) = \frac{N_a}{N_p} \sum_{i=1}^{N_p/N_a} -\log_2 \left(e^{-\beta |s_{N_a,i}(F_n,k^*) - s_{N_a,i}(F_m,k^*)|} \right), \tag{8.5}$$

where β is one hyperparameter that characterizes the impact of the distance on the penalization term and $s_{N_a,i}(F_n, k^*)$ defines the score related to the class k^* given a set of N_a traces and a parametric model F_n.
Through Equation 8.5, we want to penalize the learning process when the score $s_{N_a,i}(F_n, k^*)$ and $s_{N_a,i}(F_m, k^*)$ are different. Hence, we want to minimize the dissimilarity between the pairwise model F_n and F_m knowing Y. This will have the effect of increasing the confidence of the network on the True-Positive (TP) and True-Negative (TN) examples. Consequently, we consolidate the good predictions with more persistency. However, this loss does not interact with the False-Positive (FP) and False-Negative (FN) examples. The following redundancy loss reduces this gap.

Redundancy Loss. The redundancy $MI(F_n; F_m)$ measures the pairwise dependence between all the committee members without considering the ground truth. A large mutual information induces a strong correlation among the pairwise committee members and promotes similarities which is not desired when we want to construct an efficient ensemble model. Hence, we want to minimize this mutual information to improve the ensemble performance (see Appendix D). The redundancy loss maximizes the distance between the score distribution of the models F_n and F_m. Therefore, we propose a loss penalizing the training process when this condition does not hold. Given a profiling set $\mathcal{I}_p$ of N_p pairs $(\mathbf{t}_i, y_i)_{0\leq i\leq N_p}$, a pair of parametric models (F_n, F_m) and a number of attack traces N_a such that $N_a|N_p$, we define the empirical risk combined with the *Redundancy Loss* function as:

$$\hat{\mathcal{R}}(\mathcal{L}_{RedL}, F_n, F_m) = \frac{N_a}{N_p} \sum_{i=1}^{N_p/N_a} \sum_{k=0}^{|\mathcal{K}|-1} \sum_{k'=0}^{|\mathcal{K}|-1} -\log_2 \left(1 - e^{-\gamma |s_{N_a,i}(F_n,k) - s_{N_a,i}(F_m,k')|} \right), \tag{8.6}$$

where γ is one hyperparameter that characterizes the impact of the distance on the penalization term.

Minimizing $MI(F_n; F_m)$ is equivalent to maximizing $H(F_m|F_n)$. Consequently, we want to increase the uncertainty of F_m given F_n. Through the minimization of Equation 8.6, we promote the cluster scattering and reduce the global confidence of the committee members on the False-Positives and False-Negatives to decrease their persistency.

Ensembling Loss. We integrate the mutual information ensemble diversity during the training process to promote the diversity between the committee members. Through our individual losses provided in Equation 8.4, Equation 8.5 and Equation 8.6, we formulate an *Ensembling Loss* (EL) that maximizes an estimation of the mutual information between an ensemble $\mathcal{E}$ and a label Y. Indeed, based on Equation 8.3, this new empirical risk can be expressed as below.

Definition 8.3.1.1 (Empirical Risk combined with the Ensembling Loss)**.** Given a profiling set $\mathcal{T}$ of N_p pairs $(\mathbf{t}_i, y_i)_{1\leq i\leq N_p}$, a set of parametric models $\mathcal{E} = \{F_0, F_1, \cdots, F_{N_c-1}\}$ and a number of attack traces N_a such that $N_a|N_p$, we define the empirical risk combined with the ensembling loss function as:

$$\hat{\mathcal{R}}(\mathcal{L}_{Rel}, \mathcal{L}_{RedL}, \mathcal{L}_{CRL}, \mathcal{E}) = \frac{1}{N_c}\sum_{n=0}^{N_c-1}\hat{\mathcal{R}}(\mathcal{L}_{ReL}, F_n) + \frac{2\mu}{N_c(N_c-1)}\sum_{n=0}^{N_c-2}\sum_{m=n+1}^{N_c-1}\left(\hat{\mathcal{R}}(\mathcal{L}_{RedL}, F_n, F_m) + \hat{\mathcal{R}}(\mathcal{L}_{CRL}, F_n, F_m)\right),$$

where μ quantifies the impact of the diversity term during the training process.

We normalize each term of the empirical risk to reduce the impact of exploding gradient. Appendix E highlights the benefits of each individual loss from a training perspective. Through this study, the reader can understand how the network would train if the conditional redundancy loss or the redundancy loss is individually used. Finally, in the following sections, the number of attack traces N_a will be configured to 1 during the profiling phase as in Chapter 7.

Remark 8.3.1.2. Due to the wide range of hyperparameters (*i.e.* μ, α, β, γ), the empirical risk combined with the ensembling loss seems difficult to tune. From a practical perspective, these hyperparameters are dataset-dependent. Hence, it seems very challenging to define a generalized configuration for all types of implementations because it highly depends on the number of classes $|\mathcal{K}|$, the noise induced in each trace, the implemented countermeasures and the targeted cryptographic algorithm implementation (*e.g.* AES, RSA, ECC). However, during our experiments, the tuning process was not a pitfall. Indeed, in the following section, α, β, γ values follow the strategy observed in Appendix B. Hence, they are configured in $[0.001, 0.1]$. In opposition, μ is not optimized in this manuscript and always equals 1.

Remark 8.3.1.3. As our framework is generic, we argue it is adequate to target private-key implementations, in particular AES and DES (*i.e.* $|\mathcal{K}| = 256$). However, the training time increases exponentially with the number of output classes $|\mathcal{K}|$. Hence, from a practical perspective, the application of the *Ensembling Loss* seems more suitable for low multiclass problems (*i.e.* $|\mathcal{K}| \leq 5$). This proposition fits with asymmetric algorithm implementations which consider low multiclass problems. Finally, even if this work is only focusing on the side-channel context, the *Ensembling Loss* can be used to solve any machine learning problems (*i.e.* image classification, image recognition, fraud detection, ...).

To validate the benefits of the ensembling loss, the following section investigates its impact on the ensemble diversity through a concrete example.

8.3.2 Enhancing the Features' Diversity

Diversity among the committee members is deemed to be a key issue in ensemble learning and should reduce the global error (see Subsection 8.2.1). In this section, we want to validate the theoretical observations provided in Subsection 8.3.1. Hence, we analyze the diversity evolution depending on the loss used during the training process. Three losses are considered in the rest of this chapter: the *Negative Log-Likelihood*, the *Ranking Loss* and the *Ensembling Loss*. All these learning metrics will be considered to select the most suited ensemble model such that three processes can be considered to perform the learning process:

- **Independent Learning Strategy** – There is no interaction among the committee members during the learning process. For example, each parametric model could be trained on different training set to reduce the features' correlation [Bre96]. Thus, they are independently obtained such that their posterior probabilities are combined once the learning process is fully performed on each committee member. This is the worst solution from the Type III diversity perspective;

- **Sequential Training** – This strategy induces a set of parametric models such that the learning process is sequentially performed on each committee member. This approach is suited as, at the n^{th} iteration, the posterior probabilities obtained from parametric models $(F_i)_{0\leq i<n}$ can be used to penalize the learning process related to the parametric model F_n;

- **Simultaneous Ensemble Learning** – a set of committee members are trained interactively to promote uncorrelation and diversity during the learning process. This approach is the most natural solution to provide Type III diversity.

As the minimization of the empirical risk combined with the ensembling loss perfectly fits with the *Simultaneous Ensemble Learning* strategy, the rest of this chapter will be focused on this solution for allowing interaction between the committee members during the training process. Furthermore, it is helpful to promote the diversity between the members even if similar architectures are used. Assessing the benefits of the ensembling loss can be illustrated through the t-SNE visualization [vdMH08] and diversity measures [KW03].

Dataset setup for visualization. For that purpose, we use the secure RSA dataset presented in Section 3.6. As mentioned, the targeted index seg_{free} value is defined in $\{0, 1, 2\}$. Consequently, to solve a classification task that assigns an index value to a leakage trace, we have to consider a multi-class classification problem with 3 outputs. Furthermore, the ensemble model is configured with 5 parametric models, as illustrated in Figure 8.1, such that each of them has the same neural network architecture. Generating 5 parametric models with the same architecture is helpful to efficiently evaluate the suitability of the ensembling loss in contrast with the negative log-likelihood and the ranking loss from a diversity perspective. These committee members are convolutional neural networks with 1 convolutional block based on 2 filters of size 1 and an average pooling layer with stride 2. Then, a flatten layer is applied to reduce the space dimension of the feature selection part. Finally, a predictive layer is applied with a softmax function. The optimizer hyperparameters are set such that each neural network is trained during 40 epochs, with a batch-size of 128, a learning rate set to 0.001 and the Adam optimizer [KB15].

Remark 8.3.2.1. A deeper investigation on the number of committee members is performed in Subsection 8.4.3 to evaluate its impact on the ensemble accuracy.

t-distributed Stochastic Neighbor Embedding [vdMH08]. Introduced by van der Maaten and Hinton, the t-SNE visualization tool maps high-dimensional data into two- or three-dimensional space while preserving local structure and revealing important global structure (*e.g.* clusters). t-SNE employs a nonlinear and iterative process to convert similarities between data points to joint probabilities and tries to minimize the KL-divergence between the joint probabilities of the low-dimensional embedding and the high-dimensional data. This representation is helpful

to evaluate the ensemble model capacity to distinguish each class and validate the theoretical approach presented in Subsection 8.3.1.

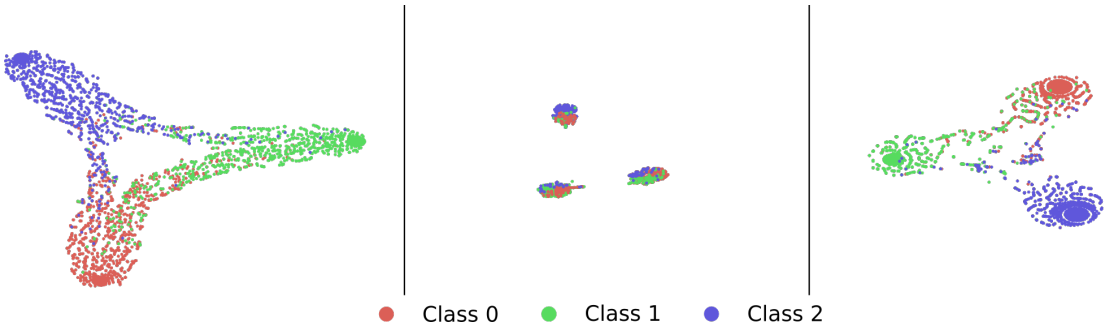


Figure 8.2: t-SNE embeddings. From left to right: Negative Log-Likelihood, Ranking Loss, Ensembling Loss.

Figure 8.2 illustrates the t-SNE visualizations depending on the loss used during the training process. When the negative log-likelihood is considered, we observe that the ensemble model is not trained enough to efficiently discriminate each class. Indeed, there are many connections between each class leading to a loss of the global performance. Hence, many FP and FN can badly influence the global performance of the model. Through this visualization, we can question the relevance of the negative log-likelihood as the best solution in our context when low complexity models are considered. This observation is in accordance with the demonstration provided in Subsection 7.2.2.
On the other hand, the ranking loss generates three separate clusters. As mentioned in Subsection 8.3.1, the ranking loss can be formulated as the relevance loss (see Equation 8.4). Through the minimization of this function, we aim at approximating the mutual information between a leakage trace T and a targeted label Y (see Subsection 7.2.1). In other words, we minimize the conditional entropy[e] $H(Y|T)$ which promotes the generation of three compact clusters. Figure 8.2 confirms this result as the ensemble model is overconfident in the features captured during the learning process. Consequently, it makes discriminative decisions to avoid connections between each cluster. However, following the t-SNE illustration, the FP and FN induced by the ranking loss are persistent and seem difficult to detect. Indeed, these errors are fully included in a wrong cluster. This result suggests that the ensemble model considering the ranking loss assigns a similar confidence to correct and wrong predictions. This phenomenon can be explained by the overfitting effect. Using more training examples could be useful to reduce this impact and reduce the error rate. However, when the number of profiling traces is limited (as often in practice), a solution has to be found to improve the ensemble model performance.
The best solution should create three separate clusters when the ensemble model is confident in its prediction while, the errors or the uncertain predictions should converge towards the equidistant point of the centroid of the clusters. These examples are called *data uncertainty*. Introduced in [MMG20], data uncertainty is the irreducible uncertainty in predictions which arises due to the complexity or noise in the data. In Figure 8.2, we can observe that the ensembling loss converges towards this best solution. Indeed, the combination of the relevance loss, the conditional redundancy loss and the redundancy loss creates three separate clusters (see Appendix E for deeper details). When the ensemble model is confident in its predictions, it will assign the related examples to the correct class. However, the ensembling loss creates some connections between the clusters which seem defined by the data uncertainty. This result tends to reduce the number of consistent FP and FN such that few errors can be detected on each cluster in contrast with the negative log-likelihood or the ranking loss. While the Evaluator's goal is to enhance the ensemble diversity, the t-SNE tool does not provide information related to this requirement. To validate the

[e]In our context, minimizing the conditional entropy $H(Y|T)$ is equivalent to minimizing $H(Y|F_n)$ as the related penalization term is similar.

suitability of the ensembling loss regarding the evaluation perspective, we evaluate its diversity growth against the negative log-likelihood and the ranking loss.

Diversity Measures. As explained in Subsection 8.2.1, diversity has a crucial impact on the ensemble model's performance. Conventionally the diversity measures can be decomposed into two categories:

- **Pairwise measures** that compute the relationship between two parametric models, and then average all the pairwise measurements to define the overall diversity of an ensemble model $\mathcal{E}$ (Disagreement measure [Ska96, KH98], Q-statistic [UY00], Correlation coefficient [SS73], κ-statistic [Coh60], Double-Fault measure [GR01], ...);
- **Non-Pairwise measures** that assess the ensemble diversity directly rather than by averaging pairwise measurements (Kohavi-Wolpert Variance [KW96], Interrater agreement [Die00b, FLP03], Entropy [CC00], ...).

One advantage of pairwise measures it that they can be easily visualized and interpreted. Choosing a specific pairwise measure does not make significant difference in our experiments, so we chose the fraction of disagreement as well as the κ-statistic as they are relatively easy to interpret.
Let $N^{ab}_{n,m}$ be a joint value between two parametric models F_n and F_m. Given a leakage trace $\mathbf{T}$, we denote $a = 0$ (resp. $b = 0$) if F_n (resp. F_m) assigns a wrong prediction to $\mathbf{T}$ and $a = 1$ (resp. $b = 1$) otherwise. For example, $N^{01}_{n,m}$ defines the number of elements such that F_n provides a wrong prediction for a given input $\mathbf{T}$ while F_m correctly predicts its related class.

Definition 8.3.2.1 (Disagreement Measure [Ska96, KH98])**.** Given two parametric models F_n and F_m, the disagreement measure defines the proportion of inputs on which these parametric models make different predictions:

$$Dis(F_n, F_m) = \frac{N^{01}_{n,m} + N^{10}_{n,m}}{N^{11}_{n,m} + N^{10}_{n,m} + N^{01}_{n,m} + N^{00}_{n,m}}.$$

This metric is 0 when two functions are making identical predictions, and 1 when they differ on every single example in the test set. Hence, the larger the value, the larger the diversity. From an ensembling perspective, we want to generate a set of parametric models $\mathcal{E}$ maximizing the disagreement measure such that each individual model keeps a high performance for classifying unseen examples. In [FHL19], Fort *et al.* propose to plot a normalized disagreement measure with respect to the accuracy of each classifier. The diversity measure is normalized by the error rate to prevent the case where random predictions provide the best diversity. From the set of parametric models $\mathcal{E}$, one member is randomly picked to be considered as the basis model. This model is denoted as F_{basis}. Then, we calculate the diversity measure of other ensembling members against the basis model.
Figure 8.3 illustrates the diversity of each model of $\mathcal{E}$ against F_{basis}. In this figure, the y-axis characterizes the fraction of labels, returned by each model of $\mathcal{E}$, which differs from F_{basis} while the x-axis defines their validation accuracy. Consequently, the sample with a 0 y-axis value defines F_{basis}. Three ensemble models are generated with the three losses used in order to investigate the benefits of the ensembling loss. In [FHL19], Fort *et al.* propose a theoretical approach to explain the results obtained in Figure 8.3. Let F_{basis} and F_n be two committee members from $\mathcal{E}$. If F_{basis} and F_n have identical validation accuracy and high diversity then, they converge towards different local optimum with identical depth. In opposition, if F_{basis} and F_n have identical validation accuracy and a low diversity then, they converge towards the same local optimum. Consequently, from a loss landscape perspective, it sounds beneficial to construct an ensemble model with high diversity members such that their prediction distributions and their selected features differ from each other. In contrast, the accuracy of individual models does not reflect the performance of the ensemble committee. Indeed the combination of poor performance (*i.e. weak*), but perfectly complementary, parametric models can generate a very effective ensemble model. While the same

configuration with effective, but correlated, parametric models is not beneficial for an ensembling approach (see Subsection 8.2.1). Consequently, an ensemble model composed by weak parametric models can outperform a combination of effective correlated models.

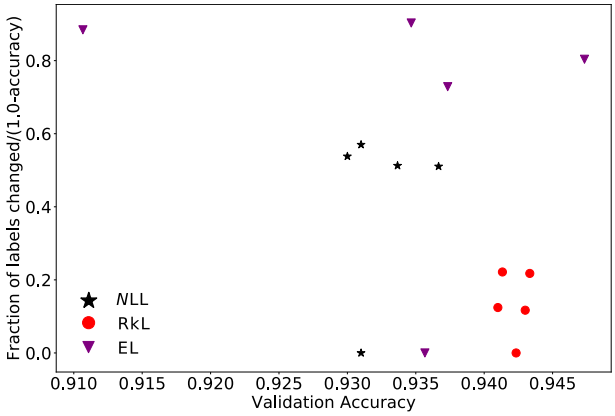


Figure 8.3: Diversity versus label accuracy plots for 3 ensemble models trained on Negative Log-Likelihood (NLL), Ranking Loss (RkL) and Ensembling Loss (EL).

The ensemble model trained with the ranking loss provides the worst diversity scenario. Even if individual parametric models are more efficient than most of the other committee members (*i.e.* validation accuracy $> 94\%$), the lack of diversity is an issue for developing uncorrelated models. Indeed, following [FHL19], all the committee members converge towards the same local optimum. Hence, a lack of complementarity can be exposed when the adversary only considers the ranking loss. Consequently, the resulted ensemble model performance should be equal to the average accuracy of its members. For the negative log-likelihood loss function, the parametric models are more diverse than the ranking loss ensemble model. Consequently, the resulted learners are less correlated and the resulted probability combination should reduce the overall error. Finally, in comparison with the negative log-likelihood and the ranking loss, the ensembling loss provides the most diverse models. Indeed, in Figure 8.3, the normalized diversity measure is the highest for the ensembling loss model. Interestingly, even if the committee members have the same architecture, the ensembling loss provides a clear diversity benefit. Hence, from a loss landscape perspective, the ensembling loss helps the committee members to converge towards independent local optimum with different depths. However, as Figure 8.3 only provides the disagreement measure between a model basis F_{basis} and all parametric models of $\mathcal{E}$, it does not provide the disagreement measure for all pairs (F_n, F_m) in $\mathcal{E}$. To mitigate this issue, we investigate the benefits of the κ-statistic metric to highlight the pairwise diversity induced in $\mathcal{E}$.

Definition 8.3.2.2 (κ-statistic [Coh60])**.** The κ-statistic measures the degree of agreement between two parametric models F_n, F_m as follows:

$$\kappa(F_n, F_m) = \frac{2\left(N^{11}_{n,m}N^{00}_{n,m} - N^{01}_{n,m}N^{10}_{n,m}\right)}{\left(N^{11}_{n,m} + N^{10}_{n,m}\right)\left(N^{01}_{n,m} + N^{00}_{n,m}\right) + \left(N^{11}_{n,m} + N^{01}_{n,m}\right)\left(N^{10}_{n,m} + N^{00}_{n,m}\right)}.$$

This metric is 1 when F_n and F_m correctly/wrongly predict the same examples, and 0 when their predictions differ on every single example in the test set. Hence, the larger the value, the lower the diversity. Based on the parametric models used for visualizing the t-SNE embeddings (see Figure 8.2), we compute the κ-statistic metric for each scenario, *i.e.* application of the negative log-likelihood, the ranking loss and the ensembling loss. Figure 8.4 illustrates the evolution of this diversity metric depending on the loss used. When the ensembling loss is used, the overall κ-statistic metric is reduced in comparison with the negative log-likelihood or the ranking loss. This figure confirms that the ensembling loss decorrelates the errors between the committee members.

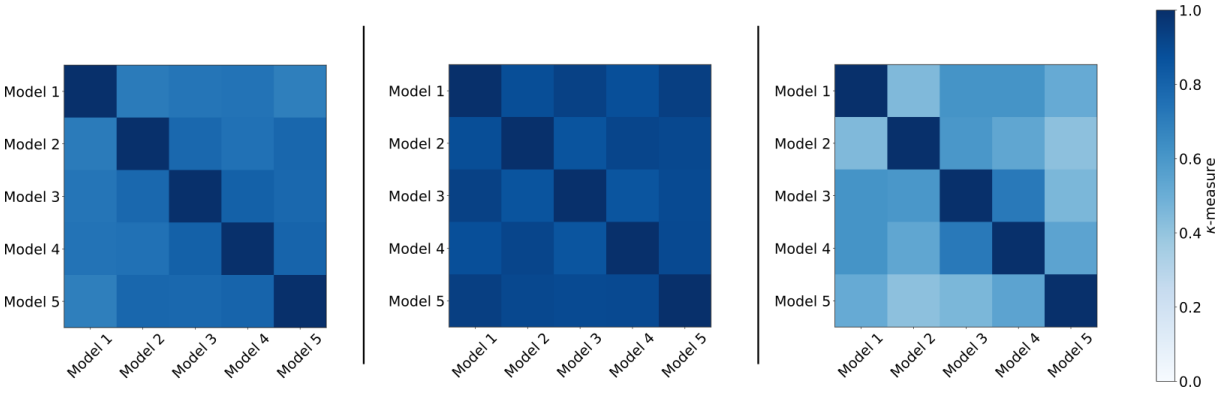


Figure 8.4: κ-statistic. From left to right: Negative Log-Likelihood, Ranking Loss, Ensembling Loss.

These observations validate the theoretical observations introduced in Subsection 8.3.1. Indeed, using the ensembling loss increases the diversity between the parametric models in order to reduce the correlation between their errors and propose an efficient ensemble model. In the next section, we evaluate this diversity gain on the side-channel attack performance.

8.4 A Case Study on Asymmetric Implementations

To assess the benefits of the diversity on the side-channel attacks targeting public-key implementations, we conduct investigations on two datasets:

- **Secure RSA dataset** – Introduced in Section 3.6, this dataset targets the manipulation of an index named $seg_{free} \in \{0, 1, 2\}$. Thus, we have to solve a multiclass classification problem with 3 output classes. As this implementation considers a RSA signature computation with blinding countermeasure, the resulted exploitation phase considers N leakage traces.

- **Secure ECC dataset** – Defined in Section 3.6, the attack consists in retrieving all the conditional swap bits with a single leakage trace. Hence, we have to solve a binary classification problem such that the output classes denote the hypothetical value of one bit (*i.e.* 0 or 1).

As mentioned in Subsection 3.3.4 and Subsection 8.1.2, the Evaluator can consider the accuracy as metric performance.

Evaluation metrics. We denote α_{bit} the accuracy expressing the capacity of the parametric model to retrieve the amount of correct bit values. Its related error rate is denoted ϵ_{bit}. As this metric is natural to solve binary classification problems with a single leakage trace, the Evaluator can consider this evaluation metric to assess the suitability of a parametric model to defeat the Secure ECC implementations.
However, when the Secure RSA dataset is considered, the Evaluator does not directly predict the bits related to the targeted private key. Indeed, he exploits the index seg_{free} such that 3 values can be assigned. We denote α_{label} the accuracy expressing the capacity of the parametric model to retrieve the correct value of seg_{free} for a given leakage trace. This metric is used to mitigate the underfitting and overfitting issues. However, while the Evaluator's goal is to retrieve the bits related to the private key, he has to convert[f] the balanced ternary representation (*i.e.* $\{0, 1, 2\}$) into a binary representation (*i.e.* $\{0, 1\}$). We also use the notation α_{bit} to denote the ability of the parametric model to retrieve the bits of the targeted private key.

[f]The interested readers may refer to [CCC+19, Section 2.3] to get an insight into how this conversion should be performed.

Models' notation. This section proposes an experimental comparison between the negative log-likelihood, the ranking loss and the ensembling loss when ensemble models are considered. Thus, in the following sections, $NLL_{i,j}$ (resp. $RkL_{i,j}$, $EL_{i,j}$) denotes an ensemble model trained with the negative log-likelihood (resp. the ranking loss, the ensembling loss), composed by i committee members such that the type j diversity is performed. Due to the interactions between the committee members during the training process, the ensembling loss can be considered as the only learning metric promoting the type III diversity.

Remark 8.4.1. In the following, we only consider convolutional neural networks because its benefit has been demonstrated in the side-channel context (see Chapter 6). Obviously, the combination of diverse neural network (*e.g.* multi-layer perceptrons, fully-connected neural networks, recurrent neural networks, residual neural networks) can also be considered to promote diversity.

Remark 8.4.2. In [DDFP21], Destouet *et al.* investigate a solution that consists in the aggregation of multiple models targeting different sensitive value (*i.e.* Hamming weight, first big-endian bit, identity). In this thesis, we assume that all the learners are trained on the same single label.

In Subsection 8.4.1 (resp. Subsection 8.4.2), we evaluate the diversity of committee members depending on the loss used when the Secure RSA dataset (resp. Secure ECC dataset) is considered.

8.4.1 Application on N Traces Exploitation

First, we design the neural network architecture that we used to construct the ensemble model. This parametric model is repeated 5 times in order to constitute an ensemble $\mathcal{E}$ of 5 similar committee members. This approach is beneficial to evaluate the complementarity of committee members when the ensemble model is trained with the different losses. Then, we combine the type I diversity with the different learning metrics to illustrate its impact of the resulted performance. Finally, all the diversity types are combined to exploit the entire benefits of the ensemble methods and highlight the improvement in the resulted side-channel attack complexity.

Neural Network Architecture. While the original network provided in [CCC+19] performs very well on the Secure RSA dataset (= 99.91%), we decide to reduce its complexity while preserving the performance. We use the methodology introduced in Subsection 6.3.1 such that the related neural network is composed of one convolutional block with 2 filters of size 1 and an average pooling layer. Then, a flatten layer is applied to connect the extracted points of interest to a predicting layer configured with 3 outputs defining the value of seg_{free}. Optimization is done using the Adam optimizer [KB15] approach on a batch-size of 128 and the learning rate is set to 10^{-3}. The batch-size and the learning rate follow the values provided in [CCC+19]. The optimization of these hyperparameters is not considered in this study. We use the SELU activation function to avoid vanishing and exploding gradient problems (see Equation 4.8). In the following sections, we only keep the parametric model achieving its best performance (*i.e.* accuracy) over 100 epochs. This new parametric model has similar performance to the one proposed in [CCC+19] (= 99.89%) while being much more efficient computational wise (*i.e.* $1,950,323$ against $39,015$ trainable parameters). In this manuscript, we want to evaluate the suitability of the ensemble models when the number of profiling traces is limited (as often in practice). Hence, we only use $30,000$ profiling traces and $3,000$ validation traces instead of using the $750,000$ traces considered by [CCC+19]. However, when the Evaluator trains a parametric model with the $30,000$ raw profiling traces of $13,000$ time samples, he already generates a classifier with very high performance (= 98.30%). Hence to efficiently evaluate the suitability of the ensembling loss, we add Gaussian noise $\mathcal{N}(0, \sigma^2)$ on each time sample of the leakage traces such that σ defines the standard deviation of the noise. Table 8.1 shows the evolution of the accuracy depending on the added noise on the Secure RSA dataset. In the following sections, σ is set to 6 in order to evaluate the benefits of the ensembling loss against the negative log-likelihood and the ranking loss. The

model trained with the negative log-likelihood is considered as the state-of-the-art result because it uses the classical learning metric in the side-channel context. This result will be considered as our reference in order to highlight the performance provided by the ensembling loss.

Table 8.1: Evolution of accuracy depending on σ ($30,000$ profiling traces & $3,000$ validation traces)

α_{bit} \ σ	0	10^{-3}	10^{-2}	10^{-1}	1	6	8	10	50
Negative Log-Likelihood	98.30%	97.60%	97.77%	96.33%	95.70%	92.50%	85.80%	80.60%	41.77%

Learning ensemble diversity. To assess the diversity growth, we generate an ensemble model composed of 5 committee members with the same neural network architecture introduced in the previous paragraph. Consequently, the diversity provided by the following ensemble models only depends on the loss used.
Table 8.3 illustrates the performance evolution depending on the diversity type and the learning metric applied. If the Evaluator only considers the state-of-the-art result, he trains a unique model with the negative log-likelihood (*i.e.* NLL_1) to perform its attack. However, when he applies the ranking loss function introduced in Chapter 7, the related attack performance is more efficient than the negative log-likelihood (see Table 8.3). While a parametric model obtained from the minimization of the empirical risk combined with the negative log-likelihood does not provide a useful model from the Evaluator's restrictions (*i.e.* $\mathcal{C}_{\{NC,2^b,AA\}} \geq 100$), using the ranking loss the Evaluator can potentially break the Secure RSA implementation (*i.e.* $\mathcal{C}_{2^b} \leq 100$).
Finally, we can observe a meaningful improvement when the ensembling loss is performed on the 5 committee members defined in $\mathcal{E}$. Even if the training time is multiplied by 9 in the worst case, it stays reasonable from a practical perspective. Indeed, α_{label} is increased by up to 2.69% and the Evaluator can extend its attack scenario. Following the SOG-IS recommendations, he can successfully perform an alternate attack while the state-of-the-art (*i.e.* NLL_1) result cannot. This result highlights the benefit of using the ensembling loss in terms of ensemble performance. In addition, considering the ensembling loss reduces $\mathcal{C}_{2^b}$ by 25. Hence, the theoretical features of the ensembling loss, which are validated through the visualizations of Subsection 8.3.2, translate an actual gain in model accuracy as well as a realistic improvement for a full side-channel attack scenario. The ensembling loss increases the overall diversity and reduces the global error rate induced in the ensemble model. Hence, the ensembling loss is helpful to promote the complementarity between the committee members.

Ensembling loss combined with type I diversity. As mentioned in Subsection 8.2.1, the type I diversity refers to the heterogeneity between the committee members' structure. This diversity is employed by Perin *et al.* [PCP20] to argue the generalization improvement induced by this ensemble method. In the following, we propose to combine the type I diversity with the different loss functions to evaluate the resulted gain in attack complexity. For that purpose, we randomly generate 5 networks with a wide range of hyperparameters (details are provided in Appendix F Table F.1). In [Zho12], Zhou recommends the configuration of heterogeneous neural network architectures with high individual performance. Even if this solution can be intuitive, this is not necessarily the best one as discussed in Subsection 8.3.2.
From a diversity perspective, using efficient heterogeneous neural networks increase the uncorrelated errors. Through Figure 8.5, combining the type I diversity with the ensembling loss reduces the overall κ-statistic metric. Following Definition 8.3.2.2, this observation confirms the gain in diversity. This result can also be verified with the disagreement measure (see Figure 8.6).
From a performance perspective, the individual committee members do not exceed 94.33% for retrieving the bits of blinding exponent when the ranking loss is considered (see Appendix F Table F.1). However, applying the ensembling loss adjusts the efficiency of each learner to increase their complementarity. Indeed, though Figure 8.6, the most powerful member finds 95.84% of all

bits while the least significance one finds only 88.21% of all bits. The interaction between the committee members during the training process tends to accentuate the discrepancy in order to force the gain in diversity. Table 8.3 illustrates the benefits of combining the type I diversity with the ensembling loss from a performance perspective. Adding type I diversity reduces the remaining attack complexity regardless of the attack scenario.

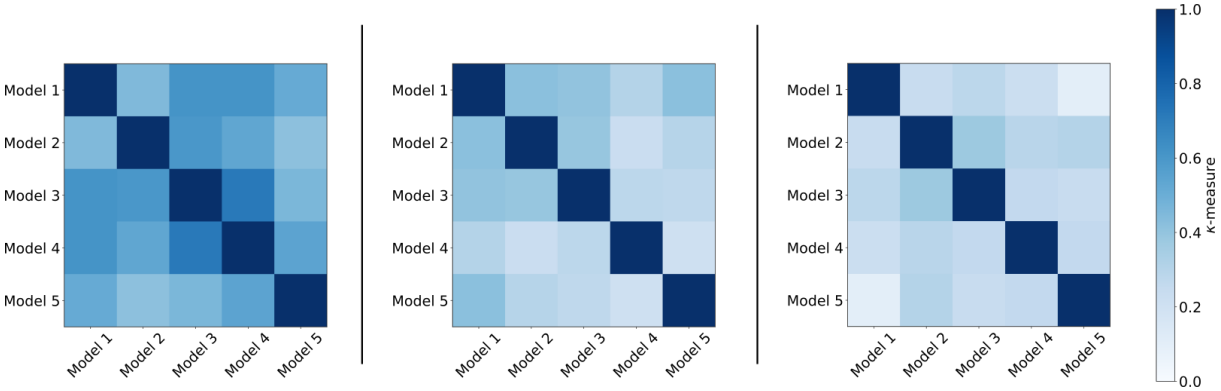


Figure 8.5: κ-statistic. From left to right: Ensembling Loss, Ensembling Loss + Type I diversity, Ensembling Loss + Type I + II diversity.

Finally, even if the resulted training time increases, it stays marginal related to the gain in attack complexity. Indeed, depending on the scenario, the attack can be performed by up to 2^{50} operations. In comparison with the previous state-of-the-art result (*i.e.* NLL_1), the number of operations is reduced by 2^{58} while the training time is only increased by 10.

Combining all types of diversity. Introduced in [LWC+19], the type I+II diversity consists of the selection of members from a pool such that the diversity measure is maximized between all the committee members. These parametric models are selected by randomly picking out the hyperparameters from ranges defined in Table 8.2. The resulted pool is composed by 100 members. As recommended in [LWC+19], we retain a set of parametric models with high performance[g] (*i.e.* $\alpha_{label} > 85\%$) such that their disagreement measure is maximized. The 5 selected neural network architectures are identified in Appendix F Table F.2.

Table 8.2: Range of hyperparameters selection

	Values
$N_{\textbf{filt.}}$	$\{2, 4, 8, 16, 32, 64\}$
Filter size	$\{1, 5, 11, 21, 43\}$
$N_{\textbf{conv.blocks}}$	$\{1, 2, 3, 4, 5\}$
Pooling operation	{Average, Max}
Pooling stride	$\{2, 4, 6\}$
$N_{\textbf{FC layers}}$	$\{0, 1, 2, 3\}$
$N_{\textbf{nodes per FC layers}}$	$\{2, 4, 8, 16, 32, 64\}$

The type I+II diversity promotes the error uncorrelation between the individual committee members. Following the κ-statistic measure (see Figure 8.5), the diversity brought by the type I+II is significant in comparison with the previous experiments. Indeed, the overall κ-statistic measure is reduced in comparison with the other approaches. This observation can also be made with the disagreement measure (see Figure 8.6).

[g] In some cases (*e.g.* boosting [CG16]), weak learners (*i.e.* models that are only slightly better than random guessing) can be helpful to increase the performance of the ensemble model. The benefit of these strategies is considered as out of our scope.

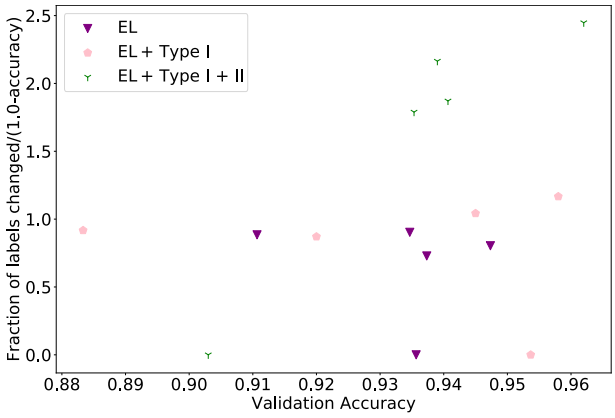


Figure 8.6: Diversity versus label accuracy plots for ensemble models trained on Ensembling Loss (EL), Ensembling Loss (EL) + Type I diversity and Ensembling Loss + Type I + II diversities.

When the ranking loss or the negative log-likelihood is considered, even if no interaction is proposed between the committee members during the training process, using the type I+II diversity is useful to bring more diversity in the ensemble model. However, combining the type I+II diversity with the ensembling loss accentuates the gain in diversity in order to generate a more powerful model. From Table 8.3, we observe a significant improvement when the ensembling loss is used in comparison to the ranking loss and the negative log-likelihood. Generating interactions between the committee members provides more consistency during the training process. As mentioned in Subsection 8.3.2, the ensembling loss leads to converge the uncertain predictions towards the equidistant point of the centroid of the clusters. Thus, the impact of the FP and FN is reduced when this loss function is performed. Combining all the diversity techniques provides the most efficient model in terms of performance. While an ensemble model trained with the ranking loss needs 2^{56} operations to retrieve the remaining bits in the best-case scenario, the addition of the ensembling loss with the type I+II diversity needs only 2^{34} operations. Even if the resulted training time is multiplied by 3, the gain in performance is significant to justify the benefits of the ensembling loss.

Table 8.3: Performance evaluation depending on the diversity's type (Average over 10 physical traces of $1,088$ bits each). Green (resp. Red) cells are considered as practicable (resp. unpracticable) following the SOG-IS recommendations.

Model	α_{label}	α_{bit}	ϵ_{bit}	$\mathcal{C}_{NC}$	$\mathcal{C}_{2^b}$	$\mathcal{C}_{AA}$	Training time (seconds)
$\boldsymbol{NLL_1}$	92.50%(±2.06%)	90.16%(±2.32%)	0.0984	500.08	108	102.74	$86s$
$\boldsymbol{RkL_1}$	94.03%(±1.85%)	91.26%(±2.21%)	0.0874	460.66	96	102.74	$81s$
$\boldsymbol{NLL_5}$	93.60%(±1.91%)	91.04%(±2.23%)	0.0896	467.39	98	102.74	$450s$
$\boldsymbol{RkL_5}$	94.33%(±1.81%)	91.66%(±2.16%)	0.0834	443.55	91	99.37	$340s$
$\boldsymbol{EL_{5,III}}$	95.19%(±1.67%)	92.45%(±2.07%)	0.0754	415.25	83	96.06	$740s$
$\boldsymbol{NLL_{5,I}}$	95.31%(±1.65%)	93.28%(±1.96%)	0.0672	381.96	74	92.43	$688s$
$\boldsymbol{RkL_{5,I}}$	95.93%(±1.55%)	94.48%(±1.79%)	0.0552	330.81	61	92.43	$780s$
$\boldsymbol{EL_{5,I+III}}$	96.57%(±1.42%)	95.49%(±1.62%)	0.0451	284.21	50	88.45	$884s$
$\boldsymbol{NLL_{5,I+II}}$	96.23%(±1.49%)	94.66%(±1.76%)	0.0534	322.58	59	88.45	$2,116s$
$\boldsymbol{RkL_{5,I+II}}$	96.27%(±1.48%)	94.94%(±1.71%)	0.0506	310.01	56	88.45	$1,092s$
$\boldsymbol{EL_{5,I+II+III}}$	97.33%(±1.26%)	96.96%(±1.34%)	0.0304	209.52	34	80.71	$3,392s$

In conclusion, combining all the diversity techniques provides a clear advantage from a side-channel point of view. Indeed, when the type I+II diversity techniques are combined with the ensembling

loss (*i.e.* type III diversity), we promote the diversity between the parametric models included in $\mathcal{E}$ in order to reduce the global error. In comparison with the previous state-of-the-art result (*i.e.* NLL_1), α_{bit} is increased by 6.8% and the number of remaining operations is reduced by $2^{290.56}$ (resp. 2^{74} and $2^{22.03}$) when the adversary wants to perform a naive attack (resp. a 2^b-attack and an alternate attack). Even if the training time is increased by up to 39.44, it stays negligible regarding the gain to perform the full attack. Indeed, following the SOG-IS recommendation and the Evaluator's restrictions (see Subsection 8.1.1), the previous state-of-the-art result considers the RSA implementation as secure while combining the different diversity techniques leads the Evaluator to reconsider its security assessment.

While the Evaluator has to deal with different attack scenario, we extend the application of the diversity approach on the Secure ECC dataset that requires a single leakage trace to retrieve all the private key bits.

8.4.2 Application on 1 Trace Exploitation

To emphasize the benefits of the ensembling loss, we evaluate its suitability on a classical binary classification problem. While the secure RSA dataset can be defined as a multi-class classification task (3 outputs), we perform the same experimental process on the Secure ECC dataset. Note that remaining brute-force attacks that require N_a exploitation traces, such as [SW14], cannot be used in this context.

Neural Network Architecture. Similarly to the secure RSA dataset, we have to add Gaussian noise $\mathcal{N}(0, \sigma^2)$ to characterize the benefits of the ensembling loss. Table 8.4 shows the evolution of the accuracy depending on the added noise and the loss used when $20,000$ profiling traces are used. To evaluate the suitability of each network, $2,000$ validation traces are considered and the evolution of the accuracy is used to limit the overfitting/underfitting effect. For our analysis, we set the added noise to $\sigma = 30$. Once again, we clearly evaluate the benefits of the ranking loss when the added noise is high in comparison to the negative log-likelihood loss function. Indeed, we increase by up to 3.5% the resulted performance.

Table 8.4: Evolution of accuracy depending on σ ($20,000$ profiling traces & $2,000$ validation traces)

α_{bit} \ σ	0	10	20	30	50	100
Negative Log-Likelihood	99.43%	98.20%	93.63%	89.10%	83.87%	72.10%
Ranking Loss	99.47%	99.00%	96.77%	92.60%	86.15%	74.30%

Ensemble diversity. First, we validate the theoretical observations provided in Subsection 8.3.1 for the binary classification problem. For that purpose, we visualize the t-SNE maps for the models trained with the different losses. Figure 8.7 confirms all the theoretical results introduced in Section 8.3. The negative log-likelihood representation does not seem relevant to efficiently discriminate each cluster. The resulted ensemble model seems to select joint patterns such that many FP and FN can deteriorate the overall performance. In opposition, the model trained with the ranking loss creates compact clusters such that the false positives and false negatives can be considered as consistent. Finally, from a theoretical perspective, the ensembling loss seems the most suitable. Indeed, the data uncertainty seems to converge towards the equidistant point of the centroid of the clusters. Hence, the resulted ensemble model tends to gather the uncertain examples towards a uniform probability distribution. Furthermore, using the ensembling loss provides a clear benefit from a diversity perspective (see Figure 8.8).

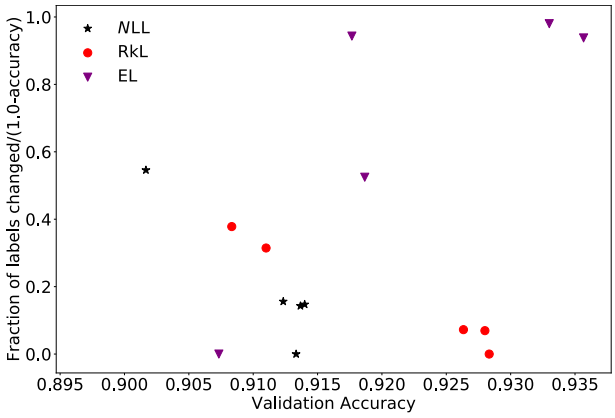


Figure 8.8: Diversity versus label accuracy plots for 3 ensemble models trained on Negative Log-Likelihood (NLL), Ranking Loss (RkL) and Ensembling Loss (EL) for the binary classification.

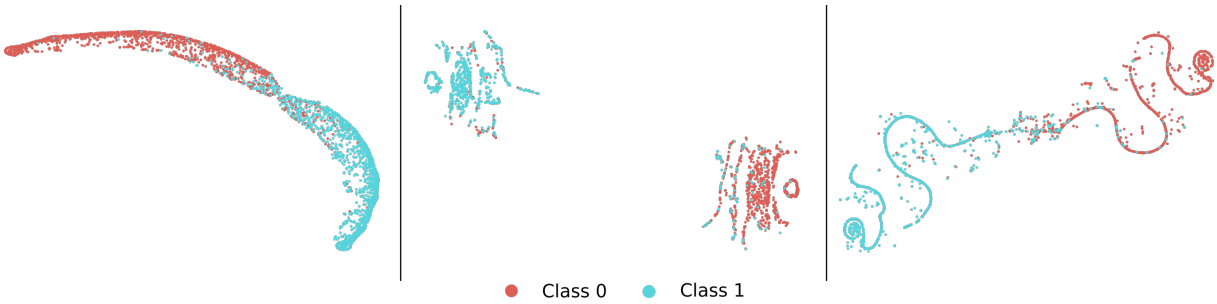


Figure 8.7: t-SNE embeddings. From left to right: Negative Log-Likelihood, Ranking Loss, Ensembling Loss.

Through Table 8.5, we confirm the benefits of the ensembling loss for increasing the performance of the ensemble model. In comparison with the previous state-of-the-art result (*i.e.* NLL_1), the accuracy expressing the performance to retrieve the conditional swap bit value is increased by 6.5% when the ensembling loss is combined with the type I and II diversities. From a side-channel attack perspective, we reduce the overall number of remaining operations by $2^{58.41}$ (resp. 2^{16}) for naive attack (resp. 2^b-attack). Hence, using the ensembling loss against a binary classification problem still performs well.

Table 8.5: Performance evolution depending on the diversity applied (Average over 10 physical traces of 256 bits each). Green (resp. Red) cells are considered as practicable (resp. unpracticable) following the SOG-IS recommendations.

Model	α_{bit}	ϵ_{bit}	$\mathcal{C}_{NC}$	$\mathcal{C}_{2^b}$	Training time (seconds)
$\boldsymbol{NLL_1}$	89.10%(±2.44%)	0.109	120.91	28	27s
$\boldsymbol{RkL_1}$	92.60%(±2.05%)	0.074	90.72	19	28s
$\boldsymbol{NLL_5}$	91.60%(±2.17%)	0.084	101.45	22	440s
$\boldsymbol{RkL_5}$	92.63%(±2.04%)	0.0737	90.72	19	387s
$\boldsymbol{EL_{5,III}}$	93.33%(±1.95%)	0.0667	86.98	18	588s
$\boldsymbol{NLL_{5,I+II}}$	94.13%(±1.84%)	0.0587	79.24	16	618s
$\boldsymbol{RkL_{5,I+II}}$	94.70%(±1.75%)	0.053	71.1	14	750s
$\boldsymbol{EL_{5,I+II+III}}$	95.60%(±1.60%)	0.044	62.50	12	884s

From a naive attack perspective, an adversary using the previous state-of-the-art result (*i.e.*

NLL_1) considers the ECC implementation as secure following the SOG-IS's recommendations ($C_{NC} > 100$). However, if the Evaluator combines all the diversity types, including the ensembling loss, he can reconsider the security of the targeted device.

8.4.3 Discussion

This discussion evaluates the classical ensemble methods (*i.e.* Bagging [Bre96], Boosting [FS96, CG16]), the parametric model fusion's techniques (*i.e.* average accuracy, voting) and the impact of the number of committee members.

Ensemble Methods. Traditionally, the methods considered in ensembling are the *Boostrap Aggregating* [Bre96], also known as *Bagging*, and the *Boosting* [FS96, CG16] techniques. Through this discussion, we evaluate the benefits of these techniques in addition to the current ensemble models.
The bagging and boosting approaches are not new in side-channel context [MPP16, PSK+18, PCP20]. While these algorithms are essentially performed with *Random Forest* (RF) [Bre01], it can also be proposed for enhancing the performance of neural networks. The details on the hyperparameters selection are provided in Appendix F Table F.3 for the bagging selection and in Appendix F Table F.4 for the *eXtreme Gradient Boosting* (XGBoost) [CG16] and the CNN-XGBoost [RGL+17].
The best results for all the models are reported in Table 8.6. In our experiment, this table illustrates that bagging and XGBoost do not provide a clear advantage when they are added to the standard proposition introduced in Table 8.3. However, if an improvement is observed, these algorithms can be combined with those introduced in this chapter (*i.e.* Type I+II+III diversity) in order to generate a more powerful ensemble model.

Table 8.6: α_{label} for each ensemble method (Average over 10 physical traces of $1,088$ bits each)

	Bagging			**XGBoost & CNN-XGBoost**			
	$RMSE$	NLL_5	RkL_5	$RMSE$	NLL_5	RkL_5	$EL_{5,III}$
$\boldsymbol{\alpha_{label}}$	84.97% (±2.80%)	93.02% (±1.99%)	93.70% (±1.90%)	91.43% (±2.19%)	93.70% (±1.90%)	94.13% (±1.84%)	94.77% (±1.74%)
Training time (seconds)	$4,073s$	$315s$	$415s$	$7,597s$	$481s$	$372s$	$775s$

Remark 8.4.3.1. The ensembling loss cannot be considered when the bagging technique is applied. Indeed, given a profiling set $\mathcal{T}$, the N_p pairs $((\mathbf{t}_i, y_i)_{0 \leq i < N_p})$ should be the same for all the committee members when the ensembling loss is computed. This condition is a limitation regarding the application of the bagging algorithm.

Combination Methods. One major issue when ensemble model is considered is to find the best way to combine the posterior probabilities of each committee member. There are several consensus methods for combining the outputs of multiple learners. We compare the two most useful combining methods:

- *Averaging* – This consensus is considered as a linear combining method. The average prediction returned by the committee members is computed. An advanced combination technique consists of weighting the average of each classifier to promote the order of the classes. However, this method stays out of our scope.

- *Voting* – This method is considered as a non-linear decision-making based on ranked information. The majority voting process predicts the value with the highest number of occurrences. Hence the collective decision has a major impact on the final prediction.

Table 8.7: α_{label} for each combination technique (Average over 10 physical traces of 1,088 bits each)

	$\boldsymbol{NLL_5}$	$\boldsymbol{RkL_5}$	$\boldsymbol{EL_{5,III}}$
Averaging	93.60%(±1.91%)	94.33%(±1.81%)	95.19%(±1.67%)
Voting	93.63%(±1.91%)	94.30%(±1.81%)	95.16%(±1.68%)

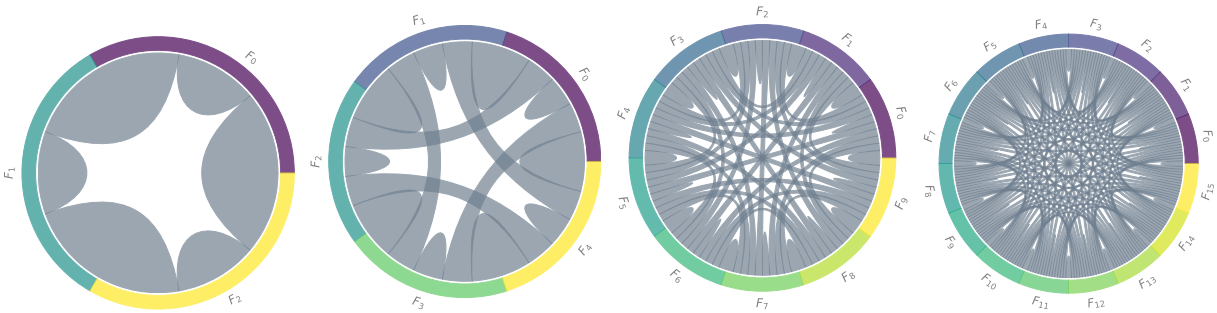
These results shown in Table 8.7 are closely correlated with those defined in the previous sections. Hence, for the experiments investigated in this chapter, these aggregating functions do not impact the performance of the ensemble model.

Number of Committee Members. The number of committee members can also be considered as an issue in ensemble methods. Indeed, no useful methods define *a priori* the best number of committee members that maximize the ensemble model performance. In the following, we explore this variable in order to identify its impact on the ensembling loss performance. To that purpose, we increase the number of committee members up to 32 in order to evaluate the gain in performance and the impact on the training time. Through Table 8.8, we can estimate the best trade-off between the training time and the ensemble performance. For the RSA implementation, the best α_{bit} value is obtained for $N = 10$ committee members.

Table 8.8: Performance evolution depending on the committee members (Average over 10 physical traces of 1,088 bits each)

Model	α_{label}	α_{bit}	ϵ_{bit}	C_{NC}	C_{2^b}	C_{AA}	Training time (seconds)
$\boldsymbol{EL_{2,I+II+III}}$	96.77%(±1.38%)	95.12%(±1.69%)	0.0488	301.54	54	88.45	$1,482s$
$\boldsymbol{EL_{5,I+II+III}}$	97.33%(±1.26%)	96.96%(±1.34%)	0.0304	209.52	34	80.71	$3,392s$
$\boldsymbol{EL_{10,I+II+III}}$	97.43%(±1.24%)	97.33%(±1.26%)	0.0267	189.23	30	80.71	$5,460s$
$\boldsymbol{EL_{16,I+II+III}}$	97.90%(±1.12%)	97.10%(±1.31%)	0.0290	199.47	32	80.71	$6,942s$
$\boldsymbol{EL_{32,I+II+III}}$	97.37%(±1.25%)	96.67%(±1.40%)	0.0333	225.26	37	84.03	$9,548s$

While increasing the number of members seems helpful to improve the ensemble model's accuracy, in our context we seem to reach the maximal possible performance. Adding too many learners can reduce the diversity effect because some committee members can share the same errors and promote irrelevant outputs. Furthermore, as illustrated in Figure 8.9, adding committee members in $\mathcal{E}$ increases the computational complexity of the ensemble model and, consequently, the *elapsed time* criterion. As no general consensus can be formulated, the best number of committee members should be defined for each case study.



(a) $N_c = 3$. **(b)** $N_c = 5$. **(c)** $N_c = 10$. **(d)** $N_c = 16$.

Figure 8.9: Mutual information ensemble diversity principle considering pairwise components with ensemble model $\mathcal{E}$ with different size N_c.

8.5 Conclusion

This chapter proposes to deal with the Evaluator's restriction when he has to perform an attack against an asymmetric cryptographic implementation. In this context, the Evaluator has to maximize the accuracy of retrieving each secret bit of the targeted private key such that, a slight improvement can be non-negligible from a side-channel attack complexity perspective.
A classical solution to enhance the performance of the deep learning approach consists in combining individual predictions from several parametric models via a consensus method (*e.g.* majority vote, average vote) in order to reduce the global error. This error reduction is known to be correlated with the parametric model's diversity. Following the work provided in [LWC+19], we describe three types of diversity and assessing their suitability to perform a side-channel attack under the Evaluator's restriction. To enhance the evaluation of the robustness of cryptographic module, we develop a new loss, namely the *Ensembling Loss*, that increases the performance of ensemble models. Promoting the interactions between the committee members during the training process, this loss increases the resulted diversity to reduce the correlation between the errors induced by the committee members. First, we link this new learning metric with the mutual information between the ensemble model and its related label introduced by Brown in [Bro09]. Then, through the disagreement measure and the t-SNE visualization, we show that ensemble models trained with the *Ensembling Loss* increase the diversity between the committee members.

To assess the benefits from a side-channel perspective, we evaluate the accuracy growth on the remaining attack complexity through multiple attack scenarios, namely *Naive Attack*, 2^b*-Attack* and *Alternate Attack*. This investigation shows that applying deep learning-based side-channel attacks can be unadapted to defeat secure RSA/ECC implementations if the previous state-of-the-art is considered (*i.e.* a single model trained with the negative log-likelihood loss function). Following the SOG-IS security guidances, the improvement provided by the combination of different types of diversity lead to a reconsideration of the targeted system's security.

Furthermore, considering the *Ensembling Loss* outperforms all the current learning metrics classically used in side-channel analysis. Hence, this loss could be considered for generating efficient ensemble models.

Part IV

Conclusion of This Thesis

Chapter 9

Conclusion & Perspectives

In this thesis, we focused on several current issues related to the deep learning-based side-channel attacks, developing new neural network architectures in order to fit with the Evaluator's restrictions as well as proposing new learning metrics for converging towards the optimal Adversary. In this chapter, we conclude the contributions made in this manuscript according to two issues: (i) the problem of modeling Side-Channel Attacks and (ii) the ability of the resulted models to converge towards the optimal Adversary. For each of these two parts, we provide a summary of the related contributions and discuss some perspectives arising from the results obtained.

Contents

9.1 Modeling Side-Channel Attacks

Modeling side-channel attacks boils down to be able to model the relationship between a leakage trace $\mathbf{T}$ and a targeted sensible variable Y which can be addressed by modeling the true unknown condition distribution $\mathsf{Pr}[\mathbf{T}|Y]$. The first part of this manuscript differentiates two traditional approaches to approximate this distribution. While a classical deep learning approach consists in considering *discriminative models* in order to directly estimate $\mathsf{Pr}[\mathbf{T}|Y]$, alternative solutions, known as *generative models*, approximate $\mathsf{Pr}[Y|\mathbf{T}]$ which can be simplified as $\mathsf{Pr}[\mathbf{T}|Y]$ up to an additive constant if the logarithm representation is considered (see Subsection 3.3.3). This thesis illustrates the benefits and the limitations of both solutions from an Evaluator point of view. While the Deep Learning field is useful to mitigate the inconsistent choices made by the Evaluator during the preprocessing phase and to provide a clear gain in attack efficiency, its lack of explainability in the side-channel context remains a tremendous issue.

Generative models. To bridge the gap between Deep Learning and Side-Channel Analysis, we describe a new generative model based on one of the most generic profiled attacks namely *stochastic attack*. Through our investigations, we emphasize the similarities with the *conditional variational autoencoder* and develop a new neural network architecture which derives from the stochastic attack scenario. Through this approach, we approximate the targeted leakage model ψ and emphasize the ability of the cVAE-ST to correctly assess the robustness of a cryptographic module against side-channel attacks. Furthermore, in order to reduce the black box issues, some visualization techniques are employed in order to assess the security flaws induced in the targeted cryptographic module. Through this approach, the Evaluator can alert the Developer on potential vulnerabilities and ease the development of adapted countermeasures. However, even if these promising results suggest that generative models are suitable to perform side-channel attacks,

some perspectives can be highlighted in order to enhance the state-of-the-art result. Indeed, an open question relates to the ability of the cVAE-ST to efficiently prevent, or at least mitigate, the countermeasures classically considered by the Developer namely data randomization and hiding countermeasures.

Perspective 9.1.1 (Hiding countermeasure)**.** *Assessing the benefits of some neural cVAE-ST properties to mitigate the hiding effect while preserving its interpretability and explainability.*

Perspective 9.1.2 (Dimensionality reduction)**.** *Finding a dimensionality reduction technique in order to focus the latent space $\mathcal{V}$ on informative time samples (*i.e. *points of interest) while preserving the interpretability and explainability of cVAE-ST.*

A first step to enhance the state-of-the-art results consists in adding convolutional blocks as mentioned in Subsection 5.4.2. Indeed, regarding the result obtained from the discriminative models, the convolutional blocks can be useful to reduce the leakage trace dimension as well as mitigating the desynchronization effect. The Evaluator can add those blocks to the encoder in order to *preprocess* the leakage traces before the estimation of the true unknown leakage model ψ. However, even if this solution can be suited for mitigating hiding countermeasures as well as reducing the dimensionality of the leakages, some verification should be made in order to assess the loss of interpretability and explainability.
Furthermore, in the black-box context, the Evaluator needs to exploit joint distribution of leakages in order to exploit information that depend on two unknown random variables X and Y (*e.g.* plaintext, output of the Sbox) [LDLL14, CR17]. In this scenario, the Evaluator has to deal with a blinded environment such that, he does not know the variables manipulated by the cryptographic module. This approach is similar to the *Unsupervised learning* approach. As mentioned in Subsection 5.1.3, the generative approach can be useful to solve such problems and computing $\mathsf{Pr}[k|X,Y]$ which denotes the probability of observing a given key hypothesis $k \in \mathcal{K}$ knowing the random variables X and Y.

Perspective 9.1.3 (Blind Side-Channel Attacks)**.** *Assessing the suitability of generative models to conduct Blind Side-Channel Attacks.*

This direction can then be extended to the non-profiled side-channel attacks.

Discriminative models. As demonstrated by Cagli *et al.* [CDP17a], the convolutional neural networks are highly beneficial to mitigate the hiding countermeasures. In Chapter 6, we validate this observation by investigating the impact of some model's hyperparameters, namely filter size, pooling operation as well as the number of convolutional blocks. Through this study, we design a new feature selection part in order to focus the interest of the network on the points of interest only without altering the performance of the model. In comparison with the state-of-the-art results, our neural network architecture suggests that a low complexity neural network can be sufficient to defeat implementations considering first-order Boolean masking schemes and/or random delay effect. Recently, this observation has been validated on other countermeasures namely code polymorphism [MBC+20], Shamir's Secret Sharing mechanism [Mag19a, Mag19b], 1-amongst-N countermeasure [Mag19a, Mag19b] and shuffling protection [Mag19a, Mag19b]. However, instead of the CNN, other neural network architectures (*e.g.* ResNets, RNNs, U-Nets, LSTM) have not been widely investigated in the side-channel context. Even if those approaches show promising results [ZS19, Mag19a, MS21], their benefits are still unknown in comparison with the convolutional neural networks.

Perspective 9.1.4 (Neural Network Architectures)**.** *Assessing the benefits of other neural network architectures in comparison with convolutional neural networks.*

For example, Cagli questions the feasibility of using *Siamese Neural Networks* in order to perform collision attacks [Cag18].

Table 9.1: cVAE-ST vs. Discriminative models from a SCA perspective.

	cVAE-ST	**Discriminative models**
Interpretability & Explainability	✓	✗
Automatically find the PoIs	✓	✓
Defeat hiding countermeasures	?	✓
Perform HO-SCA without preprocessing	✗	✓

In addition, [Mag20, ZXF+19] introduce multi-task learning in the side-channel context. This approach targets simultaneously multiple intermediate sensitive variables, subparts of the secret key, or the binary representation of a given byte. The latter solution sounds highly beneficial to conduct future works. Indeed, given a word of n bits, the Evaluator can design a neural network with n outputs such that each of them is associated with a bit of the targeted word. Thus, given a leakage trace $\mathbf{T}$ which captures the consumption related to this n-bit word, the Evaluator can assign to it n labels. Applying the multi-task learning approach to such problem is useful as the learning metrics assess the ability of the neural network to retrieve each bit of the word. Consequently, the Evaluator can define a confidence level, characterized by a threshold, such that, each bit with a score higher than this threshold is considered as a certain prediction implies that the prediction is considered certain. Based on those certain bits, the Evaluator can reduce the number of n-bit key candidates for a set of leakage traces. Furthermore, using the Ensemble approach introduced in Chapter 8 should be highly beneficial in this context if the Evaluator combines neural networks with different levels of certainty on bit predictions.

Perspective 9.1.5 (Multi-Task Learning)**.** *Assessing the benefits of the multi-task learning approach in order to target n-bit words such that $n > 8$.*

For example, if the Evaluator targets an n-bit data bus where the secret key transits (*e.g.* the bus between the microprocessor and the memory), he can use the multi-task learning approach in order to simultaneously target each bit of the secret key. Then, based on the score related to each bit, he expects to reduce the space of hypothetical keys in order to perform a practicable attack.

Hybrid discriminative-generative model. In Part II, we analyze the benefits and the limitations of the cVAE-ST and the discriminative approach. Regarding the needs of the Evaluator (*i.e.* interpretability, mitigating hiding countermeasures, points of interest selection and performing HO-SCA[a] without any preprocessing), we observe that both approaches are complementary (see Table 9.1). Furthermore, following [Las08], one interesting property of the generative models is that they independently learn each category (*i.e.* dependencies for a given label). Thus, if the Evaluator wants to add one class to a given generative model, he keeps the one already trained and just performs an additional learning phase for the new category. In opposition, for a discriminative model, the Evaluator has to start the learning process from scratch if an additional category (*e.g.* class) is added to the model. Thus, considering the generative approach provides a more flexible solution from an evaluation point of view. Moreover, following Chapter 5, the Evaluator can use the cVAE-ST to retrieve an interpretable (Θ, ϕ)-parametric model $F_{\Theta,\phi}$ but cannot perform HO-SCA without preprocessing phase. In order to automatically find a combining function defeating Boolean masking implementation, the Evaluator can consider the discriminative models as illustrated over Chapter 6. Thus, the Evaluator can question the feasibility of combining both approaches in order to preserve all the benefits introduced in Table 9.1.
In opposition to the *Generative Adversarial Network* (GAN) which can be considered as a *Generative-Discriminative model*[b], our perspective suggests designing a *Discriminative-Generative model*.

[a] This term refers to the *High-Order Side-Channel Analysis* described in Section 3.5

[b] During the training process, a GAN aims at finding the generative model's configuration such that the discriminative model cannot differentiate a fake data from a real one.

Perspective 9.1.6 (Discriminative - Generative Model)**.** *Designing a Discriminative - Generative model such that the Discriminative model automatically finds the best combining function while the cVAE-ST aims at identifying some dependencies between the leakage trace and the targeted unmasked variable Y.*

Through this approach, the Evaluator keeps the interpretability induced by the cVAE-ST model while defeating Boolean masking implementation at least. Considering this approach can be highly beneficial from an evaluation point of view because the *elapse-time* criterion can be drastically reduced if a deterministic solution exists to construct the discriminative model. This perspective needs to develop a new paradigm in order to reduce even more the gap between Deep Learning and Side-Channel Analysis fields.

On the other hand, this manuscript covers the optimal adversary's topic. The following section summarizes the contributions that have been made and it proposes some perspectives in order to enhance the attack performance.

9.2 Converging Towards the Optimal Adversary

The second part of this manuscript focuses on reducing the errors induced in Θ-parametric functions F_Θ. In particular, we reduce the gap between Deep Learning and Side-Channel Analysis by designing new loss functions that reduce the error induced by individual and collective Θ-parametric functions.

Learning metric. While classical DLSCA considers the negative log-likelihood loss function as suited to conduct the training process, we suggest an alternative, namely *Ranking Loss*, in order to reduce the gap between Machine Learning metrics and those introduced in side-channel context. Through this investigation, we highlight the correlation between the side-channel context and the learning to rank paradigm such that minimizing the ranking loss is beneficial to maximizing the Success Rate of a given neural network. This result suggests that a model trained with the ranking loss aims at retrieving the trainable parameters Θ that fit with the Optimal Adversary introduced in Objective 3.3.1.1. This new learning metric prevents the approximation error and performs, in the worse case, similarly to a parametric function F_Θ selected from the minimization of the empirical risk considering the *Negative Log-Likelihood* loss function. When the Evaluator trains a neural network, he generally computes Equation 7.5 such that $N_a = 1$. In other words, he wants to select the Θ-parametric function F_Θ that performs the best when a single leakage trace is considered during the attack phase. However, if $N_a > 1$, the Evaluator can aggregate the score $s_{N_a,i}(F_\Theta, k)$ for each key hypothesis $k \in \mathcal{K}$ and update the trainable parameters Θ accordingly.

Perspective 9.2.1 (Configuration of the Ranking Loss with $N_a > 1$)**.** *Assessing the benefits of using $N_a > 1$ leakage traces during the training phase when the ranking loss is considered and evaluating the benefits and the limitations both from a practical and a performance perspective.*

If this perspective eases the evaluation of the underfitting and overfitting issues on trivial scenarios, the Evaluator can question the feasibility of defining a bound on N_a following the SNR computation as suggested in [MSB16] for the CPA attack. It can be highly beneficial to perform *Early Stopping* techniques based on the learning metrics without having to perform a full attack [RZC+21].
On the other hand, Perspective 9.1.6 suggests the need of a new loss function in order to deal with the hybrid discriminative-generative model.

Perspective 9.2.2 (Loss function for Discriminative - Generative Model)**.** *Identifying a loss function which automatically finds a combining function that maximizes its dependency with the targeted variable Y such that the generated leakage trace $\tilde{\boldsymbol{T}}$, induced by the cVAE-ST, optimizes its similarity with the combined leakage trace.*

Finding such solution can be helpful to combine the benefits of both approaches namely interpretability and HO-SCA. The optimal goal of this research direction consists in finding a model fully interpretable and explainable from a side-channel perspective such that HO-SCA can be performed while facilitating the design of a suitable neural network based on the theoretical investigations provided in Chapter 5.

Uncertainties. To converge towards the Optimal Adversary, we also demonstrate the benefits of combining the probability distributions provided by multiple parametric models in order to reduce the *global error* induced in an ensemble model (see Chapter 8). More precisely, we design a new loss function, namely *Ensembling Loss*, in order to promote interactions between the parametric models during the training process such that correlated errors are penalized. Constructed from the mutual information between an ensemble model and the targeted sensitive variable, this loss function is beneficial to differentiate certain from uncertain predictions. This eases the location of uncertain bits and thus, promotes a full attack scenario (*i.e.* retrieving all the private key bits) against asymmetric cryptographic implementations. However, to capture differentiation between certain and uncertain predictions, the ensembling loss is confronted to the ability of the Evaluator to design suitable neural network architectures. If the parametric models are not suitably configured, the uncertain bits can be difficult to locate.

Perspective 9.2.3 (Number of Remaining Operations)**.** *Locating the uncertain predictions in order to reduce the number of remaining operations and enhance full attack scenarios against asymmetric cryptographic implementations.*

Some methods have been investigated in the Machine Learning field in order to reduce the risks of wrong diagnosis in personalized medicine or meteorological forecasting. First, the *Calibration* method [KLM19] converts the scores[c] assigned to each bit into frequentist reading. In other words, the bits with a calibrated probability of p should have an observed frequency of belonging to the correct class to be precisely p. Thus, this approach can be highly beneficial to define the score threshold such that all the bits over this score are perfectly predicted. On the other hand, the *Conformal Prediction* [CGD21] produces error bounds around the scores assigned to each private key bit in order to define a confidence interval. Based on those techniques, the Evaluator can reduce the number of remaining operations and eases the application of a full attack scenario against asymmetric cryptographic implementations.

Mitigating the desynchronization effect. As mentioned in Chapter 6, the convolutional neural networks are useful to defeat hiding countermeasures. However, through our investigations, we observe that dealing with such countermeasures make the design of a Θ-parametric function F_Θ more complex. As the model hyperparameters space increases with the complexity of the neural network architecture, the *elapsed time* criterion can be impacted during the evaluation of a targeted cryptographic module. Furthermore, in [ZS19], Zhou and Standaert suggest that the Evaluator needs to use alignment methods in order to preprocess the leakage traces and ease the exploitation of security flaws. The results obtained in Chapter 6 are consistent with this observation.

Perspective 9.2.4 (Automate the synchronization process)**.** *Finding solutions that automate the alignment process in order to ease the construction of the neural network architecture and optimizing the elapsed time criterion.*

A solution, proposed in [SWES+19], consists in automatically learn a non-linear time warping that can be applied on unseen leakage traces. Even if the Evaluator can question the benefits of this approach to prevent or mitigate the hiding countermeasures classically implemented by the Developer without any loss of information, using Deep Learning techniques on synchronized leakage traces ease non-negligibly optimization process.

[c]Here, the score is characterized by the probability provided by the parametric model or the ensemble model.

Chapter 10

Publications & Communications

10.1 Journal articles

1. [ZBHV19b] 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.

2. [ZBD+20b] Gabriel Zaid, Lilian Bossuet, François Dassance, Amaury Habrard, and Alexandre Venelli. Ranking loss: Maximizing the success rate in deep learning side-channel analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2021(1):25–55, Dec. 2020.

3. [ZBHV21b] Gabriel Zaid, Lilian Bossuet, Amaury Habrard, and Alexandre Venelli. Efficiency through diversity in ensemble models applied to side-channel attacks – a case study on public-key algorithms –. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2021.

10.2 International Conferences Proceedings

1. [RZC+21] Damien Robissout, Gabriel Zaid, Brice Colombier, Lilian Bossuet, and Amaury Habrard. Online performance evaluation of deep learning networks for profiled side-channel analysis. In Guido Marco Bertoni and Francesco Regazzoni, editors, *Constructive Side-Channel Analysis and Secure Design*, pages 200–218, Cham, 2021. Springer International Publishing.

10.3 Pre-prints

1. [ZBHV20] Gabriel Zaid, Lilian Bossuet, Amaury Habrard, and Alexandre Venelli. Understanding methodology for efficient cnn architectures in profiling attacks. Cryptology ePrint Archive, Report 2020/757, 2020.

2. [ZBC+21b] Gabriel Zaid, Lilian Bossuet, Mathieu Carbone, Amaury Habrard, and Alexandre Venelli. Conditional variational autoencoder based on stochastic models. Cryptology ePrint Archive, Report 2021/???, 2021 [**Under review at Eurocrypt 2022**].

10.4 Source Codes

1. [ZBHV19a] Gabriel Zaid, Lilian Bossuet, Amaury Habrard, and Alexandre Venelli. Methodology for efficient cnn architectures in profiling attacks. `https://github.com/gabzai/`

Methodology-for-efficient-CNN-architectures-in-SCA, 2019.

2. [ZBD+20a] Gabriel Zaid, Lilian Bossuet, François Dassance, Amaury Habrard, and Alexandre Venelli. Ranking loss: Maximizing the success rate in deep learning side-channel analysis. https://github.com/gabzai/Ranking-Loss-SCA, 2020.

3. [ZBHV21a] Gabriel Zaid, Lilian Bossuet, Amaury Habrard, and Alexandre Venelli. Efficiency through diversity in ensemble models applied to side-channel attacks – a case study on public-key algorithms –. https://github.com/gabzai/Ensembling-Loss-SCA, 2021.

4. [ZBC+21a] Gabriel Zaid, Lilian Bossuet, Mathieu Carbone, Amaury Habrard, and Alexandre Venelli. Conditional variational autoencoder based on stochastic models. https://github.com/gabzai/cVAE-ST, 2021. [Unpublished].

10.5 Communications

1. [RZC+19] Damien Robissout, Gabriel Zaid, Brice Colombier, Vincent Grosso, Lilian Bossuet, et al. Improved Deep-Learning Side-Channel Attacks using Normalization layers. *Workshop on Randomness and Arithmetics for Cryptography on Hardware (WRAC'H)*, Apr 2019, Roscoff, France.

2. [ZBHV19c] Gabriel Zaid, Lilian Bossuet, Amaury Habrard and Alexandre Venelli. Methodology for efficient cnn architectures in profiling attacks. *Workshop on Practical Hardware Innovations in Security Implementation and Characterization (PHISIC)*, Oct 2019, Gardanne, France.

3. [Zai20a] Gabriel Zaid. Application des techniques de deep learning dans la réalisation d'attaques par canaux auxiliaires. *Journée "Nouvelles Avancées en Sécurité des Systèmes d'Information"*, Jan 2020, Toulouse, France.

4. [CRZ+20] Brice Colombier, Damien Robissout, Gabriel Zaid, Lilian Bossuet, Amaury Habrard. Apprentissage profond pour les attaques par analyse de canaux auxiliaires des implémentations de fonctions cryptographiques. *Ecole d'hiver Francophone sur les Technologies de Conception des Systèmes embarqués Hétérogènes, FETCH,* Feb 2020, Montréal, Canada.

5. [Zai20b] Gabriel Zaid. Ranking loss: Maximizing the success rate in deep learning side-channel analysis. *Journée Machine Learning & SCA GDR SoC2 et Sécurité Informatique*, Dec 2020, Virtual.

6. [ZBHV21c] Gabriel Zaid, Lilian Bossuet, Amaury Habrard and Alexandre Venelli, Methodology for efficient cnn architectures in profiling attacks, *Journée de la recherche de l'École doctorale EDSIS*, June 2021, Virtual.

7. [ZR21] Gabriel Zaid and Damien Robissout, Using deep learning against threats by side-channel analysis, *Summer School - ARQUS European University Alliance*, September 2021, Virtual.

Chapter 11

Bibliography

Bibliography

[Abd21] Karim Abdellatif. Mixup data augmentation for deep learning side-channel attacks. Cryptology ePrint Archive, Report 2021/328, 2021. `https://eprint.iacr.org/2021/328`.

[AFSW13] Arnold Abromeit, Dirk Feldhusen, Lex Schoonen, and Guntram Wicke. Minimum requirements for evaluating side-channel attack resistance of rsa, dsa and diffie-hellman key exchange implementations. Technical report, Bundesamt für Sicherheit in der Informationstechnik, November 2013.

[ANS05] Ansi x9.62: Public key cryptography for the financial services industry: The elliptic curve digital signature algorithm (ecdsa). Technical report, November 2005.

[ANS14] ANSSI. Le référentiel général de sécurité. Technical report, ANSSI, June 2014.

[APSQ06] Cédric Archambeau, Eric Peeters, François-Xavier Standaert, and Jean-Jacques Quisquater. Template attacks in principal subspaces. In Louis Goubin and Mitsuru Matsui, editors, *Cryptographic Hardware and Embedded Systems - CHES 2006*, pages 1–14, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[AZLS19] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via over-parameterization. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 242–252. PMLR, 09–15 Jun 2019.

[BBD+14] Shivam Bhasin, Nicolas Bruneau, Jean-Luc Danger, Sylvain Guilley, and Zakaria Najm. Analysis and improvements of the dpa contest v4 implementation. In Rajat Subhra Chakraborty, Vashek Matyas, and Patrick Schaumont, editors, *Security, Privacy, and Applied Cryptography Engineering*, pages 201–218, Cham, 2014. Springer International Publishing.

[BBL04] Olivier Bousquet, Stéphane Boucheron, and Gábor Lugosi. *Introduction to Statistical Learning Theory*, pages 169–207. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[BBM+15] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLOS ONE*, 10(7):1–46, 07 2015.

[BCD+13] George Becker, Jeremy Cooper, Elke DeMulder, Gilbert Goodwill, Joshua Jaffe, Gary Kenworthy, Tim Kouzminov, Andrew Leiserson, Mark Marson, Pankaj Rohatgi, and Sami Saab. Test vector leakage assessment (tvla) methodology in practice (extended abstract). 2013.

[BCH+20] Shivam Bhasin, Anupam Chattopadhyay, Annelie Heuser, Dirmanto Jap, Stjepan Picek, and Ritu Ranjan. Mind the portability: a warriors guide through realistic profiled side-channel analysis. In *NDSS 2020*, 2020.

[BCO03] Eric Brier, Christophe Clavier, and Francis Olivier. Optimal statistical power analysis. Cryptology ePrint Archive, Report 2003/152, 2003. `https://eprint.iacr.org/2003/152`.

[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*, pages 16–29, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[BDL97] Dan Boneh, Richard DeMillo, and Richard Lipton. On the importance of checking cryptographic protocols for faults. In Walter Fumy, editor, *Advances in Cryptology — EUROCRYPT '97*, pages 37–51, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.

[BECN+06] Hagai Bar-El, Hamid Choukri, David Naccache, Michael Tunstall, and Claire Whelan. The sorcerer's apprentice guide to fault attacks. *Proceedings of the IEEE*, 94(2):370–382, 2006.

[Ber06] Daniel Bernstein. Curve25519: New diffie-hellman speed records. In Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin, editors, *Public Key Cryptography - PKC 2006*, pages 207–228, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[BFOS84] Leo Breiman, Jerome Friedman, Richard Olshen, and Charles Stone. *Classification and Regression Trees*. Wadsworth, 1984.

[BGH+15] Nicolas Bruneau, Sylvain Guilley, Annelie Heuser, Damien Marion, and Olivier Rioul. Less is more. In Tim Güneysu and Helena Handschuh, editors, *Cryptographic Hardware and Embedded Systems – CHES 2015*, pages 22–41, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[BGV92] Bernhard Boser, Isabelle Guyon, and Vladimir Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, pages 144–152. ACM Press, 1992.

[BHM+19] Olivier Bronchain, Julien 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*, pages 713–737, Cham, 2019. Springer International Publishing.

[BJPW13] Aurélie Bauer, Eliane Jaulmes, Emmanuel Prouff, and Justine Wild. Horizontal and vertical side-channel attacks against secure rsa implementations. In Ed Dawson, editor, *Topics in Cryptology – CT-RSA 2013*, pages 1–17, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[BPK92] Horst Bischof, Axel Pinz, and Walter Kropatsch. Visualization methods for neural networks. In *Proceedings., 11th IAPR International Conference on Pattern Recognition. Vol.II. Conference B: Pattern Recognition Methodology and Systems*, pages 581–585, Aug 1992.

[BPS+20] 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*, 10(2):163–188, 2020.

[Bre96] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, August 1996.

[Bre01] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, October 2001.

[BRL07] Christopher Burges, Robert Ragno, and Quoc Le. Learning to rank with nonsmooth cost functions. In Bernhard Schölkopf, John Platt, and Thomas Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 193–200. MIT Press, 2007.

[Bro09] Gavin Brown. An information theoretic perspective on multiple classifier systems. In Jón Atli Benediktsson, Josef Kittler, and Fabio Roli, editors, *Multiple Classifier Systems*, pages 344–353, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[BS90] Eli Biham and Adi Shamir. Differential cryptanalysis of des-like cryptosystems. In *Proceedings of the 10th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '90, page 2–21, Berlin, Heidelberg, 1990. Springer-Verlag.

[BSR+05] Christopher Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In *Proceedings of the 22Nd International Conference on Machine Learning*, ICML '05, pages 89–96, New York, NY, USA, 2005. ACM.

[BT00] Dimitri Bertsekas and John Tsitsiklis. Gradient convergence in gradient methods with errors. *SIAM Journal on Optimization*, 10(3):627–642, January 2000.

[Bur10] Christopher Burges. From ranknet to lambdarank to lambdamart: An overview. Technical Report MSR-TR-2010-82, June 2010.

[BZBN19] Sebastian Bruch, Masrour Zoghi, Michael Bendersky, and Marc Najork. Revisiting approximate metric optimization in the age of deep neural networks. In *Proceedings of the 42Nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR'19, pages 1241–1244, New York, NY, USA, 2019. ACM.

[C+47] Augustin Cauchy et al. Méthode générale pour la résolution des systemes d'équations simultanées. *Comptes Rendus Hebdomadaire Seances Académie des Sciences de Paris*, 25(1847):536–538, 1847.

[Cag18] Eleonora Cagli. *Feature Extraction for Side-Channel Attacks*. Theses, Sorbonne Université, December 2018.

[Car10] Claude Carlet. *Boolean Functions for Cryptography and Error-Correcting Codes*, page 257–397. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 2010.

[CC00] Pádraig Cunningham and John Carney. Diversity versus quality in classification ensembles based on feature selection. In Ramon López de Mántaras and Enric Plaza, editors, *Machine Learning: ECML 2000*, pages 109–116, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

[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.

[CCD00] Christophe Clavier, Jean-Sébastien Coron, and Nora Dabbous. Differential power analysis in the presence of hardware countermeasures. In Çetin Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems — CHES 2000*, pages 252–263, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

[CDP16] 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*, pages 15–33, Cham, 2016. Springer International Publishing.

[CDP17a] 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.

[CDP17b] 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*, pages 1–22, Cham, 2017. Springer International Publishing.

[CFG+10] Christophe Clavier, Benoit Feix, Georges Gagnerot, Mylène Roussellet, and Vincent Verneuil. Horizontal correlation analysis on exponentiation. In Miguel Soriano, Sihan Qing, and Javier López, editors, *Information and Communications Security*, pages 46–61, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[CG16] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 785–794, New York, NY, USA, 2016. Association for Computing Machinery.

[CGD21] Maxime Cauchois, Suyash Gupta, and John Duchi. Knowing what you know: valid and validated confidence sets in multiclass and multilabel prediction. *Journal of Machine Learning Research*, 22(81):1–42, 2021.

[CH67] Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.

[Che18] Eloi de Cherisey. *Towards a better formalisation of the side-channel threat.* Theses, Université Paris-Saclay, December 2018.

[Chm20] Lukasz Chmielewski. Reassure (h2020 731591) ecc dataset, January 2020. Contact: chmielewski@riscure.com.

[Cho15] François Chollet. Keras. `https://keras.io`, 2015.

[CHX+19] Fatih Cakir, Kun He, Xide Xia, Brian Kulis, and Stan Sclaroff. Deep metric learning to rank. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[CJ01] Christophe Clavier and Marc Joye. Universal exponentiation algorithm a first step towards provable spa-resistance. In Çetin Koç, David Naccache, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems — CHES 2001*, pages 300–308, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[CJRR99] Suresh Chari, Charanjit Jutla, Josyula Rao, and Pankaj Rohatgi. Towards sound approaches to counteract power-analysis attacks. In Michael Wiener, editor, *Advances in Cryptology — CRYPTO' 99*, pages 398–412, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[CK09] Jean-Sébastien Coron and Ilya Kizhvatov. An efficient method for random delay generation in embedded software. In Christophe Clavier and Kris Gaj, editors, *Cryptographic Hardware and Embedded Systems - CHES 2009*, pages 156–170, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[CK14] Omar Choudary and Markus G. Kuhn. Efficient template attacks. In Aurélien Francillon and Pankaj Rohatgi, editors, *Smart Card Research and Advanced Applications*, pages 253–270, Cham, 2014. Springer International Publishing.

[CK15] Marios Choudary and Markus Kuhn. Efficient stochastic methods: Profiled attacks beyond 8 bits. In Marc Joye and Amir Moradi, editors, *Smart Card Research and Advanced Applications*, pages 85–103, Cham, 2015. Springer International Publishing.

[CLL+09a] Wei Chen, Tie-Yan Liu, Yanyan Lan, Zhi-Ming Ma, and Hang Li. Ranking measures and loss functions in learning to rank. In Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 315–323. Curran Associates, Inc., 2009.

[CLL+09b] Wei Chen, Tie-Yan Liu, Yanyan Lan, Zhiming Ma, and Hang Li. Essential loss: Bridge the gap between ranking measures and loss functions in learning to rank. Technical Report MSR-TR-2009-141, October 2009.

[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*, pages 144–162, Cham, 2020. Springer International Publishing.

[CM02] Dorin Comaniciu and Peter Meer. Mean shift: a robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002.

[CMZG09] Olivier Chapelle, Donald Metlzer, Ya Zhang, and Pierre Grinspan. Expected reciprocal rank for graded relevance. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, CIKM '09, pages 621–630, New York, NY, USA, 2009. ACM.

[Coh60] Jacob Cohen. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46, 1960.

[Cor99] Jean-Sébastien Coron. Resistance against differential power analysis for elliptic curve cryptosystems. In Çetin K. Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems*, pages 292–302, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[CR17] Christophe Clavier and Léo Reynaud. Improved blind side-channel analysis by exploitation of joint distributions of leakages. In Wieland Fischer and Naofumi Homma, editors, *Cryptographic Hardware and Embedded Systems – CHES 2017*, pages 24–44, Cham, 2017. Springer International Publishing.

[Cra09] Nick Craswell. *Mean Reciprocal Rank*, pages 1703–1703. Springer US, Boston, MA, 2009.

[CRR03] Suresh Chari, Josyula Rao, and Pankaj Rohatgi. Template attacks. In *Revised Papers from the 4th International Workshop on Cryptographic Hardware and Embedded Systems*, CHES '02, pages 13–28, London, UK, UK, 2003. Springer-Verlag.

[CRZ+20] Brice Colombier, Damien Robissout, Gabriel Zaid, Lilian Bossuet, and Amaury Habrard. Apprentissage profond pour les attaques par analyse de canaux auxiliaires des implémentations de fonctions cryptographiques. In *Ecole d'hiver Francophone sur les Technologies de Conception des Systèmes embarqués Hétérogènes, FETCH*, Montréal, Canada, February 2020.

[CUH16] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

[CV95] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, September 1995.

[Cyb89] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2(4):303–314, December 1989.

[CZ06] Zhimin Chen and Yujie Zhou. Dual-rail random switching logic: A countermeasure to reduce side channel leakage. In Louis Goubin and Mitsuru Matsui, editors, *Cryptographic Hardware and Embedded Systems - CHES 2006*, pages 242–254, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[CZLG21] Pei Cao, Chi Zhang, Xiangjun Lu, and Dawu Gu. Cross-device profiled side-channel attack with unsupervised domain adaptation. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2021(4):27–56, Aug. 2021.

[dCGRP19] Eloi de Chérisey, Sylvain Guilley, Olivier Rioul, and Pablo Piantanida. Best information is most successful. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2019(2):49–79, Feb. 2019.

[DDF14] Alexandre Duc, Stefan Dziembowski, and Sebastian Faust. Unifying leakage models: From probing attacks to noisy leakage. In Phong Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, pages 423–440, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[DDFP21] Gabriel Destouet, Cécile Dumas, Anne Frassati, and Valérie Perrier. Wavelet scattering transform and ensemble methods for side-channel analysis. In Guido Marco Bertoni and Francesco Regazzoni, editors, *Constructive Side-Channel Analysis and Secure Design*, pages 71–89, Cham, 2021. Springer International Publishing.

[DDP13] Guillaume Dabosville, Julien Doget, and Emmanuel Prouff. A new second-order side channel attack based on linear regression. *IEEE Transactions on Computers*, 62(8):1629–1640, 2013.

[Des77] Des. Data encryption standard. In *In FIPS PUB 46, Federal Information Processing Standards Publication*, pages 46–2, 1977.

[DFS15] Alexandre Duc, Sebastian Faust, and François-Xavier Standaert. Making masking security proofs concrete. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015*, pages 401–429, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[DFS18] 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. *Journal of Cryptology*, 32(4):1263–1297, January 2018.

[DGH+13] Jean-Luc Danger, Sylvain Guilley, Philippe Hoogvorst, Cédric Murdica, and David Naccache. A synthesis of side-channel attacks on elliptic curve cryptography in smart-cards. *Journal of Cryptographic Engineering*, 3(4):241–265, October 2013.

[DH76] Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[DHS11] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(61):2121–2159, 2011.

[Die00a] Thomas Dietterich. Ensemble methods in machine learning. In *Multiple Classifier Systems*, pages 1–15, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

[Die00b] Thomas Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 40, 12 2000.

[DPG+14] Yann N Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In Zoubin Ghahramani, Max Welling, Corinna Cortes, Neil Lawrence, and Kilian Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[DPRS11] Julien Doget, Emmanuel Prouff, Matthieu Rivain, and François-Xavier Standaert. Univariate side channel attacks and leakage modeling. *Journal of Cryptographic Engineering*, 1(2):123–144, August 2011.

[DR02] Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Information Security and Cryptography. Springer, 2002.

[DS16] François Durvaux and François-Xavier Standaert. From improved leakage detection to the detection of points of interests in leakage traces. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016*, pages 240–262, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[DSVC14] François Durvaux, François-Xavier Standaert, and Nicolas Veyrat-Charvillon. How to certify the leakage of a chip? In Phong Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, pages 459–476, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[DY14] Li Deng and Dong Yu. Deep learning: Methods and applications. Technical Report MSR-TR-2014-21, Microsoft, May 2014.

[DZD+18] Adam Ding, Liwei Zhang, Francois Durvaux, François-Xavier Standaert, and Yunsi Fei. Towards sound and optimal leakage detection procedure. In Thomas Eisenbarth and Yannick Teglia, editors, *Smart Card Research and Advanced Applications*, pages 105–122, Cham, 2018. Springer International Publishing.

[ES16] Ronen Eldan and Ohad Shamir. The power of depth for feedforward 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 907–940, Columbia University, New York, New York, USA, 23–26 Jun 2016. PMLR.

[Fan61] Robert Fano. Transmission of information: A statistical theory of communications. *American Journal of Physics*, 29(11):793–794, 1961.

[FGG97] Nir Friedman, Dan Geiger, and Moises Goldszmidt. *Machine Learning*, 29(2/3):131–163, 1997.

[FGI+16] Dirk Feldhusen, Max Gebhardt, Georg Illies, Michael Kasper, Manfred Lochter, Richard Petri, Oliver Stein, Wolfgang Thumser, and Guntram Wicke. Minimum requirements for evaluating side-channel attack resistance of elliptic curve implementations. Technical report, Bundesamt für Sicherheit in der Informationstechnik, November 2016.

[FHL19] Stanislav Fort, Huiyi Hu, and Balaji Lakshminarayanan. Deep ensembles: A loss landscape perspective. In *Bayesian Deep Learning workshop, NIPS*, 2019.

[FIP01] Security requirements for cryptographic modules. Technical report, May 2001.

[FIP09] Security requirements for cryptographic modules (revised draft). Technical report, April 2009.

[FLP03] Joseph Fleiss, Bruce Levin, and Myunghee Cho Paik. *Statistical methods for rates and proportions; 3rd ed.* Wiley Series in Probability and Statistics. Wiley, Hoboken, NJ, 2003.

[FMPR11] Guillaume Fumaroli, Ange Martinelli, Emmanuel Prouff, and Matthieu Rivain. Affine masking against higher-order side channel analysis. In Alex Biryukov, Guang Gong, and Douglas Stinson, editors, *Selected Areas in Cryptography*, pages 262–280, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[FR05] Giorgio Fumera and Fabio Roli. A theoretical and experimental analysis of linear combiners for multiple classifier systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(6):942–956, 2005.

[FS96] Yoav Freund and Robert Schapire. Experiments with a new boosting algorithm. In *Proceedings of the Thirteenth International Conference on International Conference on Machine Learning*, ICML'96, page 148–156, San Francisco, CA, USA, 1996. Morgan Kaufmann Publishers Inc.

[FV03] Pierre-Alain Fouque and Frederic Valette. The doubling attack – why upwards is better than downwards. In Colin Walter, Çetin Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2003*, pages 269–280, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[FV12] Junfeng Fan and Ingrid Verbauwhede. *An Updated Survey on Secure ECC Implementations: Attacks, Countermeasures and Cost*, pages 265–282. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. Adaptive computation and machine learning. MIT Press, 2016.

[GBTP08] Benedikt Gierlichs, Lejla Batina, Pim Tuyls, and Bart Preneel. Mutual information analysis. In Elisabeth Oswald and Pankaj Rohatgi, editors, *Cryptographic Hardware and Embedded Systems – CHES 2008*, pages 426–442, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[GHMR17] Sylvain Guilley, Annelie Heuser, Tang Ming, and Olivier Rioul. Stochastic side-channel leakage analysis via orthonormal decomposition. In Pooya Farshim and Emil Simion, editors, *Innovative Security Solutions for Information Technology and Communications*, pages 12–27, Cham, 2017. Springer International Publishing.

[GJJR11] Gilbert Goodwill, Benjamin Jun, Josh Jaffe, and Pankaj Rohatgi. A testing methodology for side-channel resistance validation, niat, 2011.

[GJS20] Aron Gohr, Sven Jacob, and Werner Schindler. Subsampling and knowledge distillation on adversarial examples: New techniques for deep learning based side channel evaluations. Cryptology ePrint Archive, Report 2020/165, 2020. `https://eprint.iacr.org/2020/165`.

[GKLD11] Sylvain Guilley, Karim Khalfallah, Victor Lomne, and Jean-Luc Danger. Formal framework for the evaluation of waveform resynchronization algorithms. In Claudio Ardagna and Jianying Zhou, editors, *Information Security Theory and Practice. Security and Privacy of Mobile Devices in Wireless Communication*, pages 100–115, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[GMGA17] Anil Goyal, Emilie Morvant, Pascal Germain, and Massih-Reza Amini. Pac-bayesian analysis for a two-step hierarchical multiview learning approach. In Michelangelo Ceci, Jaakko Hollmén, Ljupčo Todorovski, Celine Vens, and Sašo Džeroski, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 205–221, Cham, 2017. Springer International Publishing.

[GMGH19] Christophe Genevey-Metat, Benoît Gérard, and Annelie Heuser. Combining sources of side-channel information. In *CESAR 2019*, Rennes, France, November 2019.

[GMGH20] Christophe Genevey-Metat, Benoît Gérard, and Annelie Heuser. On what to learn: Train or adapt a deeply learned profile? Cryptology ePrint Archive, Report 2020/952, 2020. `https://eprint.iacr.org/2020/952`.

[GMO01] Karine Gandolfi, Christophe Mourtel, and Francis Olivier. Electromagnetic analysis: Concrete results. In Çetin Koç, David Naccache, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems — CHES 2001*, pages 251–261, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[Goo17] Ian Goodfellow. NIPS 2016 tutorial: Generative adversarial networks. *CoRR*, abs/1701.00160, 2017.

[GP99] Louis Goubin and Jacques Patarin. Des and differential power analysis the "duplication" method. In Çetin Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems*, pages 158–172, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[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 Zoubin Ghahramani, Max Welling, Corinna Cortes, Neil Lawrence, and Kilian Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[GR01] Giorgio Giacinto and Fabio Roli. Design of effective neural network ensembles for image classification purposes. *Image and Vision Computing*, 19:699–707, 08 2001.

[GST16] Daniel Genkin, Adi Shamir, and Eran Tromer. Acoustic cryptanalysis. *Journal of Cryptology*, 30(2):392–443, February 2016.

[GT03] Jovan Golić and Christophe Tymen. Multiplicative masking and power analysis of aes. In Burton Kaliski, çetin Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002*, pages 198–212, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[GVS15] Ian Goodfellow, Oriol Vinyals, and Andrew Saxe. Qualitatively characterizing neural network optimization problems. In *International Conference on Learning Representations*, 2015.

[HFL+20] Benjamin Hettwer, Daniel Fennes, Sebastien Leger, Jan Richter-Brockmann, Stefan Gehrer, and Tim Güneysu. Deep learning multi-channel fusion attack against side-channel protected hardware. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6, 2020.

[HG16] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

[HGG19] Benjamin Hettwer, Stefan Gehrer, and Tim Güneysu. Profiled power analysis attacks using convolutional neural networks with domain knowledge. In Carlos Cid and Michael J. Jacobson Jr., editors, *Selected Areas in Cryptography – SAC 2018*, pages 479–498, Cham, 2019. Springer International Publishing.

[HGG20] Benjamin Hettwer, Stefan Gehrer, and Tim Güneysu. Deep neural network attribution methods for leakage analysis and symmetric key recovery. pages 645–666, 2020.

[HHGG20] Benjamin Hettwer, Tobias Horn, Stefan Gehrer, and Tim Güneysu. Encoding power traces as images for efficient side-channel analysis. In *2020 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 46–56. IEEE, 2020.

[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.

[HHS17] Elad Hoffer, Itay Hubara, and Daniel Soudry. Train longer, generalize better: Closing the generalization gap in large batch training of neural networks. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 1729–1739, Red Hook, NY, USA, 2017. Curran Associates Inc.

[HMP+17] Irina Higgins, Loïc Matthey, Arka Pal, Christopher Burges, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[HNI+06] Naofumi Homma, Sei Nagashima, Yuichi Imai, Takafumi Aoki, and Akashi Satoh. High-resolution side-channel attack using phase-based waveform matching. In Louis Goubin and Mitsuru Matsui, editors, *Cryptographic Hardware and Embedded Systems - CHES 2006*, pages 187–200, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[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*, pages 239–252, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[Hor91] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257, 1991.

[HOT06] Geoffrey Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, July 2006.

[HOWT06] Geoffrey Hinton, Simon Osindero, Max Welling, and Yee-Whye Teh. Unsupervised discovery of nonlinear structure using contrastive backpropagation. *Cognitive science*, 30:725–31, 07 2006.

[HR70] Martin Hellman and Josef Raviv. Probability of error, equivocation, and the chernoff bound. *IEEE Transactions on Information Theory*, 16(4):368–372, 1970.

[HRG14] Annelie Heuser, Olivier Rioul, and Sylvain Guilley. Good is not good enough. In Lejla Batina and Matthew Robshaw, editors, *Cryptographic Hardware and Embedded Systems – CHES 2014*, pages 55–74, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[HS90] Lars Kai Hansen and Peter Salamon. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10):993–1001, October 1990.

[HSAM21] Suvadeep Hajra, Sayandeep Saha, Manaar Alam, and Debdeep Mukhopadhyay. Transnet: Shift invariant transformer network for power attack. Cryptology ePrint Archive, Report 2021/827, 2021. `https://eprint.iacr.org/2021/827`.

[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)*, pages 770–778, 2016.

[IFLF+20] Hassan Ismail Fawaz, Benjamin Lucas, Germain Forestier, Charlotte Pelletier, Daniel Schmidt, Jonathan Weber, Geoffrey Webb, Lhassane Idoumghar, Pierre-Alain Muller, and François Petitjean. InceptionTime: Finding AlexNet for time series classification. *Data Mining and Knowledge Discovery*, 34(6):1936–1962, September 2020.

[ISO] ISO15408. Information technology — security techniques — evaluation criteria for it security. *Eurocrypt2004 Rump Session, May*.

[ISUH21] Akira Ito, Kotaro Saito, Rei Ueno, and Naofumi Homma. Imbalanced data problems in deep learning-based side-channel attacks: Analysis and solution. *IEEE Transactions on Information Forensics and Security*, 16:3790–3802, 2021.

[ISW03] Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003*, pages 463–481, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[IW18] Ehsan Imani and Martha White. Improving regression performance with distributional losses. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2157–2166. PMLR, 10–15 Jul 2018.

[JK00] Kalervo Järvelin and Jaana Kekäläinen. Ir evaluation methods for retrieving highly relevant documents. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '00, pages 41–48, New York, NY, USA, 2000. ACM.

[JK02] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems*, 20(4):422–446, October 2002.

[Jol86] Ian Jolliffe. *Principal Component Analysis*. Springer Verlag, 1986.

[JPS05] Marc Joye, Pascal Paillier, and Berry Schoenmakers. On second-order differential power analysis. In Josyula Rao and Berk Sunar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2005*, pages 293–308, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[JZHY20] Minhui Jin, Mengce Zheng, Honggang Hu, and Nenghai Yu. An enhanced convolutional neural network in side-channel attacks and its visualization. *CoRR*, abs/2009.08898, 2020.

[KA14] Justin Kinney and Gurinder Atwal. Equitability, mutual information, and the maximal information coefficient. *Proceedings of the National Academy of Sciences*, 111(9):3354–3359, 2014.

[KB15] Diederik 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.

[Ker83] Auguste Kerckhoffs. La cryptographie militaire. *Journal des Sciences Militaires*, pages 161–191, 1883.

[KFJP21] Karlo Knezevic, Juraj Fulir, Domagoj Jakobovic, and Stjepan Picek. Neurosca: Evolving activation functions for side-channel analysis. Cryptology ePrint Archive, Report 2021/249, 2021. `https://eprint.iacr.org/2021/249`.

[KH98] Tin Kam Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844, 1998.

[Kim14] Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar, October 2014. Association for Computational Linguistics.

[KJJ99] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael Wiener, editor, *Advances in Cryptology — CRYPTO' 99*, pages 388–397, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[KK99] Oliver Kömmerling and Markus Kuhn. Design principles for tamper-resistant smartcard processors. In *Proceedings of the USENIX Workshop on Smartcard Technology on USENIX Workshop on Smartcard Technology*, WOST'99, page 2, USA, 1999. USENIX Association.

[KKH21] Donggeun Kwon, Heeseok Kim, and Seokhie Hong. Non-profiled deep learning-based side-channel preprocessing with autoencoders. *IEEE Access*, 9:57692–57703, 2021.

[KL51] Salomo Kullback and Richard Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 03 1951.

[Kle17] Matthew Kleinsmith. CNNs from different viewpoints - prerequisite : Basic neural networks, 2017.

[KLM19] Ananya Kumar, Percy Liang, and Tengyu Ma. Verified uncertainty calibration. In Hanna Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché Buc, Emily Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[Kob87] Neal Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48(177):203–209, January 1987.

[Koc96] Paul Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In Neal Koblitz, editor, *Advances in Cryptology — CRYPTO '96*, pages 104–113, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg.

[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.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. Imagenet classification with deep convolutional neural networks. In Fernando Pereira, Christopher Burges, Léon Bottou, and Kilian Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[KSS10] Michael Kasper, Werner Schindler, and Marc Stöttinger. A stochastic method for security evaluation of cryptographic fpga implementations. In *2010 International Conference on Field-Programmable Technology*, pages 146–153, 2010.

[KUMH17] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 971–980. Curran Associates, Inc., 2017.

[Kun04] Ludmila Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms*. Wiley-Interscience, USA, 2004.

[KW96] Ron Kohavi and David Wolpert. Bias plus variance decomposition for zero-one loss functions. In *Proceedings of the Thirteenth International Conference on International Conference on Machine Learning*, ICML'96, page 275–283, San Francisco, CA, USA, 1996. Morgan Kaufmann Publishers Inc.

[KW03] Ludmila Kuncheva and Christopher Whitaker. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine Learning*, 51(2):181–207, May 2003.

[KW14] Diederik Kingma and Max Welling. Auto-encoding variational bayes. 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.

[KW19] Diederik Kingma and Max Welling. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, 2019.

[KWKT15] Tejas Kulkarni, William Whitney, Pushmeet Kohli, and Josh Tenenbaum. Deep convolutional inverse graphics network. In Corinna Cortes, Neil Lawrence, Daniel Lee, Masashi Sugiyama, and Roman Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[Lam00] Louisa Lam. Classifier combinations: Implementations and theoretical issues. In *Multiple Classifier Systems*, pages 77–86, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

[Las08] Julia Lasserre. *Hybrids of generative and discriminative methods for machine learning*. PhD thesis, Cambridge University, England, 2008.

[LBBH98] Yann Lecun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[LBD+89] Yann LeCun, Bernhard Boser, John Denker, Donnie Henderson, Richard Howard, Wayne Hubbard, and Lawrence Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

[LBLL09] Hugo Larochelle, Yoshua Bengio, Jérôme Louradour, and Pascal Lamblin. Exploring strategies for training deep neural networks. *Journal of Machine Learning Research*, 10:1–40, June 2009.

[LBOM12] Yann LeCun, Léon Bottou, Genevieve Orr, and Klaus-Robert Müller. *Efficient BackProp*, pages 9–48. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[LDLL14] Yanis Linge, Cécile Dumas, and Sophie Lambert-Lacroix. Using the joint distributions of a cryptographic function in side channel analysis. In Emmanuel Prouff, editor, *Constructive Side-Channel Analysis and Secure Design*, pages 199–213, Cham, 2014. Springer International Publishing.

[LH20] JongHyeok Lee and Dong-Guk Han. Dlddo: Deep learning to detect dummy operations. In Ilsun You, editor, *Information Security Applications*, pages 73–85, Cham, 2020. Springer International Publishing.

[Li11a] Hang Li. *Learning to Rank for Information Retrieval and Natural Language Processing*. Morgan & Claypool Publishers, 2011.

[Li11b] Hang Li. A short introduction to learning to rank. *IEICE Transactions on Information and Systems*, E94.D(10):1854–1862, 2011.

[Liu09] Tie-Yan Liu. Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval*, 3(3):225–331, March 2009.

[LKP20] Huimin Li, Marina Krček, and Guilherme Perin. A comparison of weight initializers in deep learning-based side-channel analysis. 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*, pages 126–143, Cham, 2020. Springer International Publishing.

[Llo82] Stuart Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[LLUZ16] Wenjie Luo, Yujia Li, Raquel Urtasun, and Richard Zemel. Understanding the effective receptive field in deep convolutional neural networks. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[LLY+20] Guanlin Li, Chang Liu, Han Yu, Yanhong Fan, Libang Zhang, Zongyue Wang, and Meiqin Wang. Scnet: A neural network for automated side-channel attack. *CoRR*, abs/2008.00476, 2020.

[LSJR16] Jason Lee, Max Simchowitz, Michael 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.

[LWC+19] Ling Liu, Wenqi Wei, Ka-Ho Chow, Margaret Loper, Emre Gursoy, Stacey Truex, and Yanzhao Wu. Deep neural network ensembles against deception: Ensemble diversity, accuracy and robustness. In *2019 IEEE 16th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, pages 274–282, 2019.

[LZC+21] Xiangjun Lu, Chi Zhang, Pei Cao, Dawu Gu, and Haining Lu. Pay attention to raw traces: A deep learning architecture for end-to-end profiling attacks. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2021(3):235–274, Jul. 2021.

[LZW+21] Zhimin Luo, Mengce Zheng, Ping Wang, Minhui Jin, Jiajia Zhang, and Honggang Hu. Towards strengthening deep learning-based side channel attacks with mixup. Cryptology ePrint Archive, Report 2021/312, 2021. `https://eprint.iacr.org/2021/312`.

[Mag19a] Houssem Maghrebi. Assessment of common side channel countermeasures with respect to deep learning based profiled attacks. In *2019 31st International Conference on Microelectronics (ICM)*, pages 126–129, 2019.

[Mag19b] Houssem Maghrebi. Deep learning based side channel attacks in practice. Cryptology ePrint Archive, Report 2019/578, 2019. `https://eprint.iacr.org/2019/578`.

[Mag20] Houssem Maghrebi. Deep learning based side-channel attack: a new profiling methodology based on multi-label classification. Cryptology ePrint Archive, Report 2020/436, 2020. `https://eprint.iacr.org/2020/436`.

[MAM+03] Simon Moore, Ross Anderson, Robert Mullins, George Taylor, and Jacques Fournier. Balanced self-checking asynchronous logic for smart card applications. *Journal of Microprocessors and Microsystems*, 27:421–430, 2003.

[Mas20] Loïc Masure. *Towards a Better Comprehension of Deep Learning for Side-Channel Analysis.* PhD thesis, Sorbonne University, France, 2020.

[Mat94] Mitsuru Matsui. Linear cryptanalysis method for des cipher. In Tor Helleseth, editor, *Advances in Cryptology — EUROCRYPT '93*, pages 386–397, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg.

[MBC+20] Loïc Masure, Nicolas Belleville, Eleonora Cagli, Marie-Angela Cornélie, Damien Couroussé, Cécile Dumas, and Laurent Maingault. Deep learning side-channel analysis on large-scale traces. In Liqun Chen, Ninghui Li, Kaitai Liang, and Steve Schneider, editors, *Computer Security – ESORICS 2020*, pages 440–460, Cham, 2020. Springer International Publishing.

[MBPK21] Naila Mukhtar, Lejla Batina, Stjepan Picek, and Yinan Kong. Fake it till you make it: Data augmentation using generative adversarial networks for all the crypto you need on small devices. Cryptology ePrint Archive, Report 2021/991, 2021. `https://ia.cr/2021/991`.

[McG54] William McGill. Multivariate information transmission. *Transactions of the IRE Professional Group on Information Theory*, 4(4):93–111, 1954.

[MDP19a] Loïc Masure, Cécile Dumas, and Emmanuel Prouff. Gradient visualization for general characterization in profiling attacks. In Ilia Polian and Marc Stöttinger, editors, *Constructive Side-Channel Analysis and Secure Design*, pages 145–167, Cham, 2019. Springer International Publishing.

[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.

[MDS99] Thomas Messerges, Ezzy Dabbish, and Robert H. Sloan. Power analysis attacks of modular exponentiation in smartcards. In Çetin Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems*, pages 144–157, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[Mes00] Thomas Messerges. Using second-order power analysis to attack dpa resistant software. In Çetin Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems — CHES 2000*, pages 238–251, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

[Mes01] Thomas Messerges. Securing the aes finalists against power analysis attacks. In Gerhard Goos, Juris Hartmanis, Jan van Leeuwen, and Bruce Schneier, editors, *Fast Software Encryption*, pages 150–164, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[MHA14] Emilie Morvant, Amaury Habrard, and Stéphane Ayache. Majority vote of diverse classifiers for late fusion. In *Proceedings of the Joint IAPR International Workshop on Structural, Syntactic, and Statistical Pattern Recognition - Volume 8621*, S+SSPR 2014, page 153–162, Berlin, Heidelberg, 2014. Springer-Verlag.

[MHM14] Zdenek Martinasek, Jan Hajny, and Lukas Malina. Optimization of power analysis using neural network. In Aurélien Francillon and Pankaj Rohatgi, editors, *Smart Card Research and Advanced Applications*, pages 94–107, Cham, 2014. Springer International Publishing.

[MHN13] Andrex Maas, Awni Hannun, and Andrew Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proceedings of the International Conference on Machine Learning*, Atlanta, Georgia, 2013.

[Mil86] Victor Miller. Use of elliptic curves in cryptography. In Hugh Williams, editor, *Advances in Cryptology — CRYPTO '85 Proceedings*, pages 417–426, Berlin, Heidelberg, 1986. Springer Berlin Heidelberg.

[Mit97] Tom Mitchell. *Machine Learning*. McGraw-Hill Education, mar 1997.

[MMG20] Andrey Malinin, Bruno Mlodozeniec, and Mark Gales. Ensemble distribution distillation. In *International Conference on Learning Representations*, 2020.

[MMOS16] Daniel 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*, pages 548–572, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[MO14] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *CoRR*, abs/1411.1784, 2014.

[MOP07] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power analysis attacks - revealing the secrets of smart cards*. Springer, 2007.

[MP43] Warren McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4):115–133, December 1943.

[MPP16] Houssem Maghrebi, Thibault Portigliatti, and Emmanuel Prouff. Breaking cryptographic implementations using deep learning techniques. In Claude Carlet, 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.

[MS21] Loïc Masure and Rémi Strullu. Side channel analysis against the anssi's protected aes implementation on arm. Cryptology ePrint Archive, Report 2021/592, 2021. `https://eprint.iacr.org/2021/592`.

[MSB16] Houssem Maghrebi, Victor Servant, and Julien Bringer. There is wisdom in harnessing the strengths of your enemy: Customized encoding to thwart side-channel attacks. In Thomas Peyrin, editor, *Fast Software Encryption*, pages 223–243, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[MU49] Nicholas Metropolis and Stanislaw Ulam. The monte carlo method. *Journal of the American Statistical Association*, 44(247):335–341, 1949. PMID: 18139350.

[MWM21] Thorben Moos, Felix Wegener, and Amir Moradi. Dl-la: Deep learning leakage assessment: A modern roadmap for sca evaluations. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2021(3):552–598, Jul. 2021.

[MZ13] Zdenek Martinasek and Vaclav Zeman. Innovative method of the power analysis. *Radioengineering*, 22(2):586–594, 2013.

[NCOS17] Erick Nascimento, Łukasz Chmielewski, David Oswald, and Peter Schwabe. Attacking embedded ecc implementations through cmov side channels. In Roberto Avanzi and Howard Heys, editors, *Selected Areas in Cryptography – SAC 2016*, pages 99–119, Cham, 2017. Springer International Publishing.

[NJ02] Andrew Ng and Michael Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In Thomas Dietterich, Suzanna Becker, and Zoubin Ghahramani, editors, *Advances in Neural Information Processing Systems*, volume 14. MIT Press, 2002.

[NLB+19] Behnam Neyshabur, Zhiyuan Li, Srinadh Bhojanapalli, Yann LeCun, and Nathan Srebro. The role of over-parametrization in generalization of neural networks. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[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*, pages 243–260, Cham, 2014. Springer International Publishing.

[OH08] Simon Osindero and Geoffrey Hinton. Modeling image patches with a directed hierarchy of markov random fields. In J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 1121–1128. Curran Associates, Inc., 2008.

[OM06] Elisabeth Oswald and Stefan Mangard. Template attacks on masking—resistance is futile. In Masayuki Abe, editor, *Topics in Cryptology – CT-RSA 2007*, pages 243–256, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[ON15] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks. *CoRR*, abs/1511.08458, 2015.

[PBP20] Guilherme Perin, Ileana Buhan, and Stjepan Picek. Learning when to stop: a mutual information approach to fight overfitting in profiled side-channel analysis. Cryptology ePrint Archive, Report 2020/058, 2020. `https://eprint.iacr.org/2020/058`.

[PCBP20] Guilherme Perin, Łukasz Chmielewski, Lejla Batina, and Stjepan Picek. Keep it unsupervised: Horizontal attacks meet deep learning. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2021(1):343–372, Dec. 2020.

[PCP20] Guilherme Perin, Łukasz Chmielewski, and Stjepan Picek. Strength in numbers: Improving generalization with ensembles in machine learning-based profiled side-channel analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020(4):337–364, Aug. 2020.

[PGH18] Karlson Pfannschmidt, Pritha Gupta, and Eyke Hüllermeier. Deep architectures for learning context-dependent ranking functions. *CoRR*, abs/1803.05796, 2018.

[PHG19] Stjepan Picek, Annelie Heuser, and Sylvain Guilley. Profiling side-channel analysis in the restricted attacker framework. Cryptology ePrint Archive, Report 2019/168, 2019. `https://eprint.iacr.org/2019/168`.

[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.

[Pin99] Allan Pinkus. Approximation theory of the mlp model in neural networks. *Acta Numerica*, 8:143–195, 1999.

[PP10] Christof Paar and Jan Pelzl. *Understanding Cryptography*. Springer Berlin Heidelberg, 2010.

[PP21] Guilherme Perin and Stjepan Picek. On the influence of optimizers in deep learning-based side-channel analysis. In Orr Dunkelman, Michael Jacobson, and Colin O'Flynn, editors, *Selected Areas in Cryptography*, pages 615–636, Cham, 2021. Springer International Publishing.

[PR13] Emmanuel Prouff and Matthieu Rivain. Masking against side-channel attacks: A formal security proof. In Thomas Johansson and Phong Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, pages 142–159, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[PRA20] Servio Paguada, Unai Rioja, and Igor Armendariz. Controlling the deep learning-based side-channel analysis: A way to leverage from heuristics. 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*, pages 106–125, Cham, 2020. Springer International Publishing.

[PRB09] Emmanuel Prouff, Matthieu Rivain, and Régis Bevan. Statistical analysis of second order differential power analysis. *IEEE Transactions on Computers*, 58(6):799–811, 2009.

[PSG16] Romain Poussier, François-Xavier Standaert, and Vincent Grosso. Simple key enumeration (and rank estimation) using histograms: An integrated approach. In Benedikt Gierlichs and Axel Y. Poschmann, editors, *Cryptographic Hardware and Embedded Systems – CHES 2016*, pages 61–81, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[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*, pages 157–176, Cham, 2018. Springer International Publishing.

[PWP21] Guilherme Perin, Lichao Wu, and Stjepan Picek. Gambling for success: The lottery ticket hypothesis in deep learning-based sca. Cryptology ePrint Archive, Report 2021/197, 2021. `https://eprint.iacr.org/2021/197`.

[Qia99] Ning Qian. On the momentum term in gradient descent learning algorithms. *Neural Networks*, 12(1):145–151, 1999.

[QLL10] Tao Qin, Tie-Yan Liu, and Hang Li. A general approximation framework for direct optimization of information retrieval measures. *Information Retrieval*, 13(4):375–397, 2010.

[QS01] Jean-Jacques Quisquater and David Samyde. Electromagnetic analysis (ema): Measures and counter-measures for smart cards. In Isabelle Attali and Thomas Jensen, editors, *Smart Card Programming and Security*, pages 200–210, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[RBA20] Unai Rioja, Lejla Batina, and Igor Armendariz. When similarities among devices are taken for granted: Another look at portability. In Abderrahmane Nitaj and Amr Youssef, editors, *Progress in Cryptology - AFRICACRYPT 2020*, pages 337–357, Cham, 2020. Springer International Publishing.

[RDS+15] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[RGL+17] Xudie Ren, Haonan Guo, Shenghong Li, Shilin Wang, and Jianhua Li. A novel image classification method with cnn-xgboost model. In Christian Kraetzer, Yun-Qing Shi, Jana Dittmann, and Hyoung Joong Kim, editors, *Digital Forensics and Watermarking*, pages 378–390, Cham, 2017. Springer International Publishing.

[RHW86] David Rumelhart, Geoffrey Hinton, and Ronald Williams. *Learning Internal Representations by Error Propagation*, page 318–362. MIT Press, Cambridge, MA, USA, 1986.

[RMN09] Rajat Raina, Anand Madhavan, and Andrew Ng. Large-scale deep unsupervised learning using graphics processors. ICML '09, page 873–880, New York, NY, USA, 2009. Association for Computing Machinery.

[RO05] Christian Rechberger and Elisabeth Oswald. Practical template attacks. In Chae Hoon Lim and Moti Yung, editors, *Information Security Applications*, pages 440–456, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[Ros58] Frank Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.

[RSA78] Ronald Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.

[RSVC+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*, pages 109–128, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[RWPP21] Jorai Rijsdijk, Lichao Wu, Guilherme Perin, and Stjepan Picek. Reinforcement learning for hyperparameter tuning in deep learning-based side-channel analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2021(3):677–707, Jul. 2021.

[RZC+19] Damien Robissout, Gabriel Zaid, Brice Colombier, Vincent Grosso, Lilian Bossuet, and Amaury Habrard. Improved Deep-Learning Side-Channel Attacks using Normalization layers. In *Workshop on Randomness and Arithmetics for Cryptography on Hardware (WRAC'H)*, Roscoff, France, April 2019.

[RZC+21] Damien Robissout, Gabriel Zaid, Brice Colombier, Lilian Bossuet, and Amaury Habrard. Online performance evaluation of deep learning networks for profiled side-channel analysis. In Guido Marco Bertoni and Francesco Regazzoni, editors, *Constructive Side-Channel Analysis and Secure Design*, pages 200–218, Cham, 2021. Springer International Publishing.

[SA03] Sergei Skorobogatov and Ross Anderson. Optical fault induction attacks. In Burton Kaliski, çetin Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002*, pages 2–12, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[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*, pages 411–425, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[SBdMVC08] François-Xavier Standaert, Philippe Bulens, Giacomo de Meulenaer, and Nicolas Veyrat-Charvillon. Improving the rules of the dpa contest. Cryptology ePrint Archive, Report 2008/517, 2008. `https://eprint.iacr.org/2008/517`.

[Sha49] Claude Elwood Shannon. Communication theory of secrecy systems. *The Bell System Technical Journal*, 28(4):656–715, 1949.

[Shi00] Robert Shirey. Internet security glossary. Technical report, May 2000.

[SI11] Werner Schindler and Kouichi Itoh. Exponent blinding does not always lift (partial) spa resistance to higher-level security. In Javier Lopez and Gene Tsudik, editors, *Applied Cryptography and Network Security*, pages 73–90, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[SI13] SOG-IS. Application of attack potential to smartcards. Technical report, SOG-IS, January 2013.

[SI20] SOG-IS. Crypto evaluation scheme agreed cryptographic mechanisms. Technical report, SOG-IS, January 2020.

[Ska96] David Skalak. The sources of increased accuracy for two proposed boosting algorithms. In *In Proc. American Association for Arti Intelligence, AAAI-96, Integrating Multiple Learned Models Workshop*, pages 120–125, 1996.

[SLP05] Werner Schindler, Kerstin Lemke, and Christof Paar. A stochastic model for differential side channel cryptanalysis. In Josyula Rao and Berk Sunar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2005*, pages 30–46, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[SLY15] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In Corinna Cortes, Neil Lawrence, Daniel Lee, Masashi Sugiyama, and Roman Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[SM15] Tobias Schneider and Amir Moradi. Leakage assessment methodology. In Tim Güneysu and Helena Handschuh, editors, *Cryptographic Hardware and Embedded Systems – CHES 2015*, pages 495–513, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[Smi17] Leslie Smith. Cyclical learning rates for training neural networks. In *2017 IEEE Winter Conference on Applications of Computer Vision, WACV 2017, Santa Rosa, CA, USA, March 24-31, 2017*, pages 464–472. IEEE Computer Society, 2017.

[Smi18] Leslie Smith. A disciplined approach to neural network hyper-parameters: Part 1 - learning rate, batch size, momentum, and weight decay. *CoRR*, abs/1803.09820, 2018.

[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.

[SS73] Peter Sneath and Robert Sokal. *Numerical taxonomy. The principles and practice of numerical classification.* San Francisco, W.H. Freeman and Company., USA, 1973.

[SSBD14] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms.* Cambridge University Press, USA, 2014.

[ST17] Leslie Smith and Nicholay Topin. Super-convergence: Very fast training of residual networks using large learning rates. *CoRR*, abs/1708.07120, 2017.

[Sta19] 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*, pages 65–79, Cham, 2019. Springer International Publishing.

[SVI+16] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826, 2016.

[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.

[SW14] Werner Schindler and Andreas Wiemers. Power attacks in the presence of exponent blinding. *Journal of Cryptographic Engineering*, 4, 06 2014.

[SW17] Werner Schindler and Andreas Wiemers. Generic power attacks on rsa with crt and exponent blinding: new results. *Journal of Cryptographic Engineering*, 7, 01 2017.

[SWES+19] Ron Shapira Weber, Matan Eyal, Nicki Skafte, Oren Shriki, and Oren Freifeld. Diffeomorphic temporal alignment nets. In Hanna Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché Buc, Emily Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[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.

[TAM20] Dhruv Thapar, Manaar Alam, and Debdeep Mukhopadhyay. Transca: Cross-family profiled side-channel attacks using transfer learning on deep neural networks. Cryptology ePrint Archive, Report 2020/1258, 2020. `https://eprint.iacr.org/2020/1258`.

[TB07] Michael Tunstall and Olivier Benoit. Efficient use of random delays in embedded software. In Damien Sauveron, Konstantinos Markantonakis, Angelos Bilas, and Jean-Jacques Quisquater, editors, *Information Security Theory and Practices. Smart Cards, Mobile and Ubiquitous Computing Systems*, pages 27–38, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[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.

[TG96a] Kagan Tumer and Joydeep Ghosh. Analysis of decision boundaries in linearly combined neural classifiers. *Pattern Recognition*, 29(2):341 – 348, 1996.

[TG96b] Kagan Tumer and Joydeep Ghosh. Error correlation and error reduction in ensemble classifiers. *Connection Science*, 8(3-4):385–404, 1996.

[TH12] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 2012.

[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.

[Tro04] Eran Tromer. Acoustic cryptanalysis : on nosy people and noisy machines. *Eurocrypt2004 Rump Session, May*, 2004.

[TV03] Kris Tiri and Ingrid Verbauwhede. Securing encryption algorithms against dpa at the logic level: Next generation smart card technology. In Colin Walter, Çetin Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2003*, pages 125–136, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[UY00] George Udny Yule. On the association of attributes in statistics: With illustrations from the material of the childhood society. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 194:257–319, 1900.

[VC71] Vladimir Vapnik and Alexey Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability & Its Applications*, 16(2):264–280, January 1971.

[VCGRS13] 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 Knudsen and Huapeng Wu, editors, *Selected Areas in Cryptography*, pages 390–406, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[VCGS13] Nicolas Veyrat-Charvillon, Benoît Gérard, and François-Xavier Standaert. Security evaluations beyond computing power. In Thomas Johansson and Phong Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, pages 126–141, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[vdMH08] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[vdODZ+16] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. In *9th ISCA Speech Synthesis Workshop*, pages 125–125, 2016.

[vdOKE+16] Aaron van den Oord, Nal Kalchbrenner, Lasse Espeholt, koray kavukcuoglu, Oriol Vinyals, and Alex Graves. Conditional image generation with pixelcnn decoders. In Daniel Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[vdVKPB20] Daan van der Valk, Marina Krcek, Stjepan Picek, and Shivam Bhasin. Learning from a big brother - mimicking neural networks in profiled side-channel analysis. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6, 2020.

[vdVP19] Daan van der Valk and Stjepan Picek. Bias-variance decomposition in machine learning-based side-channel analysis. Cryptology ePrint Archive, Report 2019/570, 2019. `https://eprint.iacr.org/2019/570`.

[vdVPB21] 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. In Guido Marco Bertoni and Francesco Regazzoni, editors, *Constructive Side-Channel Analysis and Secure Design*, pages 175–199, Cham, 2021. Springer International Publishing.

[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*, pages 44–62, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[vWWB11] Jasper van Woudenberg, Marc Witteman, and Bram Bakker. Improving differential power analysis by elastic alignment. In Aggelos Kiayias, editor, *Topics in Cryptology – CT-RSA 2011*, pages 104–119, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[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.

[Wal01] Colin Walter. Sliding windows succumbs to big mac attack. In Çetin Koç, David Naccache, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems — CHES 2001*, pages 286–299, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[WBFD19] Huanyu Wang, Martin Brisfors, Sebastian Forsmark, and Elena Dubrova. How diversity affects deep-learning side-channel attacks. In *2019 IEEE Nordic Circuits and Systems Conference (NORCAS): NORCHIP and International Symposium of System-on-Chip (SoC)*, pages 1–7, 2019.

[WBSG10] Qiang Wu, Christopher Burges, Krysta Svore, and Jianfeng Gao. Adapting boosting for information retrieval measures. *Information Retrieval*, 13:254–270, June 2010.

[WCL+20] Ping Wang, Ping Chen, Zhimin Luo, Gaofeng Dong, Mengce Zheng, Nenghai Yu, and Honggang Hu. Enhancing the performance of practical profiling side-channel attacks using conditional generative adversarial networks. Cryptology ePrint Archive, Report 2020/867, 2020. `https://eprint.iacr.org/2020/867`.

[Wei20] Léo Weissbart. Performance analysis of multilayer perceptron in profiling side-channel analysis. 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*, pages 198–216, Cham, 2020. Springer International Publishing.

[Wel47] Bernard Lewis Welch. The generalization of 'student's' problem when several different population variances are involved. *Biometrika*, 34(1/2):28–35, 1947.

[WHJ+21a] Yoo-Seung Won, Dong-Guk Han, Dirmanto Jap, Shivam Bhasin, and Jong-Yeon Park. Non-profiled side-channel attack based on deep learning using picture trace. *IEEE Access*, 9:22480–22492, 2021.

[WHJ+21b] Yoo-Seung Won, Xiaolu Hou, Dirmanto Jap, Jakub Breier, and Shivam Bhasin. Back to the basics: Seamless integration of side-channel pre-processing in deep neural networks. *IEEE Transactions on Information Forensics and Security*, 16:3215–3227, 2021.

[WJB20] Yoo-Seung Won, Dirmanto Jap, and Shivam Bhasin. Push for more: On comparison of data augmentation and smote with optimised deep learning architecture for side-channel. In Ilsun You, editor, *Information Security Applications*, pages 227–241, Cham, 2020. Springer International Publishing.

[WP20] Lichao Wu and Stjepan Picek. Remove some noise: On pre-processing of side-channel measurements with autoencoders. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020(4):389–415, Aug. 2020.

[WP21] Lichao Wu and Guilherme Perin. On the importance of pooling layer tuning for profiling side-channel analysis. Cryptology ePrint Archive, Report 2021/525, 2021. `https://eprint.iacr.org/2021/525`.

[WPB19] Léo 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*, pages 86–105, Cham, 2019. Springer International Publishing.

[WPP20] Lichao Wu, Guilherme Perin, and Stjepan Picek. I choose you: Automated hyperparameter tuning for deep learning-based side-channel analysis. Cryptology ePrint Archive, Report 2020/1293, 2020. `https://eprint.iacr.org/2020/1293`.

[WPP21] Lichao Wu, Guilherme Perin, and Stjepan Picek. The best of two worlds: Deep learning-assisted template attack. Cryptology ePrint Archive, Report 2021/959, 2021. `https://eprint.iacr.org/2021/959`.

[WWJ+21] Lichao Wu, Yoo-Seung Won, Dirmanto Jap, Guilherme Perin, Shivam Bhasin, and Stjepan Picek. Explain some noise: Ablation analysis for deep learning-based physical side-channel analysis. Cryptology ePrint Archive, Report 2021/717, 2021. `https://eprint.iacr.org/2021/717`.

[XKS92] Lei Xu, Adam Krzyzak, and Ching Suen. Methods of combining multiple classifiers and their applications to handwriting recognition. *IEEE Transactions on Systems, Man, and Cybernetics*, 22(3):418–435, 1992.

[XL07] Jun Xu and Hang Li. Adarank: A boosting algorithm for information retrieval. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '07, page 391–398, New York, NY, USA, 2007. Association for Computing Machinery.

[XLW+08] Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. Listwise approach to learning to rank: Theory and algorithm. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, page 1192–1199, New York, NY, USA, 2008. Association for Computing Machinery.

[YAGF21] Rabin Yu Acharya, Fatemeh Ganji, and Domenic Forte. Infoneat: Information theory-based neuroevolution of augmenting topologies for side-channel analysis. *CoRR*, abs/2105.00117, 2021.

[YLMZ19] 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*, pages 1–17, Cham, 2019. Springer International Publishing.

[Yu20] Ronald Yu. A tutorial on vaes: From bayes' rule to lossless compression, 2020.

[YZC+17] Wei Yang, Yongbin Zhou, Yuchen Cao, Hailong Zhang, Qian Zhang, and Huan Wang. Multi-channel fusion attacks. *IEEE Transactions on Information Forensics and Security*, 12(8):1757–1771, 2017.

[Zai20a] Gabriel Zaid. Application des techniques de deep learning dans la réalisation d'attaques par canaux auxiliaires. In *Journée "Nouvelles Avancées en Sécurité des Systèmes d'Information", INSA*, Toulouse, France, January 2020.

[Zai20b] Gabriel Zaid. Ranking loss: Maximizing the success rate in deep learning side-channel analysis. In *Journée Machine Learning & SCA GDR SoC2 et Sécurité Informatique*, Virtual, December 2020.

[ZBC+21a] Gabriel Zaid, Lilian Bossuet, Mathieu Carbone, Amaury Habrard, and Alexandre Venelli. Conditional variational autoencoder based on stochastic models. `https://github.com/gabzai/cVAE-ST`, 2021.

[ZBC+21b] Gabriel Zaid, Lilian Bossuet, Mathieu Carbone, Amaury Habrard, and Alexandre Venelli. Conditional variational autoencoder based on stochastic models. Cryptology ePrint Archive, Report 2021/???, 2021. `https://eprint.iacr.org/2021/???`

[ZBD+20a] Gabriel Zaid, Lilian Bossuet, François Dassance, Amaury Habrard, and Alexandre Venelli. Ranking loss: Maximizing the success rate in deep learning side-channel analysis. `https://github.com/gabzai/Ranking-Loss-SCA`, 2020.

[ZBD+20b] Gabriel Zaid, Lilian Bossuet, François Dassance, Amaury Habrard, and Alexandre Venelli. Ranking loss: Maximizing the success rate in deep learning side-channel analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2021(1):25–55, Dec. 2020.

[ZBH+17] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[ZBH+21] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115, February 2021.

[ZBHV19a] Gabriel Zaid, Lilian Bossuet, Amaury Habrard, and Alexandre Venelli. Methodology for efficient cnn architectures in profiling attacks. `https://github.com/gabzai/Methodology-for-efficient-CNN-architectures-in-SCA`, 2019.

[ZBHV19b] 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.

[ZBHV19c] Gabriel Zaid, Lilian Bossuet, Amaury Habrard, and Alexandre Venelli. Methodology for efficient cnn architectures in profiling attacks. In *Workshop on Practical Hardware Innovations in Security Implementation and Characterization (PHISIC)*, Gardanne, France, October 2019.

[ZBHV20] Gabriel Zaid, Lilian Bossuet, Amaury Habrard, and Alexandre Venelli. Understanding methodology for efficient cnn architectures in profiling attacks. Cryptology ePrint Archive, Report 2020/757, 2020. `https://eprint.iacr.org/2020/757`.

[ZBHV21a] Gabriel Zaid, Lilian Bossuet, Amaury Habrard, and Alexandre Venelli. Efficiency through diversity in ensemble models applied to side-channel attacks – a case study on public-key algorithms –. `https://github.com/gabzai/Ensembling-Loss-SCA`, 2021.

[ZBHV21b] Gabriel Zaid, Lilian Bossuet, Amaury Habrard, and Alexandre Venelli. Efficiency through diversity in ensemble models applied to side-channel attacks: – a case study on public-key algorithms –. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2021(3):60–96, Jul. 2021.

[ZBHV21c] Gabriel Zaid, Lilian Bossuet, Amaury Habrard, and Alexandre Venelli. Methodology for efficient cnn architectures in profiling attacks. In *Journée de la recherche de l'École doctorale EDSIS*, Virtual, 2021.

[Zei12] Matthew Zeiler. Adadelta: An adaptive learning rate method, 2012.

[ZF14] Matthew Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 818–833, Cham, 2014. Springer International Publishing.

[Zho12] Zhi-Hua Zhou. *Ensemble Methods: Foundations and Algorithms*. Chapman & Hall/CRC, 1st edition, 2012.

[ZL10] Zhi-Hua Zhou and Nan Li. Multi-information ensemble diversity. In Neamat El Gayar, Josef Kittler, and Fabio Roli, editors, *Multiple Classifier Systems*, pages 134–144, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[ZOB18] Yevhenii Zotkin, Francis Olivier, and Eric Bourbao. Deep learning vs template attacks in front of fundamental targets: experimental study. Cryptology ePrint Archive, Report 2018/1213, 2018. `https://eprint.iacr.org/2018/1213`.

[ZR21] Gabriel Zaid and Damien Robissout. Using deep learning against threats by side-channel analysis. In *Summer School - ARQUS European University Alliance*, Virtual, 2021.

[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*, 10(1):85–95, April 2019.

[ZXF+19] Libang Zhang, Xinpeng Xing, Junfeng Fan, Zongyue Wang, and Suying Wang. Multi-label deep learning based side channel attack. In *2019 Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*, pages 1–6, 2019.

[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.

Part V

Appendix

Appendix A

Additional Simulations on Leakage Model Estimation

To evaluate the benefits of the cVAE-ST to retrieve the correct basis, we simulate four sets of $10,000$ leakage traces through different scenarios ($9,000$ for the profiling phase and $1,000$ for the validation phase):

- **Scenario 1** – We assume that each leakage trace is configured by 3 time samples such that only 1 point of interest is considered. The leakage model induces the maximum amount of interactions between bits (*i.e.* $\mathcal{G}_9$) such that all bits influencing the leakage model have the same weighting. Hence, the i^{th} time sample of the simulated trace $\mathbf{T}$ is defined as follows:

$$\mathbf{T}[i] = \begin{cases} 1 \cdot Y[1] + 1 \cdot Y[3] + 1 \cdot Y[6] \\ + 1 \cdot \oplus_{b=0}^{1} Y[b] + 1 \cdot \oplus_{b=0}^{2} Y[b] + 1 \cdot \oplus_{b=0}^{3} Y[b] \\ + 1 \cdot \oplus_{b=0}^{4} Y[b] + 1 \cdot \oplus_{b=0}^{5} Y[b] + 1 \cdot \oplus_{b=0}^{6} Y[b] \\ + 1 \cdot \oplus_{b=0}^{7} Y[b] + \mathbf{Z}[i] & \text{if} \quad i = 1, \\ \\ \mathbf{Z}[i] & \text{otherwise,} \end{cases}$$

  where $\oplus_{b=0}^{n} Y[b] = Y[0] \oplus \ldots \oplus Y[n]$, $Y[b] = Sbox[X \oplus k^*][b]$ denotes the b^{th} bit of the output of the Sbox, and $\mathbf{Z}[i]$ is a Gaussian noise following $\mathcal{N}(0, \sigma^2)$ such that $\sigma^2 \in \{0.1, 1, 10\}$.

- **Scenario 2** – We assume that each leakage trace is configured by 4 time samples such that only 2 points of interest are considered. The leakage model does not induce interactions between bits (*i.e.* $d = 1$ and $\mathcal{G}_2$) but differs from the location of the points of interest. Hence, the i^{th} time sample of the simulated trace $\mathbf{T}$ is defined as follows:

$$\mathbf{T}[i] = \begin{cases} 1 \cdot Y[3] + 1 \cdot Y[6] + \mathbf{Z}[i] & \text{if} \quad i = 1, \\ 1 \cdot Y[1] + 1 \cdot Y[7] + \mathbf{Z}[i] & \text{if} \quad i = 2, \\ \mathbf{Z}[i] & \text{otherwise,} \end{cases}$$

  where $Y[b] = Sbox[X \oplus k^*][b]$ and $\mathbf{Z}[i]$ is a Gaussian noise following $\mathcal{N}(0, \sigma^2)$ such that $\sigma^2 \in \{0.1, 1, 10\}$.

- **Scenario 3** – We assume that each leakage trace is configured by 3 time samples such that only 1 point of interest is considered. The leakage model does not induce interactions between bits (*i.e.* $d = 1$ and $\mathcal{G}_2$) such that all bits influencing the leakage model have the same weighting. Hence, the i^{th} time sample of the simulated trace $\mathbf{T}$ is defined as follows:

$$\mathbf{T}[i] = \begin{cases} 1 \cdot Y[3] + 1 \cdot Y[6] + \mathbf{Z}[i] & \text{if} \quad i = 1, \\ \mathbf{Z}[i] & \text{otherwise,} \end{cases}$$

where $Y[b] = Sbox[X \oplus k^*][b]$ denotes the b^{th} bit of the output of the Sbox and $\mathbf{Z}[i]$ is a Gaussian noise following $\mathcal{N}(0, \sigma^2)$ such that $\sigma^2 \in \{0.1, 1, 10\}$.

- **Scenario 4** – We assume that each leakage trace is configured by 3 time samples such that only 1 point of interest is considered. The leakage model does not induce interactions between bits (*i.e.* $d = 1$ and $\mathcal{G}_2$) such that all bits influencing the leakage model have different weighting. Hence, the i^{th} time sample of the simulated trace $\mathbf{t}$ is defined as follows:

$$\mathbf{T}[i] = \begin{cases} 1 \cdot Y[3] + 0.5 \cdot Y[6] + \mathbf{Z}[i] & \text{if} \quad i = 1, \\ \mathbf{Z}[i] & \text{otherwise,} \end{cases}$$

where $Y[b] = Sbox[X \oplus k^*][b]$ and $\mathbf{Z}[i]$ is a Gaussian noise following $\mathcal{N}(0, \sigma^2)$ such that $\sigma^2 \in \{0.1, 1, 10\}$.

Different levels of noise are considered for all scenarios in order to get an overview into how cVAE-ST performs depending on the SNR result (see Table A.1). Whatever the scenario, the cVAE-ST can retrieve the leakage model (see Figure A.2, Figure A.3, Figure A.4 and Figure A.5). Hence, the cVAE-ST can be used to evaluate the security flaws when large bits' interactions are observed, when the deterministic part is different for all PoIs and when different weighting occurs. As a consequence, large use-cases can be considered when the cVAE-ST is applied. However, for a high level of noise, the extraction of the leakage model becomes more difficult. To retrieve the leakage model, the evaluator has to increase the number of profiling traces in order to reduce the noise effect. While the impact of the noise highly depends on the underlying implementation, no general rules can be formulated to predefine the number of profiling traces that are needed for a given level of noise.

Table A.1: SNR value for each scenario

	SNR value over $10,000$ **simulated traces**		
	$\sigma^2 = 0.1$	$\sigma^2 = 1$	$\sigma^2 = 10$
Scenario 1	258.51	2.578	0.0577
Scenario 2	50.58	0.509	0.0319
Scenario 3	51.28	0.549	0.0368
Scenario 4	32.38	0.346	0.0262

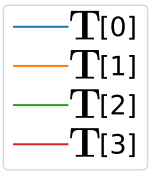


Figure A.1: Legend.

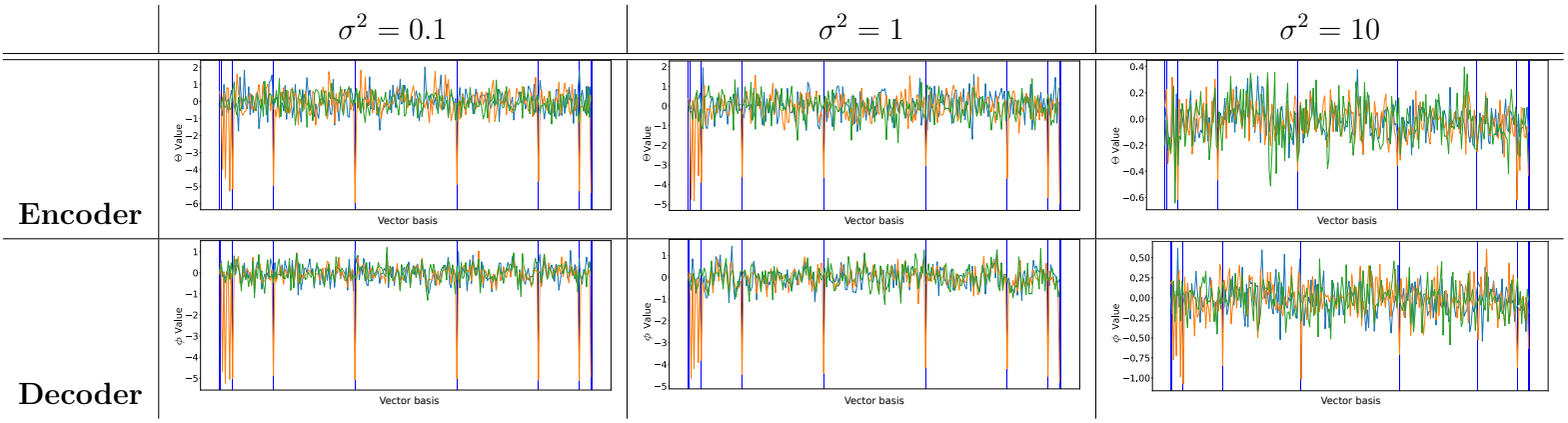


Figure A.2: Weight visualization for **Scenario 1** (Legend: see Figure A.1)

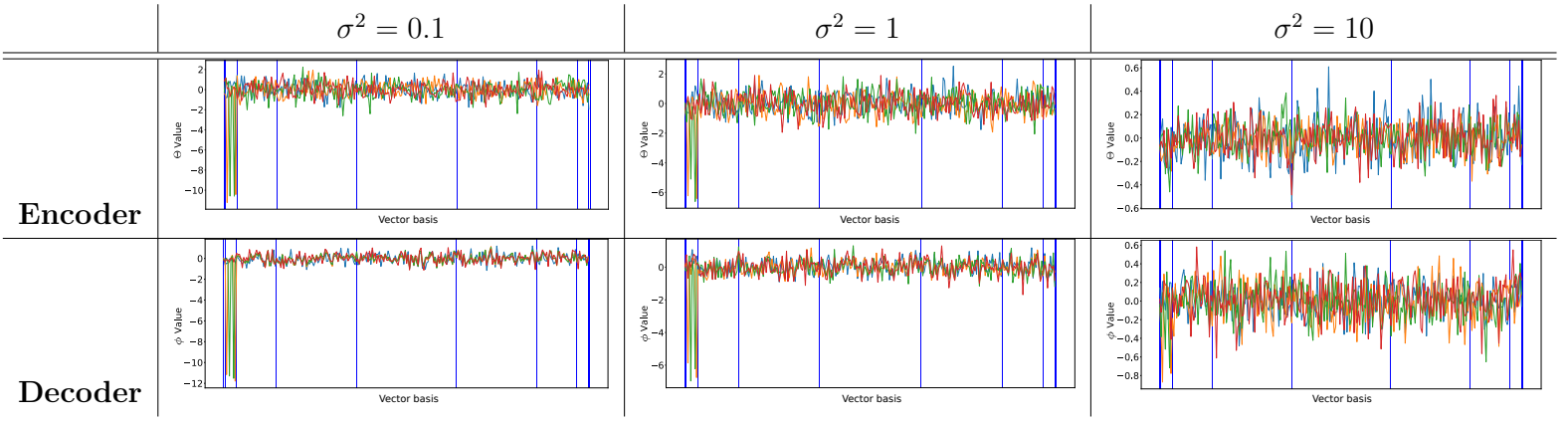


Figure A.3: Weight visualization for **Scenario 2** (Legend: see Figure A.1)

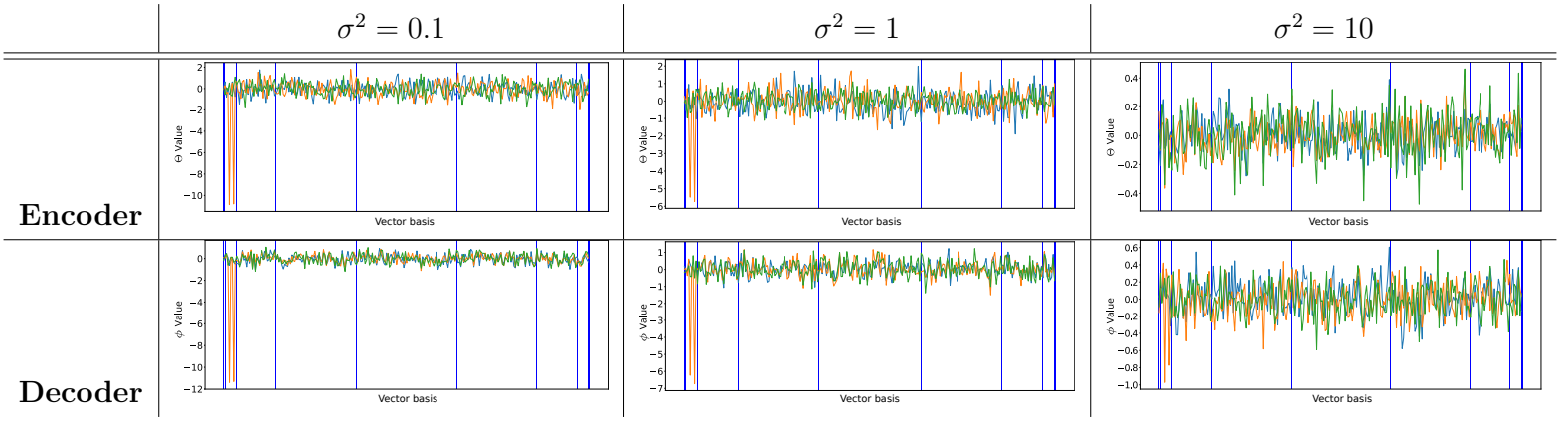


Figure A.4: Weight visualization for **Scenario 3** (Legend: see Figure A.1)

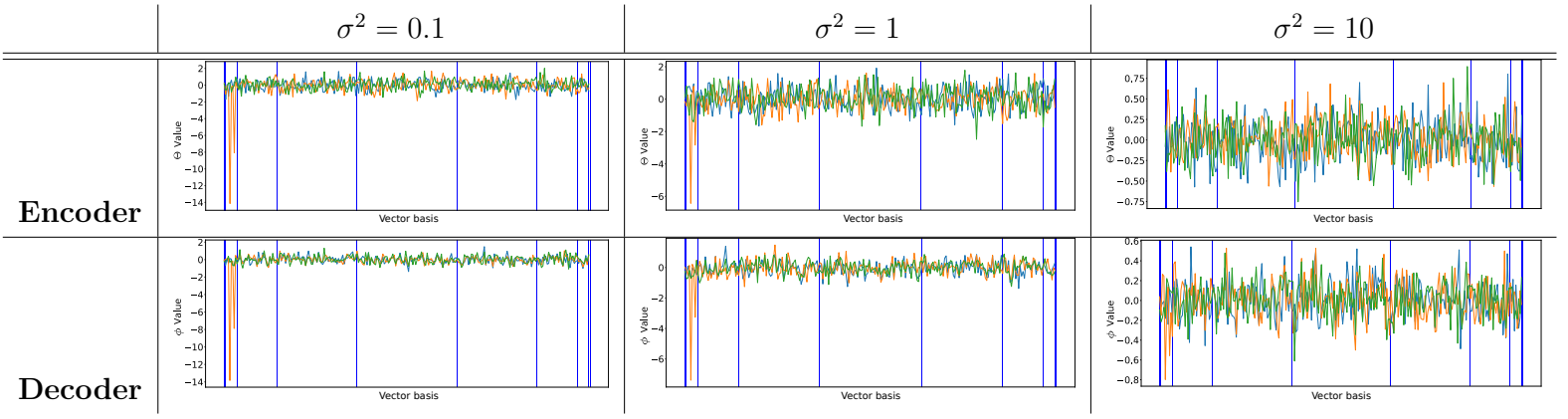


Figure A.5: Weight visualization for **Scenario 4** (Legend: see Figure A.1)

Appendix B

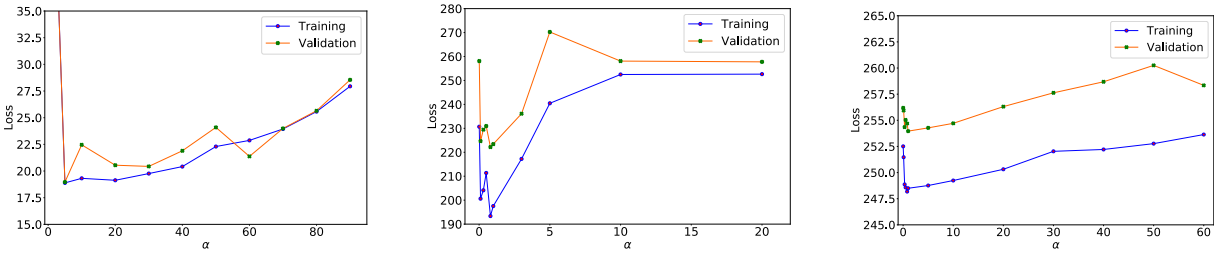
Impact of α on the Empirical Risk combined with the Ranking Loss

Through this appendix, we evaluate the impact of α on the empirical risk combined with the ranking loss. Given a profiling set $\mathcal{I}_p$ of N_p pairs $(\mathbf{t}_i, y_i)_{0 \leq i \leq N_p}$, a Θ-parametric model F_Θ and a number of attack traces N_a such that $N_a | N_p$, the empirical risk combined with ranking loss function is defined as:

$$\hat{\mathcal{R}}(\mathcal{L}_{RkL}, F_\Theta) = \frac{N_a}{N_p} \sum_{i=1}^{N_p/N_a} \sum_{\substack{k \in \mathcal{K} \\ k \neq k^*}} \left(\log_2 \left(1 + e^{-\alpha(s_{N_a,i}(F_\Theta, k^*) - s_{N_a,i}(F_\Theta, k))} \right) \right),$$

where $s_{N_a,i}(F_\Theta, k) = \sum_{j=1}^{N_a} F_\Theta \left(\mathbf{t}_{j+N_a \cdot (i-1)} \right) [f(x_j, k)]$ defines the output score of the hypothesis $k \in |\mathcal{K}|$ for a given plaintext $(x_j)_{1 \leq j \leq N_a}$ while α denotes one hyperparameter related to the sigmoid and approximates the identity function needed for estimating the success rate.

We select a wide range of α values in order to efficiently evaluate the impact of this optimizer hyperparameter on the training process. Figure B.1 illustrates the impact of α on the loss function depending on the dataset used for training the model. The architectures are the same as in Section 7.3. For the Chipwhisperer dataset, if α is small (*e.g.* less than 10), the sigmoid function approximates the indicator function less accurately [BZBN19]. Consequently, the minimization of the empirical risk does not provide a Θ-parametric model with high performance. When α is too large, the gradient tends to vanish (see Subsection 4.3.2) and the training process results in a model with poor performance. It appears that a relatively small value of α could provide a good trade-off between the learning optimization and the approximation of the indicator function.
The same observation can be made when the ASCAD-v1 with synchronized leakage traces (see Figure B.1b) and AES_HD (see Figure B.1c) are considered. However, we have to note that α should be carefully configured depending on the dataset and the score returned by the Θ-parametric model.



(a) Chipwhisperer **(b)** ASCAD (Synchronized traces) **(c)** AES_HD

Figure B.1: Impact of α on the loss function during the training phase

Appendix C

Bounds in Learning to Rank Metrics

In [CLL+09a], Chen *et al.* reveal the relationship between classical information retrieval measures (*i.e.* MAP, DCG, NDCG) and the loss functions used in "Learning to Rank" approach (*i.e.* Pairwise and Listwise losses). In this appendix, we exploit their results in order to argue the inequalities introduced in Equation 7.6 and Equation 7.7. Let $s_{N_a}(g_{N_a}, i)$ be the score associated with the key hypothesis ranked at the i^{th} position in the ranking vector g_{N_a}, introduced in Subsection 3.3.4. In [CLL+09a], the authors define a loss function, called *essential loss*, that characterizes the sum of the classification error involved during the training process,

$$l_{ess}\left(s_{N_a}(g_{N_a}, i)\right) \triangleq \sum_{i=0}^{|\mathcal{K}|-2}\left(1 - \prod_{j=i+1}^{|\mathcal{K}|-1} \mathbf{1}_{s_{N_a}(g_{N_a},i) > s_{N_a}(g_{N_a},j)}\right).$$

First, the essential loss can be defined as an upper bound of $(1 - MAP@|\mathcal{K}|)$ where $|\mathcal{K}|$ denotes the number of key hypotheses and $MAP@|\mathcal{K}|$ is the mean average precision over the top $|\mathcal{K}|$ relevant position in $\mathbf{g}$.

Theorem C.1 (From item (2) of Theorem 1 [CLL+09a])**.** *Given a 2-level rating data with n_1 elements with the label 1 and $n_1 > 1$, then, the following inequality holds,*

$$1 - MAP@|\mathcal{K}| \leq \frac{1}{n_1}\sum_{i=0}^{|\mathcal{K}|-2}\left(1 - \prod_{j=i+1}^{|\mathcal{K}|-1} \mathbf{1}_{s_{N_a}(g_{N_a},i) > s_{N_a}(g_{N_a},j)}\right).$$

Proof. See the proof of *Theorem 1* in [CLL+09a] □

Following Chen *et al.*, the essential loss can also be defined as a lower bound of many ranking loss functions (*e.g.* pairwise loss or listwise loss).

Theorem C.2 (From item (1) of Theorem 2 [CLL+09a])**.** *The pairwise loss function is an upper bound of the essential loss such as,*

$$\sum_{i=0}^{|\mathcal{K}|-2}\left(1 - \prod_{j=i+1}^{|\mathcal{K}|-1} \mathbf{1}_{s_{N_a}(g_{N_a},i) > s_{N_a}(g_{N_a},j)}\right) \leq \sum_{i=0}^{|\mathcal{K}|-2}\sum_{\substack{j=0 \\ gr(j)<gr(i)}}^{|\mathcal{K}|-1} \log_2\left(1 + e^{-\alpha(s_{N_a}(F_\Theta,i) > s_{N_a}(F_\Theta,j))}\right).$$

Proof. See the proof of *Theorem 2* in [CLL+09a] □

Interestingly, combining the results obtained in C.1 and C.2 is helpful to distinguish a relation between the classical learning to rank loss functions and the information retrieval measures used to evaluate the training process. Indeed, this combination gives us the following result,

$$(1 - MAP@|\mathcal{K}|) \leq \frac{1}{n_1}\sum_{i=0}^{|\mathcal{K}|-2}\sum_{\substack{j=0 \\ gr(j)<gr(i)}}^{|\mathcal{K}|-1} \log_2\left(1 + e^{-\alpha(s_{N_a}(F_\Theta,i) - s_{N_a}(F_\Theta,j))}\right),$$

where $gr(i)$ defines the grade associated to the i^{th} key hypothesis (*i.e.* 0 or 1).

Appendix D

Approximation of the Pairwise Mutual Information Ensemble Diversity

In [Bro09], Brown proposes a solution to compute the mutual information between an ensemble model composed by a set of parametric models $\mathcal{E} = \{X_0, X_1, \ldots, X_{N_c-1}\}$ and a set of true unknown labels Y. However, as mentioned in Subsection 8.2.2, it is quite difficult to estimate higher-order interaction information. Thus, Brown proposes to simplify Equation 8.2 by considering only pairwise components as follows:

$$MI(\mathcal{E};Y) \approx \sum_{n=0}^{N_c-1} MI(F_n;Y) - \sum_{n=0}^{N_c-2}\sum_{m=n+1}^{N_c-1} MI(F_n;F_m) + \sum_{n=0}^{N_c-2}\sum_{m=n+1}^{N_c-1} MI(F_n;F_m|Y), \quad \text{(D.1)}$$

where $MI(F_n;Y)$ computes the mutual information between the n^{th} model of $\mathcal{E}$ and the target Y, $MI(F_n;F_m)$ measures the mutual information between two models F_n and F_m, and $MI(F_n;F_m|Y)$ measures the conditional redundancy between two models F_n and F_m knowing Y.

Based on an ensemble model composed by a set $\mathcal{E}$ of parametric models, we aim at approximating $MI(\mathcal{E};Y)$ such that each component of Equation D.1 are optimized during the training process. In order to achieve a general-purpose estimator, we base our propositions on the characterization of the mutual information as the Kullback-Leibler (KL-) divergence [KL51] between the joint distribution and the product of the marginals such that, given two random variables X and Y, we have:

$$\begin{aligned} MI(X;Y) &= H(Y) - H(Y|X) = D_{KL}(\mathsf{Pr}[X,Y]||\mathsf{Pr}[X]\mathsf{Pr}[Y]) \\ &= \sum_{(x,y)\in\mathcal{X}\times\mathcal{Y}} \mathsf{Pr}[X=x]\mathsf{Pr}[Y=y|X=x]\ \log\left(\frac{\mathsf{Pr}[Y=y|X=x]}{\mathsf{Pr}[Y=y]}\right). \end{aligned} \quad \text{(D.2)}$$

Furthermore, the conditional mutual information of random variables X and Y given Z can be expressed as follows:

$$\begin{aligned} MI(X;Y|Z) &= H(Y|Z) - H(Y|X,Z) \\ &= \sum_{z\in\mathcal{Z}} \mathsf{Pr}[Z=z] \sum_{(x,y)\in\mathcal{X}\times\mathcal{Y}} \mathsf{Pr}[X=x,Y=y|Z=z]\ \log\left(\frac{\mathsf{Pr}[X=x,Y=y|Z=z]}{\mathsf{Pr}[X=x|Z=z]\mathsf{Pr}[Y=y|Z=z]}\right). \end{aligned} \quad \text{(D.3)}$$

Based on these definitions, we develop a loss function for each term of Equation D.1.

Approximation of $MI(F_n;Y)$. The mutual information $MI(F_n;Y)$ is used to compute the relevancy term (*i.e.* $\sum_{n=0}^{N_c-1} MI(F_n;Y)$) such that, following Equation D.1 and Equation D.2, we want to maximize the probability of Y knowing F_n. If $F_n : \mathcal{T} \to \mathcal{S}$, such that $\mathcal{S} \subseteq \mathbb{R}^{|\mathcal{Y}|}$, we have to solve a similar problem to the ranking loss defined in Chapter 7. Thus, given a profiling set $\mathcal{I}_p$ of N_p pairs $(\mathbf{t}_i, y_i)_{0\leq i\leq N_p}$, a parametric model F_n and a number of attack traces N_a such that $N_a|N_p$, we define the empirical risk combined with the *Relevance Loss* function as:

$$\hat{\mathcal{R}}(\mathcal{L}_{ReL}, F_n) = \frac{N_a}{N_p} \sum_{i=1}^{N_p/N_a} \sum_{\substack{k\in\mathcal{K} \\ k\neq k^*}} \left(\log_2\left(1 + e^{-\alpha(s_{N_a,i}(F_n,k^*) - s_{N_a,i}(F_n,k))}\right)\right), \tag{D.4}$$

where $s_{N_a,i}(F_n,k) = \sum_{j=1}^{N_a} F_n\left(\mathbf{t}_{j+N_a\cdot(i-1)}\right)[f(x_j,k)]$ defines the output score of the hypothesis $k \in |\mathcal{K}|$ for a given plaintext $(x_j)_{1\leq j\leq N_a}$ while α approximates the sigmoid function needed for estimating the success rate.
The interested readers may refer to Subsection 7.1.2 in order to obtain more details on Equation D.4.

Approximation of $MI(F_n;F_m|Y)$. The conditional interaction information $MI(F_n;F_m|Y)$ is used to compute the conditional redundancy term (*i.e.* $\sum_{n=0}^{N_c-2}\sum_{m=n+1}^{N_c-1} MI(F_n;F_m|Y)$), such that, following Equation D.1 and Equation D.3, we want to maximize the probability of observing F_n and F_m knowing the sensitive variable Y. This probability always equals 1 if F_n and F_m provide the same score related to the class Y. Consequently, if $F_n : \mathcal{T} \to \mathcal{S}$ and $F_m : \mathcal{T} \to \mathcal{S}$, such that $\mathcal{S} \subseteq \mathbb{R}^{|\mathcal{Y}|}$, we want to find a solution that maximizes $\mathsf{Pr}[s_{N_a,i}(F_n,k^*) = s_{N_a,i}(F_m,k^*)]$. In other words, maximizing an approximation of $MI(F_n;F_m|Y)$ consists in minimizing the distance between the scores $s_{N_a,i}(F_n,k^*)$ and $s_{N_a,i}(F_m,k^*)$. Following [IW18, Section 2.3], the probability $\mathsf{Pr}[s_{N_a,i}(F_n,k^*) = s_{N_a,i}(F_m,k^*)]$ can be approximated as follows:

$$\mathsf{Pr}\left[s_{N_a,i}(F_n,k^*) = s_{N_a,i}(F_m,k^*)\right] = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\left(\frac{\left|s_{N_a,i}(F_n,k^*) - s_{N_a,i}(F_m,k^*)\right| - \mu}{\sigma}\right)^2}, \tag{D.5}$$

such that the parameters $\sigma = \frac{1}{\sqrt{2\pi}}$ and $\mu = 0$ respect the condition $\mathsf{Pr}\left[s_{N_a,i}(F_n,k^*) = s_{N_a,i}(F_m,k^*)\right] = 1$ if and only if $|s_{N_a,i}(F_n,k^*) - s_{N_a,i}(F_m,k^*)| = 0$. Thus, Equation D.5 can be rewritten as follows:

$$\mathsf{Pr}\left[s_{N_a,i}(F_n,k^*) = s_{N_a,i}(F_m,k^*)\right] = e^{-\pi|s_{N_a,i}(F_n,k^*) - s_{N_a,i}(F_m,k^*)|^2}. \tag{D.6}$$

Then, similarly to the ranking loss, we apply the binary cross-entropy in order to penalize the deviation of the model probabilities from the desired prediction. In other words, we want to penalize the loss function when $s_{N_a}(F_n,k^*)$ differs from $s_{N_a}(F_m,k^*)$. Thus, we define the conditional redundancy loss function $\mathcal{L}_{CRL}(F_n,F_m,Y)$ as:

$$\mathcal{L}_{CRL}(F_n,F_m,k^*) = -P_{F_n,F_m,k^*}\cdot\log_2\left(\bar{P}_{F_n,F_m,k^*}\right) - (1 - P_{F_n,F_m,k^*})\cdot\log_2\left(1 - \bar{P}_{F_n,F_m,k^*}\right), \tag{D.7}$$

where $\bar{P}_{F_n,F_m,k^*} = \mathsf{Pr}\left[s_{N_a}(F_n,k^*) = s_{N_a}(F_m,k^*)\right]$ and P_{F_n,F_m,k^*} defines the true unknown probability that $s_{N_a}(F_n,k^*)$ equals $s_{N_a}(F_m,k^*)$.
Similarly to the ranking loss, we assume that P_{F_n,F_m,k^*} is deterministically known such that, $P_{F_n,F_m,k^*} = \frac{1}{2}(1 + rel_{F_n,F_m,k^*})$ where $rel_{F_n,F_m,k^*} \in \{-1,1\}$ defines the relation between $s_{N_a}(F_n,k^*)$ and $s_{N_a}(F_m,k^*)$ such that $rel_{F_n,F_m,k^*} = 1$ if $s_{N_a}(F_n,k^*)$ equals $s_{N_a}(F_m,k^*)$ and $rel_{F_n,F_m,k^*} = -1$ otherwise. As we want to force the equality between $s_{N_a}(F_n,k^*)$ and $s_{N_a}(F_m,k^*)$, we assume that $P_{F_n,F_m,k^*} = 1$. From Equation D.6 and Equation D.7, we can deduce the following conditional redundancy loss function:

$$\mathcal{L}_{CRL}(F_n,F_m,k^*) = -\log_2\left(e^{-\pi|s_{N_a,i}(F_n,k^*) - s_{N_a,i}(F_m,k^*)|^2}\right). \tag{D.8}$$

However, from a computational perspective, a large gap between $s_{N_a,i}(F_n, k^*)$ and $s_{N_a,i}(F_m, k^*)$ can be problematic to solve Equation D.8 as the exponential value converges quadratically towards 0. To reduce this issue, we simplify the distance measurement such that the empirical risk combined with the conditional redundancy loss function can be written as follows: given a profiling set $\mathcal{I}_p$ of N_p pairs $(\mathbf{t}_i, y_i)_{0 \leq i \leq N_p}$, a pair of parametric models $(F_n,\ F_m)$ and a number of attack traces N_a such that $N_a | N_p$, we define the empirical risk combined with the *Conditional Redundancy Loss* function as:

$$\hat{\mathcal{R}}(\mathcal{L}_{CRL}, F_n, F_m) = \frac{N_a}{N_p} \sum_{i=1}^{N_p/N_a} -\log_2 \left(e^{-\beta |s_{N_a,i}(F_n,k^*) - s_{N_a,i}(F_m,k^*)|} \right),$$

where β is one hyperparameter that characterizes the impact of the distance on the penalization term and $s_{N_a,i}(F_n, k^*)$ defines the score related to the class k^* given a set of N_a traces and a parametric model F_n.

Remark D.1. Furthermore, through the maximization of the probability of observing F_n and F_m knowing the sensitive variable Y, we force the ensemble model to assign similar score to the class related to Y such that, the probability of observing F_n knowing Y is similar to the probability of observing F_m knowing Y. Thus, we encourage each term of the denominator of Equation D.3 to converge towards the same solution.

Approximation of $MI(F_n; F_m)$. The mutual information $MI(F_n; F_m)$ is used to compute the redundancy term (*i.e.* $\sum_{n=0}^{N_c-2} \sum_{m=n+1}^{N_c-1} MI(F_n; F_m)$) such that, following Equation D.1 and Equation D.2, we want to minimize the probability of observing F_n knowing F_m. Following the discussion provided on the conditional redundancy loss, we have to maximize the dissimilarity between F_n and F_m. If $F_n : \mathcal{T} \rightarrow \mathcal{S}$ and $F_m : \mathcal{T} \rightarrow \mathcal{S}$, such that $\mathcal{S} \subseteq \mathbb{R}^{|\mathcal{Y}|}$, we want to find a solution that maximizes $\mathsf{Pr}[s_{N_a,i}(F_n, k) \neq s_{N_a,i}(F_m, k')]$ for each pair $(Y_k, Y_{k'})_{(k,k') \in \mathcal{K} \times \mathcal{K}}$ such that $Y_k = f(X, k)$ (resp. $Y_{k'} = f(X, k')$) for a given plaintext X, a cryptographic primitive f and a key hypothesis k (resp. k').
As $\mathsf{Pr}[s_{N_a,i}(F_n, k) \neq s_{N_a,i}(F_m, k')] = 1 - \mathsf{Pr}[s_{N_a,i}(F_n, k) = s_{N_a,i}(F_m, k')]$, we can reuse the results obtained from the conditional redundancy loss in order to compute a partial redundancy loss, denoted $\mathcal{L}_{RedL}^{(p)}$, such that:

$$\mathcal{L}_{RedL}^{(p)}(F_n, F_m, k, k') = -\log_2 \left(1 - e^{-\gamma |s_{N_a,i}(F_n,k) - s_{N_a,i}(F_m,k')|} \right),$$

where γ is one hyperparameter that characterizes the impact of the distance on the penalization term. Then, as the partial redundancy loss function $\mathcal{L}_{RedL}^{(p)}(F_n, F_m, k, k')$ has to be performed on each pair $(k, k') \in \mathcal{K} \times \mathcal{K}$, it is possible to generalize the result to compute the resulted empirical risk. Indeed, given a profiling set $\mathcal{I}_p$ of N_p pairs $(\mathbf{t}_i, y_i)_{0 \leq i \leq N_p}$, a pair of parametric models $(F_n,$ $F_m)$ and a number of attack traces N_a such that $N_a | N_p$, the empirical risk combined with the *Redundancy Loss* function can be defined as:

$$\hat{\mathcal{R}}(\mathcal{L}_{RedL}, F_n, F_m) = \frac{N_a}{N_p} \sum_{i=1}^{N_p/N_a} \sum_{k=0}^{|\mathcal{K}|-1} \sum_{k'=0}^{|\mathcal{K}|-1} -\log_2 \left(1 - e^{-\gamma |s_{N_a,i}(F_n,k) - s_{N_a,i}(F_m,k')|} \right).$$

Approximation of the pairwise mutual information ensemble diversity. Based on Equation D.1, we can introduce a new empirical risk that maximizes an approximation of the mutual information between a set of parametric models $\mathcal{E} = \{F_0, F_1, \cdots, F_{N_c-1}\}$ and a targeted information Y. Indeed, given a profiling set $\mathcal{T}$ of N_p pairs $(\mathbf{t}_i, y_i)_{1 \leq i \leq N_p}$ and a number of attack traces N_a such that $N_a | N_p$, we define the empirical risk combined with the ensembling loss function as:

$$\hat{\mathcal{R}}(\mathcal{L}_{Rel}, \mathcal{L}_{RedL}, \mathcal{L}_{CRL}, \mathcal{E}) = \frac{1}{N_c} \sum_{n=0}^{N_c-1} \hat{\mathcal{R}}(\mathcal{L}_{ReL}, F_n) + \frac{2\mu}{N_c(N_c-1)} \sum_{n=0}^{N_c-2} \sum_{m=n+1}^{N_c-1} \left(\hat{\mathcal{R}}(\mathcal{L}_{RedL}, F_n, F_m) + \hat{\mathcal{R}}(\mathcal{L}_{CRL}, F_n, F_m) \right), \quad \text{(D.9)}$$

where μ quantifies the impact of the diversity term during the training process. We normalize each term of the empirical risk to reduce the impact of exploding gradient. Appendix E highlights the benefits of each individual loss from a training perspective. Through this study, the reader can understand how the network would train if the conditional redundancy loss or the redundancy loss is individually used.

Appendix E

t-SNE Ensembling Loss

Figure E.1 illustrates the evolution of the t-SNE visualizations [vdMH08] in order to evaluate the impact of the *Conditional Redundancy Loss* and the *Redundancy Loss* during the training process. First, as mentioned in Subsection 8.3.1, the ranking loss, introduced in Definition 7.1.2.1, can be formulated as the relevance loss (see Equation 8.4). Through its minimization, we minimize the conditional entropy $H(Y|F_n)$ which promotes the generation of three compact clusters. Figure E.1 confirms this observation. The ensemble model is overconfident in the features captured during the training process. Consequently, it detects discriminative patterns to avoid connections between each cluster. However, following the t-SNE illustration, the False Positives (FP) and the False Negatives (FN) induced by the ranking loss are persistent and seem difficult to detect. Indeed, these errors are fully included in a wrong cluster. For a given number of profiling leakage traces, a solution is to promote the interaction between the committee members in order to reduce this overconfidence and enhance the ensemble model.

In Equation 8.5, the *Conditional Redundancy Loss* function minimizes $(1 - \mathsf{Pr}\left[F_n, F_m|Y\right])$ which defines the probability of observing the model F_n and F_m given Y. Hence, maximizing $\mathsf{Pr}\left[F_n, F_m|Y\right]$ is asymptotically equivalent to minimizing $H\left(F_n, F_m|Y\right)$. Therefore, we force the network to generate three compact clusters given the correct sensitive variable (*i.e.* label). This loss tends to increase the confidence of the network on the True Positives (TP) and the True Negatives (TN) while reducing the impact of the FP and the FN. This observation can be made on Figure E.1. Indeed, adding the conditional redundancy to the ranking loss is helpful to distinguish TPs and FPs for each cluster. Hence, each cluster is divided into two parts: a part with high level of confidence in prediction and a part with uncertain predictions. This phenomenon highlights the benefits of the conditional redundancy loss function to reduce the intra-class variance and makes an easier distinction between confident and uncertain predictions. However, as illustrated in Figure E.1, the conditional redundancy loss function does not clearly separate the confident and uncertain predictions into different clusters. Hence, an additional partial loss should be considered in order to increase the dissociation between these samples. This is provided by the *Redundancy Loss* function.

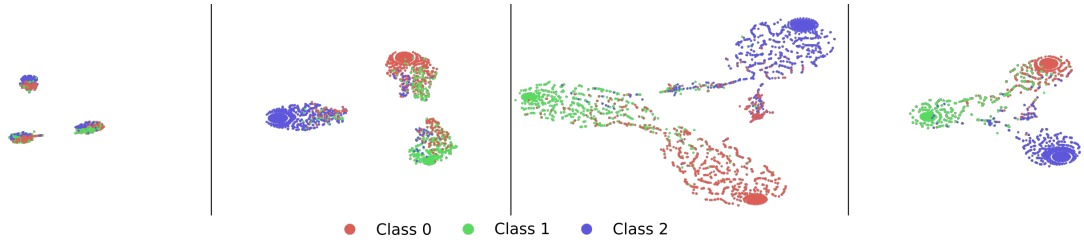


Figure E.1: t-SNE embeddings. From left to right: Ranking loss, Ranking Loss + Conditional Redundancy Loss, Ranking Loss + Redundancy Loss, Ensembling Loss (= Ranking Loss + Conditional Redundancy Loss + Redundancy Loss).

In Equation 8.6, the *Redundancy Loss* function minimizes $\mathsf{Pr}[F_m|F_n]$ which defines the output probability of the model F_m given F_n. From an information theory perspective, this can be considered as a minimization of $H(F_m|F_n)$ In other words, we want to maximize the inter-class variance between the models F_m and F_n. Hence, adding the redundancy loss to the ranking loss should increase the distance between each cluster by diversifying the features representation of each cluster. This observation can be validated thanks to Figure E.1. Indeed, the third t-SNE visualization illustrates a model trained with the ranking and the redundancy losses. In comparison with the first t-SNE visualization, we can highlight the benefits of the redundancy loss to increase the distance between each cluster and make the FN and FP less persistent. However, in some extent, this approach generates sparse representation of a given cluster and also reduces the confidence of the networks on some TP. Hence a good trade-off has to be found between maximizing the confidence of the TP (*i.e.* conditional redundancy loss) and minimizing the persistence of the FP (*i.e.* redundancy loss). The *Ensembling Loss* aims at finding this solution for given α, β, γ, μ values (see Equation D.9).
In Figure E.1, the combination of the relevance loss, the conditional redundancy loss and the redundancy loss creates three separate clusters. When the network is confident in its predictions, it will assign the related examples to the correct cluster. Thanks to the conditional redundancy loss, we know that the predictions with a high level of confidence will be assigned to the same compact cluster. However, the ensembling loss also creates some connections between the clusters which seem defined by the data uncertainty. This result tends to reduce the number of consistent FP and FN such that few errors can be detected on each cluster in contrast with the ranking loss. This observation highlights the benefits of the redundancy loss during the training process. In Figure E.1, the ensembling loss finds a good trade-off between maximizing the confidence of the TP and minimizing the persistence of the FP.

Appendix F

Neural Network Architectures for Enhancing Diversity

Type I diversity. The architectures used for the type I diversity are randomly selected such that the number of convolutional layers and fully-connected layers (FC) do not exceed 2. Hence, we evaluate the type I diversity with the restriction of small network complexity. We select 5 neural network architectures with high individual α_{label} value (*i.e.* $\geq 85\%$) to limit the impact of the outliers and preserve an overall good performance. All the architectures used for the type I diversity investigation are detailed in Table F.1.

Table F.1: Architectures and performance related to the networks used for the type I diversity (models trained with the ranking loss)

Type I diversity	$\boldsymbol{Model_1}$	$\boldsymbol{Model_2}$	$\boldsymbol{Model_3}$	$\boldsymbol{Model_4}$	$\boldsymbol{Model_5}$
1^{st} **Conv. layer (+ BN)**	2 filters (size 1)	10 filters (size 5)	5 filters (size 15)	2 filters (size 1)	2 filters (size 1)
1^{st} **Pool. layer**	Avg (stride 2)	Avg (stride 5)	Max (stride 5)	Avg (stride 2)	Max (stride 2)
2^{nd} **Conv. layer (+ BN)**	-	-	-	2 filters of size 25	-
2^{nd} **Pool. layer**	-	-	-	Avg (stride 2)	-
Flatten	Yes	Yes	Yes	Yes	Yes
1^{st} **FC layer**	-	2 nodes	-	-	5 nodes
2^{nd} **FC layer**	-	-	-	-	5 nodes
Prediction layer	3 classes	3 classes	3 classes	3 classes	3 classes
$\boldsymbol{\alpha_{label}}$	94.03%	92.50%	93.80%	94.33%	89.87%
$\boldsymbol{\alpha_{bit}}$	90.62%	89.24%	90.62%	91.73%	86.66%
$\boldsymbol{\mathcal{C}_{NC}}$	483.93	531.28	483.93	440.07	611.61
Training time (seconds)	$400s$	$140s$	$100s$	$60s$	$80s$

Type I + II diversity. For the type I + II diversity study, we randomly generate 100 models from a range of hyperparameter selection introduced in Table 8.2. From the resulted pool of classifiers, we pick out those with a high individual performance (*i.e.* $\alpha_{label} \geq 85\%$) such that their pairwise diversity measure (*i.e.* disagreement measure or κ-statistic) is maximized. The resulted architectures are details in Table F.2.

Table F.2: Architectures and performance related to the networks used for the type I + II diversity (models trained with the ranking loss)

Type I + II diversity	$\boldsymbol{Model_1}$	$\boldsymbol{Model_2}$	$\boldsymbol{Model_3}$	$\boldsymbol{Model_4}$	$\boldsymbol{Model_5}$
1^{st} **Conv. layer (+ BN)**	16 filters (size 11)	4 filters (size 5)	16 filters (size 11)	32 filters (size 21)	32 filters (size 1)
1^{st} **Pool. layer**	Max (stride 4)	Avg (stride 2)	Avg (stride 6)	Avg (stride 2)	Avg (stride 2)
2^{nd} **Conv. layer (+ BN)**	64 filters (size 1)	16 filters (size 21)	32 filters (size 11)	-	-
2^{nd} **Pool. layer**	Avg (stride 6)	Avg (stride 4)	Avg (stride 2)	-	-
3^{rd} **Conv. layer (+ BN)**	-	8 filters (size 5)	64 filters (size 43)	-	-
3^{rd} **Pool. layer**	-	Max (stride 2)	Max (stride 6)	-	-
4^{th} **Conv. layer (+ BN)**	-	-	32 filters (size 11)	-	-
4^{th} **Pool. layer**	-	-	Max (stride 6)	-	-
5^{th} **Conv. layer (+ BN)**	-	-	16 filters (size 21)	-	-
5^{th} **Pool. layer**	-	-	Avg (stride 4)	-	-
Flatten	Yes	Yes	Yes	Yes	Yes
1^{st} **FC layer**	8 nodes	-	-	32 nodes	-
2^{nd} **FC layer**	8 nodes	-	-	-	-
3^{rd} **FC layer**	16 nodes	-	-	-	-
Prediction layer	3 classes	3 classes	3 classes	3 classes	3 classes
$\boldsymbol{\alpha_{label}}$	90.04%	95.43%	94.53%	92.83%	93.83%
$\boldsymbol{\alpha_{bit}}$	83.16%	93.84%	93.01%	90.16%	89.79%
$\boldsymbol{\mathcal{C}_{NC}}$	704.45	358.84	393.24	500.08	512.73
Training time (seconds)	$940s$	$200s$	$400s$	$700s$	$540s$

Bagging and Boosting Hyperparameters Selection The hyperparameter selection is performed on *Random Forest* (RF) [Bre01] and *Convolutional Neural Networks* (CNN). Table F.3 (resp. Table F.4) identifies the ranges selected to configure the bagging (resp. boosting) models. Let N_{split} be the minimum number of samples required to split an internal node and the boostraping factor r denotes the number of side-channel traces n used to train a classifier (*i.e.* $n = r \cdot N_p$ with $N_p = 30,000$). For the Random Forest models, the nodes are expanded until all leaves contain less than N_{split} samples.

Table F.3: Range of hyperparameters selection for Bagging models

	Variables	**Values**
RF	**Bootstraping factor** (r)	$\{0.5, 0.8, 1.0\}$
	Objective function	Root Mean Square Error (RMSE)
	Number of trees	$\{5, 10, 25, 50, 100, 500, 1,000\}$
	N_{split}	$\{2, 3, 5, 10\}$
	Depth	until all leaves contain less than N_{split} samples
CNN	**Bootstraping factor** (r)	$\{0.5, 0.8, 1.0\}$
	Loss function	$\{CCE, RkL\}$
	Number of models	$\{1, 2, 3, 4, 5\}$
	Architecture	same as Subsection 8.4.1

Table F.4: Range of hyperparameters selection for XGBoost models

	Variables	Values
RF	**Objective function**	Root Mean Square Error (RMSE)
	Number of trees	$\{5, 10, 25, 50, 100, 500, 1,000\}$
	Learning rate	$\{10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1\}$
	Depth	until all leaves contain less than N_{split} samples
CNN-XGB	**Number of trees**	$\{5, 10, 25, 50, 100, 500, 1,000\}$
	Number of CNN	$\{1\}$
	Architecture	same as Subsection 8.4.1
	Learning rate	$\{10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1\}$