Variously supervised deep learning methods for side-channel attacks

Dorian Llavata

N° d'ordre NNT : 2025UJMO0064

THÈSE DE DOCTORAT
DE L'UNIVERSITÉ JEAN MONNET SAINT-ÉTIENNE

Membre de l'Université de Lyon

École Doctorale N° 488
SIS - Sciences Ingénierie Santé

Spécialité de doctorat : Informatique

Soutenue publiquement le 09/12/2025, par :
DORIAN LLAVATA

Variously Supervised Deep Learning Methods for Side-Channel Attacks

---

Devant le jury composé de :

Cécile Capponi, Professeure, Université d'Aix-Marseille	Présidente du Jury
Philippe Maurine, Professeur, Université Montpellier	Rapporteur
Christophe Clavier, Professeur, Université de Limoges	Rapporteur
Gabriel Zaid, Docteur, CryptoExperts	Examinateur
Lilian Bossuet, Professeur, Université Jean Monnet	Directeur de thèse
Rémi Eyraud, Maître de conférences HDR, Université Jean Monnet	Co-directeur de thèse
Eleonora Cagli, Docteure, CEA-Leti Grenoble	Co-encadrante de thèse
Benoît Gérard, Docteur, ANSSI	Invité

Acknowledgments

Je tiens à exprimer mes premiers remerciements à l'ensemble des membres de mon encadrement. En particulier, **Eleonora Cagli**, qui m'a accueilli et accompagné au quotidien au sein du CESTI du CEA-Leti. Sa bienveillance n'a d'égal que sa patience, qui a parfois dû être mise à l'épreuve, notamment face à mon enthousiasme débordant pour le deep learning. Ses conseils toujours avisés et sa disponibilité, aussi efficace que mesurée, m'ont permis d'orienter mes idées et de surmonter l'ensemble des obstacles rencontrés. De même, cette thèse ne se serait pas aussi bien déroulée sans les échanges stimulants, les idées inspirantes et les remarques constructives de l'équipe d'encadrement du laboratoire Hubert Curien, composée de mes deux directeurs de thèse, **Lilian Bossuet** et **Rémi Eyraud**, ainsi que de **Vincent Grosso**, co-encadrant de cette thèse.

Je me considère extrêmement chanceux d'avoir été encadré par vous tous. La confiance que vous m'avez accordée dans la direction de mes travaux a fortement contribué à mon épanouissement scientifique et personnel. Ce fut un immense plaisir de partager ces trois années à vos côtés. Merci infiniment.

J'adresse également ma sincère gratitude à **Philippe Maurine** et **Christophe Clavier** pour m'avoir fait l'honneur d'être les rapporteurs de ce travail, à **Gabriel Zaid** et **Benoît Gérard** pour avoir accepté d'être membres du jury de soutenance, ainsi qu'à **Cécile Capponi** pour en avoir assuré la présidence.

Merci pour toutes vos questions, vos remarques constructives et l'intérêt que vous avez porté à mes travaux. Grâce à la richesse de nos échanges, ce moment a été particulièrement agréable et restera un souvenir précieux, concluant de la plus belle des manières ce parcours doctoral.

Je souhaite également adresser mes sincères remerciements à l'ensemble de mes collègues du CEA-Leti. Merci pour les moments partagés au quotidien, les nombreuses discussions, qu'elles soient professionnelles ou informelles, notamment lors des nombreuses pauses café, qui ont souvent donné lieu à des échanges scientifiques enrichissants. Toutes ces interactions ont contribué à la qualité de ce travail tout en rendant ce parcours doctoral bien plus humain et agréable.

Mes derniers remerciements vont naturellement à ma famille, et tout particulièrement à mes parents. Les mots ne suffisent pas à exprimer toute ma gratitude pour votre amour inconditionnel, votre patience et vos encouragements tout au long de ce chemin parfois exigeant. Votre soutien indéfectible a été un repère précieux, une source essentielle de réconfort et de motivation, me permettant d'avancer avec sérénité et détermination. Cette thèse est aussi le reflet de la confiance profonde que vous avez toujours placée en moi, et je vous en dédie l'aboutissement avec une immense reconnaissance.

Abstract

The use of neural networks for profiling side-channel attacks on open systems is nowadays well established and has become an integral part of security evaluations. However, these techniques still exhibit notable weaknesses: they remain largely exploratory and costly in terms of hyperparameter tuning, making them difficult to deploy in time-constrained evaluation contexts, and the study of their applicability in non-profiling contexts remains limited. In this thesis, we explore strategies to make these approaches more practical and robust. Firstly, we show that ensemble learning through stacking can turn incomplete hyperparameter exploration into a well-performing architecture, offering a simple and cost-effective solution to ease the configuration of profiling attacks. Secondly, we investigate the use of siamese networks to conduct horizontal collision attacks in both profiling and non-profiling contexts. Among other aspects, we introduce two unsupervised learning methods using siamese networks to project pairs of traces into latent representations that are more prone to collision detection, without relying on systematic points of interest selection or realignment pre-processing. These contributions greatly simplify the practical implementation of horizontal collision attacks and reveal that vulnerabilities previously considered difficult to exploit can become easily exploitable, raising the need for the consideration of dedicated countermeasures.

Supervision and funds

This thesis results from a collaboration between the "CNRS Laboratoire Hubert Curien UMR 5516 F-42023", part of the University of Saint-Étienne (France) and the University of Lyon (France), and CEA-Leti ITSEF (Grenoble, France), and was financially supported by the Defense Innovation Agency (AID) of the French Ministry of Armed Forces.

Résumé en Français

L'utilisation de réseaux de neurones pour effectuer des attaques par canaux auxiliaires profilées sur des systèmes ouverts est aujourd'hui bien établie et fait partie intégrante des évaluations de sécurité. Toutefois, ces techniques présentent encore des faiblesses notables : elles restent largement exploratoires et coûteuses en matière de réglage des hyperparamètres, ce qui les rend difficiles à déployer dans des contextes d'évaluation soumis à des contraintes de temps, et l'étude de leur applicabilité dans des contextes d'attaques non profilées reste limitée. Dans cette thèse, nous avons exploré des stratégies visant à rendre ces approches plus pratiques et plus robustes. Tout d'abord, nous montrons que l'apprentissage ensembliste par stacking peut permettre à une exploration incomplète des hyperparamètres de se concrétiser en une architecture performante, offrant ainsi une solution simple et peu coûteuse pour faciliter la configuration des attaques profilées. Ensuite, nous étudions l'utilisation des réseaux siamois pour mener des attaques horizontales par collision dans des contextes profilés et non profilés. Entre autres aspects, nous introduisons deux méthodes d'apprentissage non supervisé utilisant des réseaux siamois pour projeter des paires de traces dans des représentations latentes plus propices à la détection des collisions, sans s'appuyer sur une sélection systématique des points d'intérêt ou un prétraitement de réalignement. Ces contributions simplifient considérablement la mise en œuvre pratique des attaques horizontales par collision et révèlent que des vulnérabilités auparavant considérées comme difficiles à exploiter peuvent devenir facilement exploitables, ce qui soulève la nécessité de prendre en considération des contre-mesures dédiées.

Supervision et financements

Cette thèse est le fruit d'une collaboration entre le Laboratoire Hubert Curien (CNRS UMR 5516, F-42023), rattaché à l'Université Jean Monnet de Saint-Étienne et à l'Université de Lyon, et le CESTI du CEA-Leti (Grenoble, France) et a été partiellement financée par l'Agence de l'Innovation de Défense (AID) du Ministère des Armées Français.

Contents

List of Figures

List of Tables

List of Abbreviations

AdaGrad *Adaptive Gradient*

Adam *Adaptive Moment Estimation*

AES *Advanced Encryption Standard*

ANSSI *Agence Nationale de la Sécurité des Systèmes d'Information*

BTCCA *Barlow Twins Collision Correlation Analysis*

CARDIS *Smart Card Research and Advanced Application Conference*

CCA *Canonical Correlation Analysis*

CC *Common Criteria*

CDF *Cumulative Distribution Function*

CEPACA *Correlation-Enhanced Power Analysis Collision Attack*

CiC *IACR Communications in Cryptology*

CMOS *Complementary Metal Oxide Semiconductor*

CNN *Convolutional Neural Network*

DCCA *Deep Canonical Correlation Analysis*

DDLA *Differential Deep Learning Analysis*

DES *Data Encryption Standard*

DLSCA *Deep Learning-based Side-Channel Analysis*

DPA *Differential Power Analysis*

DSA *Digital Signature Algorithm*

EAL *Evaluation Assurance Level*

ECC *Elliptic Curve Cryptography*

ECDSA *Elliptic Curve Digital Signature Algorithm*

EEG *Electroencephalogram*

ElU *Exponential Linear Unit*

EM *Electromagnetic Emissions*

FC *Fully-Connected*

FIPS *Federal Information Processing Standard*

FPGA *Field Programmable Gate Arrays*

GAN *Generative Adversarial Network*

GE *Guessing Entropy*

GelU *Gaussian Error Linear Unit*

GPU *Graphic Processing Unit*

HD *Hamming Distance*

HW *Hamming Weight*

ITSEF *Information Technology Security Evaluation Facilities*

LIM *Long Integer Multiplication*

LReLU *Leaky Rectified Linear Unit*

ML *Machine Learning*

MLP *Multi-Layer Perceptron*

MMM *Montgomery Modular Multiplication*

NIST *National Institute of Standards and Technology*

PCA *Principal Component Analysis*

PoI *Points of Interest*

PoNI *Points of No Interest*

PQC *Post Quantum Cryptography*

ReLU *Rectified Linear Unit*

RMSProp *Root Mean Square Propagation*

RSA *Rivest Shamir Adleman*

SCA *Side-Channel Attacks*

SeLU *Scaled Exponential Linear Unit*

SNN *Siamese Neural Network*

SNR *Signal-to-Noise Ratio*

SOG-IS *Senior Officials Group Information Systems Security*

SPA *Simple Power Analysis*

TCHES *IACR Transactions on Cryptographic Hardware and Embedded Systems*

Dedicated to my parents, who supported me every step of the way and never stopped believing in me. And to those brave enough to venture beyond Chapter 1...

Part I

State of the art

Chapter 1 The Phantom Menace

Side-Channel Attacks

The more I learn, the more I realize how much I don't know.

– Albert Einstein

Resume

In this first chapter, the context of side-channel attacks in the field of embedded cryptography is presented. In particular, a categorization of the main types of attack is proposed, along with the introduction of some metrics for assessing their feasibility and efficiency. An overview of the main countermeasures implemented by manufacturers to mitigate these threats is also provided. A particular focus is placed on unsupervised horizontal attacks, which are the main study of this work. The main contributions of this thesis are also detailed.

1.1 Security of Embedded Cryptography

Nowadays, embedded devices are deeply integrated into our daily lives, occurring in technologies such as banking and access control smart cards, electronic passports, smartphones, smartwatches, and many others. In many cases, these devices handle sensitive personal data, making them prime targets for attackers. Strengthening their security is therefore fundamental to ensure a trustworthy and secure operating environment, in the face of growing threats. In this context, cryptography plays a central role in providing essential security properties such as confidentiality, authentication, data integrity, non-repudiation and secure key exchange. All of these properties rely on robust cryptographic primitives, enabling the design and standardization of advanced algorithms and protocols with strong theoretical security guarantees. Among the most widely used standards are those for symmetrical cryptography, such as the Advanced Encryption Standard (AES) [Dwo+01], alongside those for asymmetrical cryptography, such as Rivest-Shamir-Adleman (RSA) [RSA78] and Elliptic Curve Cryptography (ECC) [Mil85]. In addition, new asymmetric cryptographic standards based on post-quantum cryptography problems, whose first standards were adopted in August 2024 by the National Institute of Standards and Technology (NIST), are about to be deployed in order to address the anticipated threat posed by quantum computers to classical algorithms. Although these developments are crucial for the future of secure systems, they fall outside the scope of this thesis, as the NIST standardization process was still ongoing at the beginning of this PhD work.

By now, the cryptographic algorithms mentioned above are considered secure from a cryptanalytic point of view. In other words, they offer computational security (excluding quantum computing [Sho97]) in a theoretical model where the attacker can only observe the inputs and outputs of the algorithm, without access to its internal computations. However, these algebraic guarantees alone are no longer sufficient to ensure the security of embedded systems. Indeed, theoretical proofs do not guarantee the practical security of their physical implementations, which may be vulnerable to the so-called physical attacks [Koc96], seeking to exploit weaknesses in hardware execution. Among them, side-channel attacks are particularly feared. They consist in exploiting physical leakage, such as the device's power consumption or electromagnetic emissions, in order to extract sensitive information relating to cryptographic operations. The emergence of these attacks profoundly impacted the cryptographic device manufacturing industry, which had to react by integrating dedicated countermeasures into their designs. Consequently, it became essential to guarantee the reliability of the security mechanisms claimed by manufacturers. For this purpose, these new attack vectors have been included in international security evaluation standards such as the common criteria (ISO/IEC 15408). These security evaluations involve rigorous tests performed by accredited laboratories, known as Information Technology Security Evaluation Facilities (ITSEF), which follow technical guidelines issued by national or international certification authorities. For example, in France, the Agence Nationale de la Sécurité des Systèmes d'Information (ANSSI) is responsible for delivering certification and validating the relevance of evaluations carried out by French ITSEF laboratories. The European Senior Officials Group Information Systems Security (SOG-IS) agreement aims to harmonize evaluation requirements and procedures, particularly for the cryptographic aspects of common criteria certifications. In the United States, comparable requirements are defined by the

Federal Information Processing Standard (FIPS 140-2), under the supervision of the NIST.

During a security evaluation, the purpose of the evaluator is to assess the threat as accurately as possible in order to quantify the effort required to exploit a vulnerability. A vulnerability whose exploitation requires a great amount of time, a high level of expertise, and significant financial resources will be deemed less critical for the device's security and will, therefore receive a higher quotation. This is why ITSEFs must constantly stay up to date with attack developments, as the discovery of new attacks can have a significant impact on quotations during evaluation. A good example of such an impact is the emergence of deep learning techniques [MPP16; CDP17], which allowed the development of more effective attacks while considerably reducing the complex pre-processing that was usually done by experts. Accordingly, these deep learning-based approaches are the subject of growing interest in both academic and industrial communities and are the focus of a continuing and intensive research effort [Bur+24; IUH25; ZL25]. This thesis is in line with this dynamic, exploring new attack strategies based on various deep learning methods, with the aim of simplifying their implementation and improving their efficiency.

1.2 Introduction to Side-Channel Attacks

Secure embedded devices using built-in cryptographic algorithms can be vulnerable to side-channel attacks (SCA). Unlike traditional cryptanalysis, which focuses on exploiting mathematical weaknesses in algorithms, SCA target their implementations by leveraging information extracted from physical quantities measured during the device operation. Indeed, once implemented on resource-constrained devices such as smart cards, cryptographic algorithms can unintentionally emit physical leakage during execution. A temporal sequence of leakage measurements, commonly referred as *traces*, can take various forms, including execution time [Koc96], power consumption [KJJ99] and electromagnetic emissions [GMO01], and are likely to be statistically dependent on the intermediate values manipulated during computation.

1.2.1 Notations

In this thesis, the calligraphic letter, *e.g.* $\mathcal{X}$ is used to denote sets. The uppercase letter X (resp. X_i when an indexation is needed) is used to denote random variables and the corresponding lowercase letter x (resp. x_i) to denote their realizations. Vectors are denoted with an arrow letter $\vec{X}$ whereas matrices are denoted with bold letters $\boldsymbol{X}$. The i-th entry of a vector $\vec{X}$ (resp. $\vec{x}$) is denoted by $\vec{X}[i]$ (resp. $\vec{x}[i]$).

1.2.2 Side-Channel Leakage

SCA rely on the idea that a detailed analysis of physical leakages can reveal information about intermediate computation steps involving sensitive variables, *i.e.*, variables that depend directly or indirectly on a secret parameter. In targeting these sensitive variables and

analyzing how they influence the observed leakages, an evaluator may recover some secret data, such as chunks of the secret key processed by the device. This opens the way for a divide-and-conquer strategy: rather than attempting to recover the entire key at once, the evaluator proceeds incrementally, extracting small chunks (*e.g.*, bytes or some bits), which are then combined to reconstruct the full key. This strategy significantly reduces the overall complexity of the key recovery process compared to classical algebraic cryptanalysis methods.

In the rest of this section, we briefly explain how leakage traces are collected, how leakage containing information about sensitive variables can manifest in them, and how an evaluator can verify the actual presence of such leakage.

Leakage Measurements. This thesis focuses on SCA that exploit power consumption or electromagnetic emissions (EM). To observe and collect these leakages, a dedicated measurement setup, called an acquisition bench, is required. Figure 1.1 represents a typical example of such a bench, typically comprising a computer used by the evaluator to control acquisition and record data, as well as a high-resolution digital oscilloscope equipped with various probes to capture the leakages emitted by the targeted device when it performs cryptographic operations.

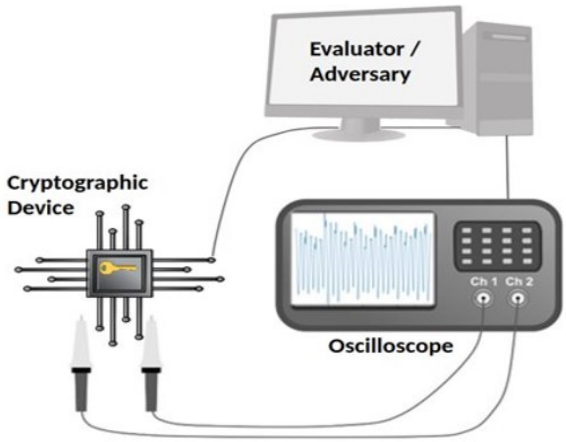


Figure 1.1: SCA bench.

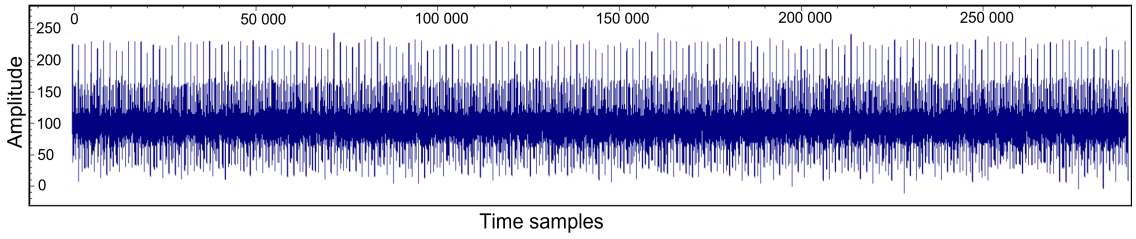
The leakage traces are stored on the computer in the form of temporal vectors and are considered as realizations of a random vector $\vec{T} \in \mathbb{R}^d$, where d denotes its dimension, *i.e.*, the number of temporal points it contains. For example, Figure 1.2a depicts an EM trace of a modular multiplication operation, which is part of an acquisition resulting from RSA decryption. During acquisition, each trace is associated with a value of the sensitive variable Z manipulated by the cryptographic device. The measured traces generally reflect all the leakage emitted by the device, which means that they depend not only on the targeted cryptographic function but also on other internal device activities. In this way, a trace $\vec{T}$ can be considered as a noisy observation of Z and can be modeled as the sum of two independent components, which can be expressed as:

$$\vec{T} = M(Z) + B, \tag{1.1}$$

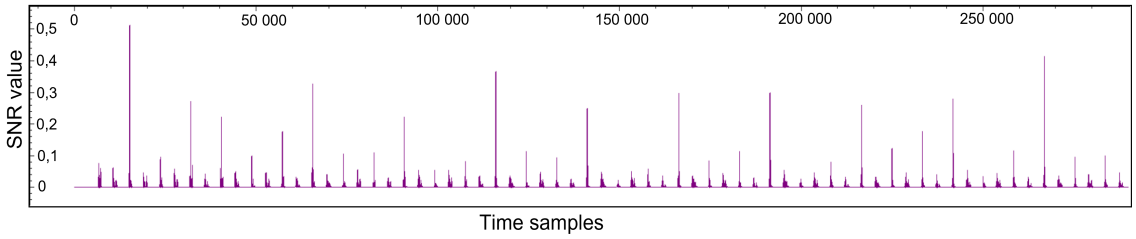
where $M(Z)$ represents the deterministic component related to the sensible variable Z and B denotes the random component independent of Z, representing noise (including measure-related disturbances or other irrelevant operations). It is common to assume that this noise follows a multivariate Gaussian distribution $\mathcal{N}(\vec{\mu}, \mathbf{\Sigma})$ with unknown mean $\vec{\mu}$ and standard deviation $\mathbf{\Sigma}$.

The leakage model $M(.)$ defines the relationship between the sensitive variables processed by the device and the observed leakage traces, *i.e.* the way in which this sensitive information manifests itself in the leakage traces. In general, it is unrealistic to assume that the values of the sensitive variable Z directly influence the leakage trace (as an identity leakage model would suggest). In practice, the actual leakage model remains unknown to the evaluator, which poses a major issue for side-channel analysis. Typically, the evaluator makes an assumption about the leakage model based on his prior knowledge of the cryptographic device and the hardware technology used. Classical leakage models are inspired by the behavior of CMOS circuits, where power consumption mainly depends on logic state transitions, such as going from "0" to "1" or conversely. Two widely used leakage models are the Hamming Weight (HW), representing the number of "1" bits in a value of the manipulated sensitive variable, and the Hamming Distance (HD) between two successive values, representing the number of bit changes between this two values.

During the evaluation process, the aim of the evaluator is to extract the deterministic component present in the leakage traces in order to recover the secret used by the cryptographic algorithm. In order to assess the presence of an informative leakage in the trace time points, the evaluator can rely on some leakage assessment techniques discussed in the following section.



(a) EM trace of a modular multiplication operation, *i.e.*, operation of the form $A \times B \bmod N$.



(b) Signal-to-Noise Ratio using 20 000 traces (Targeted value: 12th bit of the least significant 32-bit word in the left-hand operand (A).)

Figure 1.2: Example of an SCA trace and Signal-to-Noise Ratio.

Leakage Assessment. Leakage assessment aims to determine whether sensitive information can be revealed in leakage traces. Typically, this assessment is based on statistical

tests applied to a set of traces $\boldsymbol{T} \in \mathbb{R}^{n \times d}$, where n represents the number of traces and d the number of temporal points per trace. Each trace is associated with a known value of the target sensitive variable, called a label, denoted Y. The aim is to detect a significant statistical dependence between the set of traces $\boldsymbol{T}$ and the labels Y, in order to reveal any exploitable leakage.

A commonly used statistical test is the univariate signal-to-noise ratio (SNR) [MOP07]. It measures at each time point i of the set of traces (with i ranging from 1 to d), the ratio between the variance of the data related to the sensitive variable and the variance of the noise. In the context of SCA, SNR can be expressed as:

$$SNR[i] = \frac{\text{Var}_y(\mathbb{E}_i(\boldsymbol{T}[i]|Y = y)}{\mathbb{E}_y(\text{Var}_i(\boldsymbol{T}[i]|Y = y))}, \tag{1.2}$$

where $\mathbb{E}$ and Var denote respectively the esperance and variance. The numerator corresponds to the useful signal (the average per class), while the denominator provides an estimate of the noise variance (the intra-class variance). A high SNR value indicates the presence of a potentially exploitable leakage. Accordingly, the aim of the evaluator (or the attacker) is to optimize his attack setup in order to maximize the SNR, while the aim of the defender is to minimize it by burying the useful signal in noise (see Section 1.2.4). For example, Figure 1.2b depicts a mono-bit SNR during some modular multiplication operations, which are part of an RSA decryption acquisition.

Beyond the detection of leakage, this analysis also helps to reduce the dimensionality of traces, which are often very extensive and difficult to exploit. In practice, only some temporal points, called Points of Interest (PoI), contain relevant information related to the manipulation of the sensitive variable. Their identification makes it possible to focus on the moments when the leakage is most significant, thereby simplifying analysis and improving the efficiency of the attacks. Carefully identifying these PoI is therefore usually a crucial pre-processing step for most SCA.

Other leakage assessment methods exist, such as the Welch's T-test [GGJR+11]. A recent survey of statistical techniques for leakage detection and assessment can be found in [WT23]. Interestingly, a recent work explores the use of advanced statistical techniques based on deep learning to perform multivariate leakage assessments [MWM21].

1.2.3 Birth of a Threat: The seminal Attacks

Side-channel attacks first appeared in the literature in 1996, when Paul Kocher reported that the execution time of a cryptographic implementation could reveal information about the secret key involved, thus introducing the first timing attacks [Koc96]. Following this, Kocher *et al.* introduced two power analysis techniques: Simple Power Analysis (SPA) and Differential Power Analysis (DPA) [KJJ99].

The SPA is based on the direct observation of temporal variations in the physical leakage of a device, without the use of complex statistical tools. The aim is generally to identify, with the naked eye, characteristic patterns in leakage, depending on the operations performed

by the device. Indeed, different operations within a cryptographic algorithm can produce distinct leakage profiles (*e.g.* different shape or length). For example, the RSA algorithm requires a modular exponentiation (*i.e.* operation of the form $m^d \mod N$), whose execution flow may depend on the secret exponent d. A straightforward modular exponentiation is the binary *Square and Multiply* exponentiation algorithm (depicted in Algorithm 1), which processes the secret exponent bit by bit, performing a squaring operation at each iteration, followed by a conditional multiplication if the current bit is 1. In constrained devices, the squaring operation is often optimized and implemented differently from the multiplication, resulting in distinguishable execution times and leakage profiles. This difference makes it easier for an evaluator to visually identify which operation is being performed. In this way, by visually inspecting the leakage trace of a modular exponentiation and identifying the sequence of operations, an evaluator can potentially infer the values of the processed secret exponent bits (as depicted in Figure 1.3). It is worth noting that, similarly, some ECC algorithms such as Double and Add present a similar vulnerability.

Algorithm 1: Square and Multiply

Input: $m, N \in \mathbb{N}$,
$d = (d[k-1], d[k-2], \cdots, d[0])_2$
Output: $S = m^d \mod N$

1 $R_0 \leftarrow 1$
2 **for** $i = k-1$ ***to*** 0 **do**
  `// Square`
3 $R_0 \leftarrow R_0 \times R_0 \mod N$
4 **if** $d_i = 1$ **then**
    `// Multiply`
5 $R_0 \leftarrow R_0 \times m \mod N$
6 **end**
7 **end**
8 **return** R_0

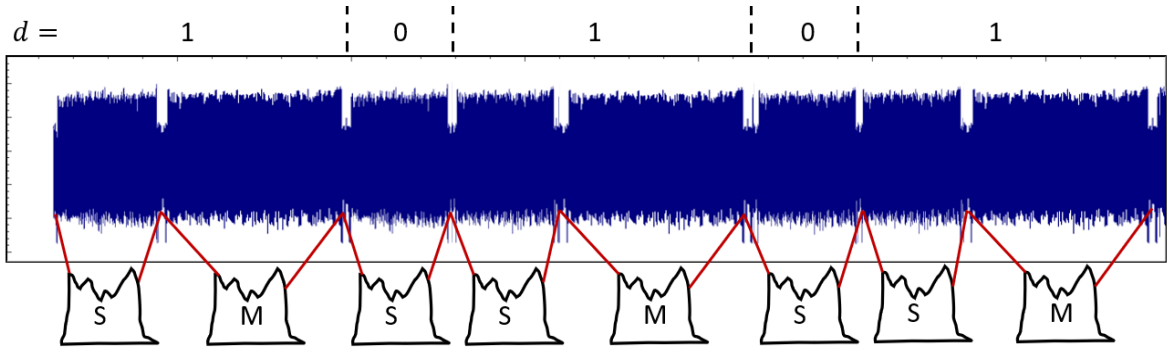


Figure 1.3: SPA on binary Square and Multiply exponentiation (Algorithm 1)

Nowadays, modern implementations are generally well protected against this kind of attack. A basic countermeasure, called *atomicity*, consists in using the same operation for both types of modular operation (*i.e.* processing the square as a multiply), thus making their

physical leakage indistinguishable [CCJ04]. Furthermore, it is usual to avoid conditional branching dependent on secret data, in favor of using so-called *regular algorithms*, whose execution flow remains independent of the manipulated secrets.

Common example of this type of regular exponentiation with constant operation sequences is the *Square and Multiply Always* with the addition of dummy multiplications (depicted in Algorithm 2) [Cor99]. This exponentiation always performs squaring and multiplication operations, regardless of bit values. For this purpose, when the current bit is 0 a dummy multiplication (which is useless in the computation) is performed which does not impact the exponentiation flow. This ensures a consistent operation sequence, making SPA inapplicable. Other regular algorithms may also be used such as the *Montgomery Ladder* [JY02] which uses interleaved variables. It should be noted that, such regular algorithms remain vulnerable to certain more advanced attacks, such as DPA, which involves statistical methods applied to a large number of traces to detect statistical dependence between secret data being processed and observed leakage. Additional countermeasures, such as exponent randomization, are generally implemented to protect RSA (for which a part of secret key may correspond to the exponent) against this kind of attack. The following section provides a more detailed discussion of countermeasures against side-channel attacks.

1.2.4 The Usual Countermeasures

Faced with the rise in side-channel attacks, protecting cryptographic implementations has become a priority. Thus, alongside research aimed at identifying new attacks, significant efforts are being dedicated to developing countermeasures to thwart them. Those countermeasures generally aim to introduce variability into leakage or make it independent of the processed sensitive data. They can be implemented at several levels:

- At the hardware level, countermeasures can be implemented to alter the physical behavior of the device, with the aim of reducing observable leakage.
- At the software level, countermeasures can be implemented by modifying the implementation of the cryptographic algorithm, *e.g.*, as mentioned in the previous section, by removing conditional branches in favor of regular algorithms, or by integrating masking techniques, which we will discuss later in this section.
- At the protocol level, countermeasures involving frequent renewal of the cryptographic key can be considered in order to limit its reuse.

The countermeasures can be algorithm-specific, such as atomicity or regularity to protect modular exponentiation, or more generic and applicable to several types of algorithms. There are two main types of countermeasures: those that hide sensitive data and those that mask it. In practice, these two approaches are often combined in protected implementations to make side-channel attacks as difficult as possible.

Hiding. This type of countermeasure does not modify sensitive variables, but aims to blurring their processing by reducing the statistical dependence between their values and the observed leakages, *i.e.*, in reducing the SNR.

A simple method consists of running additional operations in parallel that are independent of the sensitive variable at the moment it is being processed. The PoI associated with information leakage are thus blurred by irrelevant operations, making analysis more complex for the attacker. Another commonly used hiding method is to spread the processing of sensitive variables over different time periods, randomly changing the moment when they are processed. In this way, this approach leads to leakage traces that are misaligned with respect to their PoI. There are various approaches that introduce this temporal randomization: for example, at the software level, the addition of random delays through the random insertion of dummy instructions [CK09], or at the hardware level, the use of jitter induced by an unstable clock frequency [Moo+02].

In general, the misalignment of traces significantly reduces the efficiency of most side-channel attacks, forcing the evaluator to carry out a pre-processing step to realign them. These often rely on peak detection algorithms or advanced signal processing techniques, such as elastic alignment [WWB11].

Masking. This type of countermeasure aims to divide the sensitive variable Z into several parts, called shares, using random masks. The combination of these masks creates a masked representation of Z, and all operations are then performed not on the values of Z but on the shares, thus breaking the relationship between Z and the resulting leakage. The most commonly used masking schemes are Boolean masking [GP99] and Arithmetic masking [Cor99] which we describe below:

- *Boolean masking.* This kind of masking consists in manipulating a sensitive variable z by computing $z \oplus r$ where r is picked randomly and $\oplus$ is the XOR operation.
- *Arithmetic masking.* This kind of masking consists in manipulating a sensitive variable z by computing some form of arithmetic operation, *e.g.*, $z + r \bmod q$ where r is picked randomly and $+$ is the addition modulo q.

Both boolean and arithmetic masking can be applied at the first order or at a higher order N. An N-order masking involves the use of N masks during calculations, thereby enhancing robustness against sophisticated side-channel attacks. Indeed, in order to retrieve Z-dependent leakage, an attacker must find a sufficient number of shares, requiring them to identify, extract, and combine several portions of the leakage traces that manipulate these masks. However, this approach comes with significantly higher computational costs for the implementation.

Other masking methods exist, such as inner product masking [Bal+17] and affine masking [Fum+10]. In this thesis, the implementations studied are based on first-order masking techniques. For symmetric algorithms, Boolean masking is commonly used, while for asymmetric schemes, a form of arithmetic masking can be applied to sensitive intermediate values such as messages, exponents, or the modulus.

An essential distinction exists between the masking strategies employed in symmetric and asymmetric algorithms. For symmetric algorithms, masking is generally applied to sensitive intermediate variables, *e.g.* the outputs of substitution boxes in the case of AES. In this way, although masking is implemented, the key stays fixed from one execution to

the other. By contrast, in the context of asymmetric algorithms, masking can be applied directly to the secret key (*e.g* onto the secret exponent for RSA). In this way, the secret key is turned into a random one, differing at every execution of the implementation. This kind of masking considerably strengthens security against SCA, by making it virtually impossible to aggregate exploitable information from multiple observations. This distinction leads to the emergence of fundamentally different attack methodologies, which will be formalized in the next section.

1.2.5 Modern Attack Categories

Since Kocher's seminal work [Koc96], research in SCA has advanced rapidly, leading to a diverse range of attacks and numerous countermeasures developed to thwart them. Nowadays, SCA are commonly classified according to attacker models that define the capabilities, knowledge, and resources assumed to be available to an attacker exploiting a vulnerability.

This section does not aim to provide an exhaustive state-of-the-art review but rather to introduce the fundamental concepts behind modern attack categorization.

Attack Shape. Depending on the targeted implementation, different attack methods are available to the evaluator. A fundamental distinction lies in the type of leakage that can be exploited and the number of observations available per sensitive operation. This consideration has led to the development of two main strategies [Bau+13] which we describe below:

- *Vertical attacks.* This kind of attack seeks to exploit, using statistical methods, the variations observed between several traces from different executions of the implementation (usually by varying the input), but using the same fixed secret key. This scenario particularly concerns symmetrical algorithms, such as AES, for which the key remains fixed, regardless of the type of implemented countermeasures.

- *Horizontal attacks.* This kind of attack seeks to thoroughly exploit the information in a single trace (usually divided into several parts), resulting from a single execution of the implementation. Typically, this involves analyzing recurrent leakage from similar operations, based on the algebraic dependencies of several intermediate calculations of the algorithm. This scenario particularly concerns asymmetric algorithms handling an ephemeral key, such as DSA/ECDSA signatures or RSA/ECC implementations using a random ones [Cor99].

Nowadays, modern asymmetric implementations generally prevent vertical attacks, making it mandatory to resort to horizontal attacks, as will be discussed in more detail in Section 1.3. In contrast, vertical attacks remain the main approach for targeting symmetric implementations.

Attacker Knowledge. The attacker's knowledge of the targeted implementation can significantly influence the strategy adopted for the attack. Although it is generally assumed

in an evaluation context that the evaluator has access to the details of the implementation, some attacks require a particularly thorough understanding of the physical leakage behavior of the device. In practice, there are generally two main attack scenarios, which we describe below:

- *Profiling attacks.* In this kind of attack (*a.k.a.* supervised attacks), it is assumed that the evaluator has access to another device identical to the target one (called a clone device), over which it has partial or total control. This enables the use of chosen or known secrets and the observation of the corresponding sensitive variables. As part of the profiling phase, the evaluator collects a set of traces whose labels (corresponding to the values of the targeted sensitive variable) are known. This set of labeled traces enables accurate leakage assessment, as described in Section 1.2.2, and a supervised characterization of an empirical leakage model. Following this, the empirical leakage model can be used to evaluate the traces acquired from the real target, whose secret remains unknown. The issue of such a characterization may be, for example, an approximate of the trace distribution by a multivariate Gaussian distribution (template attacks [CRR02]) or a classification-task-inspired deep learning attacks [MPP16; CDP17].

- *Non-Profiling attacks.* In this kind of attack (*a.k.a.* unsupervised attacks), the evaluator must directly target the device under evaluation, whose access is limited. In general, only the ability to send inputs (*e.g.* plaintext) and collect corresponding outputs and leakage traces is available. There is no assumed knowledge regarding the secret, the values of sensitive variables, or the leakage model. Some non-profiling attacks rely on a hypothetical leakage model, such as DPA [KJJ99], where the evaluator assumes a specific leakage behavior (*e.g.*, Hamming weight). Others, like SPA (described in Section 1.2.3) or the unsupervised horizontal attacks discussed in Section 1.3 do not rely on any explicit leakage model, but instead exploit patterns or regularities within the traces in an unsupervised way.

In an evaluation context, profiling attacks provide a worst-case estimate of the security risk, based on the most powerful attacker model. Nevertheless, several profiling scenarios are possible. As an example, it is not unusual for an evaluator to have only partial access to the profiling device. In this case, the evaluator may be able to set the secret used by the implementation, but may not be able to deactivate some of the built-in software or hardware countermeasures, thereby considerably restricting, or even making impossible, the profiling phase. In the case of asymmetric implementations, these limitations are particularly problematic. For instance, if the implementation includes an exponent masking countermeasure [Cor99], the evaluator must be able to disable it, or at least access the random values used during execution to carry out profiling. However, in practice, such access is often unrealistic, as these values are generated securely and remain inaccessible, even to the evaluator. This is particularly the case in a composite evaluation context, where the target under evaluation use of a pre-evaluated secure component that has been previously certified and is not part of the evaluation scope.

1.3 Horizontal Attacks

A well-known example of a horizontal attack, previously introduced, is the SPA, representing its most elementary form. Another representative technique, applicable in the absence of robust countermeasures, is the chosen message collision attack, such as the doubling attack [FV03]. The latter exploits the leakage horizontally by searching for collisions between two related executions. However, as it requires the acquisition of leakage from two distinct executions of the implementation, it is effectively thwarted by the implementation of countermeasures designed to prevent vertical attacks.

Nowadays, modern asymmetric implementations of RSA and ECC-based algorithms are generally well protected against most side-channel attacks. Notably, masking countermeasures that randomize the secret key usually prevent both vertical and profiling attacks [Cor99]. Furthermore, elementary single-trace attacks, such as the SPA, can be effectively thwarted thanks to the well-established atomicity principle [CCJ04] and the use of regular algorithms [Cor99; JY02]. Consequently, in many evaluation scenarios targeting asymmetric implementations, only unsupervised single-trace horizontal attacks remain practicable. For the sake of clarity, we will refer to these attacks simply as *horizontal attacks* in the remainder of this section.

1.3.1 Horizontal Attacks Categories

Among the horizontal attacks presented in the literature, two main approaches can be distinguished. One is *horizontal collision attack* [Wal01; Cla+10; Cla+12], which exploit leakage resulting from the manipulation of the same operand during two computationally expensive operations (*e.g.*, long integer multiplications), in order to compare them and identify similarities. On the other hand, *horizontal clustering attack* [Hey+13; Spe+15; PC15], which rely on leakage resulting from directly bit-dependent operations (*e.g.*, the bit reading itself), in order to distinguish these operations by regrouping them according to their leakage behavior.

Clustering Attacks. In [Hey+13], Heyszl *et al.* are the first to propose the use of clustering algorithms in horizontal side-channel attacks targeting an ECC implementation. The exploited leakage stemmed from conditional register accesses, depending on the value of the key bit being processed. The authors applied a multidimensional K-means algorithm to automatically identify similar patterns in the traces, grouping them into two clusters (corresponding to bits “0” and “1” respectively). In a subsequent work, Specht *et al.* [Spe+15] enhanced this approach by incorporating dimension reduction via Principal Component Analysis (PCA), and investigating the use of the expectation maximization algorithm as an alternative to K-means. Although these methods can be effective in low-noise environments, their performance drops significantly in the presence of high noise, making discrimination between clusters more challenging. To address this issue, the authors of both works proposed the use of an acquisition bench equipped with multiple probes, to amplify the measured leakage in noisier contexts.

Subsequently, another clustering-based approach was proposed, aiming in this case at identifying PoI rather than directly exploiting multivariate leakage [Per+14; PC15; NC17; COM23]. In these works, the authors first use clustering algorithms (or other heuristics) to perform a univariate analysis on each temporal point in the traces, with the aim to identify those that exhibit statistically discriminative behavior. Then, a multivariate clustering attack is performed on this reduced subset, allowing the focus on the most informative points while limiting the impact of noise. In this kind of univariate analysis, the quality of the traces has a major influence on the reliability of PoI identification. Indeed, measurement disturbances, such as extreme values or outliers, can strongly affect the results of univariate clustering. To address this issue, Cler *et al.* proposed a pre-processing approach based on unsupervised deep learning [COM25]. In addition, post-processing methods have been proposed to fine-tune the estimate of bits after the clustering attack, based on the use of unsupervised deep learning, as described in [Per+21; BA23]. However, the effectiveness of this post-processing remains highly dependent on the percentage of bits properly estimated during the initial attack.

Collision Attacks. One common strategy in horizontal attacks is the search for collisions. This type of attack employs an analogous methodology to the chosen-message collision attack [FV03; Hom+08]. Instead of looking for collisions between two related executions of exponentiation, the horizontal collision attack aims to determine whether or not two distinct modular operations performed at different times during the same execution share common operands. It should be noted, as mentioned earlier, that chosen-message collision attacks are not effective in presence of countermeasures such as masking, while the horizontal collision attacks can be.

These attacks are based on the fact that each modular operation involves multiplications of very large integers. For example, in the case of RSA, these are integers of $1\,024$, $2\,048$ or $4\,096$ bits. In an embedded device, such multiplications are performed using a Long Integer Multiplication (LIM) algorithm, which performs multiplication by repeatedly calling an internal multiplier operating on t-bit words. Thus the evaluator can exploit the leakage resulting from these elementary multiplications on t-bit words, and recover substantially more information from a single modular operation. For instance, the *Schoolbook* method (depicted in Algorithm 3) represents the most straightforward way to realize a long integer multiplication. Let x and y be long integers of the same size and their respective decompositions are $x = (x[l-1], x[l-2], \cdots, x[0])_b$ and $y = (y[l-1], y[l-2], \cdots, y[0])_b$ in base $b = 2^t$ with $l = \lceil \log_b(x) \rceil$. The product $x \times y$ is performed by the algorithm with a nested-loop structure and performs l^2 inner-products on t-bit integers $x[i] \times y[j]$. Thus, this algorithm gives l^2 intermediate results on $2t$-bits words. Other more advanced LIM algorithms may also be used such as Comba [Com90], Karatsuba [Kar63], and Montgomery multiplication [Mon85]. For a detailed analysis of the different LIM, the reader may refer to [RMH17].

Walter presents the first horizontal collision attack known as the *Big Mac* attack [Wal01], which exploit leakage information coming from each of the elementary operations involved in the LIM. This attack aims to compare the modular operation traces with a so-called template trace, for which the message handling is known. In order to perform these comparisons accurately using statistical distinguishers (*e.g.* euclidean distance or correlation), the

Algorithm 2: Square-and-Multiply Always

Input: $m, n \in \mathbb{N}, m < n$,
$d = (d[k-1], d[k-2], \cdots, d[0])_2$
Output: $m^d \mod n$

1 $R_0 \leftarrow 1$
2 $R_1 \leftarrow 1$
3 **for** $i = k-1$ ***to*** 0 **do**
  // Square $R_1 \times R_1$
4 $R_1 \leftarrow \text{LIM}(R_1, R_1) \mod n$
  // Multiply $R_1 \times m$
5 $R_{d_i} \leftarrow \text{LIM}(R_1, m) \mod n$
6 **end**
7 **return** R_1

Algorithm 3: Schoolbook Long Integer Multiplication (LIM)

Input: x, y
$x = (x[l-1], x[l-2], \cdots, x[0])_b$
$y = (y[l-1], y[l-2], \cdots, y[0])_b$
Output: $x \times y = w$
$= (w[2l-1], w[2l-2], \cdots, w[0])_b$

1 $w \leftarrow (00 \cdots 0)$
2 **for** $i = 0$ ***to*** $l-1$ **do**
3 $c \leftarrow 0$
4 **for** $j = 0$ ***to*** $l-1$ **do**
5 $(uv)_b \leftarrow w[i+j] + x[i] \times y[j] + c$
6 $w[i+j] \leftarrow v$
7 $c \leftarrow u$
8 **end**
9 $w[i+l] \leftarrow c$
10 **end**
11 **return** w

proposed approach involves signal pre-processing, which consists in averaging the traces of modular operations according to one of the operands of the long integer multiplication. This pre-processing improves collision detection and makes it easier to distinguish between, for instance, square and multiplication operations, even in the presence of an atomicity countermeasure. Clavier *et al.* [Cla+10] extend this work with the so-called horizontal correlation analysis, which exploits correlations between intermediate values and modular operation traces (assuming knowledge of the message). Subsequent works have proposed other horizontal collision attacks [WWM11; HKT15; SSS15] some of them targeting regular algorithm implementations.

As an example, the square and multiply always exponentiation (depicted in Algorithm 2), although designed to be regular and resistant against various side-channel attacks, has an intrinsic vulnerability that may be exploited through a horizontal collision attack. Indeed, Algorithm 2 always performs squaring and multiplication operations, regardless of bit values. For this purpose, when the current exponent bit d_i treated at loop index i is 0 (line 5 of Algorithm 2), a dummy multiplication is performed and stored in a register dedicated to dummy values (R_0), which does not impact the exponentiation flow stored in the valid register (R_1). This implies that the subsequent squaring operation for the processing of the next exponent bit d_{i-1} shares an operand (the left-hand one) with this dummy multiplication since the algorithm will manipulate the same value of $R1$. At the opposite, if the current exponent bit d_i is 1, then the multiplication and the subsequent squaring are not likely to share the same left-hand operand since the result of one is the entry of the other one. Therefore, by detecting these collisions (as depicted in Figure 1.4), the evaluator can infer all bits of the secret exponent except the last one.

In the context of a profiling attack, Carbone *et al.* [Car+19] realized this attack scheme with a deep learning technique against an RSA implementation running on a certified arithmetic co-processor and equipped with full masking countermeasures *i.e.*, masking of the message, masking of the exponent and masking of the modulus, as we will present in Section 4.2.3. Concerning the unsupervised ones, early works explored this attack path using cross-correlation techniques [WWM11; HKT15; SSS15].

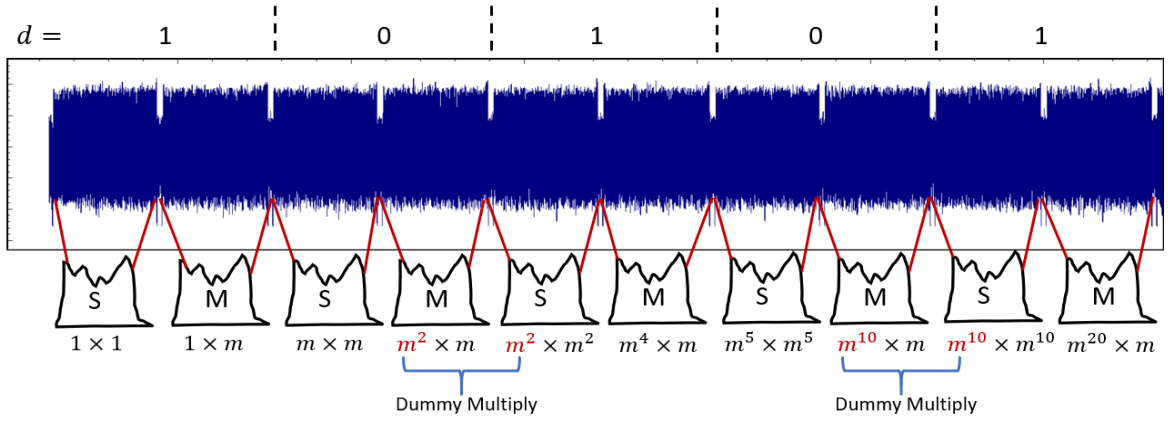


Figure 1.4: Collision attack on binary square and multiply always exponentiation (Algorithm 2)

Witteman *et al.* [WWM11] are the first to propose the use of correlation to exploit the collision of operands between two consecutive operations (multiply, square). However, their attack requires several exponentiation traces to work. This attack was generalized to use a single exponentiation trace by Hanley *et al.* [HKT15], and then improved by Sugawara *et al.* [SSS15] with the integration of the Big Mac signal pre-processing which we describe below.

Let $M = R1 \times m$ be a modular multiplication and let $S = R1 \times R1$ be the following modular squaring operation in Algorithm 2. Both include long integer multiplication in the form $X \times Y$ and their partial products $x[i] \times y[j]$ (line 5 of Algorithm 3). The leakage of each $x[i] \times y[j]$ is denoted by $t_{i,j}$. To highlight the left-hand operand and improve collision detection, the modular operation trace is compressed into l-dimensional vectors $\vec{t_x}$ such that:

$$\vec{t_x}[i] = \frac{1}{l} \sum_{j=0}^{l-1} t_{i,j}. \tag{1.3}$$

The averaged trace $\vec{t_x} = [\vec{t_x}[0], \ldots, \vec{t_x}[l-1]]$ is the concatenation of each compressed segment trace. The averaged traces of multiplication and squaring operations, denoted by $\vec{t_x}$ and $\vec{t'_x}$, are compared with correlation distinguisher (as depicted in Figure 1.5). The correlation coefficient is assumed to be high if there are collisions. In this way, by analyzing the resulting correlation coefficients for each pair of modular operation traces, a threshold can be defined at which a collision is considered to occur (as depicted in Figure 1.5).

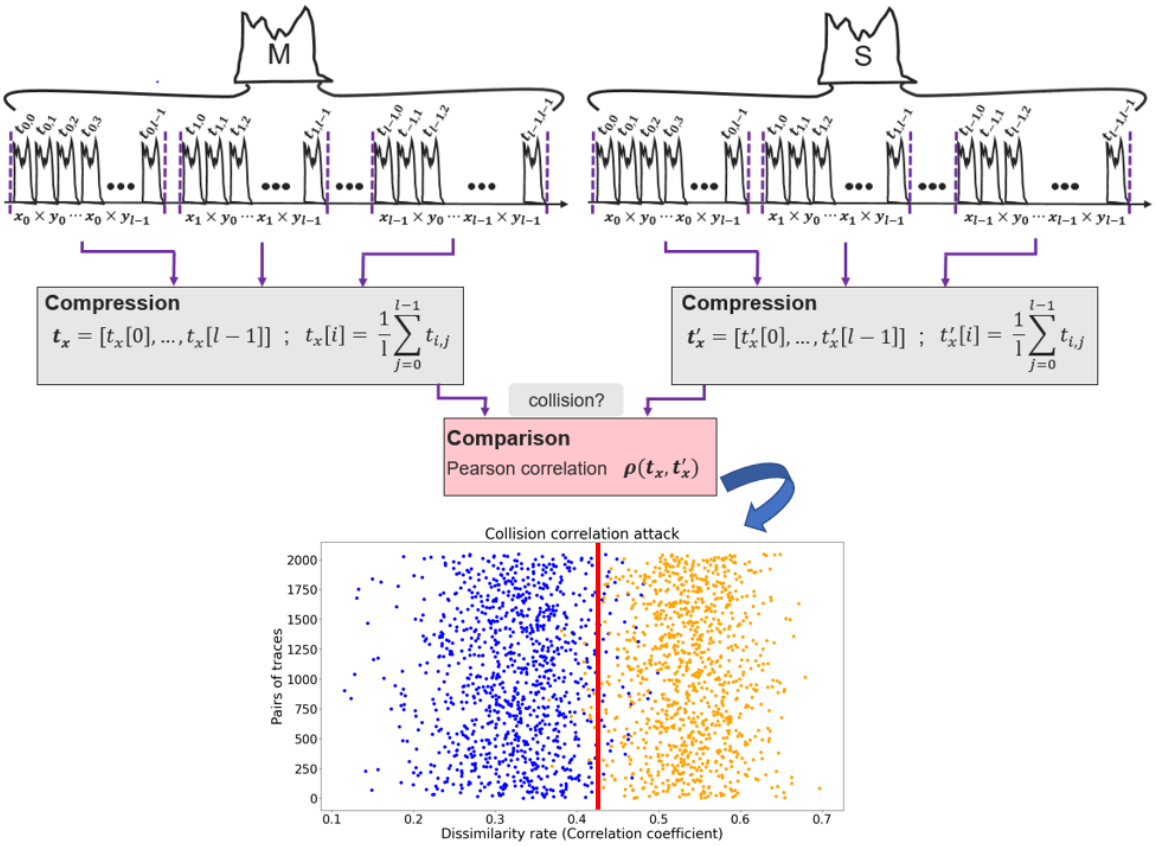


Figure 1.5: Collision correlation attack with Big Mac pre-processing between pair of modular operation traces (Multiply/Square). The correlation coefficients are colored with their ground-truth labels. Blue (resp. orange) values represent pairs of modular operation traces without collision (resp. with collision).

1.3.2 Limits of the Horizontal Attacks

Horizontal attacks are theoretically effective approaches, but their implementation remains a major issue, whose effectiveness heavily depends on the prior success of many critical pre-processing steps. Firstly, to the best of our knowledge, all horizontal attacks in the literature assume that trace cutting and realignment have been perfectly performed, which is a strong assumption for practical use in evaluations. Furthermore, an accurate extraction of PoI is mandatory, as a high proportion of non-informative points can compromise the success of both clustering and collision attacks. In particular, the feasibility of a horizontal collision attack, which exploits leakage information from LIM multiplier, depends heavily on the assumption that the evaluator is able to extract these leakages from a raw trace. For this purpose, the evaluator must have specific informations, such as detailed knowledge of the implementation of modular multiplication. Furthermore, even with this knowledge, performing this analysis in a non-profiling context is challenging, as the leakage assessment techniques traditionally used, such as SNR, cannot generally be applied. Instead, the evaluator has to rely upon less robust techniques which can be heavily impacted by the presence of significant noise or some measurement disturbances [COM25].

For instance, Sugawara *et al.* [SSS15] performed unsupervised leakage assessment using a correlation matrix obtained by applying Witteman *et al.* collision correlation attack [WWM11]. However, the authors had to disable the masking countermeasure in order to exploit many exponentiation traces. When such a scenario is not feasible, unsupervised selection of PoI is currently mainly performed by projecting the modular operation traces into a lower-dimensional subspace using linear transformations such as PCA [Spe+15] or univariate analysis with some heuristic algorithms [Per+14; COM23]. Applying these methods to highly dimensional data is also a challenge. As a result, these methods may not be applicable in some evaluation scenarios, as the typical length of a modular operation trace is often very high.

An additional challenge lies in the fact that, even when all the aforementioned preprocessing steps are correctly performed, the success of these attacks still relies on recovering nearly all bits of the secret exponent with a very low error rate, in order to keep the brute-force of the remaining uncertain bits computationally feasible. This requirement is particularly crucial for large keys, such as those commonly used in RSA. Due to these practical constraints, none of the above-mentioned horizontal attacks has proven effective against an implementation running on a defensive arithmetic co-processor such as the protected RSA dataset described in Section 4.2.2 and introduced in [Car+19].

As a result, horizontal attacks are often considered impractical by developers, and the importance of equipping implementations with dedicated countermeasures such as those proposed in [Bau+13] is largely underestimated.

1.4 Efficiency Metrics

In order to measure the robustness of a cryptographic implementation against side-channel attacks, the evaluator must assess the difficulty of recovering the key. For this purpose, several metrics have been proposed in the literature. This section focuses on the presentation of the metrics that are directly used in the contributions developed in this thesis. Notably, in this manuscript, the definition of a successful attack depends on both the cryptographic algorithm and the attack scenario under consideration (*i.e.* vertical or horizontal).

1.4.1 Efficiency of Horizontal SCA

In a horizontal attack targeting an asymmetric implementation (*e.g.* RSA-based exponentiation), the main objective is to recover as many bits of the secret exponent as possible from a single exponentiation trace. Therefore, to assess the performance of this kind of attack, the *accuracy* metric, commonly used in machine learning applications, is a well suited approach. In this context, this metric provides a precise estimate of the percentage of key bits successfully retrieved through a given attack. Nevertheless, it is unusual for a horizontal attack to succeed in recovering all of the bits of the secret key. Indeed, some parts of the key may remain unknown or uncertain, and the missing bits must be recovered by some remaining operation (*e.g.* brute-force). This residual complexity needs to be estimated, in order to assess whether the attack can be carried out in a reasonable amount of time.

The assessment of the complexity of the remaining operations following a horizontal attack remains an open issue in literature, lacking a clear consensus either in academic research or among certification bodies. In this regard, the European SOG-IS scheme proposes a measure of time complexity, defined as the amount of offline calculations required to complete the key recovery attack, *i.e.* the number of invocations of the cryptographic algorithm under consideration (DES, AES, modular exponentiation, scalar multiplication, etc.). However, this measure does not take into account the actual runtime of each invocation, nor the computing power available to the evaluator. According to this scheme, a maximum brute-force complexity of around 2^{100} operations is considered to be practical [SI+23]. It is also worth noting that, under the French scheme, ANSSI considers a time complexity of 2^{70} or less to be negligible [AWG+19]. Accordingly, in this manuscript, we adopt the 2^{100} threshold in terms of time complexity to assess the feasibility of an attack targeting asymmetric cryptographic implementations.

In a recent work, Zaid *et al.* [Zai+21a] experimentally evaluate how the accuracy impacts the final attack complexity and find that increasing the accuracy of a side-channel attack is crucial in order to reduce the remaining operations required for a successful key recovery. Following the same approach as the authors, in this manuscript we define the brute-force complexity for N remaining operations as $\log_2(N)$, and uses two distinct remaining complexity metrics introduced in [Zai+21a]: the naive remaining operation complexity (C_{NC}) and the 2^n remaining operation complexity (C_{2^n}), both described below.

- C_{2^n}. In the case where the evaluator has recovered K bits of the secret exponent and the total size of the exponent is N bits. Assuming he perfectly knows each position of the uncertain bits, the evaluator only needs to explore the remaining brute-force complexity for the $N - K$ bits, which can be expressed as:

$$C_{2^n} = \log_2\left(2^{N-K}\right).$$

  Note that, this scenario represents the most optimistic case for the evaluator, and is unrealistic in regard to the actual knowledge an attacker could have.

- C_{NC}. In the case where the evaluator has no knowledge of the position of the uncertain bits, he must exhaustively try all possible combination of wrong assumptions, which considerably increases the search space and makes the attack quotation far more costly. This complexity can be expressed as:

$$C_{NC} = \log_2\left(\sum_{i=0}^{N-K}\binom{N}{i}\right). \tag{1.4}$$

  Note that, this scenario which represents a sub-optimal strategy for the evaluator, actually comes closer to the point of view of an attacker.

Considering these two metrics provides a valuable comparison of different evaluation strategies and offers a broader perspective on the security of the targeted implementation. Notably, in this manuscript, we treat the 2^n-complexity as a lower bound and the naive

complexity as an upper bound for attack quotation. Note that, in practice post-analysis techniques for key recovery from partial information may enable a potential attacker to position their attack somewhere within these two bounds. These may include techniques based on lattice reduction [MH24] or key enumeration algorithms [PSG16]. However, such approaches are out of the scope of this thesis.

1.4.2 Efficiency of Vertical SCA

In the case of a vertical attack, where the evaluator can exploit multiple traces, the accuracy metric is not sufficient to correctly assess the effectiveness of the attack. In this case, specific metrics related to side-channel attacks should be used, such as the success rate or the empirical Guessing Entropy (GE) [SMY09]. The latter is the most commonly used, as it allows to estimate, once the attack has been carried out, the average number of keys that need to be tested before identifying the correct one.

Typically, an attack which used N traces outputs a sorted logarithmic vector of key candidates, called the rank vector $\vec{g} = (\vec{g}[0], \vec{g}[1], ..., \vec{g}[C])$. In the vector, $\vec{g}[0]$ is considered the most likely key hypothesis and $\vec{g}[C]$ as the least likely. The position of the good key k^* in the vector is called the rank of the key, which can be expressed as:

$$\text{Rank}(k^*, N) = \text{i such that } \vec{g}[i] = k^* \tag{1.5}$$

The guessing entropy [SMY09] is defined as the expected rank of the correct key, which can be expressed as:

$$\text{GE}(k^*, N) = \mathbb{E}[\text{Rank}(k^*, N)] \tag{1.6}$$

This metric is estimated empirically, by performing the attack several times from different subsets of N traces. As usual in the literature, in this thesis we randomly select subsets of all available attack traces and set the number of attacks to 100.

Another metric commonly used in the literature, derived from GE, corresponds to the number of traces generally noted N_a required for an attack to be considered successful. As expressed below an attack is considered successful using N_a traces if the GE is stably equal to 1.

$$N_a = min(N | GE_Q(k^*, N) = 1) \tag{1.7}$$

1.5 Contributions

This section summarizes the contributions presented in this thesis and outlines the overall structure of the manuscript. In this thesis, we explored several deep learning paradigms

to simplify the implementation of side-channel attacks, covering both supervised and unsupervised approaches. Among these, we paid particular attention to fully unsupervised method for addressing non-profiling horizontal collision attacks, which represent an essential scenario for assessing the practical security of asymmetric implementations.

To introduce these contributions, Chapter 2 provides the foundations of deep learning and its application to side-channel analysis.

Deep Ensemble Learning for Hyperparameterization. Nowadays, Deep Learning is widely used in profiling side-channel attacks. It reduces the need for pre-processing, such as the realignment of traces or the manual selection of points of interest. However, as a paradox, deep learning-based attacks are highly complex to implement, because of the large number of hyperparameters that need to be configured (see Section 2.2.4). In many cases, this hyperparameterization requires a considerable human effort, or significant computational resources when a random search or an algorithm-based one (*e.g.* bayesian or reinforcement learning optimization) is involved. Another approach proposed to mitigate this hyperparameterization issue is the use of Ensemble Learning. It consists in using multiple neural networks in a single predictive task and aggregating the several predictions in an opportune way. The intuition behind is to use the power of numbers to build a powerful model with an ensemble of individually weak models, whose hyperparameterization is slight or even random. In this way, an evaluator may use a posteriori ensemble learning for combining already trained models, in order to improve their attack by combining them. In the SCA context, Perin *et al.* [PCP20] proposed the use of bagging ensemble, which consists of aggregating the weak model predictions by an averaging process. In Chapter 3, we extend these preliminary results and propose the use of stacking as an aggregation method, in which a meta-model is trained to learn the best way to combine the weak model ensemble predictions. Our experimental exploration on several commonly used AES implementations highlights some of the limitations of the bagging process and shows that stacking can significantly improve attack performance while providing a flexible solution to address these limitations. Notably, we experimentally observed that staking ensemble can provide with less effort attack performances similar to those of rigorously hyperparameterized architectures proposed in the litterature. Hence confirming that stacking ensemble may be considered as a suitable approach to avoid the need for the security evaluator to finely tune the neural network architecture.

Siamese Neural Networks for Horizontal Collision Attacks. State-of-the-art non-profiling horizontal attacks are extremely difficult to implement in practice, as they often require advanced trace pre-processing, including realignment and the selection of points of interest. Few studies have explored the potential of unsupervised deep learning to address these challenges [Per+21; LHK22; COM25], and those exploring it have mainly focused on horizontal clustering attacks. To the best of our knowledge, no previous work has attempted to simplify the pre-processing required for collision-based attacks using unsupervised deep learning. Chapter 4 of this manuscript aim to address this gap. Inspired by the profiling deep learning attack on operand manipulation present in [Car+19], and the design of traditional non-profiling horizontal collision attacks such as the Big Mac [Wal01], we investigated several deep learning methodologies by redefining the operand collision detection between pairs of traces as a verification task that can be processed using siamese neural

networks. For this purpose, we bridged traditional non-profiling collision attacks based on cross-correlation and siamese networks, which have the ability to automatically extract common characteristics from their inputs in a latent space, *i.e.* new trace representation using neural network mapping. In this respect, we introduced in the side-channel context two methods for training unsupervised siamese networks to pre-process pairs of traces, projecting them into highly correlated latent representations, that are more prone to collision detection, without relying on systematic points of interest selection or realignment pre-processing. Our first approach, called DCCA, maximizes the correlation between pairs of traces using canonical correlation analysis. This approach considerably simplifies trace pre-processing, making it possible to exploit high-dimensional and noisy traces without the need for any manual selection of points of interest. Nevertheless, this attack has a number of practical limitations. In particular, its performance depends heavily on the random initialization of DCCA model weights, leading to unstable results. In addition, DCCA performs poorly in the presence of misalignments in the traces. To mitigate this, we equipped DCCA with a weight-initialization search framework that runs multiple random restarts and selects the most promising attacks in an unsupervised way, enabling reliable and complete attacks even if many of them may fail individually. Secondly, we go beyond the limitations of DCCA by proposing BTCCA, that tailors the Barlow Twins learning paradigm [Zbo+21] to the collision attack context. This new approach simplifies the attack process by improving stability and reducing the need for prior trace realignment. Several experimental results on the protected RSA implementation presented in Section 4.2.2 running on a defensive arithmetic coprocessor with up-to-date countermeasures, show how our proposals outperform state-of-the-art attacks. Following the SOG-IS guidances, the improvement in remaining complexity provided by DCCA/BTCCA attack should lead to a reconsideration of the security level of the targeted system, raising the need to implement dedicated countermeasures against horizontal attacks. Notably, BTCCA significantly improves previous works by being simpler to implement and by extending the portability of horizontal collision attacks to evaluation contexts where realignment methods are either impractical or not satisfying. Extremely simple to implement, both DCCA and BTCCA represents a valuable addition to the evaluator's toolbox for assessing the security of asymmetric implementations, complementing existing methods.

Chapter 2 The Rise of DLSCA

Deep Learning for Side-Channel Attacks

All models are wrong, but some are useful.

– George Box

Resume

In this chapter, the fundamental principles of deep learning are presented, including the training process of neural networks, the design of their architectures, and common regularization techniques used to improve generalization. Attention is then given to approaches based on unsupervised deep learning. The chapter concludes with an overview of the application of deep learning in cybersecurity, highlighting recent contributions in the context of side-channel attacks, both in vertical and horizontal scenarios, and in profiling as well as non-profiling settings.

2.1 Machine Learning Foundation

As early as 1959, Arthur Samuel [Sam59] defined the Machine Learning (ML) as a research field that gives machines the ability to learn without being explicitly programmed. A few decades later, Tom Mitchell [MCM83] proposed a more formal definition, stating that a machine learns when its performance on a certain task improves with new experiences. Both of these definitions highlight the main idea behind ML: the creation of algorithms that do not simply follow predefined instructions, but improve performance by using the experience gains from data. More specifically, during the learning process, ML algorithms aim to build a mathematical function, referred to as a *model*, that captures the underlying structure of the data in order to solve a predictive/generative task. The learning process can be divided into several paradigms, which are mainly distinguished by the nature of the predictive task and the way in which the ML algorithm exploits the available data. Among these, we can particularly distinguish between *supervised learning* and *unsupervised learning*.

Supervised Learning. This learning paradigm relies on the use of labeled data, *i.e.*, a set of examples for which the expected output is known. The aim of such an approach is to learn a mapping function between inputs and corresponding labels, in order to generalize this correspondence to new data. There are mainly three categories of supervised learning tasks:

- *Regression.* This task consists in predicting a continuous value from input data.
- *Classification.* This task consists in predicting a discrete value (*i.e.*, a class) from input data. Classification task can be either binary (involving two possible classes) or multi-class (involving more than two distinct classes).
- *Verification.* This task consists in comparing two distinct input data to determine whether or not they belong to a same class.

Unsupervised Learning. This learning paradigm does not rely on the use of labeled examples. The goal of such an approach is to discover intrinsic structures in the data without the use of direct supervision to perform the desired predictive task. There are two main categories of unsupervised learning tasks:

- *Clustering.* This task is an unsupervised classification that involves grouping together similar data in order to reveal implicit categories in them.
- *Dimensionality reduction* This task involves projecting data into a lower-dimensional space while preserving the most relevant information, which makes it easier to analyze the data and use them for subsequent learning tasks (*e.g.* clustering or supervised learning).

Self-Supervised Learning. Between supervised and unsupervised learning, self-supervised learning has recently emerged as a popular trend. It relies on defining pretext tasks built directly from data. The term *pretext* means that these tasks do not constitute the final predictive objective but provide a way to pre-train a model that can use the underlying structure of the data and extract robust and transferable representations for subsequent tasks, such as supervised classification or unsupervised clustering. For instance, these pretext tasks may consist of predicting a missing part of data, determining whether two artificial augmentations come from the same data, or reordering a disturbed part of the data. In some cases, intermediate pseudo-labels are generated by the pretext process, but they are not essential: the key lies in the self-generated constraint that leads the learning.

Reinforcement Learning. In contrast to supervised learning, where models are trained using labeled examples, and unsupervised learning, which aims to discover hidden structures in data, reinforcement learning focuses on sequential decision-making through interaction with an environment. In this approach, the model (*a.k.a. agent*) picks an action and gets feedback in the form of a reward or penalty. The goal is to maximize the cumulative reward, which leads the model to discover an optimal strategy for solving the optimization task.

As mentioned in Section 1.2.5, supervised and unsupervised learning paradigms find direct applications in SCA context. Supervised classification is typically used in profiling attacks, whereas unsupervised methods are mainly applied in non-profiling scenarios, for example as pre-processing for dimensionality reduction or through clustering-based attacks, particularly in horizontal attacks as discussed in Section 1.3. One of the contributions of this thesis further extends this perspective by proposing, for the first time, to formulate a horizontal collision attack as an unsupervised verification task relying on self-supervised learning.

2.2 Neural Networks

Deep learning is a subfield of machine learning, based on neural network models that can automatically learn complex hierarchical representations from large amounts of data. Nowadays, neural networks are particularly valued for their ability to model complex non-linear relationships and are among the tools most commonly used by evaluators to carry out SCA. Prior to discussing their application in this context, we introduce some fundamental concepts related to their use.

2.2.1 A First Look at an Artificial Neuron: the Perceptron

The first formalization of an artificial neuron was the perceptron, presented by Rosenblatt [Ros58]. In this model, a neuron processes an input vector $\vec{x}$ using an aggregation function that corresponds to a weighted sum using a weight vector $\vec{w}$. Then, an activation function f is applied to the weighted sum to produce an output value y, which can be expressed as:

$$y = f\left(\sum_{i=1}^{N} x[i]w[i] + b\right) \tag{2.1}$$

Among the most commonly used activation functions there is the sigmoid function, which produces an output between 0 and 1, defined as $f(x) = \frac{1}{1+e^{-x}}$. However, there exists a large number of activation functions that can be used, depending on the characteristics of the data and the predictive task. Another parameter of the perceptron is the bias b, which corresponds to an additional weight that is added to the weighted sum before the activation function is applied. It allows for increased flexibility of the neuron by more finely adjusting the conditions for its activation.

The perceptron model can be used for example to perform binary classification. A supervised learning process was also proposed by Rosenblatt, based on an iterative process that adjusts the weights to determine an hyperplane that separates the two classes. However, this model has a fundamental limitation: it can only handle linearly-separable problems. This limitation was overcome by networking several neurons to form the so called *multilayer perceptron*, opening the way to modern neural networks.

2.2.2 Neurons Team Up: the Multi-Layer Perceptron

The Multi-Layer Perceptron (MLP) [RHW85] is one of the earliest forms of deep neural networks. The principle of such model consists in organizing several simple perceptrons in a collaborative manner. In this way, neurons are grouped side by side to form neural layers, and each layer is connected to the adjacent one by weighted connections. An MLP has at least one input layer, which receives the information to be processed, and one output layer, which produces the predictions. Between these two layers, one or several hidden layers can be inserted, whose number defines the depth of the network. A greater depth allows for the modeling of more complex relationships, but with the price of making learning more difficult. An example of MLP is depicted in Figure 2.1. In this kind of model, all the neurons in a layer are connected to all the neurons in the adjacent layers, but there are no connections between neurons in the same layer. Furthermore, the information flows in one direction, from the input to the output, which is characteristic of feed-forward neural networks.

The parameters of the MLP (*i.e.* weights), are generally initialized randomly and then automatically adjusted from the training data using a learning algorithm, whose behavior is itself governed by a set of hyperparameters defined by the evaluator prior to training, as will be detailed in the following sections.

2.2.3 Learning Process

The training of a neural network consists in adjusting its weights $\boldsymbol{\theta}$ in order to find a suitable configuration for efficiently solving the predictive task. This adjustment, generally performed in a supervised setting, relies on a training dataset of labelled examples establishing

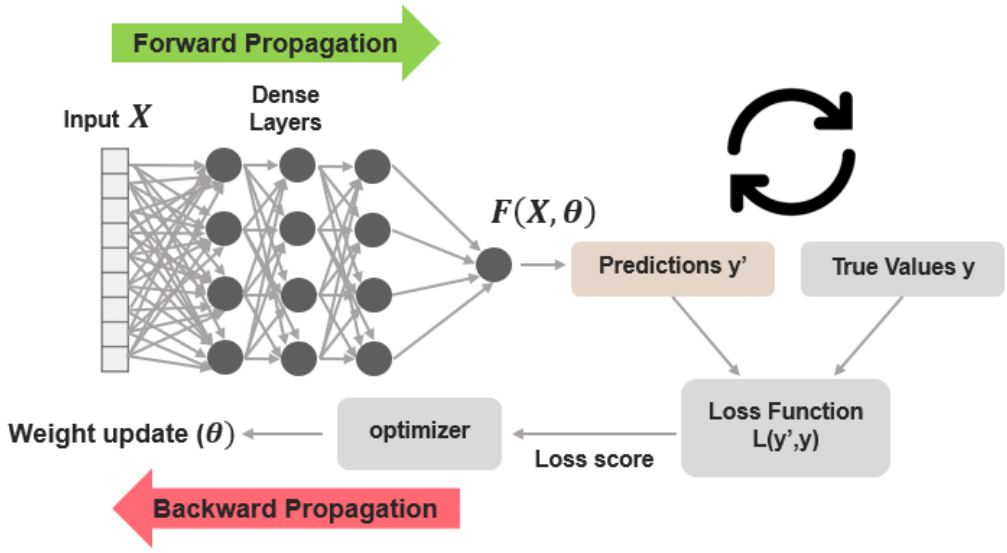


Figure 2.1: Neural network learning.

the correspondence between inputs and expected outputs. One of the most commonly used methods for training neural networks is the backpropagation algorithm [RHW86]. It relies on gradient descent to minimize a loss function L, that measures the difference between the predicted outputs of the model and the expected values. For instance, in a typical binary classification task, a widely used loss function is the binary cross-entropy, which can be expressed as:

$$L_{BC} = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot \log(y_i') + (1 - y_i) \cdot \log(1 - y_i'), \tag{2.2}$$

where y_i denotes the actual labels associated with the training data (*i.e.*, 0 or 1 in this case), and y_i' corresponds to the probabilities predicted by the model that these data belong to class 1. The error is calculated for each training data, then averaged over N of them to estimate the loss function.

The learning process (depicted in Figure 2.1), aimed at minimizing the loss, involves several steps. Firstly, a forward propagation computes the model outputs in respect with the current weights $\boldsymbol{\theta}$, which are then compared to the reference values in order to assess the error. Subsequently, backpropagation spreads this error through the layers by calculating the partial derivatives of the loss function with respect to the weights. Allowing them to be updated iteratively, thereby gradually reducing the error and improving the model's ability to perform its predictive task, which can be expressed as:

$$\boldsymbol{\theta}' = \boldsymbol{\theta} - \alpha\frac{\partial L}{\partial \boldsymbol{\theta}}, \tag{2.3}$$

where α denotes the learning rate, a hyperparameter that controls the amplitude of updates applied to the weights. This hyperparameter must be defined carefully, as its choice

can be crucial to the efficiency of learning. For instance, a too-low learning rate can significantly slow down learning and trap the model in a local minimum of the loss function, whereas a too-high learning rate can lead to learning divergence.

There are several variants of gradient descent, differing in the frequency of weight updates, which affects the convergence speed and stability toward a minimum of the loss function. While *stochastic gradient descent* updates the weights after each training data, *batch gradient descent* only updates them after processing all available data, using the average of the gradients, which ensures more stable but often slower convergence. Nowadays, the most widely used method is *mini-batch gradient descent*, which combines the two approaches by updating the weights after processing a small subset of training data.

The learning process is composed of several steps called *epochs*, themselves comprising several iterations. According to the kind of gradient descent involved, an iteration relates either to the application of the learning algorithm to a single training data, to a whole batch, or to a mini-batch of them. An epoch refers to the set of iterations covering the entire training dataset, *i.e.*, a complete cycle during which the model has processed all available training data. Several epochs are performed in order to gradually reduce the error of the loss function. The number of epochs, which is a hyperparameter to be defined, directly affects the learning: a large number of epochs allows the model to further fine-tune its performance, but increases the execution time and computational cost.

2.2.4 Architecture Design

This section focuses on the design of a neural network model, an essential part of the implementation of deep learning strategies.

The learning and generalization capabilities, as well as the computational efficiency of a neural network model, are directly dependent on a set of decisions taken during its design. Notably, the configuration of its *hyperparameters*, representing the explicit configurations of the model, fixed before the start of training (*e.g.*, learning rate, mini-batch size, number of layers or neurons per layer, etc.). In contrast to learnable parameters (*i.e.* weights), they are not automatically adjusted based on the data, but are defined manually.

An appropriate configuration of hyperparameters is essential to ensure the effective convergence of neural networks and their ability to generalize. Some hyperparameters can be defined relatively easily based on expertise, such as the choice of loss function or activation function for the output layer, which are directly related to the learning task under consideration. For instance, in binary classification, it is coherent to use the binary cross-entropy loss function with an output neuron activated by a sigmoid function. For multi-class classification, categorical cross-entropy is favored, along with an output layer composed of n neurons activated by a softmax function [Bis95], which converts the outputs into normalized pseudo-probabilities providing an estimate of the likelihood of each class. On the other hand, most hyperparameters, such as network depth, number of neurons per layer, learning rate, or hidden layer activation functions, are not based on any fixed rules, and their selection is often performed empirically.

Because of the dynamic research environment surrounding deep learning, new methods and alternatives regularly emerge, broadening the choice of possible solutions. Among the many avenues explored in the literature, activation functions have been the subject of numerous proposals. Since they are responsible for enabling the network to model nonlinear relationships, their choice can significantly influence the learning process and results. Notably, when training neural networks, two gradient propagation-related problems can arise: the vanishing gradient, when gradients become too small and prevent deeper layers from learning, and the exploding gradient, when gradients become too large, causing unstable updates and training divergence. While the use of ReLU activation [NH10], defined by $\text{ReLU}(x) = \max(0, x)$, helps mitigate vanishing gradients by maintaining a constant gradient for positive values, it has the drawback of saturating neurons for negative values. To overcome this problem, several variants have been proposed, such as LReLU [MHN+13], PReLU [He+15], ElU [CUH16], GelU [HG16], and SeLU [Kla+17]. In most cases, the choice of activation function remains empirical, tailored by the observed training efficiency.

In a similar way, optimizers, which determine how the network weights are updated, have been the subject of extensive research and improvement. Classical stochastic gradient descent maintains a single learning rate for all weight updates, which remains constant throughout training. Although simple, this approach limits adaptation to different gradient scales and can slow down convergence. Many variants have been proposed to adapt the learning rate to each weight in order to accelerate and stabilize the learning process. Among these, AdaGrad [DHS10], Adadelta [Zei12], RMSProp [HSS12], and Adam [KB15] are the most widely used today.

Determining the most appropriate hyperparameter configuration is not a straightforward task and is a well-known open problem in the literature, referred to as the *hyperparameterization problem*. It may require human expertise or computational effort, using techniques such as random search, grid search, or more sophisticated optimization methods. One of the most delicate aspects is choosing the complexity of the model. A deep architecture, with many layers and a large number of neurons, is able to model complex relationships and to handle difficult problems. However, if the architecture is too complex with respect to the amount of data or their characteristics, it can hinder learning. In this regard, two learning issues can occur: overfitting and underfitting, which we describe below.

- *Overfitting*. This problem occurs when the model is too complex or has been overly trained, leading it to memorize the training data by heart. As a result, the model does not (or not only) focus on generalizable patterns but on the inherent noise in the training data, compromising its ability to provide reliable predictions on new data.
- *Underfitting*. This problem occurs when the model fails to learn properly, resulting in poor performance even on the training data. A possible cause of this problem may be that the model is not complex enough to effectively capture the underlying patterns in the data.

In order to limit overfitting and improve model generalization, a number of research studies proposed learning regularization methods, which we discuss in the next section.

2.2.5 Learning Regularization

Regularization techniques are a set of methods aimed at improving the generalization ability of a neural network by reducing the risk of overfitting and promoting more stable convergence. Such techniques can be applied at various levels: during data processing, in monitoring model performance, or by penalizing the model and the loss function. In the following, we present some of the most commonly used approaches.

Early Stopping. The early stopping [MB89] is a regularization technique that consists in interrupting the training of a neural network before it starts to overfit on the training data. The principle is based on monitoring the model's performance on a validation dataset, consisting of examples not used during the training phase. In an ideal scenario, the loss function gradually decreases on both the training and validation datasets. However, as soon as the model begins to overfit, it becomes too specialized for the training data, reducing its ability to generalize. This results in an increase in validation loss, even though training loss continues to decrease. In order to preserve the model's ability to generalize, training is stopped at this precise moment. In practice, early stopping often incorporates a patience parameter, which defines the number of epochs the algorithm must wait after the last improvement in validation loss before stopping training. This prevents premature stopping due to small fluctuations in validation performance. In addition, the restore best weights property ensures that after training stops, the model returns to the state (*i.e.*, weights) corresponding to the epoch when it reached its best validation performance.

Loss Regularization. The loss function regularization [KH91] L1 (*a.k.a.* Lasso) and L2 (*a.k.a.* Ridge) aim to add penalties to the loss function in order to constrain the weights of the model, particularly those with high amplitude, which can induce high output variance and make the model overly dependent on the training data. These penalties thus favor models that are less flexible but more robust to the risk of overfitting.

Dropout. The dropout [Sri+14] regularization method acts directly on the structure of the neural network to limit its excessive specialization on training data. Applied to a given layer, it consists in temporarily and randomly deactivating some of the neurons at each iteration of training. The deactivated neurons do not participate in the propagation of information or in the weight update during backpropagation. This operation is only performed during training. In modifying the structure of the model at each iteration, dropout promotes the learning of more robust representations that are less dependent on specific combinations of neurons, thereby effectively reducing the risk of overfitting.

Batch Normalization. The batch normalization [IS15] is a technique aimed at stabilizing and accelerating the convergence of neural networks. It consists in normalizing data, whether inputs or intermediate activations, by centering and bringing them into a stable range of values, generally with a mean close to zero and a standard deviation close to one. When applied to inputs, it ensures a homogeneous distribution, thus facilitating optimization for the early stages of learning. When used in intermediate layers, it limits the phenomenon of internal covariate shift, *i.e.*, the variation in activation distributions over iterations, which otherwise tends to slow down convergence. By reducing these variations,

batch normalization simplifies weight adaptation, promoting faster convergence and improving generalization ability.

Data Augmentation. The application of data augmentation [Won+16] is a technique commonly used in deep learning, which applies a form of regularization by modifying the training dataset through the generation of additional examples, often more difficult to predict. It relies on the application of predefined transformations to the original data in order to produce new training examples. The introduction of these additional examples increases the complexity of the training task, which limits overfitting by making it more difficult for the network to learn the data by heart. Furthermore, some transformations can be specifically designed to promote a network's invariance with respect to those transformations. For instance, in the SCA context, Cagli *et al.* [CDP17] introduced tailored augmentations based on misalignment strategies to target traces that are disturbed by clock jitter during measurements.

All of these regularization methods can be combined, but their implementation, both in terms of choice and configuration, brings new hyperparameters into the model design. In the context of SCA, Robissout *et al.* [Rob+21] investigated the impact of some of these regularization techniques. Their results show that an appropriate combination can significantly improve the efficiency of attacks. In particular, they enable the use of more complex architectures while limiting the risk of overfitting, leading to better overall performance.

2.3 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) [LB98] differ from traditional MLP in the way they process input data. While MLP are based on fully connected layers, where each neuron is connected to all neurons in the previous layer, the CNN, inspired by the visual cortex, introduce an architecture that is particularly effective for extracting multidimensional local dependencies. This approach is better suited to data with a spatial or temporal structure, such as images or signals. Similarly to a MLP, the information is propagated forward, but the originality of CNN lies in the use of specific layers dedicated to feature extraction: namely, convolution layers, which automatically learn local filters, and pooling layers, which reduce dimensionality while retaining the essential information, as described below.

Convolutional Layer. The convolution layer automatically extracts features using local filtering. In contrast to fully connected layers, each neuron is connected only to a limited part of the previous layer, called the receptive field. This specificity is based on the use of filters (*a.k.a.* kernels) representing the set of learnable parameters (*i.e.* weights) applied locally to the data in order to detect characteristic patterns. The main operation consists in sliding this filter over the input, shifting it with a step size called the *stride*, and calculating a scalar product between the receptive field and the filter. A convolution layer typically involves several filters, each producing a feature map that represents the filter's response for all positions of the receptive field.

Pooling Layer. A pooling layer aims to consolidate the features extracted by the convolutional layers. It gradually reduces the spatial dimension of the feature maps while

preserving the most relevant information. In this way, both the number of parameters in the network and its computational cost are reduced. Similarly to convolution, a filter is applied to the input, shifting according to a given stride. However, in contrast to convolutional layers, this filter does not contain any learnable parameters, but simply aggregates local information, usually by taking either the maximum value (*i.e.* max pooling) or the average value (*i.e.* average pooling) in the receptive field under consideration.

CNN Design. An example of a convolutional neural network is depicted in Figure 2.2. It consists of two main blocks, which we describe below.

- *Feature extractor block.* This block is designed to extract meaningful features from input data, typically using convolutional layers to capture spatial and local patterns. As multiple layers are stacked, the model learns increasingly complex representations in high-dimensional feature maps. To manage computational complexity, downsizing techniques are usually applied within this block, such as pooling layers or strided convolutions [He+16; Aya+20], where filters shift across the input with a stride greater than one, allowing the model to learn how to downsize through trainable parameters. Convolution and pooling layers can be repeatedly alternated to detect increasingly complex relationships. Finally, a flatten layer converts the resulting multi-dimensional feature maps into a single dimension for downstream processing.
- *Classification block.* Once the features have been extracted, they are projected into a more compact and discriminative latent space using dense layers. These layers act as MLP classifier, operating as a global feature aggregator to model complex nonlinear relationships and perform the predictive task.

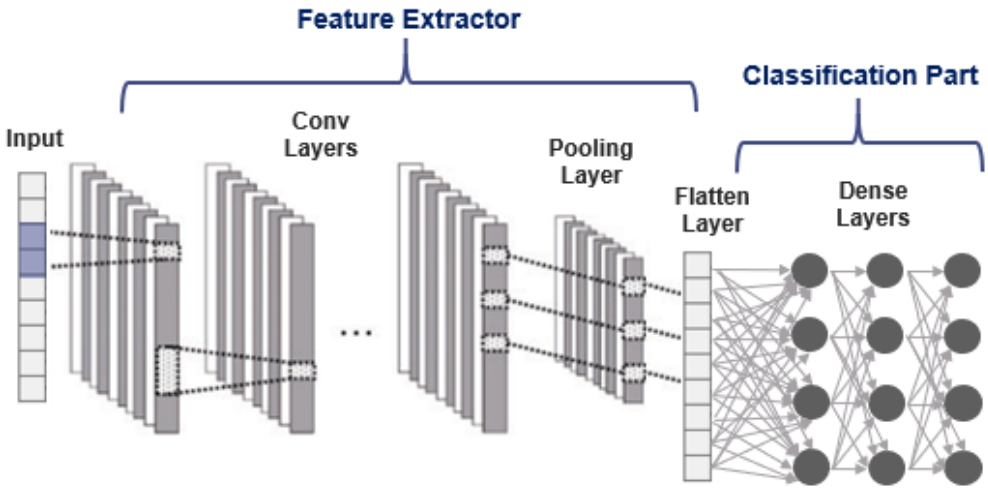


Figure 2.2: Convolutional neural network general architecture.

The CNN feature extraction approach has proved to be particularly effective, reaching state-of-the-art performance in many fields. Their ability to process a variety of data sources makes them an extremely versatile tool, whether for processing of images [RW17],

audio signals [Her+17], or videos [Kar+14]. Furthermore, CNN offer several major advantages over traditional architectures such as MLP, including parameter sharing and translation invariance property, which we discuss below.

- *Computational efficiency.* When a feature map is generated through convolution of a filter with the input data, each convolution window (*i.e.* receptive field location) applies the same weights. This property significantly reduces the number of parameters to be learned as well as the cost of updating them during backpropagation. Furthermore, by gradually reducing the dimensionality in the feature extractor, a compact representation can be provided to the dense classification layers, thus making it easier to train deeper networks.

- *Translational invariance.* A particularly interesting property of CNNs is their translation invariance. Indeed, the same pattern can be detected regardless of its position in the input, thanks to parameter sharing, which applies filters uniformly across the entire input. Such invariance is further enhanced by the downsizing performed in the feature extractor, either through pooling layers or strided convolutions. Both of these techniques help to consolidate information while making the network less sensitive to small positional variations [Jad+15; MMD20]. In the SCA context, this property is precisely what makes CNN a tool of choice for effectively processing misaligned traces [CDP17].

Although powerful, a CNN remain more complex to implement than a MLP due to the large number of additional hyperparameters that need to be configured in the feature extractor block. Indeed, the configuration of convolutional layers (*e.g.* number and size of filters, stride) and the choice of downsizing techniques directly influence the quality of the extracted representations. As these representations constitute the input to the classification block, their relevance conditions its ability to model complex interactions and make accurate predictions. As a result, particular attention must be paid when configuring CNN, especially in complex contexts such as SCA [Zai+20a].

2.4 Unsupervised Neural Networks

Unsupervised deep learning aims to train a neural network to automatically learn relevant representations of data without the use of explicit labels. Unlike supervised approaches, where the model is directly guided by the correspondence between input data and known labels, unsupervised approaches seek to exploit latent structures by taking advantage of the regularities and intrinsic patterns present in the data.

One of the most fundamental approaches is based on the use of an autoencoder [Kra91]. This kind of neural model, consisting of an encoder and a decoder, is designed to reconstruct the input data after compressing it into a latent space of reduced dimension. The reconstruction objective constrains the network to learn an internal representation that is both compact and informative, allowing it to be used in subsequent tasks such as clustering.

Deep clustering is currently a popular topic of research in the field of deep learning. We refer the reader to the recent review [Ren+25] for a comprehensive overview. Several studies have extended the principle of autoencoders by combining latent representation learning with simultaneous improvement of clustering on them [XGF16; Yan+17]. Another approach, is based on an iterative framework that exploits the ability of a CNN to extract discriminative features [Car+18]. Starting with pseudo-labels from an initial clustering, a CNN is trained on them. Following this, the latent representations from the network (the feature extractor output) are then clustered again, updating the pseudo-labels. The model is then retrained with these updated labels. This alternating mechanism, based on the ability of neural networks to generalize even in the presence of noisy labels, gradually improves the quality of the representations learned by the CNN and, consequently, the relevance of the related pseudo-labels. Nevertheless, these methods still have significant limitations. Although deep clustering has seen notable progress, its performance remains inferior to that of supervised approaches, particularly on complex tasks. Most research still focuses on relatively simple problems, such as image clustering, which could be easily solved by humans. Thus, even though it is a promising avenue for reducing dependence on massive annotations, unsupervised learning has not yet reached the necessary maturity to be deployed effectively in challenging contexts involving complex data.

Another pillar of unsupervised approaches is Generative Adversarial Networks (GANs). Introduced by Goodfellow *et al.* [Goo+14], they are based on competition between two networks: a generator, responsible for producing synthetic data, and a discriminator, responsible for distinguishing these synthetic data from real ones. This adversarial approach allows the generator to gradually learn to reproduce the underlying distribution of the data and generate realistic ones. Beyond their ability to produce high-quality data, GANs have also established themselves as powerful tools for unsupervised representation learning [Muk+19], providing rich and exploitable latent spaces for downstream tasks such as clustering or classification.

Lastly, self-supervised learning [Gui+24] has recently emerged as a particularly promising trend in deep learning. As already introduced in Section 2.1, such learning paradigm relies on defining pretext tasks that are constructed directly from the data itself, enabling models to learn meaningful representations in an unsupervised way. This approach has gained significant attention in the research community and has demonstrated strong interest across various domains, including image [Alb22], natural language [Bae+22], and speech processing [Zai+22].

2.5 Deep Learning and Cybersecurity

Deep learning is now an essential component in the field of cybersecurity, due to its ability to automatically learn complex representations from large heterogeneous datasets. Its applications cover a wide range of topics, as highlighted in a recent literature review [MG19]. Among the most notable are the following:

- *Intrusion detection* [Jav+16]. Intrusion detection systems use deep learning to analyze real-time network traffic and system behavior, enabling the identification of

suspicious or malicious activity.

- *Malware detection* [Dah+13]. Deep learning models can learn from features extracted from implementation analysis, whether static (source code, binary files) or dynamic (execution behavior), to detect malware (*e.g.* viruses, backdoors, spyware) and their polymorphic variants, even without prior signatures.

- *Phishing/Spam detection* [MTM14; AT19]. By leveraging advances in contextual analysis, deep learning is now able to identify sophisticated phishing and spam content in emails and websites, outperforming traditional filtering approaches.

- *Biometric authentication* [Min+23]. Nowadays, deep learning improves the accuracy of facial, voice, and fingerprint recognition mechanisms, thereby enhancing the reliability and robustness of authentication systems.

It is also worth noting that deep learning is not only a defensive tool but also provides a powerful way to design sophisticated offensive attacks. For instance, large language models can be exploited to automatically generate phishing campaigns [Roy+24], while voice and video generation techniques can be used to falsify identities (*e.g.* deepfakes) in order to support social engineering attacks [Ngu+22]. This duality reflects a competitive evolutionary dynamic in which the same deep learning techniques are used both to strengthen defense mechanisms and to develop increasingly sophisticated attacks. Beyond these emerging uses, one field of application stands out for its strategic importance: the SCA, which poses a major threat to the physical security of critical systems that feature embedded cryptography. The integration of deep learning in this field has profoundly changed the landscape, making these attacks more automated and more efficient. Today, this field is a particularly active topic of research, which we will discuss in the next section.

2.6 DLSCA Research Efforts

In 2016, Maghrebi *et al.* [MPP16] presented the first study dedicated to the use of Deep Learning for Side-Channel Attacks (DLSCA), marking a major breakthrough in the evolution of SCA, with a gradual shift from traditional approaches based on statistical analysis to those exploiting neural networks. DLSCA has demonstrated remarkable performance, particularly thanks to its ability to directly process large-scale traces without requiring careful PoI selection. Furthermore, these approaches are particularly effective against common countermeasures such as masking or some forms of desynchronization. In this context, DLSCA have become an extremely active topic of research, both in academia and industry. A wide variety of deep learning architectures and strategies have been explored by the SCA community, in both profiling and non-profiling attack contexts, as discussed in the remainder of this section.

2.6.1 Profiling DLSCA

In the context of profiling attacks, deep learning has been extensively studied. Early work focused on vertical attacks targeting symmetric implementations [MPP16; CDP17; Ben+20; Zai+20a], while more recent research has explored horizontal attacks on asymmetric implementations [Car+19; Zai+21a; Shi+22; Bar+22]. In most cases, the attack is formulated as a classification problem and a neural network is trained to distinguish traces based on the sensitive data associated with them.

As emphasized in Section 2.2.4, deep learning remains a highly parameterizable approach, relying on a complex choice of hyperparameters to design the neural network architecture. Thus, in the SCA context, the efficiency of an attack strongly depends on this hyperparameterization. Faced with this difficulty, numerous works have focused on the systematic study of hyperparameters and on the search for methods to design more effective neural network architectures. From this perspective, Prouff *et al.* [Ben+20] studied the impact of hyperparameters in the context of the ASCAD AES implementation in order to identify those that prove the most promising. While these results provide interesting benchmarks, their generalization to other implementations remains limited. In practice, hyperparameterization should be adapted to the specific constraints of the targeted implementation and thus, still relies heavily on empirical research to find appropriate architectures.

SCA Neural Network Design. In early work, the SCA community relied on architectures derived from computer vision, such as VGG15 or VGG16 models, which were used as starting points and then combined with regularization techniques to limit overfitting [Ben+20; Rob+21]. Some contributions introduced data augmentation strategies, like the addition of artificial noise in traces [Ben+20; Kim+19] or jitter-effect simulation [CDP17], aiming to improve the robustness and generalization ability of the models. On the other hand, some work has been focused on proposing a methodology for building CNNs specifically tailored to the targeted implementation [Zai+20a; Wou+20; Zai+20b]. Following this approach, the authors successfully designed tiny models that are particularly effective. However, these architectures remain highly dependent on the used dataset, and their design requires significant human effort as well as advanced expertise in deep learning to analyze and adjust the models. In practice, their application to new targets is therefore not straightforward. In order to reduce this dependence on human expertise, automated hyperparameter optimization has been widely explored. The principle is based on training and evaluating a large number of models on a validation dataset in order to identify the best-performing configurations. Several approaches have been proposed in this context, such as reinforcement learning [Rij+21], bayesian optimization [WPP24], and evolutionary techniques [AGF23]. Finally, an alternative to the search for a rigorously hyperparameterized architecture is to use ensemble learning, which combines several weakly hyperparameterized architectures to construct a more robust and efficient ensemble model [PCP20; Zai+21a]. The latter methodology will be discussed in more detail in Chapter 3.

SCA Tailored Deep Learning. With the progress of DLSCA research, the SCA community has gradually acquired more in-depth expertise in deep learning, paving the way for the development of methods specifically tailored to the context of side-channel attacks. This

is particularly the case for vertical attacks, where the exploitation of multiple traces during the attack introduces additional subtleties that go beyond the simple problem of individual trace classification. In this regard, various learning process strategies have been proposed. These include, for example, the use of early stopping criteria based on SCA-specific metrics [Rob+21], the design of new loss functions that more accurately reflect the objectives of the attack [Zai+21b], as well as the development of specific components such as dedicated layers [Cao+22], custom activation functions [Kne+23], and even an SCA-specific optimizer [Rou+25].

New Trends. Recently, Transformer-based models [Vas+17], which are at the origin of major advances in deep learning (*e.g.*, large language models such as ChatGPT), started to be investigated in the SCA context [Haj+22; HCM24; Bur+24]. The interest of this kind of approach lies in particular in the ability of Transformers to effectively capture long-range dependencies, allowing them to model relationships between distant PoI more accurately than traditional deep learning methods. From a different line of research, some recent works focused on the explainability and interpretability of DLSCA, drawing on theoretical results from stochastic attacks [SLP05] to propose novel neural network architecture and learning process for SCA [Zai+23; Bou+25]. This line of research bridges deep learning and classical SCA methods, aiming to reduce the black box nature of neural networks and facilitate the design of suitable architectures. By proposing a neural network model that is both explainable and interpretable, it also contributes to a better definition of security boundaries during evaluations. Nevertheless, the proposed approach still has several limitations: higher-order attacks cannot be mounted and desynchronization is not handled by this type of models.

2.6.2 Non-Profiling DLSCA

Recently, the application of deep learning to non-profiling attacks has begun to attract the interest of the community. However, unlike profiling approaches, which have been extensively studied and are now well established, research on non-profiling attacks is still in its early stages. Only a few research studies have been carried out in this context, and the proposed methods remain less mature than those used in profiling contexts. Furthermore, most of this research focuses on vertical attacks applied to symmetric implementations, while very few studies have addressed the challenge of horizontal attacks in an unsupervised setting.

Non-Profiling Vertical SCA with Supervised Deep Learning. In 2019, Timon presented a way to apply supervised deep learning in a non-profiling attack context by adapting the DPA concept [Tim19] with a neural network approach. The proposed approach, called Differential Deep Learning Analysis (DDLA), consists of training a model for each key hypothesis in an AES implementation, then identifying the correct one with the best performance indicators during training. The principle remains similar to that of profiling DLSCA, but rather than using a dataset labelled with the correct key, the different key hypotheses are used to generate the labels. In concrete terms, this method requires the training of 256 neural network models, corresponding to all possible hypotheses for a given targeted key byte. By leveraging neural networks whose ability to automatically extract relevant features, DDLA extends DPA-like attacks to masked implementations and remains effective

even in the presence of desynchronizations, particularly through the use of CNN models. Since its introduction, DDLA has become a very popular line of research. Numerous studies aim to improve its performance, [BH21; KHK22; Kur+23; DHD24], mainly by reducing the computational cost associated with training a large number of models. A frequently explored solution is to use multi-output model sharing common layers, allowing multiple key hypotheses to be processed simultaneously while limiting computational redundancy.

In a different line of research, Staib and Moradi [SM23] proposed another approach using supervised deep learning to carry out a non-profiling SCA, this time based on the collision principle. This approach can be seen as a deep learning application of classic collision attacks, such as the Correlation-Enhanced Power Analysis Collision Attack (CEPACA) [MME10]. The concept is based on the assumption that similar operations applied to the same data lead to comparable leakage behavior. For instance, observing similar patterns for two different SBox operation traces suggests that the same data has been processed. In this context, a neural network is trained in a supervised manner to identify these similarities. This involves training it on a first SBox and then reusing it to predict labels associated with another SBox, thereby exploiting the transferability of learned representations. A formal comparative study of these two non-profiling SCA approaches using supervised deep learning is presented in [Ito+23].

Non-Profiling Vertical SCA with Unsupervised Deep Learning. The use of unsupervised deep learning methods in the non-profiling SCA context remains relatively limited. Some studies have explored the use of autoencoders for traces pre-processing. In particular, Ramezanpour *et al.* [RAD20] explored their application for dimensionality reduction. The principle consists in processing a whole trace as input to an autoencoder, which projects it into a low-dimensional space before attempting to reconstruct it from this compressed representation. The aim is to extract and keep only the most relevant points of the trace, which can then be used in a DPA-like attack. Nevertheless, this approach has only been assessed on unprotected implementations, and its effectiveness on protected ones, for example through masking, remains uncertain. Alternatively, autoencoders have been proposed by Wu and Picek [WP20] as a denoising preprocessing step. The idea is to train an autoencoder on pairs of noisy and clean traces so that it learns to reconstruct the clean version from the noisy one. The cleaning of several hiding countermeasures have been explored including Gaussian noise and some desynchronization methods. However, these unsupervised deep learning approaches have several practical limitations. Firstly, autoencoder models are complex to train, especially for high-dimensional traces. Secondly, the unsupervised nature of the loss function does not guarantee that the latent space preserves the PoI that are relevant to the attack. Particularly in noisy contexts or those with few PoI, the autoencoder may focus on reconstructing noise or non-informative points in order to minimize the overall reconstruction error, to the detriment of those points that are relevant to SCA. Furthermore, denoising methods assume that the evaluator is able to access a set of clean traces, free from countermeasures, which is often not the case in a non-profiling scenario.

Non-Profiling Horizontal SCA with Unsupervised Deep Learning. Regarding horizontal attacks without profiling, only a few studies have explored the use of unsupervised deep learning, focusing on clustering-based attacks. As explained in Section 1.3, the suc-

cess of this kind of attack mainly depends on the proper selection of PoI, which itself relies heavily on the quality of the traces. Indeed, overly level of noise, outliers, or measurement artifacts can compromise the identification of PoI and decrease the efficiency of the attack. Furthermore, the success of the attack depends on recovering almost all of the bits of the secret key, in order to ensure that the remaining bits can be practically recovered. To address these limitations, pre-processing and post-processing methods based on unsupervised deep learning techniques have recently been proposed in the literature. With regard to pre-processing, Cler *et al.* [COM25] explored the use of unsupervised neural networks to mitigate noise artifacts present in the measured traces. Their approach is based on specific autoencoder and GAN architectures designed to correct anomalies and improve trace exploitation through a clustering attack. With regard to post-processing, some works have explored the use of a corrective framework designed to improve the estimation of the remaining bits of the exponent by relying on neural network training [Per+21; BA23]. Inspired by the co-teaching strategy [Han+18], such a framework consists in training two neural networks on distinct subsets of the dataset iteratively: after each training, one network is used to relabel the subset on which it was not trained, then the process is reversed with the second network. This mechanism allows for a gradual improvement in label accuracy. The effectiveness of this correction depends on the neural network's ability to generalize correctly despite the presence of noisy labels. However, the success of such a framework depends heavily on the proportion of bits correctly estimated during the initial stage, *i.e.*, from the clustering attack. Although relevant in some contexts, implementing these approaches in realistic attack scenarios remains challenging, particularly when the acquired traces are noisy and extensive. Furthermore, these approaches have only been investigated in the context of clustering-based horizontal attacks. To date, no research has yet explored the use of unsupervised deep learning methods to either simplify or improve the implementation of collision-based horizontal attacks. In this regard, Chapter 4 of this thesis presents pre-processing methods based on unsupervised neural networks to help in the detection of collisions between pairs of modular operation traces as part of a horizontal collision attack.

Part II

Contribution

Chapter 3 The Ensemble Strikes Back

Vertical Attacks with Stacking Neural Networks

Even the weak become strong when they are united.

– Friedrich Schiller

Resume

In this chapter, we address the challenge of neural network hyperparameterization in SCA. To address this issue, we investigate ensemble learning, focusing on stacking, in which a meta-model learns how to aggregate predictions, as opposed to bagging, which is based on a deterministic aggregation process. Experimental results on publicly available datasets of symmetric implementations show that stacking consistently outperforms bagging, making better use of model predictions. Moreover, our work shows that stacking can achieve performance comparable to highly hyperparameterized architectures with significantly less effort, making it a valuable tool to assist evaluators in building efficient models.

3.1 Research Issue

As shown in Chapter 2, deep learning is nowadays widely used in side-channel evaluations, in which it has proven to be particularly effective, even against protected implementations. For instance, deep learning attacks can bypass certain countermeasures such as masking or trace desynchronization. They also reduce the need for careful selection of points of interest, making the pre-processing of attacks easier. Nevertheless, this strength comes with a major drawback. As discussed in Section 2.6, the efficiency of such attacks heavily depends on the selection of a large number of hyperparameters (*e.g.* learning rate, number of layers, number of neurons, etc.), whose precise impact is often difficult to foresee. In particular, overly complex neural network architectures are prone to overfitting.

The hyperparameterization problem poses a major challenge in a time-constrained evaluation context. Indeed, the design of a well-performing neural network model relies on specific expertise and careful tuning, usually achieved through numerous empirical experiments. As mentioned in Section 2.6, several studies have focused on methodologies to support the construction of neural networks well-suited for SCA. For this purpose, Zaid *et al.* [Zai+20a] proposed designing tiny convolutional neural networks that could achieve good performance while limiting the risk of overfitting, relying in particular on visualization tools to analyze the impact of each convolutional hyperparameter. In contrast, Robissout *et al.* [Rob+21] start from complex architectures, which are naturally prone to overfitting, and investigate different regularization techniques (*e.g.*, batch normalization, dropout, loss regularization) to improve their generalization ability. Both of these approaches involve significant human effort in terms of deep learning expertise and detailed analysis through extensive experimentation. Furthermore, the resulting architecture remains specific to the targeted dataset and is not directly transferable to other settings.

To overcome these limitations, some works have attempted to automate the search for hyperparameters and neural network model design. Notably, Rijsdijk *et al.* [Rij+21] proposed using reinforcement learning techniques, Wu *et al.* [WPP24] explored Bayesian optimization, and Acharya *et al.* [AGF23] studied neural evolution approaches based on genetic algorithms. While these methods largely eliminate the need for human expertise and the analysis of the targeted dataset, they come at the cost of extremely high computational effort, as they require training hundreds or even thousands of models. This makes them difficult to apply in evaluation contexts, particularly when attacking protected symmetric implementations that demand millions of traces for training, with time and computational resources often being limited. In practice, evaluators often have neither the opportunity nor the ability to test a wide variety of architectures, and therefore limit themselves to training a few models selected by some grid search, with hyperparameters chosen based on their expertise.

Contributions. We propose in this chapter to address the problem of hyperparameterization through ensemble learning, which consists of combining several weakly or partially hyperparameterized models, rather than seeking a rigorously optimized one. This approach allows the evaluator to reduce their dependence on a specific hyperparameter configuration by leveraging the complementarity between different models to improve the robustness of the attack. For instance, Perin *et al.* investigated deep bagging ensemble in the context of

profiling vertical SCA against symmetric implementations [PCP20]. This principle relies on training several neural networks to perform the same predictive task and then combining their outputs. The underlying intuition is that, by leveraging the collective strength of multiple models, the overall effectiveness of the attack can be significantly improved.

Building on these preliminary results, we extend the application of ensemble learning to SCA by proposing the use of stacking as an aggregation method. Our proposal is supported by numerous experimental results obtained on various publicly available datasets, which demonstrate that stacking can relieve the evaluator of the need to perform fine hyperparameterization. We notably present a detailed experimental study, whose objectives are to evaluate the relevance of stacking in the profiling context, to highlight its advantages and limitations, and to compare it to bagging.

In this contribution, we consider ensemble learning only in an a posteriori setting, *i.e.*, by leveraging models that have already been trained in order to strengthen attack performance. The major practical advantage of this method is that the models in the ensemble can be directly taken from a grid search that has already been performed. Thus, rather than considering the least effective models as failures or wasted resources, this strategy allows them to be reused effectively with minimal additional computational cost. In this way, the ensemble constructed a posteriori can turn an incomplete exploration of hyperparameters into a well-performing architecture, while limiting the need for extensive tuning.

The solutions proposed in this chapter have been presented and published at the 22nd Smart Card Research and Advanced Application Conference (CARDIS 2023) [Lla+23].

3.2 Targeted Implementations and Attack Path

This section provides details on the vertical profiling attack scheme considered in this contribution, as well as a description of the datasets used to experimentally validate our contribution.

3.2.1 Description of the AES Datasets

This section presents the symmetric implementations targeted in this contribution of this thesis relating to vertical attacks. Notably, two 128-bit AES implementations were considered: an unprotected but highly noisy implementation on FPGA (AES HD) and a protected implementation using first-order masking with optional trace misalignment (ASCADv1). These publicly available datasets are widely used in the literature to benchmark new attacks [Zai+20a; Rob+20; Zai+21b; Zai+23; WPP24].

ASCADv1. This dataset was introduced in [Ben+20]. It contains traces from a masked AES-128 implementation running on a 8-bit AVR microcontroller (ATmega8515). The implementation is protected with a first-order Boolean masking scheme. The targeted sensitive variable is the third byte of the first round Sbox operation such that the label of a leakage trace $\vec{T}$ can be described as $\text{Sbox}[P[3] \oplus k]$, where $P[3]$ denotes the plaintext byte.

Without any assumption on the mask, the attack has to combine an unknown mask, denoted $r3$ and the related masked value, denoted $\text{Sbox}[P[3] \oplus k] \oplus r3$, in order to perform a high-order side channel attack. There are actually two versions of the dataset that we will name ASCADF and ASCADV. ASCADF has a fixed key for training traces and consists of $50\,000$ traces for profiling and $10\,000$ for attack. Traces contain 700 time points, *a.k.a.* features. ASCADV has variable keys for training traces and consists of $200\,000$ traces for profiling and $100\,000$ traces for attack. Traces contain $1\,400$ features. Each of these versions also includes 3 variants to add a desynchronization type countermeasure with an artificial shift of maximum amplitude of respectively 0, 50 and 100 samples. The dataset is publicly available at https://github.com/A-NSSI-FR/ASCAD. For more information on the dataset, we suggest readers refer to [Ben+20].

AES HD. This dataset was introduced in [Pic+19]. It contains traces from an unprotected hardware implementation of AES-128 implemented on Xilinx Virtex-5 FPGA of a SASEBO GII evaluation board. The targeted sensitive value is the register writing in the last round such that the label of a leakage trace $\vec{T}$ 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 $\vec{T}$, and the relation between j and j' is given by the ShiftRows operation of AES. The authors use $j = 12$ and $j' = 8$. The AES HD dataset does not include any countermeasure but has the particularity to be very noisy. This dataset is composed by a set of $75\,000$ leakage traces of $1\,250$ time points. $50\,000$ traces are used for profiling and $25\,000$ are used for attack. The dataset is publicly available at https://github.com/AESHD/AES-_HD_Dataset. For more information on the dataset, we suggest readers refer to [Pic+19].

3.2.2 Attack Path

For the sake of completeness, we briefly recall the basics for mounting a traditional vertical attack against AES implementation. During acquisition, each trace is associated with a target sensitive variable, corresponding to the output of a Sbox, denoted $Z = \text{Sbox}(K \oplus P)$, where P denotes some part of public variable, *e.g.* a plaintext, and K the part of secret key the evaluator aims to retrieve. In the profiling phase, the leakage characterization is done with a model $F(z, \vec{t})$ that provides an estimate of the posterior probability $Pr[Z = z|\vec{T} = \vec{t}]$, being Z the target sensitive variable and $\vec{T}$ the random vector representing side-channel traces. In this contribution, the model F is assumed to be either a MLP or a CNN.

Once the profiling phase is done, and the model F is able of establishes the relationship between the leakage and the corresponding value of the sensitive variable (which is linked to the target key byte), the evaluator can proceed to recover the key byte. To do this, all possible values of the targeted key byte are enumerated, and for each of them, the corresponding sensitive variable values are computed. Then, for all the key byte candidates, the evaluator exploits the set of attack traces by summing the logarithms of the output probabilities of their respective labels. This gives the following logarithmic probability vector $\vec{g} = (\vec{g}[0], \ldots, \vec{g}[C])$ used to determine the likelihood that each of the C candidates is the

correct one:

$$\vec{g}[k] = \sum_{i=1}^{Q} log(F[\text{Sbox}(k \oplus p_i), \vec{t_i}]), \tag{3.1}$$

where Q is the number of attack traces. The value $F[\text{Sbox}(k \oplus p_i), \vec{t_i}]$ denotes the $\text{Sbox}(k \oplus p_i)$-th component of output of the neural network model, given the trace $\vec{t_i}$ as an input. It is interpreted as the probability assigned by the profiling model of obtaining the sensitive value $\text{Sbox}(k \oplus p_i)$ corresponding to leakage trace $\vec{t_i}$ with a key byte hypothesis k and a plaintext byte p_i.

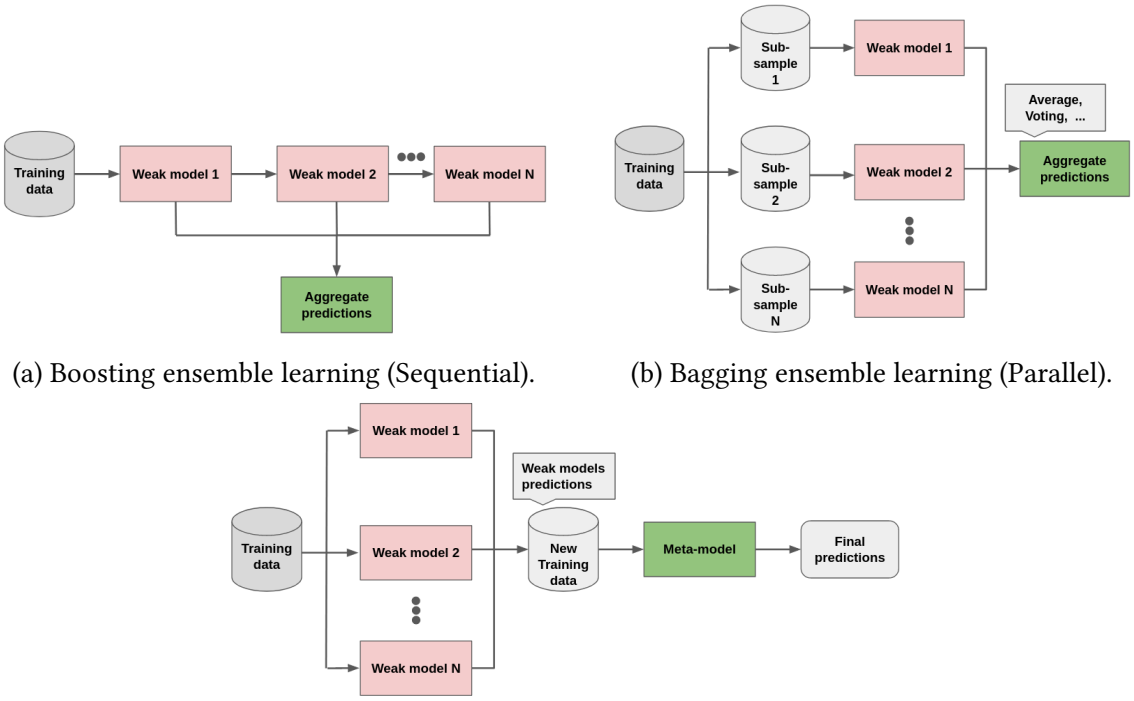
In order to assess the success of the attack, the evaluator can subsequently use this logarithmic vector to compute metrics such as the rank of the correct key hypothesis or the number of traces required to distinguish it from the others, as discussed in Section 1.4.2.

3.3 Ensemble Learning Paradigm

In ensemble learning, the models usually perform poorly individually (slightly better than random guessing) and are referred to as *weak models*. These models are used as construction blocks to design a more complex model by combining them. The principle is based on the synergistic use of several different weak models and the appropriate combination of their predictions using an aggregation method in order to reduce both their variance and bias and thus obtain a more accurate and robust model, called the *ensemble model*. As proved in [AP95], if the weak models' errors are uncorrelated, the ensemble can be more efficient than any individual weak model. Combining the predictions of complementary weak models can thus improve the generalization.

Ensemble approaches can be categorized into two main paradigms. On the one hand, parallel methods consist of training several weak models either independently of each other or jointly and simultaneously, then aggregating their predictions. On the other hand, sequential methods train weak models iteratively, with each new model correcting the errors made by the previous ones. In these two paradigms, there are three main families of ensemble meta-algorithms: bagging, boosting, and stacking, which we describe in detail below.

- *Boosting*. The principle of boosting (depicted in Figure 3.1a) is based on a sequential approach where each weak model is constructed taking into account the errors made by the previous ones. As opposed to other ensemble methods, it cannot be parallelized. The process is carried out in iterations, each corresponding to the training of a new weak model responsible for correcting the errors of its predecessors. The seminal boosting algorithm is AdaBoost [FS95]. In this approach, the training set is re-weighted at each iteration according to the errors of the previous weak model so that misclassified data are assigned a higher weight, leading the next weak model to focus on them. The final aggregation of weak models thus produces a robust ensemble model that significantly outperforms each of the individual ones.
- *Bagging*. The principle of bagging [Bre96] (depicted in Figure 3.1b) consists of training several weak models, generally of the same type, or several instances of the same



(a) Boosting ensemble learning (Sequential).

(b) Bagging ensemble learning (Parallel).

(c) Stacking ensemble learning (Parallel).

Figure 3.1: Different ensemble learning methodologies.

model. Unlike boosting, learning can be performed in parallel, independently, or jointly. In most cases, the training set is subsampled so that each weak model is trained on a different part of the dataset, promoting diversity in predictions and ensuring greater complementarity in the corresponding errors. The predictions obtained are then combined using a deterministic aggregation function, such as an average or majority vote. Although bagging is commonly used with decision trees, particularly in random forests [Bre01], this method can be applied to any type of model, including advanced ones such as neural networks.

- *Stacking.* The principle of stacking [Wol92] (depicted in Figure 3.1c) differs from bagging in several ways. In contrast to bagging, the weak models used in stacking ensemble are generally trained on the same data. The diversity between them therefore does not come from different sampling, but rather from the ability to properly exploit weak models of heterogeneous nature. Furthermore, the aggregation of weak model predictions is not based on a simple deterministic function, but on a higher-level model called a *meta-model*, which is responsible for learning how to combine the predictions of the weak models to produce the final ensemble predictions. The meta-model can be a simple algorithm like logistic regression, a decision tree, or a more complex one such as a neural network.

Due to its ability to enhance the predictive performance of models while reducing the effort required for hyperparameter tuning, ensemble learning has become a particularly active and popular research area. When combined with deep learning, it has led to major breakthroughs by enhancing the generalization capabilities, flexibility, and robustness of

models. This synergy has led to state-of-the-art results in many application domains, including text processing [MK22], image classification [Haq+22], audio analysis [Nan+20], and video analysis [Zhe+19]. Notably, CNN ensembles used as advanced feature extractors can provide much richer representations than those produced by a single traditional CNN. For an in-depth overview of ensemble learning, we refer the reader to two recent surveys: the first by Mohammed and Kora [MK23], which presents current trends in ensemble learning, and the second by Tanveer *et al.* [Tan+23], which focuses on applications to speech signal processing tasks.

Ensemble SCA. In this context, it is natural that the SCA community has also turned its attention to ensemble learning, leveraging its principles to strengthen attack performance while reducing the effort devoted to hyperparameterization. Early contributions have mainly focused on machine learning ensembles: for instance, Destouet *et al.* [Des+20] employed an ensemble of models to approximate a leakage model by targeting different sensitive values. Similarly, Gao *et al.* [Gao+21] investigated various ensemble methods based on decision trees, including Bagging, and AdaBoost, with the objective of improving attack success rates. These works illustrate the potential of ensemble strategies in SCA, paving the way for more advanced approaches relying on deep ensemble learning, particularly suited to alleviating the hyperparameterization effort required for neural networks. Notably, Perin *et al.* [PCP20] demonstrated that combining predictions from multiple neural networks of different architectures using a bagging approach can significantly improve attack performance in profiling vertical attacks against symmetric implementations. Building on this idea, Zaid *et al.* [Zai+21a] extended the concept to profiling horizontal attacks against asymmetric implementations and introduced a new loss function, called *Ensembling loss*, designed to maximize diversity among weak models during training. These studies show that leveraging multiple neural networks in an ensemble can enhance the efficiency of side-channel attacks, while also emphasizing the importance of model diversity for improving overall performance.

Following this line of research, ensemble deep learning has become an active area of research to improve profiling vertical attacks on symmetric implementations. A number of recent studies have explored new ensemble learning methodologies aimed at resolving various issues encountered when implementing this kind of attack. Notably, Hameed *et al.* [HRA24] explored the use of a bagging strategy with resampling along with data augmentation to address the problem of class imbalance in side-channel datasets. Other work has furthered the study of ensemble bagging: Ding *et al.* [Din+25] proposed new aggregation methods that incorporate specific weights for SCA during the ensemble attack phase, while Minghui and Yap [MY24] suggested the use of genetic algorithms to select complementary weak models in order to create effective ensembles. Furthermore, Yin *et al.* [YLO25] have continued the analysis of stacking ensembles. Proposing a different approach from ours, they suggest to simultaneously adjust the weak models and the meta-model during a single learning process.

3.4 Experiments Details

In this contribution, we consider specifically the scenario of a posteriori ensemble learning, *i.e.*, the combination of previously trained weak models. Accordingly, we do not consider sequential boosting-type approaches. Similarly, methods that aim to jointly train several weak models to maximize their diversity, as in the work of Zaid *et al.* [Zai+21a], are out of the scope of this study. More specifically, our contribution is in line with the work of Perin *et al.* [PCP20], which proposes using a posteriori ensemble learning to improve profiling vertical SCA on symmetric implementations. To experimentally validate the methods presented in this work, we used various publicly available datasets of AES-128 implementations, which are detailed in Section 3.2.1. For each of these datasets, the attack scheme consists of targeting a key byte, with the targeted sensitive value being an Sbox output, and the chosen leakage model is that of identity, assuming 256 possible values in the classification task.

Hardware and Software Setup. For all datasets, the experiments are implemented in Python 3.9 using the Keras 2.8 library and are run on a workstation equipped with 32GB RAM and a NVIDIA Quadro P4000 with 8GB memory.

3.4.1 Weak Models and Baselines

In practice, the implementation of ensemble learning raises several issues, including the number of weak models to aggregate, the choice of their architectures, and the aggregation function to use. In the remainder of this section, we present the experimental setup chosen for our ensemble approach and provide arguments for our choices in respect to the context and constraints related to the attack performance of our weak models. We also present the baselines used to compare our approach.

Number of Weak Models. Some works have investigated the question of how many weak models should be used in an ensemble [HS90; OM99; HMS13]. To summarize, it appears that the optimal ensemble size depends on the problem and the performance of the weak models, so generally the ensemble size can be considered as another hyperparameter that can be searched by experimental analysis. Notably, Hansen and Salamon suggested that an ensemble with as few as 10 weak models is generally sufficient to reduce test error and improve generalization [HS90]. Furthermore, the SCA bagging experiments reported by Perin *et al.* [PCP20] also suggested that an ensemble of 10 weak models offers a good trade-off between efficiency and computational complexity compared to larger ones (*i.e.*, 50 weak models). These statements are in line with our research issue: as mentioned above, in a time-constrained evaluation context, the evaluator often lacks the resources needed to train a large number of models. Therefore, in this study, we chose to limit the scope of the analysis to ensemble of up to 10 weak models

Weak Models Configuration. In order to evaluate bagging and stacking ensemble learning methods, we trained for each dataset 10 neural networks comprising both MLPs and CNNs, some of whose hyperparameters were randomly selected from predefined intervals

in a search space provided in Table 3.1. The training and validation sets are obtained from the labeled data with a split ratio of $80\%/20\%$. Each weak model was trained for a maximum of 100 epochs, with an early stopping criterion that restores the best weights after a patience of 20 epochs, monitoring the loss on the validation data.

Table 3.1: Search space for weak models.

Hyperparameter	min	max	step
Learning Rate	0.0001	0.001	0.0001
Mini Batch	100	1000	100
Nb conv layers	2	8	1
Filters	8	32	4
Kernel size	10	20	2
Stride	5	10	5
Nb FC layers	2	3	1
Nb FC neurons	100	1000	100
Activations	Relu, Elu, Selu, Gelu, Tanh		

(a) Search space for ASCADF 0d / AES HD CNN hyperparameters.

Fixed Conv part			
Conv(32, 1)			
AveragePooling(2, 2)			
Conv(64, 25)			
AveragePooling(25, 25)			
Conv(128, 3)			
AveragePooling(4, 4)			
Random Dense part			
Hyperparameter	min	max	step
Learning Rate	0.0001	0.001	0.0001
Mini Batch	100	1000	100
Nb FC layers	2	4	1
Nb FC neurons	500	4000	500
Activations	Relu, Elu, Selu, Gelu, Tanh		

(b) Search space for ASCADV 100d CNN hyperparameters.

Hyperparameter	min	max	step
Learning Rate	0.0001	0.001	0.0001
Mini Batch	100	1000	100
Kernel size	16	128	16
Nb conv layers	1	4	1
Nb FC layers	1	3	1
Nb FC neurons	500	4000	500
Filters	1		
Strides	2		
Activations	Relu, Elu, Selu, Gelu, Tanh		

(c) Search space ASCADV 50d full random CNN.

Hyperparameter	min	max	step
Learning Rate	0.0001	0.001	0.0001
Mini Batch	50	1000	100
Nb FC layers	2	8	1
Nb FC neurons	100	1000	100
Dropout	0.0	0.4	0.1
Activations	Relu, Elu, Selu, Gelu, Tanh		

(d) Search space for ASCADF 0d/ AES HD/ ASCADV 0d MLP hyperparameters.

It should be noted that, in the context of bagging as described in Section 3.3, the usual approach is to subsample the training set in order to train weak models on distinct subsets, thereby promoting diversity in predictions and complementarity in errors. However, in the context of side-channel attacks, the profiling phase already requires a large number of traces, and the publicly available datasets do not contain enough data to allow for relevant subsampling. Hence, in a similar manner to Perin *et al.* [PCP20], we use the same training set for all weak models. In order to mitigate the lack of diversity due to non-subsampling, we varied the architectures of the neural networks, allowing them to learn different features from the same training set. This also allows us to compare bagging and stacking in the same configuration.

It is also noteworthy in Table 3.1 that an exception to the random selection of hyperparameters is done in the case of the ASCADV 100d dataset. Here, the convolutional feature extractor part of the CNNs has been fixed once for all in order to deal with the high desynchronization and obtain weak models able to perform successful (even if poorly performing)

attacks independently. However, in Section 3.5.3 we explores some experimental results on ASCADV 50d where the convolutional feature extractor part has not been fixed.

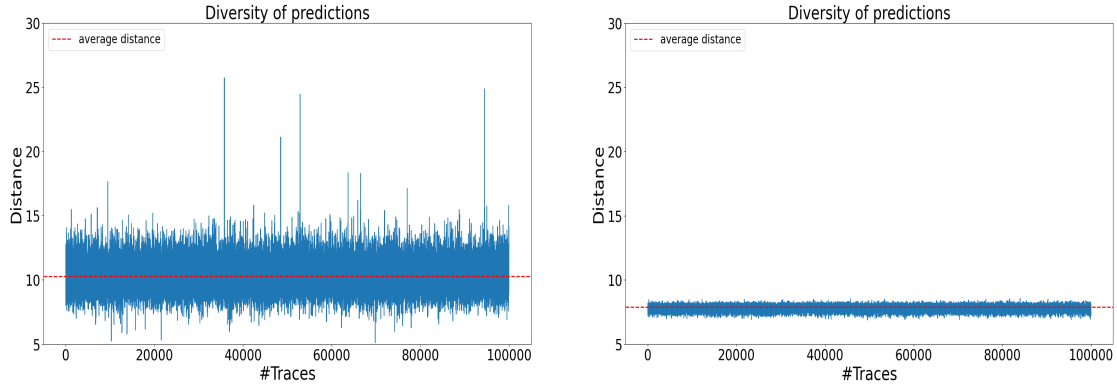
Weak Models Result. The attack performance of the weak models is presented in Table 3.2. The models are ranked according to their attack efficiency, measured by the Na metric, which corresponds to the number of traces required for an attack to be considered successful.

Table 3.2: Weak models ranked according to Na across datasets.

Weak model	ASCADF 0d	ASCADV 0d	ASCADV 100d	AES HD
	Na | Acc	Na | Acc	Na | Acc	Na | Acc
1	1109 | 0.0065	2973 | 0.0073	1792 | 0.0070	22034 | 0.0043
2	1400 | 0.0071	3340 | 0.0073	1797 | 0.0066	23784 | 0.0041
3	2037 | 0/0069	3563 | 0.0076	1950 | 0.0071	23852 | 0.0044
4	2154 | 0.0065	3814 | 0.0073	2082 | 0.0069	24290 | 0.0043
5	2782 | 0.0066	3970 | 0.0074	2174 | 0.0068	24365 | 0.0042
6	2788 | 0.0059	3992 | 0.0074	2175 | 0.0068	24587 | 0.0042
7	3766 | 0.0060	4037 | 0.0070	2200 | 0.0070	24814 | 0.0040
8	8832 | 0.0054	4051 | 0.0077	2491 | 0.0067	24846 | 0.0041
9	+10000 | 0.0041	4187 | 0.0077	2572 | 0.0064	24978 | 0.0040
10	+10000 | 0.0031	4375 | 0.0070	2729 | 0.0068	24983 | 0.0039

It is important to note that the weak models involved are at the heart of ensemble learning, as their diversity and complementarity directly condition the overall efficiency of the ensemble. In this way, a good ensemble is defined as a combination of models that individually perform correctly, while offering sufficient diversity in their predictions and complementarity in their errors. However, in our study, the analysis of the results of weak models reveals contrasting settings among the datasets. For ASCADF 0d, there is a significant gap between the performance of weak models, with some of them achieving very good results while others completely fail to carry out the attack with the available number of traces. For ASCADV 100d, all the weak models perform correctly but their predictions lack in diversity due to the fixed feature extractor part of the CNNs. This lack of diversity is depicted in Figure 3.2, which compares the cross-entropy between the predictions of the best and worst weak models on ASCADV 0d and ASCADV 100d. Here, we can clearly observe that fixing the convolutional part on ASCADV 100d leads to predictions that are much closer to each other. In contrast, on AES HD, the weak models present good diversity, but their performance remains very poor due to the high level of noise in the dataset, making the predictions close to random behavior. Finally, the ASCADV 0d dataset appears to be the most favorable scenario, combining good individual performance of weak models with enough diversity to fully exploit the benefits of ensemble learning.

These contrasting configurations provide a particularly interesting experimental environment for evaluating the relevance of ensemble learning in both favorable and challenging contexts. These latter settings correspond to typical scenarios that may be encountered in practical evaluations, notably when it is not feasible to conduct in-depth investigation into the hyperparameterization of weak models.



(a) Crossentropy between best and worst weak model predictions on ASCADV 0d

(b) Crossentropy between best and worst weak model predictions on ASCADV 100d

Figure 3.2: Comparison of diversity of weak model predictions between ASCADV 0d and ASCADV 100d

Choice of Weak Models. Once obtained 10 weak models, and in order to use a posteriori ensemble learning, we have to choose the models to use in the ensemble. Nevertheless as with the ensemble size, the choice of weak models to combine must often be found experimentally. Due to the performance gap and the lack of diversity on some datasets, we ranked our 10 weak models from the best performing in the attack (minimal Na) to the worst performing in order to increase ensemble size by decreasing attack performance.[1] We made this choice in order to take the point of view of an evaluator who knows the attack performance of his weak models and wishes to use a posteriori ensemble learning to improve his attack. This also allowed us to consider a good conditions for bagging and assess its limitations in this context. Indeed, a significant performance gap between the weak models within the ensemble can greatly impact the effectiveness of a simple bagging aggregation.

Baselines. In our experiments, in order to assess the benefits of the stacking approach, we established two comparative benchmarks. Firstly, we compared the results obtained with those of the best weak model taken individually (marked in blue in Table 3.2), which allows us to directly assess the contribution of meta-learning. In addition, we also compared our results with a bagging-based ensemble method. More specifically, we used the bagging method tailored for the vertical SCA context, as introduced by Perin *et al.* [PCP20]. Such baging aggegation consists of summing the log-probabilities during the attack phase. The new sum of the log-probabilities $\vec{e}$ based on the Equation 3.1 is calculated for each key byte hypothesis k:

$$\vec{e}[k] = \sum_{m=1}^{W} \sum_{i=1}^{Q} log(F[\text{Sbox}(k \oplus p_i), \vec{t}_i]_m), \tag{3.2}$$

where W is the number of weak models.

[1]Other criteria have been tested during the experimental campaign, but the obtained results were less performant and uninteresting in our opinion. Thus, they have been omitted.

3.4.2 Stacking Ensemble

In this section, we provide useful details on how we implemented stacking ensemble learning. In particular, we explain how the predictions of weak models were concatenated to form the input for meta-models, as well as their training procedures and the scope of the conducted experiments.

Concatenation Method. We stack the weak models predictions in depth-wise sequence: if we have N weak models and each of them produces 256 value per prediction, we finally get a stacked prediction c of shape $256 * N$ to train the meta-models:

$$c = \left[\vec{P}_0[0], \vec{P}_1[0], \cdots, \vec{P}_N[0], \cdots, \vec{P}_0[255], \vec{P}_1[255], \cdots, \vec{P}_N[255]\right], \tag{3.3}$$

where $\vec{P}_j[i]$ denotes the i-th component of output of the weak model j. Anyway, we remark that this choice should have no impact a priori on the learning process, since we use MLP as meta-models and the input layer of the MLP is fully connected (the order of the input thus not affect the possible functions it could model). Interestingly, we believe that this choice may have an important impact in cases where the meta-model is a different kind of model, for example a CNN. Indeed, CNN extracts information locally from the input, thus the proximity of the probability predictions for a same class could be a benefit for this kind of meta-model. An analysis of these impacts is left for future works.

Once trained to combine the outputs of the weak models, the meta-model provides the final predictions. In concrete terms, the weak models first generate their predictions on new data, which are then used as input for the meta-model. The meta-model then aggregates these predictions and produces the final ones

A major constraint of stacking is that the performance of the ensemble mostly depends on the ability of the meta-model to learn from the predictions of the weak models. A well-known challenge is how to divide the data between the different learning levels. It is generally recommended to split the training set, since using the same examples both to train the weak models and to generate predictions for the meta-model can promotes overfitting. A common practice is therefore to split the training set into two parts: one is used to train the weak models, and the other is used to train the meta-model to combine their predictions. However, given the limited number of traces available in the targeted datasets, we chose to train the meta-models on the same data as the weak models.[2] The potential risk of overfitting is therefore taken into consideration in our experiments, and to mitigate it, we used the early stopping mechanism.

Scope of the Experimentations. The core idea of our work is that a meta-model should be able to outperform a simple bagging-based aggregation, especially in contexts that are less favorable to ensemble methods. Nevertheless, since the objective of this study is precisely to address the issue of hyperparameterization, we deliberately avoid relying on a single carefully tuned meta-model. Instead, we train 30 meta-models with hyperparameter

[2]We also tried to train on the validation dataset, but the results were generally worse due to the small number of data. Results have thus been omitted.

configurations randomly selected from the grid search provided in Table 3.3. By analyzing the variability of the results for the 30 meta-models, we are able to verify if the stacking approach is robust or if it requires a careful meta-model hyperparameterization. Note that the 30 meta-models are the same for all ensemble sizes, in order to properly study the meta-models behavior when the ensemble size increases. Our performance criterion is the convergence of Guessing Entropy and the Na metric.

Table 3.3: Hyperparameter search space for MLP meta-models.

Hyperparameter	min	max	step
Number of layers	2	8	1
Number of neurons	100	1000	100
Activation	Relu, Elu, Selu, Gelu, Tanh		
Epoch	100 | Early stopping : Val loss Patience 20		
Learning Rate	0.0001		
Mini Batch	100		
Optimizer	RMSprop		
Loss	Categorical Crossentropy : metric accuracy		

3.5 Experiments Results

In this section, we report an experimental exploration of the stacking method on several publicly available dataset, whose goals are to assess the soundness of such a technique in profiling vertical SCA context to highlight its advantages and inconvenients and to compare it with the bagging.

3.5.1 Results on ASCADF 0d and ASCADV 0d

In this section, we present the experimental results obtained on two datasets incorporating first-order masking countermeasures: ASCADF 0d and ASCADV 0d. These two datasets provide contrasting contexts. On the one hand, ASCADV 0d represents a scenario that is particularly favorable for ensemble learning, thanks to the good individual performance of weak models and the diversity of their predictions. On the other hand, ASCADF 0d also offers good diversity in predictions, but the weak models are characterized by a significant performance gap between them, which can have a major impact on the quality of ensemble learning.

Stacking Results. The results of our experiments are summarized in Table 3.4. On ASCADF 0d, the best meta-model was trained on the predictions of the 4 best weak models. This meta-model successfully performed the attack in 203 traces, reducing the number of traces required by 81.69% (compared to 1 109 traces for the best weak model). A similar performance improvement was obtained on ASCADV 0d with a meta-model trained on the predictions of the 5 best weak models that successfully performed the attack in 582

traces, reducing the number of traces needed to succeed in the attack by 80.42% (compared to $2\,973$ traces for the best weak model). If we look at the performance of the best meta-models obtained on each ensemble size, we see that stacking improved the overall attacks performance on both datasets by 60% and 70% respectively.

Table 3.4: Stacking on ASCADF 0d and ASCADV 0d datasets. *Nb success* column refers to the number of meta-models that improved attack performance compared to the best weak model. Min, Max and Mean Na values are estimated considering only such *Nb success* meta-models. The best result is highlighted by a green cell.

Size of Ensemble	Nb success (Na <1109)	Na			Improvement in number of traces
		Min	Max	Mean	
2	30/30	371	853	576	66.54%
3	23/30	368	1098	696	66.81%
4	24/30	203	1064	680	81.69%
5	23/30	342	1062	674	69.16%
6	14/30	452	1043	588	59.24%
7	13/30	450	1070	604	59.42%
8	18/30	357	1086	666	67.80%
9	17/30	377	814	589	66.00%
10	15/30	427	989	631	61.49%

(a) Stacking on ASCADF 0d.

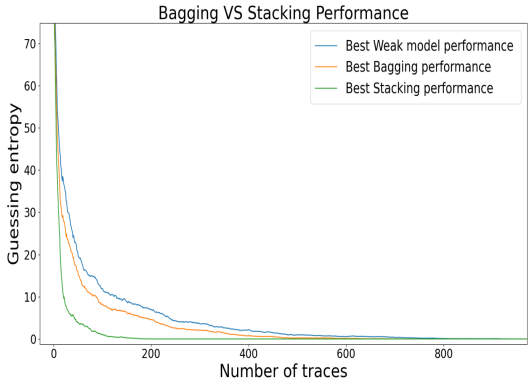
Size of Ensemble	Nb success (Na <2973)	Na			Improvement in number of traces
		Min	Max	Mean	
2	21/30	673	2448	1306	77.36%
3	11/30	635	2306	1533	78.64%
4	11/30	626	2879	1509	78.94%
5	9/30	582	2601	1341	80.42%
6	8/30	789	2693	1427	73.46%
7	7/30	604	2143	1327	79.68%
8	6/30	678	2475	1412	77.19%
9	5/30	607	2909	1606	79.58%
10	3/30	745	2502	1385	74.94%

(b) Stacking on ASCADV 0d.

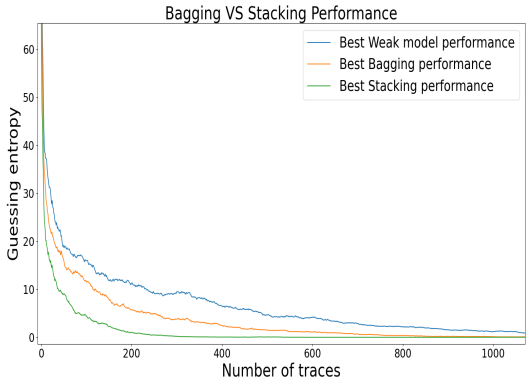
We also notice on both datasets that by increasing the ensemble size, the number of meta-models that improved attack performance tends to decrease. Interestingly, this phenomenon is not caused by the addition of less performing weak models that would degrade the overall quality of predictions. On the contrary, the integration of additional weak models, even less accurate, provides useful diversity, which makes the meta-model's learning task easier. This makes them too complex for the given task, allowing them to easily memorize the training data by heart rather than extracting generalizable information. As a result, they overfit almost immediately, without really being able to exploit the richness of the combined predictions. This may be confirmed by observing the behaviour of the early-stopping mechanism: on ASCADF 0d up to an ensemble size of 5 weak models, the meta-models began to overfit in an average of 25 epochs, but from an ensemble size of 6 weak models, the average learning epoch before overfitting is only 2 epochs. The phenomenon is even more pronounced on ASCADV 0d, where the weak models both have good diversity in their predictions and perform well individually. In this case, overfitting occurs almost from the start of training, *i.e.*, after 3 epochs for the ensemble size of 2 weak models and 2 epochs for the larger ones.

As a result, on these two datasets, the use of stacking ensemble learning resulted in significant improvements over the performance of weak models taken individually. However, in both cases, an excessive increase in the size of the ensemble leads to a high risk of overfitting, which limits the ability of meta-models to extract generalizable information.

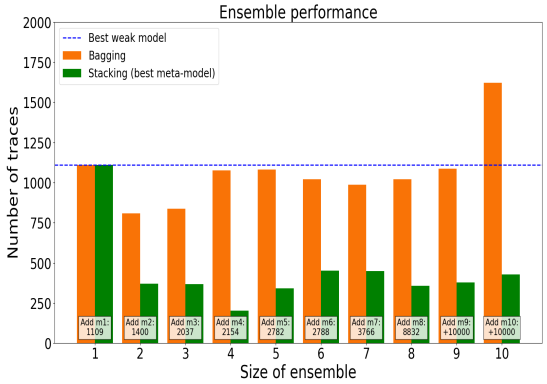
Comparison with Bagging. The best stacking and bagging attack performance on both datasets are depicted in Figure 3.3a and Figure 3.3b. The behavior of the attack performance across all ensemble sizes are depicted in Figure 3.3c and Figure 3.3d. We can see that stack-



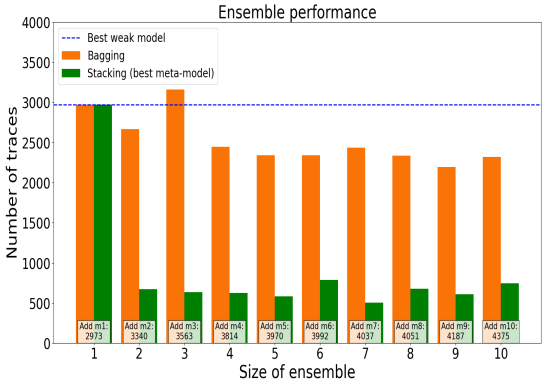
(a) ASCADF 0d GE of the best stacking and bagging ensemble.



(b) ASCADV 0d GE of the best stacking and bagging ensemble.



(c) ASCADF 0d Comparison of stacking and bagging across all ensemble sizes.



(d) ASCADV 0d Comparison of stacking and bagging across all ensemble sizes.

Figure 3.3: Comparison of stacking and bagging ensemble on ASCADF 0d and ASCADV 0d. For Figures (c) and (d), each box lying at the bottom of the i-th column, informs about the performance (in terms of Na) of the i-th weak model.

ing converges faster and allows us to obtain higher attack performance than bagging. In particular, the results on ASCADF 0d show that the bagging process is strongly affected when the ensemble contains weak models with a significant performance gap. Adding less performing weak models tends to reduce the effectiveness of bagging, to the point where the ensemble performance can become worse than that of the best individual weak model. This outcome is naturally explainable given the deterministic average-based nature of bagging aggregation. By contrast, stacking aggregation is less impacted by such variability in weak model performance, as the meta-model is able to learn the relevance of each weak model and weight their contributions accordingly. It is worth noting that, in the case of ASCADV 0d, which provides a favorable context for the use of ensemble methods, bagging aggregation tends to work more effectively. However, we observe that stacking aggregation significantly improves generalization and attack performance, regardless of the ensemble size.

As a result, by comparing the performance obtained with the best meta-model and that

obtained with bagging, our experiments on the two datasets highlight a clear benefit for stacking. The latter provides much more significant improvements in attack performance, consistently outperforming bagging. Moreover, unlike bagging, which can be strongly affected by performance disparities among the weak models within the ensemble, stacking always improves attack performance compared to the best individual weak model.

3.5.2 Results on ASCADV 100d and AES HD

In this section, we present the experimental results obtained on two datasets: ASCADV 100d, which includes a first-order masking countermeasure and a jitter simulation designed to desynchronize traces, and AES HD, which does not include any countermeasures but is characterized by a particularly high level of noise. These two datasets offer contrasting contexts for evaluating ensemble learning methods. On the one hand, ASCADV 100d is characterized by a lack of diversity in the predictions of the weak models, limiting the benefit of combining them. On the other hand, AES HD presents good diversity between the predictions of weak models, but their overall performance remains very poor, most of them barely above the random guessing. Both of these cases provide interesting unfavorable conditions for evaluating ensemble learning.

Table 3.5: Stacking on ASCADV 100d and AES HD datasets. *Nb success* column refers to the number of meta-models that improved attack performance compared to the best weak model. Min, Max and Mean Na values are estimated considering only such *Nb success* meta-models. The best result is highlighted by a green cell.

Size of Ensemble	Nb success (Na <1792)	Na			Improvement in number of traces
		Min	Max	Mean	
2	30/30	429	1172	808	76.06%
3	30/30	423	1256	735	76.39%
4	30/30	362	1160	763	79.79%
5	30/30	369	1141	711	79.40%
6	30/30	352	1070	700	80.35%
7	30/30	351	1130	742	80.41%
8	30/30	351	1333	741	80.41%
9	30/30	369	1097	737	79.40%
10	30/30	369	1137	717	79.40%

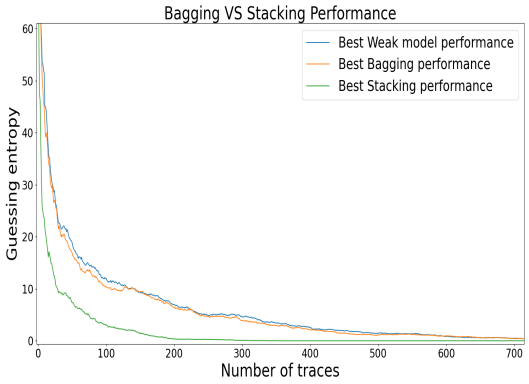
(a) Stacking on ASCADV 100d.

Size of Ensemble	Nb success (Na <22034)	Na			Improvement in number of traces
		Min	Max	Mean	
2	25/30	1365	4179	2212	93.80%
3	27/30	1507	20542	2704	93.16%
4	28/30	1324	11394	2286	93.99%
5	28/30	1251	8014	2038	94.32%
6	27/30	1253	9641	1988	94.31%
7	29/30	1324	12604	2377	93.99%
8	26/30	1315	8947	1962	94.03%
9	27/30	1220	4556	1865	94.46%
10	27/30	1318	9092	2106	94.01%

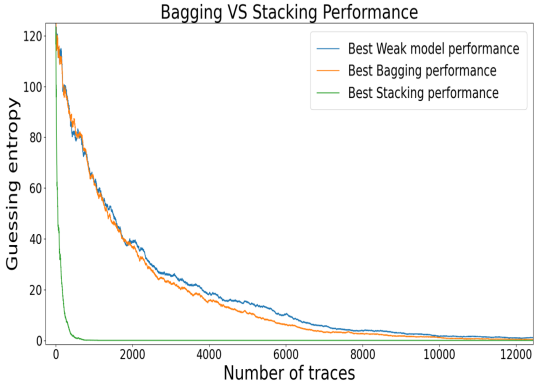
(b) Stacking on AES HD.

Stacking Results. The results of our experiments are summarized in Table 3.5. On ASCADV 100d the best meta-model was trained on the predictions of the 7 best weak models that successfully performed the attack in 351 traces, reducing the number of traces needed to succeed in the attack by 80.41% (compared to 1 792 traces for the best weak model). If we look at the performance of the best meta-models obtained on each ensemble size, we see that stacking improved the overall attacks performance by 80%. An even greater performance improvement was obtained on AES HD with a meta-model trained on the predictions of the 9 best weak models. This meta-model successfully performed the attack in

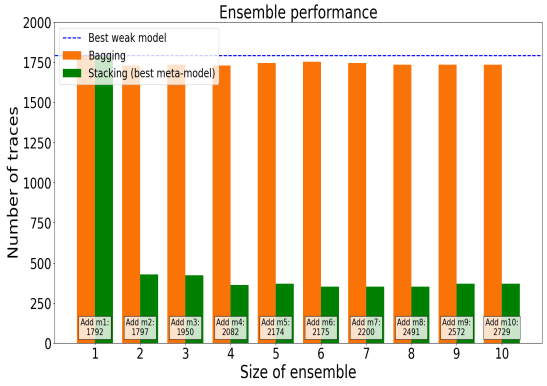
1 220 traces, reducing the number of traces required by 94.46% (compared to 22 034 traces for the best weak model). If we look at the performance of the best meta-models obtained on each ensemble size, we see that stacking improved the overall attacks performance by more than 90%. The use of stacking has shown significant interest for this dataset, allowing to obtain for all ensemble sizes performances below 2 000 traces with ensemble of weak models that have independently attack performances above 20 000 traces.



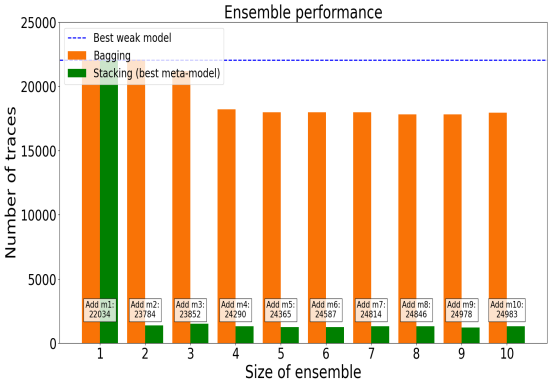
(a) ASCADV 100d GE of the best stacking and bagging ensemble.



(b) AES HD GE of the best stacking and bagging ensemble.



(c) ASCADV 100d Comparison of stacking and bagging across all ensemble sizes.



(d) AES HD Comparison of stacking and bagging across all ensemble sizes.

Figure 3.4: Comparison of stacking and bagging ensemble on ASCADV 100d and AES HD. For Figures (c) and (d), each box lying at the bottom of the i-th column, informs about the performance (in terms of Na) of the i-th weak model.

Unlike previous experiments, on these two datasets, stacking proved to be more robust, adding weak models did not decrease the number of meta-models that improved attack performance. On ASCADV 100d, we even observe that all meta-models improved attack performance for all ensemble sizes. This behavior can be explained by the fact that, in both cases, adding weak models brings few additional useful information to the meta-model. In the case of ASCADV 100d, this is due to the lack of diversity among weak models, while for AES HD, it results from their poor performance.

As a result, on these two datasets, the use of stacking ensemble learning also resulted

in significant improvements over the performance of weak models taken individually. Interestingly, in these experiments, the constraints related to the lack of diversity and poorly individual performance of the weak models acted as a form of implicit regularization during the training of the meta-model. This effect helped mitigate the overfitting observed in the previous experiments, thereby enabling the stacking approach to extract more robust and generalizable informations from the ensemble predictions.

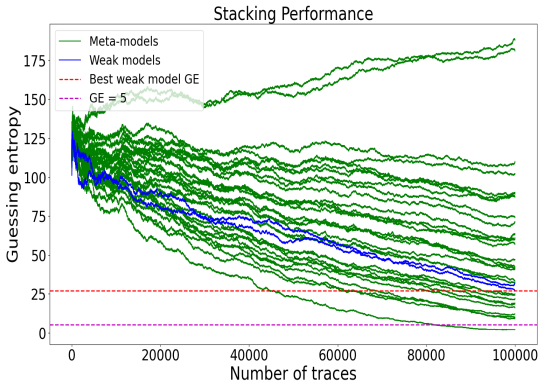
Comparison with Bagging. The best stacking and bagging attack performance on both datasets are depicted in Figure 3.4a and Figure 3.4b. The behavior of the attack performance across all ensemble sizes are depicted in Figure 3.4c and Figure 3.4d. We can see that stacking converges faster and allows us to obtain higher attack performance than bagging. On ASCADV 100d, another limit of the bagging process can be visualized. As reported in Section 3.4.1, in this experiment we fixed the convolutional part of the CNNs in order to quickly obtain weak models able to perform attacks independently. This resulted in weak models with very close predictions. Since the bagging process draws its strength from the diversity of the weak models, the lack of diversity in this case leads to a poorly performing or even worthless ensemble attack. Interestingly, compared to bagging, we noticed that stacking was not affected by the lack of diversity between the weak models. On AES HD, the weak models present naturally a good diversity, thus bagging process is performant. However, as for ASCADV 0d, the performance of bagging is limited by the individual performance of the weak models. In comparison, meta-model training results in better aggregation of weak model predictions and much greater improvement in attack performance.

3.5.3 Results on ASCADV 50d with Full Random CNNs

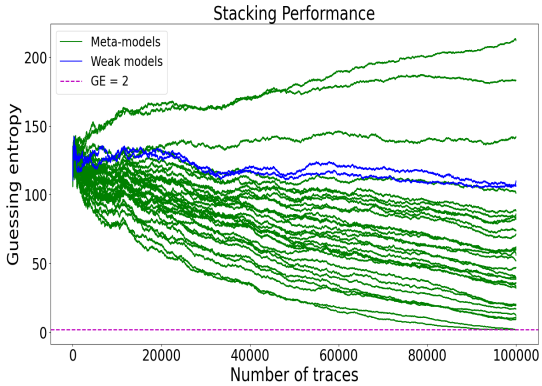
In previous experiments on the ASCADV dataset including desynchronization, in order to obtain individually performing weak models, we fixed the convolutional part of our weak models. In this experiment, in order to have a more objective analysis of the applicability of stacking in the presence of desynchronization, we trained new weak models varying randomly the convolutional part (*i.e.*, whose hyperparameters were randomly selected from a search space provided in Table 3.1c).

As a result, we obtained very weak models that are not able to independently succeed in the attack with the $100\,000$ available attack traces. After applying stacking on different groups of two weak models, we found that in each case several meta-models were anyway able to improve GE convergence. The results of this experiment are shown in Figure 3.5. In the first scenario, the two weak models show a converging trend, but had not enough traces to succeed in the attack. We observe that the best trained meta-model reaches the same performance as the best weak model in less than $50\,000$ traces (instead of $100\,000$ traces). Moreover, when we consider all traces, it obtains an average rank under 5 instead of an average rank of 25 for the best weak model. In the second scenario, stacking is particularly interesting: the two weak models are just starting to converge after $100\,000$ attack traces, while some meta-models reached a rank of 2 with the available traces. Finally, in the third scenario, one of the two weak models converged by placing the correct key guess in last position. Interestingly, stacking was able to correct this effect to provide a correct convergence of the GE. Therefore, stacking appears to be robust independently of the weak

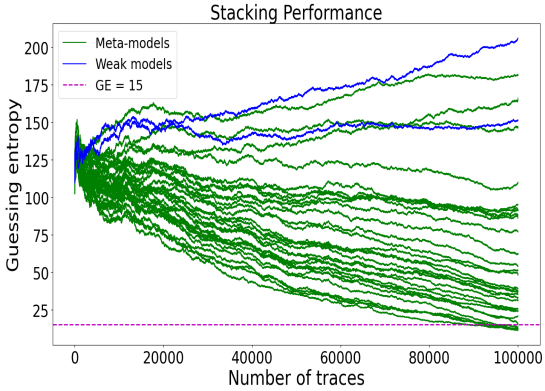
models performance. This is encouraging for the interest of the method and its application in practical evaluation context.



(a) Scenario 1.



(b) Scenario 2.



(c) Scenario 3.

Figure 3.5: Stacking on ASCADV 50d with poorly random weak models.

3.6 Experimental Findings and Interpretation

In this section, we deepen our analysis of the results and emphasize the benefits of stacking as opposed to bagging. We also provide practical recommendations for implementing stacking ensembles, particularly with regard to ensemble complexity and some general properties of a suitable meta-model. Finally, we illustrate that a stacking ensemble can achieve attack performance similar to that of rigorously hyperparameterized architectures reported in the literature, but at a lower cost, thus confirming the relevance of the approach.

Stacking Vs Bagging. The comparison of the attack improvements of bagging and stacking obtained on each dataset are summarized in Table 3.6. If we compare the best performance obtained with the two aggregation methods, we observe that for all datasets, stacking has always been a better choice, allowing to obtain significant attack improvements compared to bagging and non-ensemble models. This significant improvement is due to the fact that stacking allows more sophisticated combinations and transformations of the weak

model predictions by the training of the meta-model. Indeed, the meta-model is able to capture non-linear relationships between weak model predictions, which allows for more granular and accurate ensemble predictions. When the bagging was constrained by the individual performance of the weak models or the lack of diversity within the ensemble, some meta-models proved to be robust and allowed to improve the attack performance. This can be explained by the fact that a properly trained meta-model is able to correctly learn the relevance of each weak model while being able to learn consistent information even over small variations in ensemble predictions. Thus, the choice as well as the number of weak models is less determining when considering the stacking ensemble, which makes it a more flexible method for evaluators interested in using a posteriori ensemble learning. Interestingly, our experiments revealed that even with very few weak models, significant performance gains can already be achieved. This suggests that stacking has the ability to extract relevant information from a small subset of diverse weak models and that there is no need to consider an overly complex ensemble model. On the other hand, stacking has some drawbacks. Since the meta-model training is determinant in the success of the ensemble, this adds a new constraint to the success of the attack. Finally, we observed that the ensemble model often proved to be too complex for the problem. Therefore, the meta-models tends to overfit quickly. Furthermore, the meta-model needs an important amount of data to generalize properly. For example, our omitted experiments in training the meta-model on the validation dataset did not work well due to the lack of data.

Table 3.6: Comparison of bagging and stacking results on all datasets.

Dataset	Best weak model	Bagging improvement in number of traces	Stacking (best meta-model) improvement in number of traces
AES HD	22034	17798 (20%)	1220 (94%)
ASCADF 0d	1109	709 (28%)	203 (81%)
ASCADV 0d	2973	2194 (26%)	582 (80%)
ASCADV 100d	1792	1730 (3%)	351 (80%)

Generalizable Meta-model. A quick analysis of the different meta-models used in our experiments allows us to deduce some general properties for constructing an effective meta-model. In particular, we identified two architectures that consistently proved suitable across all datasets. These two architectures, reported in Table 3.7, did not always achieve the best results, but they always improved attack performance, regardless of the dataset or ensemble size. Interestingly, they are the only two-layer architectures. We interpret this as a consequence of overfitting phenomenon: more complex meta-models tended to overfit in our experiments. Thus, a tiny architecture with only a few layers appears to be the most appropriate choice for a meta-model.

Relieving Hyperparameterization Effort. In Table 3.8, we report a comparison of the performance of our best meta-model across all datasets with various finely tuned architectures proposed in the literature. Even if performance optimization was not at the core of the experiments, we observed that stacking ensemble can provide with less effort similar attack performance to rigorously hyperparameterized architectures. Notably, for ASCADF 0d and ASCADV 0d, we observe that stacking ensembles built from weak models, whose

Table 3.7: Meta-models that always improve attack performance.

Hyperparameter	Architecture 1	Architecture 2
Number of layers	**2**	**2**
Number of neurons	600	300
Activation	elu	tanh
Epoch	Early stopping : Val loss Patience 20	
Learning Rate	0.0001	
Mini Batch	100	
Optimizer	RMSprop	
Loss	Categorical Crossentropy : metric accuracy	

Table 3.8: Comparison in terms of performance with state-of-the-art architectures.

Dataset	Reference	Hyperparameterization method	Na
ASCADF 0d	Arch. in [Ben+20]	-	1146
	Arch. in [Rij+21]	Reinforcement learning	202
	Our best Meta-model	4 random weak models with Na between [1109-2154]	203
ASCADV 0d	Arch. in [Ben+20]	-	1275
	Arch. in [Rij+21]	Reinforcement learning	490
	Our best Meta-model	5 random weak models with Na between [2973-3970]	582
ASCADV 100d	Arch. 1 in [Rob+21]	-	3333
	Arch. 2 in [Rob+21]	Regularization technique	347
	Our best Meta-model	7 random weak models with Na between [1792-2200]	351
AES HD	Arch. in [Kim+19]	-	25000
	Arch. in [Zai+20a]	Visualisation tools	1050
	Our best Meta-model	9 random weak models with Na between [22034-24983]	1220

individual performance is comparable to the slightly hyperparameterized architecture proposed in [Ben+20], can reach performances similar to those of high-performance architectures extensively tuned with reinforcement learning [Rij+21]. The interest of stacking is even more striking for AES HD, where with an ensemble of weak models (with individual performance higher than 20 000 traces), we obtain with less effort similar performance to high-performance architecture proposed by Zaid *et al.* [Zai+20a] that are properly hyperparameterized. The main interest of stacking ensemble is to limit the hyperparameterization effort. The methodology proposed by Zaid *et al.* [Zai+20a] make it possible to efficiently hyperparameterize its architecture and thus obtain a high-performance attack. However, they also assume much knowledge about datasets and an in-depth study of the impact of hyperparameters using data visualization tools (which can be time-consuming). Alternatively, the use of reinforcement learning proposed by Rijsdijk *et al.* [Rij+21] to automate

hyperparameter search is effective, but the related process is extremely time-consuming. Furthermore, the experiments described in Section 3.5.3 are very representative: stacking is able to improve the attack performance even with very weak models that had (almost) not started to converge. This indicates an interest in the method in a more realistic attack scenario. Therefore, stacking may be considered as a suitable approach to avoid the need for the security evaluator to perform a fine tuned hyperparameterization of the neural network architecture.

3.7 Conclusion

In this chapter, we propose a new study of deep ensemble learning applied to profiling vertical SCA. We extend the preliminary results of Perin *et al.* [PCP20] who used bagging to aggregate the predictions of the weak models and we propose to use stacking as a more suitable choice in the aggregation method. Our experimental exploration on several publicly available datasets highlights some of the limitations of the bagging process and shows that stacking can significantly improve attack performance while providing a flexible solution to address these limitations. During our experiments, we observed that stacking ensemble can provide with less effort attack performances similar to those of rigorously hyperparameterized architectures. Therefore, stacking may be considered as a suitable approach to avoid the need for the security evaluator to finely tune the neural network architecture. However, stacking ensemble has proven to be extremely sensitive to overfitting, making it crucial to avoid using overly complex meta-models. In our experiments, two-layer meta-models have always succeeded in improving attack performance. We also noticed that the improvement in attack performance was not correlated with the number of weak models in the ensemble. Indeed, we often found similar improvements across all ensemble sizes. Thus, since the complexity increases with the addition of weak models, we recommend using an ensemble with few weak models.

Chapter 4 Attack of the Clones

Horizontal Collision Attacks with Siamese Neural Networks

Everything is related to everything else, but near things are more related than distant things.

– Waldo Tobler

Resume

In this chapter, we address horizontal side-channel attacks based on collision techniques. We reformulate the problem as a verification task and propose using siamese networks to detect collisions between pairs of traces. Experimental results are presented on a protected RSA implementation featuring state-of-the-art countermeasures, evaluated in both profiling and non-profiling contexts. We focus on attack methodologies suited to constrained evaluation settings with limited preprocessing and profiling, introducing unsupervised methods capable of handling high-dimensional traces without PoI selection or realignment. Simple to implement, the proposed approaches offer valuable tools for evaluators assessing the security of protected RSA implementations.

4.1 Research Issue

In order to protect against side-channel attacks, masking countermeasure is widely considered. Its application on asymmetric cryptographic algorithms, such as RSA implementations, rendered vertical attacks [KJJ99; BCO04] impractical. In addition, elementary single-trace attacks, such as SPA [Koc96], can be effectively thwarted thanks to the well-established principle of atomicity [CCJ04] and the regular exponentiation algorithms [Cor99; JY02]. The research community has therefore turned its attention to more advanced single-trace horizontal attacks. One clear attack path known as collision technique, consists in understanding whether two related operations share a common input operand or not [Wal01].

Several works have explored the use of supervised deep learning to carry out horizontal attacks on asymmetric cryptographic algorithms, targeting for instance RSA and ECC implementations [Car+19; Zai+21a; Shi+22; Bar+22]. However, all these attacks involve a profiling scenario, assuming that the evaluator has access to an open clone of the targeted device, enabling them to fine-tune all of the attack parameters. For instance, if the implementation includes an exponent masking countermeasure [Cor99], the evaluator needs to be able to disable it, or access the random values used during the execution to complete the profiling phase. In practice, such access is often unrealistic, as these values are generated securely and remain inaccessible, even to the evaluator. This limitation is particularly true in composite evaluation contexts, where the target of evaluation integrates a pre-evaluated secure component that lies outside the evaluation scope. Consequently, in many evaluation scenarios, only non-profiling horizontal attacks are practicable.

As discussed in Section 1.3.2, despite their potential, such unsupervised attacks suffer from significant practical constraints: their success strongly depends on the effectiveness of many prior critical pre-processing steps, including cutting, realignment and the selection of PoI. Indeed, all non-profiling horizontal attacks in the literature assume that trace cutting and realignment have been perfectly performed, which is a strong assumption for practical use in evaluations. Furthermore, unsupervised PoI selection remains an open issue, as in unsupervised scenarios it is not possible to benefit from advanced leakage assessment techniques such as SNR or Welch's T-test [SM15]. Instead, the evaluator has to rely upon less robust techniques such as PCA [Spe+15], univariate analysis with some heuristic algorithms [Per+14] or statistical tests [COM23], which are often thwarted by both noise and high dimensionality. Due to these practical constraints, none of the existing unsupervised attacks has proven effective against an implementation running on a defensive arithmetic co-processor. As a result, horizontal attacks are often considered impractical by developers, and the importance of equipping implementations with dedicated countermeasures such as those proposed in [Bau+13] is largely underestimated.

Contributions. In this chapter, we investigate several deep learning methodologies for conducting horizontal collision attacks. Our works are founded upon the one of Carbone *et al.* [Car+19], which currently represents the state-of-the-art in profiling horizontal collision attacks but relies on a highly optimistic scenario for the evaluator: extensive profiling, precise realignment, and careful selection of PoI. In contrast to these ideal conditions, our goal is to address progressively more constrained evaluation contexts, where pre-processing and

profiling are limited.

The contributions in this chapter focus on learning methods based on siamese networks. First, we present a supervised attack reformulated as a verification task that can operate on high-dimensional traces without requiring PoI selection or prior realignment pre-processing. Then we propose two unsupervised approaches that take advantage of structural similarities between traces without relying on any profiling: the first one, named DCCA and published in Communications in Cryptology journal (CiC 2025) [Lla+25], requires prior realignment but avoids any PoI selection, whereas the second one, named BTCCA, fully removes any pre-processing steps at all, and, to our knowledge, constitutes the first horizontal collision attack capable of operating directly on misaligned traces. Together, these results highlight the relevance of siamese networks as a versatile tool for horizontal collision attacks.

4.2 Targeted Implementation and Attack Path

This section provides details on the collision attack path used in this contribution, together with a description of the datasets used to experimentally validate the novel horizontal attacks presented in this thesis. One of them is a simulated dataset, whose advantage lies in its independence from physical measurements or the hardware involved, making it easily reproducible. In addition, the simulation offers total control over the parameters (*e.g.* leakage model, noise level), allowing accurate assessment of attacks in different settings. The other dataset is based on real traces from an implementation protected by state-of-the-art countermeasures [Car+19]. This latter allows to validate the efficiency of our approaches in a realistic environment, which is representative of real-world security evaluation.

4.2.1 Description of the Simulated RSA Dataset

This dataset consists of simulated exponentiation traces of a square and multiply always $2\,048$-bit exponentiation, implementing schoolbook LIM (depicted in Algorithm 3) with an internal multiplier of 32×32 bits. For each exponentiation trace, we randomly generated the message, modulus and exponent to simulate a fully masked environment. As generally considered in the literature, we assume a linear leakage model with respect to the Hamming weight $HW(.)$ of the intermediate values manipulated during algorithm execution. We deliberately chose to place the leakage only on the LIM output.[1] In this way, each t-bit words multiplication of the LIM generates a single leakage point $HW(x[i] \times y[j])$. The result is a modular operation trace of $4\,096$ time points. The complete exponentiation trace is $2\,048 \times 2 \times 4\,096$ points long (*i.e.* $16\,777\,216$ time points). Additionally, Gaussian noise with mean $\mu = 0$ and standard deviation σ is added to the traces to test the behavior of the attacks in different noise conditions. To get closer to the experimental setup of Clavier *et al.* [Cla+12], we used similar levels of noise (*i.e.* $0 \leq \sigma \leq 7$). In the experiments, we notably consider $\sigma = 0$ a null noise, $\sigma = 2$ a moderate noise, $\sigma = 4$ a substantial noise and

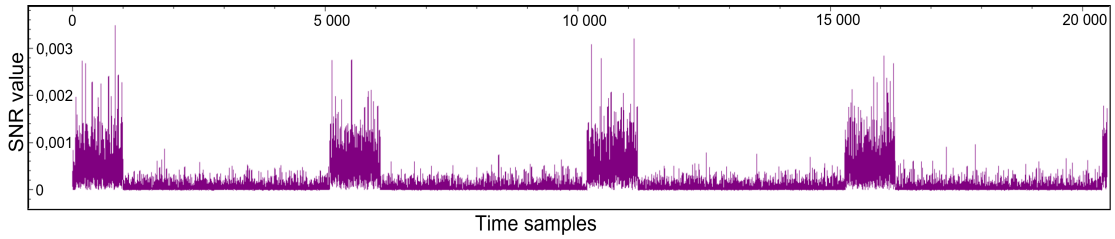
[1] By removing the leakage from the input operands $HW(x[i])$, $HW(y[j])$, we get a more complex problem, as operand collisions will have to be identified only upon the output.

$\sigma = 7$ a strong noise.

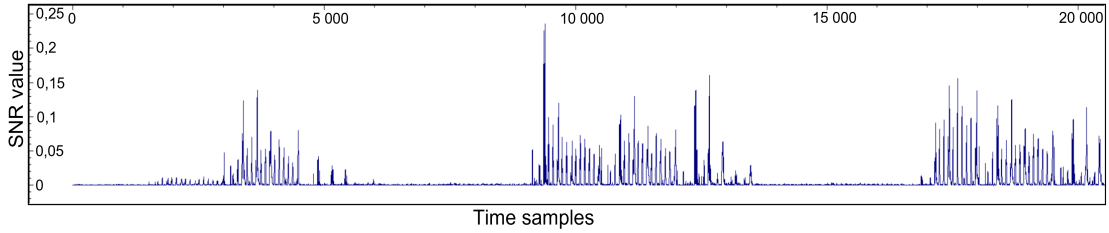
Table 4.1: Configuration of the addition of non-informative points in traces (PoNI).

Dataset	Modular trace dimension	Nb PoNI	% of PoNI
Without PoNI	4 096	-	-
Moderate PoNI	8 192	$4 \times 1\,024$	50
Substantial PoNI	12 288	$4 \times 2\,048$	66
Strong PoNI	20 480	$4 \times 4\,096$	80

In order to evaluate different scenarios, we also added non-informative points within the modular operation traces that will be denoted *PoNI*. For this purpose, we used a normal distribution $\mathcal{N}(\mu, \sigma)$ with a mean $\mu = 30$ and a standard deviation σ according to the noise. These parameters have been arbitrarily chosen to ensure that non-informative points blend visually with the rest of the trace, making them indistinguishable from visual inspection. We tested several configurations with moderate, substantial and strong addition of PoNI. Table 4.1 illustrates the different configurations and their impact on trace dimensions. In order to simulate an environment close to the traces of the protected RSA dataset, we added these non-informative points in distinct blocks of the modular operation traces (*i.e.* every 1 000 time points of the original traces). Figure 4.1 illustrates the operand collision leakage on the resultant simulated traces with strong PoNI configuration and a null noise $\sigma = 0$, with respect to the collision leakage on the protected RSA dataset.



(a) Simulated dataset with Strong PoNI and Null noise ($\sigma = 0$).



(b) Protected RSA dataset.

Figure 4.1: SNRs of operand collision leakage computed with the time-sample wise absolute difference between the modular operation traces of multiplications and their respective consecutive squares. Here the labelling is correct, obtained with the perfect knowledge of the colliding and non-colliding consecutive modular operations.

4.2.2 Description of the Protected RSA Dataset

Introduced in [Car+19], the targeted constant-time RSA implementation runs on a 0.13um 32-bit contact Smartcard IC that features an ARM core SC 100 with an EAL4+ arithmetic co-processor. The implementation (depicted in Algorithm 4) is based on a $1\,088$ bits left-to-right binary square and multiply always exponentiation combined with up-to-date countermeasures: message randomization, modulus randomization and exponent randomization.

For two 512-bit primes p and q, the combination of the three masking countermeasures corresponds to the following equation:

$$m^d \bmod N = \Big((m + r_1 \cdot N)^{d+r_2 \cdot \phi(N)} \bmod (r_0 \cdot N)\Big) \bmod N, \tag{4.1}$$

where m is the plaintext, d is the private exponent, r_0, r_1, r_2 denote three random positive integers of bit-length 64, $N = p \times q$ the modulus of $1\,024$ bits and $\phi(N) = (p-1)(q-1)$, is the Euler's totient function. By consequence all the values involved in the exponentiation have size $n = 1\,024 + 64 = 1\,088$ bits.

The processing flow is regular and independent of the exponent value thanks to the addition of dummy multiplications. For this purpose, the implementation uses 4 memory segments that respectively contain the input of the exponentiation and the intermediate values resulting from the bit-by-bit exponentiation processing. These addresses are denoted by $@(j)$ with $j \in [1..4]$. The input is stored at address $@(j)$ with $j = \text{seg}_{\text{in}}$ and the value of seg_{in} does not vary during the processing. At each loop, a squaring then a multiplication are always executed: the true multiplication is stored at address $@(j)$ with $j = \text{seg}_{\text{acc}}$, whereas the dummy one is stored at address $@(j)$ with $j = \text{seg}_{\text{dum}}$. The values of the two indices seg_{acc} and seg_{dum} vary according to the value of the exponent bit which is treated. Their updating is done without conditional branch thanks to the use of a third index seg_{free}.

This co-processor includes a dedicated memory area based on Montgomery arithmetic and performs elementary operations on 32-bit words. Note that the multiplications and squarings are performed using the same operation, referred to as MMM in Algorithm 4. As squarings are not optimized, both modular operations exhibit similar patterns and identical processing times.

The software part of the targeted RSA implementation does not provide specific security mechanisms to defeat horizontal attacks. For more information on the embedded arithmetic co-processor, and the acquisitions campaign we suggest readers refer to [Car+19].

Dataset Insights. For the sake of completeness, the electromagnetic (EM) exponentiation traces we used have been cut by modular operation, and each modular operation trace consists of around $300\,000$ time points. Since the latter size is unnecessarily large, we sometimes apply a subsampling procedure. For this purpose, we first retain only the window of the first $200\,000$ points, then perform subsampling by a factor of 4. This yields modular operation traces of $50\,000$ time points, which allows us to stay in high dimension while reducing the complexity of our attacks.

A timing desynchronization is clearly observable on raw traces (depicted in Figure 4.2a), in particular around the three large EM peaks marked in red. Hence we perform a realign-

Algorithm 4: Exponentiation Square-and-Multiply Always (from [Car+19])

Input: the masked input m' in Montgomery representation, the masked exponent d' of size n bits, the function MMM initialized with the modulus N' and the Montgomery factor R, and four memory segments $@(j)$ with $j \in [1..4]$.

Output: address $@(1)$ contains $m'^{d'} \bmod N'$

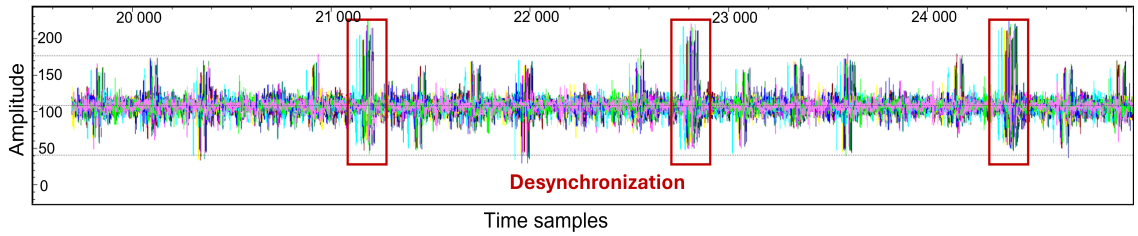
1 $@(1) \leftarrow m$
2 $@(2) \leftarrow 1$
3 $\text{seg}_{\text{in}} \leftarrow 1$
4 $\text{seg}_{\text{acc}} \leftarrow 2$
5 $\text{seg}_{\text{dum}} \leftarrow 3$
6 **for** $i = 0$ ***to*** $n - 1$ **do**
7 $\quad s \leftarrow d'[n-1-i]$ ▷ Read from left to right
8 $\quad \text{seg}_{\text{free}} \leftarrow 9 - \text{seg}_{\text{acc}} - \text{seg}_{\text{dum}}$
9 $\quad \text{MMM}(\text{seg}_{\text{free}}, \text{seg}_{\text{acc}}, \text{seg}_{\text{acc}})$ ▷ SQUARE
10 $\quad \text{seg}_{\text{acc}} \leftarrow \text{seg}_{\text{free}}$
11 $\quad \text{seg}_{\text{free}} \leftarrow 9 - \text{seg}_{\text{acc}} - \text{seg}_{\text{dum}}$
12 $\quad \text{MMM}(\text{seg}_{\text{free}}, \text{seg}_{\text{acc}}, \text{seg}_{\text{in}})$ ▷ MULTIPLY
13 $\quad \text{seg}_{\text{acc}} \leftarrow s \times \text{seg}_{\text{free}} + (1 - s) \times \text{seg}_{\text{acc}}$
14 $\quad \text{seg}_{\text{dum}} \leftarrow s \times \text{seg}_{\text{dum}} + (1 - s) \times \text{seg}_{\text{free}}$
15 **end**
16 $@(\text{seg}_{\text{dum}}) \leftarrow 1$
17 $\text{MMM}(\text{seg}_{\text{acc}}, \text{seg}_{\text{acc}}, \text{seg}_{\text{dum}})$ ▷ Montgomery representation to integer normal form
18 **return** $@(1) \leftarrow @(\text{seg}_{\text{acc}})$

ment pre-processing using a peak detection algorithm. As a result, the realignment was satisfying and all the modular operation traces are well aligned (as depicted in Figure 4.2b).

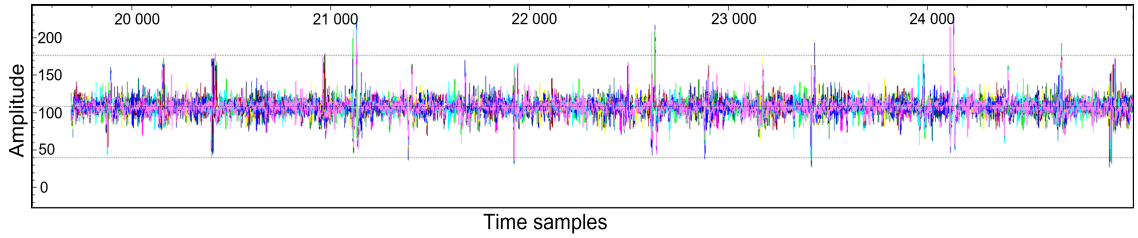
As reported in [Car+19], the leakage of the 32 bits of a single word of the left-hand side operand during a MMM processing is unbalanced: the amplitude of the SNR for the 12-th least significant bit is significantly higher than the others. This characteristic is found for each of the 34 words composing the 1088-bit operand. It may be observed that the SNRs (depicted in Figure 4.2c) corresponding to the 12-th bit of the 34 words are not exactly located at the same time points. This observation was exploited by the authors of [Car+19] to design a horizontal collision attack, which we will detail in the following section.

4.2.3 Collision Attack Path

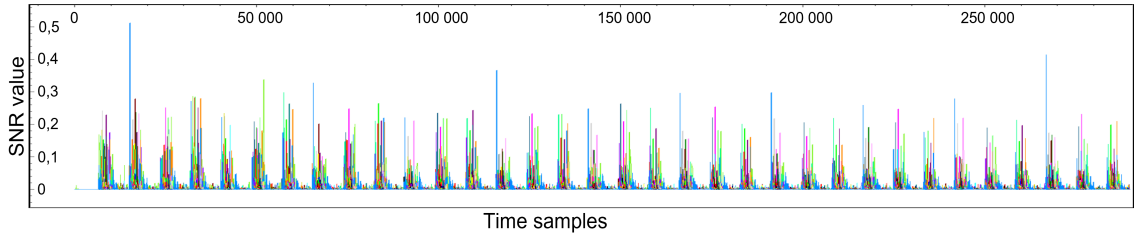
As introduced in Section 1.3.1, the square and multiply always exponentiation, although designed to be regular and resistant against various side-channel attacks, has an intrinsic vulnerability that may be exploited through a horizontal collision attack. Indeed, when the current exponent bit $d'[n-1-i]$ treated at loop index i is 0 (line 7 of algorithm 1), a dummy multiplication is performed and stored in a register dedicated to dummy values at address



(a) Raw misaligned traces overlapped.



(b) Realigned traces overlapped.



(c) Monobit SNRs of the 12-th bit for each of the 34 words of the MMM operand (on 50 000 modular operation traces). Each one is depicted with a different color.

Figure 4.2: Protected RSA traces analysis

$@(\text{seg}_{\text{dum}})$, which does not impact the exponentiation flow stored in the valid register at address $@(\text{seg}_{\text{acc}})$. This implies that the subsequent squaring operation shares the left-hand operand with this dummy multiplication. Therefore, by detecting these collisions, the evaluator can infer all bits of the secret exponent except the last one.

As discussed in Section 1.3.1, early works explored this attack path using cross-correlation techniques [WWM11; HKT15; SSS15]. However, these approaches are ineffective against implementations running on a defensive arithmetic co-processor.

Attack from Carbone *et al.* [Car+19]. With the advance of deep learning techniques, the authors revisited this attack using a supervised, neural-network-based strategy. As introduced in Section 4.2.2, they exploit the fact that the SNR amplitude for the 12-th least-significant bit is significantly higher than for the other bits in the elementary operations of the co-processor. Consequently, they restrict the recovery to these most leaking bits, *i.e.*, the 12-th bit of each 34 words composing the left-hand operands of two consecutive MMM processings, and then to compare the two bit-sets to decide whether a collision occurs. A concise description of the different steps of the attack is provided below:

- *Step 1: Leakage characterization and PoI selection.* For each of the 34 words compos-

ing the operand, an SNR characterization is applied to identify the temporal points presenting the most significant leakage of the 12-th bit. This characterization leads to the selection of 5 000 PoI per word, enabling the creation of 34 distinct training sets. Each set consists of traces reduced to the corresponding PoI and is labeled by the value of the 12-th bit of the i-th word of the left-hand operand of the MMM.

- *Step 2: Supervised learning.* Then, a deep learning model is trained onto each set in order to predict the value of the 12-th bit of the corresponding word. The one-model-per-word strategy leverages the PoI specific to each targeted bit while reducing the complexity of the predictive task.

- *Step 3: Inference and collision detection.* Eventually, for two consecutive MMM operations, the 34 trained models are applied twice. This results in two guessed 34-bit vectors. The collision decision is made by comparing these two vectors: if the number of matching bits exceeds a certain threshold, a collision is considered to have occurred. It may be verified that only 26 bits (or more) among the 34 ones have to be equal to reliably decide whether a collision (or non-collision) took place[2], and therefore to determine whether the corresponding exponent bit is 0 or 1.

Following this attack method, the authors reported a success rate close to 100% for bit recovery (except for the least significant bit, which was retrieved through exhaustive testing), breaking the security of the protected implementation.

It should be noted that this scenario is particularly favorable for the evaluator. Firstly, it assumes the possibility of conducting a profiling attack as well as careful pre-processing: meticulous re-alignment of traces, thorough leakage assessment, and selection of PoI for each of the 34 words in the operand. However, in a time-constrained evaluation context, it is not unusual for the evaluator to lack the resources or time to perfectly execute these tasks. The leakage assessment can be extremely time-consuming, or a satisfyingly accurate realignment may not be achievable. Thus, in this chapter we consider those conditions as the ideal scenario. In the following sections, we progressively consider less favorable contexts by reducing or even removing pre-processing and profiling capabilities, with the aim of exploring methods applicable to more challenging and realistic evaluation settings.

4.3 Supervised Collision Attacks with SNN

In this section, we propose a supervised approach for horizontal collision attacks based on siamese network and contrastive learning. Following the intuition of Cagli [Cag18], we reformulate the collision detection task as a verification problem and train a model to learn a similarity metric that is able to state if some pairs of modular operation traces share the same operand.

After reviewing the principles of siamese network and contrastive learning, we present an experimental exploration of our attack on the protected RSA dataset described in Section

[2] A detailed argumentation of this point is provided in [Car+19] Appendix A.

4.2. To assess the robustness of the approach under realistic conditions, we conduct experiments under two intentionally pessimistic conditions: high-dimensional realigned traces without PoI selection, and then directly onto the raw misaligned traces.

The objective here is to illustrate a resource-constrained evaluator who cannot perform a thorough leakage assessment nor obtain a satisfying realignment. Under these conditions, the profiling neural network attack from Carbone *et al.* (described in Section 4.2.3) is not applicable. Accordingly, we propose an alternative attack methodology designed to remain applicable even when pre-processing is reduced to a minimum.

4.3.1 Siamese Neural Network Learning Paradigm

Let us assume a profiling scenario where an evaluator has acquired a training set of modular operation traces $\vec{X} \in \mathbb{R}^d$ already assigned to the correct labels $Y \in \mathcal{Y}$ (*e.g.* the value of the target variable handled during the acquisition). In contrast with the so-called classification task, that consists in assigning to each trace $\vec{x}_i$ the corresponding label y_i, let us consider the so-called verification task. It consists in assigning to each couple of traces ($\vec{x}_i$,$\vec{x}_j$) a label $y_{i,j}$ that indicates whether or not $\vec{x}_i$ and $\vec{x}_j$ belong to a same class, *i.e.* share the same label $y_i = y_j$.

In deep learning context, a Siamese Neural Network (SNN), *a.k.a.* twin network [CHL05], is specifically designed for the verification task by processing two input data, $\vec{X}_1$, $\vec{X}_2$, through two instances of a unique neural network F, producing latent space representations $F(\vec{X}_1, \boldsymbol{\theta})$ and $F(\vec{X}_2, \boldsymbol{\theta})$. The trainable parameters $\boldsymbol{\theta}$ of the two network instances are constrained to be the same, to ensure that both network instances learn similar representation and map both inputs in a common latent space while providing a symmetric mapping function. The role of such a model is to extract class discriminative features from these two latent spaces and apply to them a distance function D (*e.g.* Euclidean or Manathan distance) measuring similarity or dissimilarity. The output of the distance function $D(F(\vec{X}_1, \boldsymbol{\theta}), F(\vec{X}_2, \boldsymbol{\theta}))$ is used to compute a loss function L and generally seeks to minimize this distance for class-colliding inputs and maximize it for non-class-colliding inputs. To achieve this, the Contrastive loss introduced by Chopra *et al.* [CHL05] is widely used, as described below:

$$L_C = \frac{1}{N} \sum_{i,j=1}^{N} (1 - y_{i,j}) \cdot {y'_{i,j}}^2 + y_{i,j} \cdot \max(0, m - y'_{i,j})^2, \tag{4.2}$$

where, $y'_{i,j}$ is the distance between the learned representations of the two inputs, and $m > 0$ is a margin defining the minimum desired distance between dissimilar inputs. This loss function therefore encourages the model to build a latent space in which similar data are grouped together, while dissimilar ones are pushed sufficiently apart.

Siamese Architecture Design. Modern siamese architectures [Zbo+21; CH21] generally consist of three distinct blocks (as depicted in Figure 4.3): a feature extractor block, typically involving convolutional layers for their strong ability to efficiently capture spatial/temporal information, following by a projector block, usually consisting of dense layers

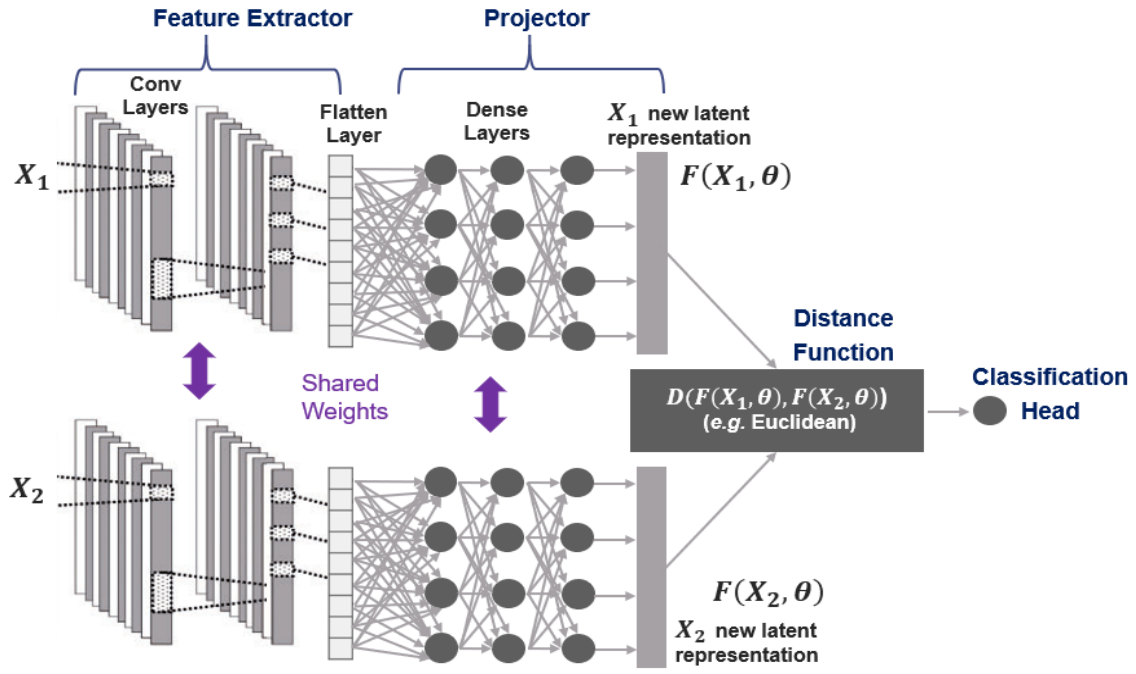


Figure 4.3: Siamese neural network general architecture.

to project the data into a discriminant latent space favorable for the application of a similarity function. Additionally, a classification head may be added onto the projector. This head, often implemented in supervised context as a single neuron output, enables the siamese model to perform specific tasks such as similarity measurement to determine whether or not inputs are similar. By leveraging this modular design, siamese network models can efficiently learn meaningful representations, making them well-suited for supervised contrastive learning [Kho+20] and self-supervised learning paradigms [Zbo+21; CH21]. For some applications of such models in the context of SCA, but in scenarios different from collision attacks, the reader may refer to [Muk+22; LHK22; LLO24].

Behind Convolution Lies Cross-correlation. It is noteworthy that the convolution operation is very similar to the cross-correlation [Lie+16]. Both slide a kernel window through an input by computing a weighted combination with the kernel values. The two operations mainly differ because the convolution flips the kernel along all spatial dimensions before computing the weighted combination while the cross-correlation does not. Here are the equations for a basic example of cross-correlation ($\star$) and convolution ($*$) side-by-side for h-size signal $\vec{S}$ and kernel $\vec{K}$:

$$\begin{aligned} \vec{G} = \vec{S} \star \vec{K} : \vec{G}[i] &= \sum_{u=0}^{h} \vec{S}[i+u]\vec{K}[u], \\ \vec{G} = \vec{S} * \vec{K} : \vec{G}[i] &= \sum_{u=0}^{h} \vec{S}[i-u]\vec{K}[u]. \end{aligned} \tag{4.3}$$

When using the well-known convolutional layers in a deep learning model, the kernel weights are trainable parameters learned during the training process. Thus, these layers

operations may be interpreted as both convolutions or cross-correlations. Interestingly, the commonly used TensorFlow[3] and Pytorch[4] deep learning libraries are implemented using cross-correlation for efficient implementation. Since the kernel parameters in a siamese CNN model are shared for both inputs, the learning process consists in finding the kernel parameters such that cross-correlation on both inputs minimizes the learning objective.

This strong relationship between convolution and cross-correlation makes a siamese CNN a well-suited tool for performing correlation collision attacks, assuming that we can properly control the learning of the kernel parameters. In this chapter, we propose several methods for implementing different horizontal collision attack scenarios. More specifically, in the remainder of this section, we present the use of contrastive loss to conduct a supervised collision attack. Later in Section 4.4 and Section 4.5, we present two unsupervised loss functions that maximize correlation in a latent space to conduct non-profiling horizontal collision attacks.

4.3.2 Experiments Results on Protected RSA Dataset

This section presents an experimental exploration of the use of siamese neural networks and supervised contrastive learning for profiling collision attacks, conducted on the protected RSA dataset described in Section 4.2.

Experiments Settings. For this experimental campaign, we randomly selected $10\,000$ pairs of modular operation traces (multiplication/squaring) for training the siamese models, as well as $2\,000$ distinct pairs for the validation. We conducted our experiments on both raw misaligned traces and realigned traces. For computational reasons, in both cases, we retained only the first $200\,000$ points of the modular operation traces.

The RMSProp optimizer is employed with a learning rate of 0.0001 and a mini-batch size of 100 to optimize the contrastive loss defined in Equation 4.2. We used the weighted Manhattan distance as our distance function [KZS+15], *i.e.*, the sum of the absolute values of the coordinate-wise differences between the two latent representations. In this setup, the weighting sum is applied via an output neuron with a sigmoid activation.

In order to get closer to a real-world evaluation scenario with time constraints, we conducted hyperparameter search by training 20 models whose configurations were randomly picked from the search space depicted in Table 4.2. This exploration was performed directly onto the misaligned traces, our goal being to keep pre-processing steps (*i.e.* PoI selection, realignment) to a minimum. By following this procedure, we were able to evaluate the robustness of the candidate architectures under challenging conditions and assess the ability of the proposed methods to counterbalance the lack of careful pre-processing.

All the experiments are implemented in Python 3.9 using the Keras 2.8 library with TensorFlow 2.8 backend and are run on a workstation equipped with 32GB RAM and an NVIDIA 4080 SUPER with 16GB memory.

[3] https://www.tensorflow.org/api_docs/python/tf/nn/convolution
[4] https://pytorch.org/docs/stable/generated/torch.nn.Conv2d.html

Table 4.2: Hyperparameter search space for SNN models.

Hyperparameter	min	max	step
Projector layers	0	3	1
Projector neurons	100	500	100
Conv/Pool layers	1	4	1
Activation	Relu, Selu, Tanh		
Filters	2, 4, 8, 16		
Kernel size	3,7,11,33		
Pool size	2,4,8		
Epoch	100 | Early stopping : Val loss Patience 20		

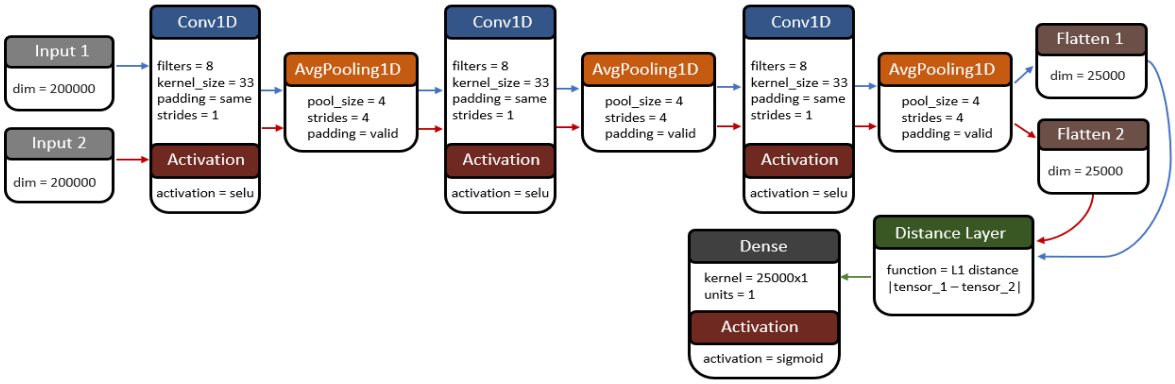


Figure 4.4: Best SNN architecture.

Experiments Results. As a result of this experimental campaign, the performance on the validation data of the 20 trained siamese models ranged from a random decision to an accuracy close to 100%. The best model, depicted in Figure 4.4, is a projector-less model, whose feature extractor block consists of three convolutional layers of 8 filters with a kernel size of 33, each followed by an average pooling layer of size 4.

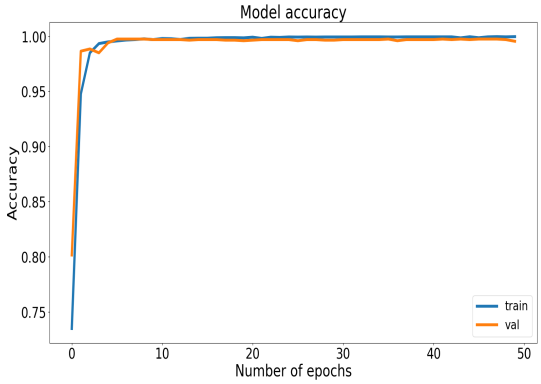
Interestingly, we observed that models without projectors tend to provide better results, although they required a longer convergence time. In addition, architectures with a large convolutional window achieved better performance. We interpret this as being due to an increased ability to deal with desynchronization due to a wider receptive field.

The training metrics for the best SNN model are depicted in Figure 4.5. This model was also retrained on realigned traces to assess its performance in a more favorable scenario, where misalignment is no longer an issue and only high-dimensionality and PoI learning remain as constraints.

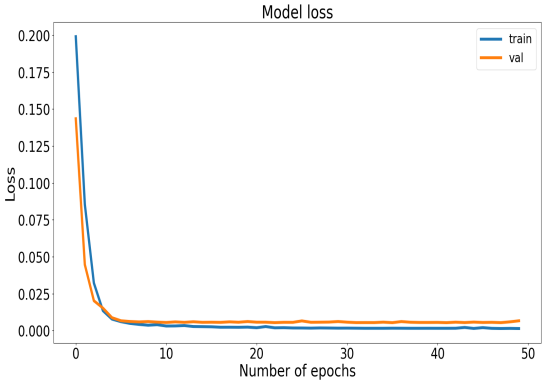
In such a context, the model achieved excellent performance from the first few epochs and remained stable throughout training. This indicates that, thanks to its ability to effectively exploit the information present in high-dimensional traces, the model can easily be dispensed with the PoI selection pre-processing. When dealing with misaligned traces, the model required more epochs to converge (*i.e.*, around 30 epochs to effectively address the desynchronization) and the learning process was less stable. However, it ultimately

achieved performance very close to that obtained on realigned traces, indicating that with this approach, realignment pre-processing can also be omitted safely.

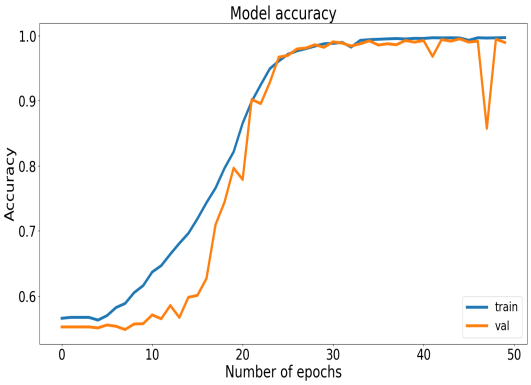
Eventually, we applied the best model to other attack traces and successfully recovered almost all the bits of the secret exponent: *i.e.*, 99.85% on realigned traces and 99.35% on misaligned traces. The remaining bits may be recovered by exhaustive search, as discussed in Section 4.6.



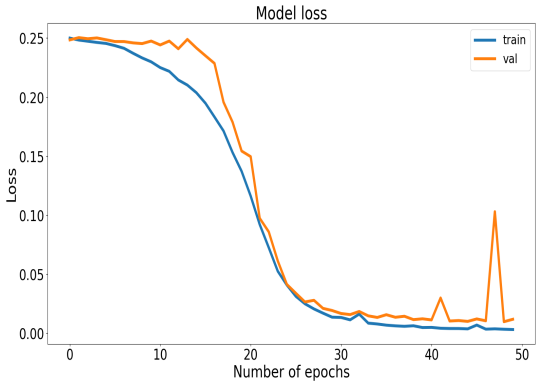
(a) Validation accuracy onto realigned traces.



(b) Validation loss onto realigned traces.



(c) Validation accuracy onto misaligned traces.



(d) Validation loss onto misaligned traces.

Figure 4.5: Training metrics of the best SNN model.

4.3.3 Take-Away Messages

In this section, we presented a collision-based profiling horizontal attack relying on a siamese network and supervised contrastive learning, deliberately putting us in pessimistic evaluation scenarios where the evaluator has neither the time for an in-depth leakage assessment nor a satisfying realignment.

In contrast to the ideal profiling scenario considered by Carbone *et al.* (described in Section 4.2.3), which decomposes collision detection into 34 binary classifications targeting the 12-th bit of each word composing the operand, and requires a monobit leakage model, our reformulation of the problem as a verification task allows us to train a single model that

learns a similarity metric across the entire pairs of traces. In this way, the siamese network simultaneously exploits the leakage of the 34 bits (and other potentially useful information), getting rid of the dependence on PoI selection and making the attack applicable to high-dimensional traces without pre-processing.

We also observed that, although convergence is slower on misaligned traces, the model's performance approaches that obtained on realigned traces, suggesting that realignment pre-processing can also be omitted.

Furthermore, the verification approach reduces profiling requirements: unlike the Carbone *et al.* approach, which requires knowledge of the operand values for each trace, our method only needs information about the presence of collisions between pairs of traces, making it applicable when an evaluator has only partial information (*e.g.*, from another prior attack) or can make strong assumptions about the presence of collisions.

In the following sections, we explore this avenue by introducing two unsupervised variants of siamese networks that leverage structural similarities between traces without resorting to any kind of profiling, thus confirming the relevance of siamese networks as a generic tool for horizontal collision attacks.

4.4 Unsupervised Collision Attacks with Siamese DCCA

In this section, we present a novel horizontal collision attack based on an unsupervised siamese network, designed to address the challenges of evaluation scenarios without profiling capabilities. Inspired by the supervised attack described in Section 4.3 and classical cross-correlation approaches [WWM11; HKT15; SSS15], our method overcomes the limitations of existing unsupervised techniques, which are often highly sensitive to noise and rely on costly PoI selection.

Analogous to the Big Mac pre-processing presented in Section 1.3.1, which compresses traces to enhance collision detection [Wal01], we introduce a learned compression using an unsupervised siamese network trained to maximize the correlation between pairs of modular operation traces through canonical correlation analysis. This model projects high-dimensional traces into a low-dimensional and highly correlated latent space, better suited for identifying operand collisions. In this way, this method makes it possible to attack, in a fully unsupervised way, high-dimensional noisy traces without the need to select points of interest, with the realignment of the modular operation traces as the only required pre-processing step.

4.4.1 DCCA Learning Paradigm

In this section, we begin with a brief introduction to the canonical correlation analysis (CCA), then we provide details of the deep learning extension known as deep canonical correlation analysis (DCCA). In particular, we provide a detailed description of the canonical correlation loss used to train the DCCA models, which aims to extract maximally correlated

latent spaces.

Basics of Canonical Correlation Analysis. The CCA [Ket71] is a multivariate statistical technique used to project two sets of variables into a common latent space that maximizes their correlation through linear combinations called *canonical variates*. CCA is often used in the field of multi-view learning, which aims to exploit information from multiple data sources (sometimes called views) to improve model performances. In this context, CCA can notably help to combine information from heterogeneous sources (*e.g.* text, audio and video [Sun+20]) by projecting them into a common latent space which is as correlated as possible, in order to obtain a more complete and accurate representation of a given problem.

Let $\vec{X}_1 \in \mathbb{R}^{d_1}$ and $\vec{X}_2 \in \mathbb{R}^{d_2}$ denote random vectors. CCA aims to find pairs of linear combinations from the two vectors, $(\vec{W}_1^* \vec{X}_1, \vec{W}_2^* \vec{X}_2)$, such that the resultant projections are maximally correlated:

$$\begin{aligned}(\vec{W}_1^*, \vec{W}_2^*) &= \text{argmax}_{\vec{W}_1, \vec{W}_2} \text{Corr}\Big(\vec{W}_1^T \vec{X}_1, \vec{W}_2^T \vec{X}_2\Big), \\ &= \text{argmax}_{\vec{W}_1, \vec{W}_2} \frac{\vec{W}_1^T \mathbf{\Sigma_{12}} \vec{W}_2}{\sqrt{\vec{W}_1^T \mathbf{\Sigma_{11}} \vec{W}_1 \vec{W}_2^T \mathbf{\Sigma_{22}} \vec{W}_2}},\end{aligned} \tag{4.4}$$

where $(\mathbf{\Sigma_{11}}, \mathbf{\Sigma_{22}})$ and $\mathbf{\Sigma_{12}}$ are the covariance and cross-covariance matrices, and $(\vec{W}_1 \in \mathbb{R}^{d_1}, \vec{W}_2 \in \mathbb{R}^{d_2})$ are the pairs of canonical directions. Several solutions exist to solve the problem. One of the methods proposed by Martin and Maes [MM79] performs singular value decomposition (SVD) on a matrix $\boldsymbol{T} = \mathbf{\Sigma_{11}}^{-1/2} \mathbf{\Sigma_{12}} \mathbf{\Sigma_{22}}^{-1/2}$.

Classical CCA is limited due to its ability to discover only *linear* relationships between two views. To overcome this limitation, various non-linear extensions of CCA have been introduced, including Kernel CCA [LF00] and Deep CCA [And+13].

Siamese Deep Canonical Correlation Analysis. The DCCA, is a non-linear extension of canonical correlation analysis using neural networks [And+13], which has grown in popularity due to its ability to handle complex non-linear relationships, as opposed to traditional CCA. DCCA has been explored in various feature learning applications, including image and text matching [YM15] and cross-modal subspace clustering [Gao+20]. A DCCA model conjointly trains two distinct neural networks with the aim of finding non-linear transformations that maximize the correlation of the two latent spaces. For this purpose, each input is passed through a neural network (encoder) and a canonical correlation analysis is performed on their outputs.

Note that in this contribution, we modified the traditional DCCA approach by using a siamese architecture. This choice results from the fact that in our targeted implementation, squares are not optimized and are performed as multiplications (to avoid trivial simple power analysis). This leads squares and multiplications execution to be enough similar to bring to indistinguishable patterns with the same processing time. Therefore, unlike the traditional method, which aims to combine data of heterogeneous natures (*e.g.* audio/video [Sun+20] or image/text [YM15]) and requires architectures adapted to each type of input, our approach uses a single architecture adapted to both inputs. The application of this constraint of weights sharing results in the search for a single transformation for both modular

operation input traces that maximizes their correlation. Such a constrained canonical correlation analysis using a single transformation has already been suggested in the literature under the name of *common canonical variates* [NF95].

Let $\vec{X_1} \in \mathbb{R}^d$ and $\vec{X_2} \in \mathbb{R}^d$ denote two random modular operation traces. $\vec{X_1}$ and $\vec{X_2}$ are passed as inputs to the siamese DCCA model, which produce the latent representations $F(\vec{X_1}, \boldsymbol{\theta}) \in \mathbb{R}^o$ and $F(\vec{X_2}, \boldsymbol{\theta}) \in \mathbb{R}^o$, where o is the latent dimension and $\boldsymbol{\theta}$ are the parameters. The objective of the model is to determine the parameters $\boldsymbol{\theta}$ such that:

$$(\boldsymbol{\theta^*}) = \text{argmax}_{(\theta)} \, \text{Corr}\Big(F(\vec{X_1}, \boldsymbol{\theta}), F(\vec{X_2}, \boldsymbol{\theta})\Big). \tag{4.5}$$

To find $\boldsymbol{\theta}^*$, one estimates the correlation objective from the training data, using stochastic gradient descent and backpropagation algorithms, according to the Andrew *et al.* calculation method [And+13]. In this way, let $\boldsymbol{Z_1}$ and $\boldsymbol{Z_2}$ be the representation matrices produced by the two network instances for a random mini-batch of n pairs of training traces.

$$\begin{aligned} \boldsymbol{Z_1} &= \Big(F(\vec{x}_{1,1}, \boldsymbol{\theta}), \cdots, F(\vec{x}_{1,n}, \boldsymbol{\theta})\Big) \in \mathbb{R}^{o \times n} \\ \boldsymbol{Z_2} &= \Big(F(\vec{x}_{2,1}, \boldsymbol{\theta}), \cdots, F(\vec{x}_{2,n}, \boldsymbol{\theta})\Big) \in \mathbb{R}^{o \times n} \end{aligned} \tag{4.6}$$

Let $(\boldsymbol{\Sigma_{11}}, \boldsymbol{\Sigma_{22}})$ and $\boldsymbol{\Sigma_{12}}$ be the covariance and cross-covariance matrices from $\boldsymbol{Z_1}$ and $\boldsymbol{Z_2}$. As remarked in [And+13], when the dimensionality of the features is high, the covariance matrix $\boldsymbol{\Sigma_{11}}$ (or $\boldsymbol{\Sigma_{22}}$) may be singular, making the optimization problem underdetermined. To address this issue, the covariance matrices can be regularized [DBDM03] and $\boldsymbol{Z_1}$, $\boldsymbol{Z_2}$ are centralized beforehand:

$$\begin{aligned} \hat{\boldsymbol{\Sigma}}_{\boldsymbol{11}} = \frac{1}{n-1}\bar{\boldsymbol{Z}}_{\boldsymbol{1}}\bar{\boldsymbol{Z}}_{\boldsymbol{1}}^T + r_1\boldsymbol{I}, \quad \hat{\boldsymbol{\Sigma}}_{\boldsymbol{22}} = \frac{1}{n-1}\bar{\boldsymbol{Z}}_{\boldsymbol{2}}\bar{\boldsymbol{Z}}_{\boldsymbol{2}}^T + r_2\boldsymbol{I}, \quad \hat{\boldsymbol{\Sigma}}_{\boldsymbol{12}} = \frac{1}{n-1}\bar{\boldsymbol{Z}}_{\boldsymbol{1}}\bar{\boldsymbol{Z}}_{\boldsymbol{2}}^T, \\ \text{with } \bar{\boldsymbol{Z}}_{\boldsymbol{1}} = \boldsymbol{Z_1} - \frac{1}{n-1}\boldsymbol{Z_1}\boldsymbol{1}, \quad \bar{\boldsymbol{Z}}_{\boldsymbol{2}} = \boldsymbol{Z_2} - \frac{1}{n-1}\boldsymbol{Z_2}\boldsymbol{1}, \end{aligned} \tag{4.7}$$

where $r_1 > 0$ and $r_2 > 0$ are regularization constants that must be set to relatively small values,[5] $\boldsymbol{I}$ and $\boldsymbol{1}$ are respectively the identity matrix and an all-1 matrix of dimension $n \times n$.

Then, the total correlation of $\boldsymbol{Z_1}$ and $\boldsymbol{Z_2}$ can be calculated by the sum of the top k singular values of the matrix $\boldsymbol{T} = \hat{\boldsymbol{\Sigma}}_{\boldsymbol{11}}^{-1/2}\hat{\boldsymbol{\Sigma}}_{\boldsymbol{12}}\hat{\boldsymbol{\Sigma}}_{\boldsymbol{22}}^{-1/2}$. If we consider the case of $k = o$, the correlation is exactly the matrix trace tr(.) norm of $\boldsymbol{T}$ and the final optimization goal for DCCA loss function is:

$$L_{DCCA} = -\text{min}\sqrt{tr(\boldsymbol{T}^T\boldsymbol{T})}, \tag{4.8}$$

4.4.2 Deep Collision Correlation Attack

This section provides details on how to use the DCCA loss to perform unsupervised collision attacks and explains the underlying intuition that motivated its design.

[5] Following the implementation of the mvlearn library we fixed both constants at $1e^{-3}$.

It is worth noting that, unlike the supervised attack presented in Section 4.3, the siamese DCCA model does not include an output classification layer. It simply aims to project pairs of modular operation traces into a latent space where their correlation is maximized. In other words, the output of the DCCA model is not a label prediction but a new representation of the pairs of traces, strongly correlated. The application of the DCCA model can therefore be considered as a pre-processing step, analogous in role to that of the Big Mac [Wal01], which then makes it easier to apply a collision test by a given statistical distinguisher (as depicted in Figure 4.6).

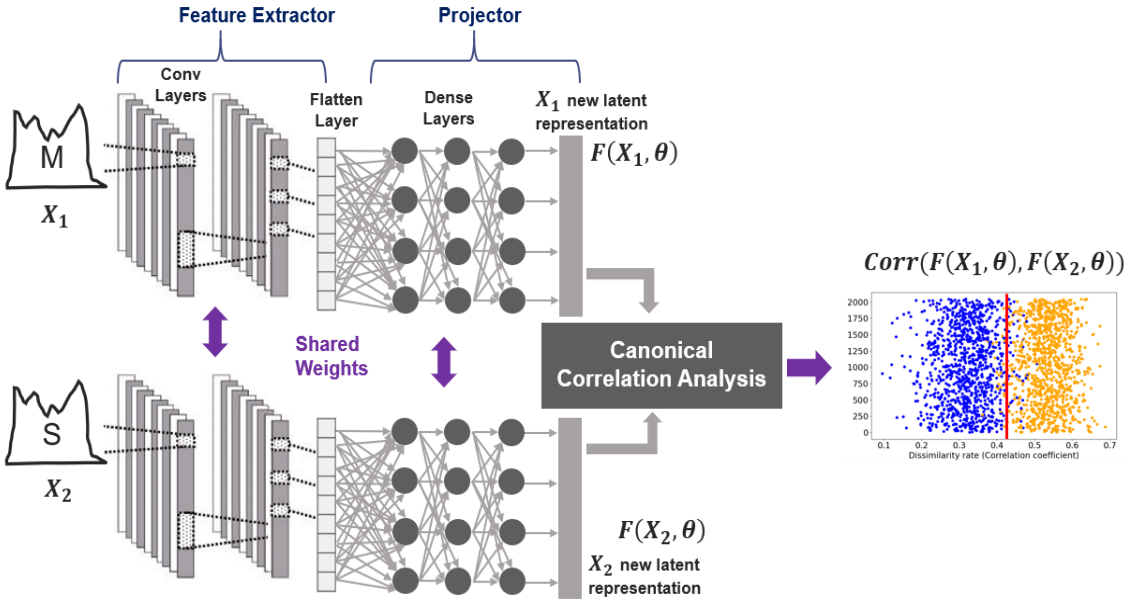


Figure 4.6: Unsupervised collision attack with DCCA.

In this contribution, we consider a collision attack on a regular square and multiply always implementation. Thus, by following the attack scheme described in Section 4.2.3, we use a siamese DCCA model to maximize the correlation in a latent space, for all pairs of modular operation traces (one multiplication and one squaring) that potentially contain collisions.

Then, for each pair of new highly correlated latent representation ($\vec{M'} = F(\vec{X}_1, \boldsymbol{\theta}), \vec{S'} = F(\vec{X}_2, \boldsymbol{\theta}) \in \mathbb{R}^o$), we compute their correlation using the Pearson correlation (obtaining a cloud of correlation values, as it is depicted in Figure 4.6).

$$\rho(\vec{M'}, \vec{S'}) = \frac{\sum_{i=1}^{o}(\vec{m}[i] - \bar{\vec{m}})(\vec{s}[i] - \bar{\vec{s}})}{\sqrt{\sum_{i=1}^{o}(\vec{m}[i] - \bar{\vec{m}})^2}\sqrt{\sum_{i=1}^{o}(\vec{s}[i] - \bar{\vec{s}})^2}} \tag{4.9}$$

Collision Threshold Identification. A threshold must be selected to determine the presence of operand collisions (*i.e.* assign colors in Figure 4.6). When attacking a regular algorithm with a random secret exponent, it is often assumed that the distribution of null and non-null bits is balanced. In this respect, some approaches have suggested dividing the correlation coefficients into upper and lower halves, using a median [SSS15].

However, even if the bit distribution is balanced, it is very unlikely that the number of 0 and 1 bits are strictly identical, so that using the median would inject errors in the threshold placement. Therefore, to propose a more generalizable solution, we decided to identify the collision/no-collision correlation threshold using a clustering algorithm. We chose to use the K-means algorithm due to its relevance in the image binarization process [GBB06; LY09]. The K-means algorithm aims to form k distinct clusters[6] from n unlabeled data. It starts by choosing k initial centroids and assigns each data point to the nearest centroid, then updates the centroids by recalculating the average distance of the points in each cluster. This process is repeated until convergence is achieved and the objective is to minimize total intra-cluster variance:

$$L_{K-means} = \sum_{j=1}^{k} \sum_{i=1}^{n} \|(x_i^j - c_j)\|^2. \tag{4.10}$$

Attack Motivation. As the DCCA is unsupervised, the loss function does not take any labeling into account, and class information is not exploited during training. In this way, some pairs contain collisions while others do not. Thus the present approach relies on the assumption that the presence of collisions should enhance the ability of the DCCA model to maximize the correlation between the two input modular operation traces. In practice, we expect latent representations that are substantially more correlated for colliding pairs than for non-colliding ones, thereby improving the effectiveness of subsequent collision-based attacks.

4.4.3 Experiments Results on Simulated Dataset

This section presents an experimental exploration of DCCA on the simulated dataset introduced in Section 4.2.1, whose advantage lies in its independence from physical measurements or the hardware involved, making it easily reproducible. We assess the relevance of DCCA on simulated traces under different conditions, including the progressive addition of Gaussian noise and non-informative points (PoNI), following the methodology described in Section 4.2.1.

Baselines. In order to evaluate the relevance of DCCA, we conducted several collision correlation attacks using the K-means algorithm to identify the collision/non-collision threshold across different trace representations stemming from state-of-the-art pre-processing techniques, which we summarize below:

- *Raw traces*: direct collision attack on raw modular operation traces as a fundamental baseline
- *PCA*: collision attack after PCA dimensionality reduction. We set the number of principal components to the same value as the dimension of the latent space of our DCCA model for fair comparison.

[6] In our proposed framework, K-means clustering is used to identify a flexible collision/no-collision threshold in the correlation coefficients. Consequently, we set $k = 2$.

- *Big Mac*: collision attack after Big Mac signal pre-processing (described in Section 1.3.1). In order to stay in a realistic scenario, we always used the method naively in a totally unsupervised way, *i.e.* without extracting the LIM leakage points beforehand.

Experiment Settings. The DCCA model used in these experiments (depicted in Figure 4.7) is a siamese CNN with a feature extractor block consisting of one convolution layer of 4 filters of size 64, in order to take the same window size as the average segment size of the Big Mac pre-processing in the scenario without PoNI. This is followed by a pooling layer of size 32 to reduce dimensionality. Finally, the projector uses a single dense layer of 400 neurons to obtain the latent representations. The same architecture was employed across all simulation experiments, including those involving the addition of non-informative points. In all cases, convolutional layers use the ReLU activation function, while the dense projector employs a Sigmoid activation function.

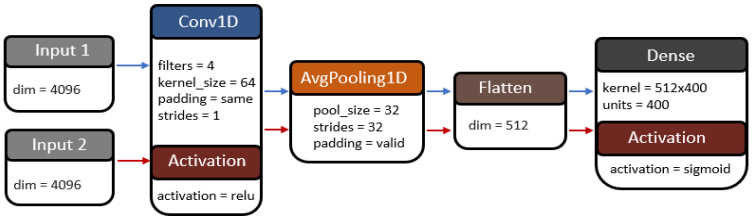
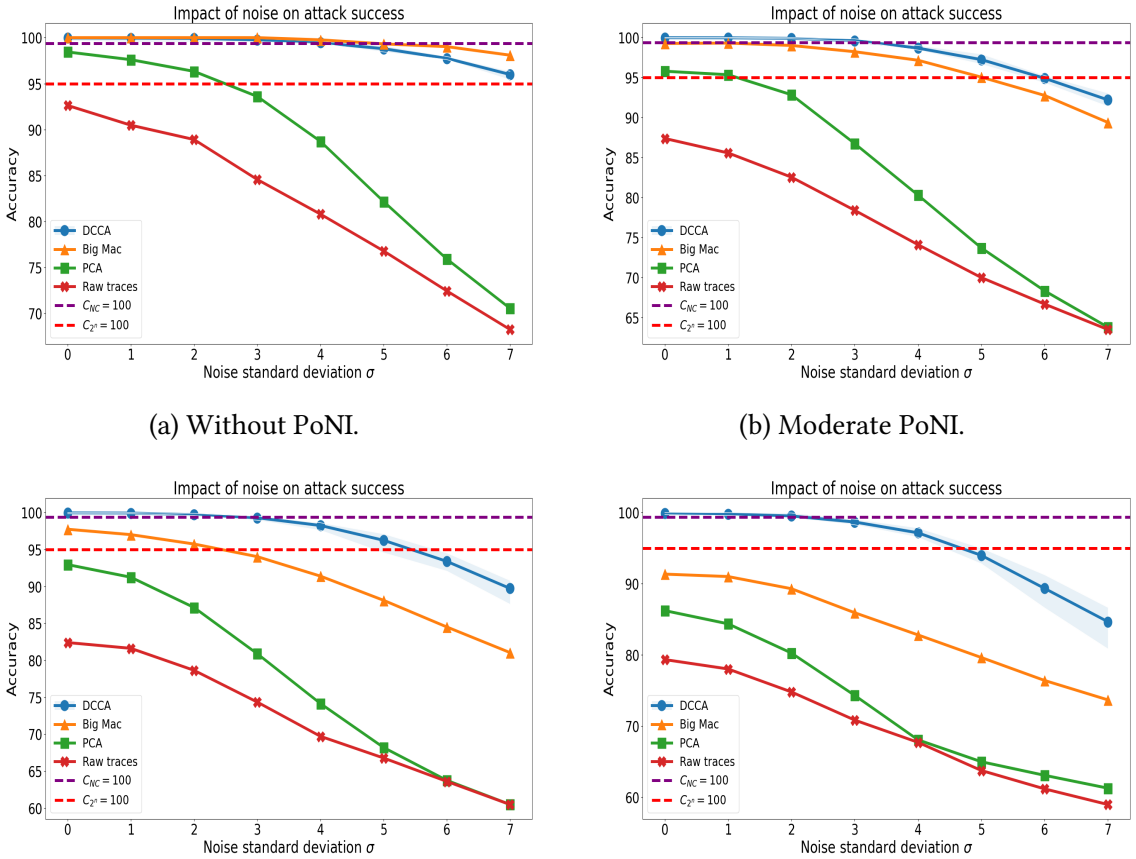


Figure 4.7: DCCA model for simulated trace experiments.

A total of 10 exponentiation traces were generated, 9 traces were used for training the DCCA models (without using label knowledge) and 1 exponentiation trace was kept and used to evaluate the attacks (test traces). In other words, our training set (resp. test set) contains $18\,423$ (resp. $2\,047$) pairs of modular operation traces (multiply/square).

The correlation objective is optimized using the RMSProp optimizer with a learning rate of 0.0001 and a mini-batch size of 100. This mini-batch size was chosen based on its proven effectiveness in previous studies applying DCCA models to EEG signals [QLL18; Liu+19]. All DCCA models were trained for 5 epochs in each experiment. To take into account the influence of weight initialization, each model was trained 10 times per experiment, and the average attack performance is reported with a 95% confidence interval. The experimental results are summarized in Figure 4.8.

Impact of Noise. First of all, we investigated the impact of noise on the performance of our attacks and observed a significant improvement with DCCA in comparison to collision attacks on raw modular operation traces or those reduced by PCA (depicted in Figure 4.8a). Note that, in this experiment, all points are informative but not independent. For instance, during the multiplication of two integers $x = (x[l-1], x[l-2], \cdots, x[0])_{32}$ and $y = (y[l-1], y[l-2], \cdots, y[0])_{32}$, intermediate operations such as $x[0] \times y[0]$ and $x[0]] \times y[1]$ have dependencies, and these dependencies extend across many operations. Consequently, we expect that a good PoI selection obtained via dimensionality reduction can extract the most relevant variance from the leakage and thus improve the effectiveness of the collision attack.



(a) Without PoNI.

(b) Moderate PoNI.

(c) Substantial PoNI.

(d) Strong PoNI.

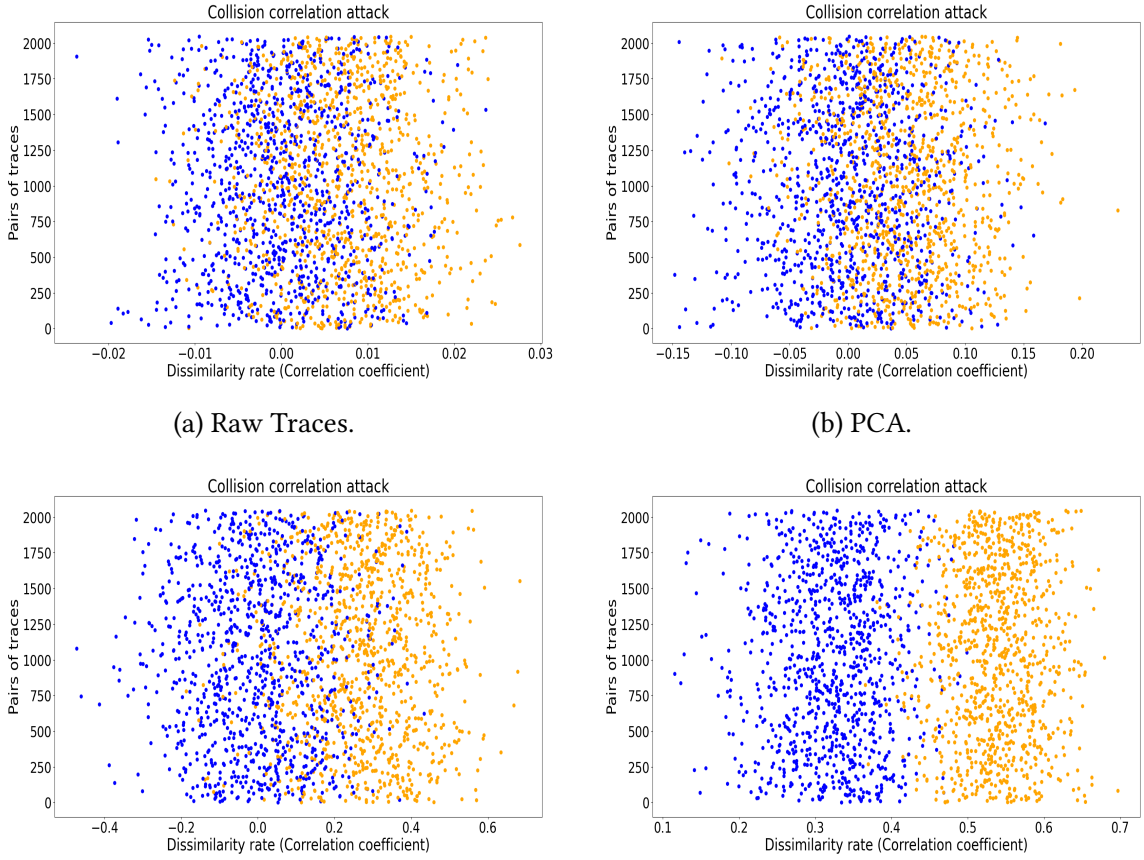
Figure 4.8: Experiment results on simulated datasets for different levels of noise and proportions of non-informative points. The blue curve represents the average performance of our DCCA attacks over 10 runs, along with the 95% confidence interval.

In this regard, we observed that the DCCA models have successfully put the modular operation traces into highly correlated low-dimensional latent spaces, amplifying collisions, even in the presence of significant noise. However, the Big Mac attack was the most effective. It is worth noting that the Big Mac attack excels since the context is favorable for it (*i.e.* the modular operation traces correspond only to the leakage of the multiplier). In addition, pre-processing involving segment averaging in the Big Mac attack reduces the impact of Gaussian noise while highlighting the targeted operand. DCCA achieves comparable results to the Big Mac attack up to a substantial level of noise ($\sigma = 4$), but is less robust in the case of a strong level of noise ($\sigma = 7$). Despite this, DCCA may result in a successful attack for all considered levels of noise (*i.e.* $C_{2^n} \leq 100$). Furthermore, in cases where the Big Mac attack is more effective ($\sigma > 4$), both of these attacks fail to allow a naive remaining attack (*i.e.* $C_{NC} \leq 100$).

Impact of Non-Informative Points (PoNI). Then we analyzed the impact of the addition of non-informative points in the modular operation traces (Figures 4.8b, 4.8c, 4.8d). Interestingly, DCCA proved to be more robust in the presence of non-informative points than the Big Mac, used here in a scenario that is no longer optimal for the latter. Indeed,

the Big Mac attack is particularly impacted by a strong level of PoNI (Figure 4.8d), while there is only a slight impact for DCCA up to a substantial level of noise ($\sigma = 4$).

The robustness of DCCA can be explained by the high ability of CNN to extract features, notably thanks to their sliding window approach on the input traces. We note that in the most extreme scenario, with strong PoNI, the DCCA is the only one able to allow successful remaining attacks (*i.e.* $C_{2^n} \leq 100$). In addition, DCCA allows remaining naive attacks (*i.e.* $C_{NC} \leq 100$) up to a moderate level of noise ($\sigma = 2$).



(a) Raw Traces.

(b) PCA.

(c) Big Mac.

(d) DCCA.

Figure 4.9: Distinguishability of collisions with Strong PONI and Substantial noise ($\sigma = 4$). The correlation coefficients are colored with their ground-truth labels. Blue (resp. orange) values represent pairs of modular operation traces without collision (resp. with collision).

Figure 4.9 depicts the distinguishability of collisions in the correlation coefficients for a substantial level of noise and a strong level of PoNI. We can clearly visualize the benefit of DCCA to distinguish the presence of collisions in comparison to the attack baselines.

It is worth noting that DCCA becomes increasingly unstable above a substantial level of noise ($\sigma = 4$). This instability is even more pronounced in the presence of a large number of non-informative points (Figure 4.8d). This suggests that DCCA could be particularly unstable in a real attack environment. This assumption is confirmed in the next section, where we present our experiments on the protected RSA dataset, thus confronting our

approach with conditions closer to a real-world application.

4.4.4 Experiments Results on Protected RSA Dataset

This section proposes an experimental exploration of DCCA on real traces from the protected RSA dataset described in Section 4.2.2. We notably conduct an analysis of several hyperparameters of the siamese DCCA model with the aim of deducing some general properties that can guide the design of a suitable architecture.

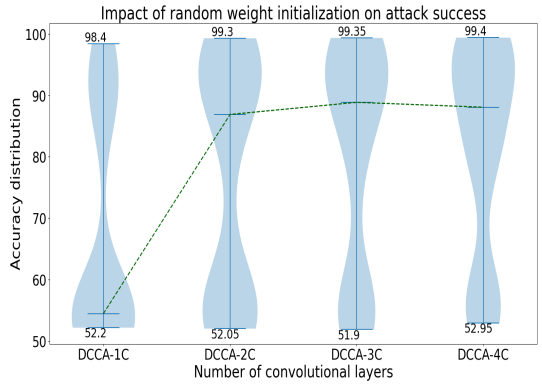
Experiments Settings. For this experimental campaign, we randomly selected $10\,000$ pairs of modular operation traces (multiplication/squaring) for training the DCCA models (without using label knowledge), as well as $2\,000$ distinct pairs for assessing the performance of our collision attacks. For the sake of completeness, each modular operation trace is subsampled using the subsampling procedure described in Section 4.2.2. This yields modular operation traces of $50\,000$ time points, allowing us to stay in a high dimension while reducing the complexity of our attacks. In addition, to enhance training stability and convergence speed, we pre-processed the modular operation traces such that all samples in a mini-batch are normalized between 0 and 1 [IS15]. Note that, all experiments were run on the realigned traces. We attempted to apply the DCCA pre-processing onto the raw misaligned traces but obtained poor results.

In order to analyze the impact of learning time, the DCCA models were trained from 5 to 40 learning epochs. In addition, the DCCA models were trained 100 times to assess the impact of random weights. The RMSProp optimizer, with a learning rate of 0.0001 and a mini-batch size of 100, is used to optimize the correlation objective. For all DCCA architectures, convolutional layers used SeLU activation function and dense layers used Sigmoid activation function.

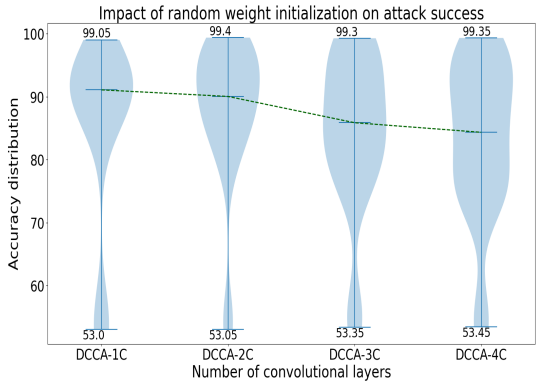
All the experiments are implemented in Python 3.9 using the Keras 2.8 library with TensorFlow 2.8 backend and are run on a workstation equipped with 32GB RAM and an NVIDIA Quadro P4000 with 8GB memory.

Feature Extractor. In order to study the impact of the feature extractor block on the success of the attack, we experimented with several DCCA models based on a convolutional feature extractor block comprising a variable number of convolutional layers (from 1 to 4) without downsizing (*i.e.* without pooling operations), each with 4 filters and a convolution window of size 8. Following this, the extracted representations are projected into a fully connected latent space in the same way as the experiments on simulated data, using a dense layer of 400 neurons acting as a projector. This exploration summarized in Figure 4.10 enabled us to identify the complexity required for the feature extractor to successfully extract fine-grained operand collision leakage.

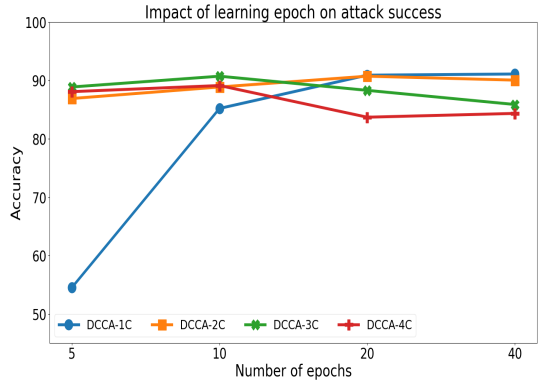
We observed that using a feature extractor with a large number of convolutional layers does not necessarily offer significant advantages. Although the addition of convolutional layers does enable finer-grained feature extraction, this advantage seems to be limited to the early learning stages, as depicted in Figure 4.10a. Indeed, during the first learning epochs, a more complex feature extractor may speed up initial convergence thanks to the richness of



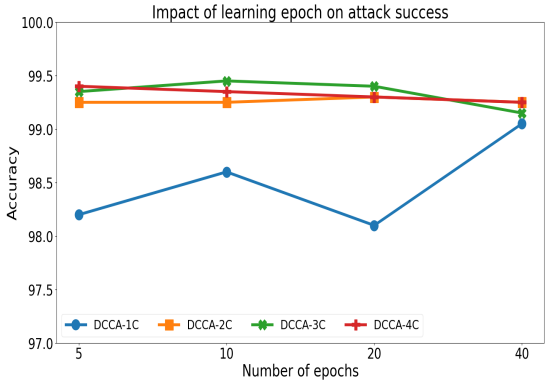
(a) DCCA results for 5 training epochs.



(b) DCCA results for 40 training epochs.



(c) Median of DCCA attacks.



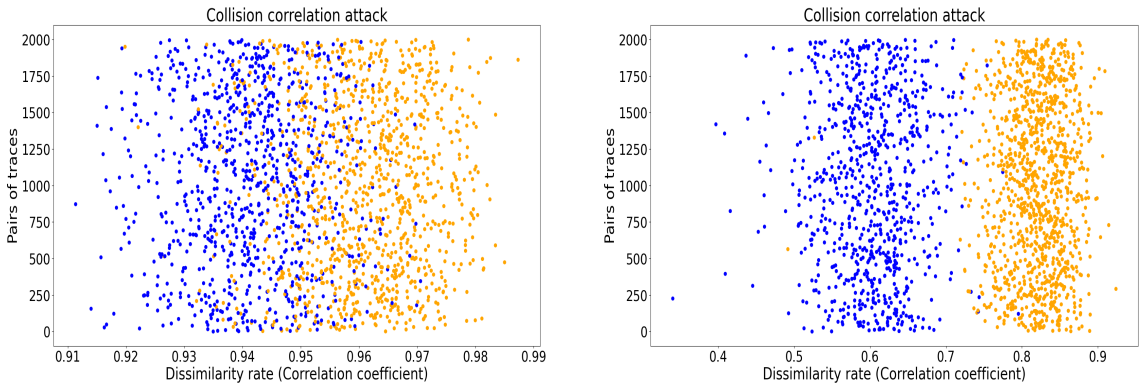
(d) Best attack.

Figure 4.10: Experiment results on protected RSA dataset for different convolutional feature extractors without downsizing.

the representations. However, with many epochs, increasing the size of the feature extractor may lead to a drop in attack stability, as depicted in Figure 4.10b, with the median attack performance decreasing as the feature extractor grows. We observed that simple architectures, such as those with two convolutional layers, tended to lead to DCCA models whose performance gradually stabilized during training (depicted in Figure 4.10c). Conversely, models with three or four layers feature extractor tended towards a gradual drop in stability beyond 10 learning epochs. Indeed, we observed that these more complex models tended to over-correlate latent spaces, preventing collision detection.

Instability. For all DCCA models, we observed a high degree of instability in our attacks, whose success depended heavily on the initialization of random weights in the DCCA models. The impact of weights initialization remains an open issue in deep learning literature [NBS22]. A poor initialization may have a significant impact on the learning process, making convergence more difficult or even impossible. In the context of an unsupervised objective, weights initialization is particularly important since there are no labels to guide the task. Thus, the unsupervised model is expected to discover patterns and representations by itself.

In our case, the main issue is that the DCCA model is only trained to maximize the correlation. Hence, the DCCA model can correlate patterns that do not refer to collisions and successfully solve its learning objective. As a result, an overly correlated latent space may result in an inability to distinguish collisions (as depicted in Figure 4.11a).



(a) Unsuccessful DCCA attack.

(b) Successful DCCA attack.

Figure 4.11: Visual interpretation of successful attack. The correlation coefficients are colored with their ground-truth labels. Blue (resp. orange) values represent pairs of modular operation traces without collision (resp. with collision).

In addition, although the K-means algorithm is effective in determining a threshold to separate correlation coefficients, in some cases, the presence of outliers (*i.e.*, points that differ significantly from the rest of the data) resulting from under-correlated or, conversely, over-correlated pairs of modular operation traces can significantly affect the result. Indeed, a single outlier with a large distance can strongly influence the clustering, leading to the formation of a cluster around it. This phenomenon explains why some of our attacks achieved results close to random guessing.

Handling Instability. To address the issue of weight initialization, most unsupervised deep learning approaches require some form of pre-training. With regard to the DCCA model, in the seminal paper, Andrew *et al.* [And+13] initializes the parameters of each layer of the DCCA model with a denoising autoencoder. However, this instability-handling approach based on autoencoder is not well suited to the collision SCA problem. In particular, training an autoencoder with high-dimensional traces requires very complex architectures with a large number of parameters to be learned. Furthermore, the reconstruction loss does not guarantee the preservation of the collision points of interest.

In our context, we consider this instability to be a manageable problem, as the results are highly interpretable. Indeed, in a one-dimensional data with only two classes, clustering becomes remarkably easy to evaluate visually. A simple scatter plot of the correlation coefficients provides an intuitive view of attack effectiveness: a successful attack produces a clear separation between the two clusters, as illustrated in Figure 4.11b.

An evaluator can therefore train the DCCA model multiple times with different weight initializations, seeking one that converges toward latent representations more favorable for detecting collisions. By quickly inspecting the correlation coefficients, it is then possible

to identify runs that exhibit a clear separation between clusters and thus a high probability of success. To automate this evaluation process, we propose a methodology based on unsupervised clustering metrics.

Firstly, under the assumption that the secret exponent is roughly balanced, a significant cluster imbalance (potentially caused by some outliers) is considered to be an unlikely attack. Therefore, in order to reject all such unlikely attacks, we propose to identify those clusters with non-negligible class imbalance, using the cumulative distribution function (CDF) which gives the probability of a random variable X taking a value less than or equal to a value k.

$$CDF(k, n, p) = P(X \leq k) = \sum_{i=0}^{k} \binom{n}{i} p^i (1-p)^{n-i}, \tag{4.11}$$

where k is the value for which we want to calculate the cumulative probability, n is the total number of trials and p is the probability of success in an individual trial.

Based on the CDF analysis, we define a threshold (arbitrarily set to 1^{-32}) beyond which we consider the cluster to be non-negligibly imbalanced. Therefore, if one of the two clusters is below this threshold, we consider the attack unlikely and we reject it.

Secondly, we applied an unsupervised cluster validity index to attacks that pass the binomial test in order to rank them and identify the most effective ones. Arbelaitz *et al.* [Arb+13] propose an experimental work that compares 30 cluster validity indices in many different environments, and the so-called *silhouette score* obtained the best results in many of them. The silhouette score [Rou87] is a measure of clustering quality that evaluates how points in the same cluster are similar to each other, and how different they are from points in other clusters. It assumes that a good clustering solution encompasses compact and well-separated clusters. The silhouette score $S(x)$ computation proceeds by evaluating the individual *silhouette coefficient* $s(x_i)$ for each data point x_i. This coefficient is calculated as the difference between *intra-cluster cohesion* $a(.)$ and *inter-cluster separation* $b(.)$, divided by the higher of these two values.

$$\begin{aligned} S(x) &= \frac{1}{n}\sum_{i=1}^{n} s(x_i), \quad \text{with } s(x_i) = \frac{b(x_i) - a(x_i)}{\max(a(x_i), b(x_i))}, \\ a(x_i) &= \frac{1}{|C_I| - 1}\sum_{j \in C_I, i \neq j} d(x_i, x_j), \quad b(x_i) = \min_{J \neq I} \frac{1}{|C_J|}\sum_{j \in C_J} d(x_i, x_j), \end{aligned} \tag{4.12}$$

where $d(x_i, x_j)$ denotes the distance between points x_i and x_j, $a(x_i)$ is the average distance between point x_i and the other points in the same cluster C_I and $b(x_i)$ is the average distance between point x_i and all points in the nearest cluster C_J, where the nearest cluster is the one that minimizes this distance. For a global evaluation of a clustering, the silhouette score $S(x)$ represents the average $s(x_i)$ coefficients for all n points x_i. An overall silhouette score close to 1 indicates a good clustering quality, while a score close to -1 indicates that the points could be better allocated to a different cluster.

One may observe in Figure 4.12 that the accuracy, although less stable in the rank, follows the same trend as the silhouette score. Therefore, ranking attacks according to silhouette score allows us to approximate in a fully unsupervised way their accuracy and correctly distinguish the most likely successful attacks.

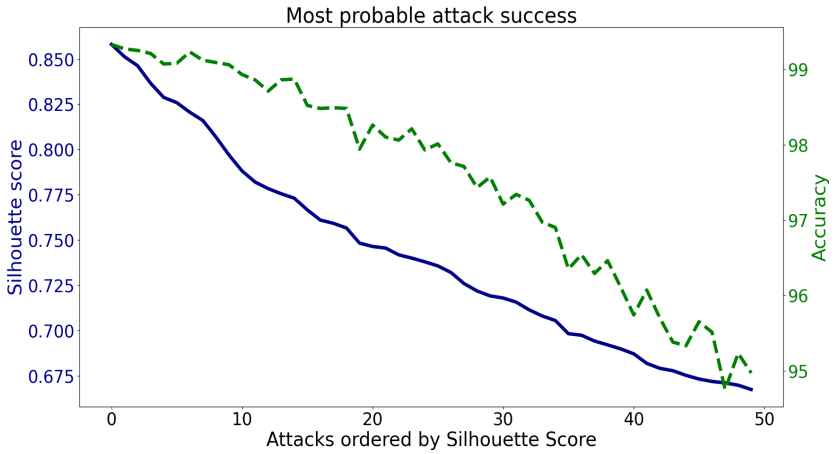


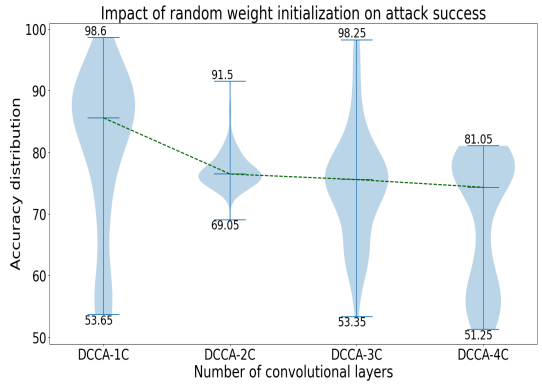
Figure 4.12: Top 50 most likely attacks ranking by silhouette score (blue curve) and their true corresponding accuracy (green dotted curve).

This procedure allowed us, across all experiments, to identify at least one of the most effective DCCA models, achieving strong attack performance at every learning stages, as depicted in Figure 4.10d. This allows us to conclude that except for single convolutional feature extractor (DCCA-1C), extended training of DCCA models had only a slight impact on the performance of the best DCCA attacks. As a result, a two convolutional feature extractor trained for 20 epochs, seems to be a good balance between stability and best attack performance.

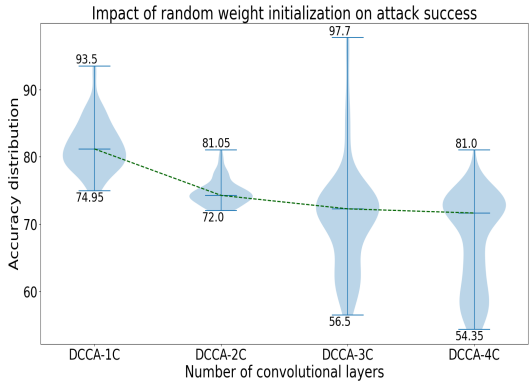
Dimensionality Reduction The feature extractor block previously considered only involved convolutional layers, without intermediate pooling layers. In this way, the spatial dimensionality is not compressed throughout the feature extractor, meaning that dimensionality reduction is entirely carried out in the fully connected dense projection layer. Thus, the projector, whose job is to produce the final latent space where the loss function is computed, acts as a genuine bottleneck, drastically compressing the extracted features. This architectural design comes with its own challenges. Although this design preserves spatial granularity up to the projection stage, which in some cases may be advantageous for capturing meaningful local relationships in the traces, the lack of downsizing in the feature extractor may limit the ability of the DCCA model to reduce noise and redundant information. Furthermore, the drastic bottleneck constraint imposed on the dense projector can lead to a significant loss of collision leakage.

In order to reduce the complexity of intermediate representations in a structured way, while at the same time reducing the downsizing task of the projector, we experimentally tried to integrate a progressive dimensionality reduction directly into the feature extractor block. To this end, we added successive averaged pooling layers with a pool size and stride of 2 after each convolutional layer.

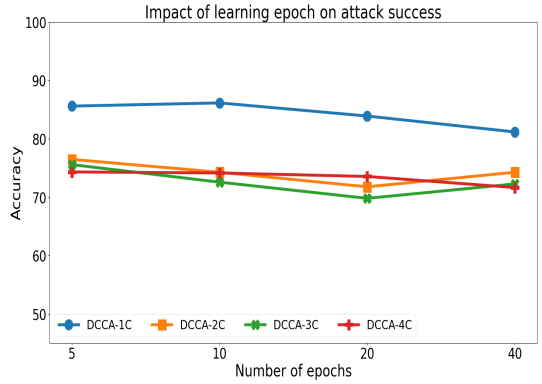
The results of these experiments are depicted in Figure 4.13. We observed significant negative impacts from this downsizing. The overall DCCA attack performance was considerably degraded with increased instability. Indeed, such a downsizing led to over-



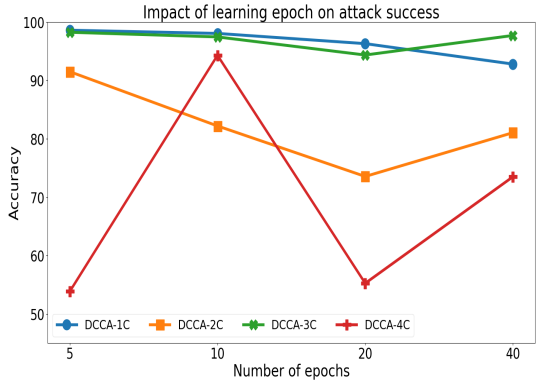
(a) DCCA results for 5 training epochs.



(b) DCCA results for 40 training epochs.



(c) Median of DCCA attacks.



(d) Best attack.

Figure 4.13: Experiment results on protected RSA dataset for different convolutional feature extractors with downsizing.

correlation of representations, which prevented collision pairs from being distinguished from non-collision ones. These results highlight the inherent difficulty of performing effective dimensionality reduction in unsupervised contexts. In comparison to the simulation, where large pooling layers efficiently reduced noise and improved collision detection (in a similar way to the Big Mac pre-processing), performing such a downsizing on real traces remains an open issue.

Need for Translational Invariance. The lack of downsizing with pooling operations in the feature extractor limits the DCCA model's ability to learn translational invariant representation [Jad+15; MMD20]. Indeed, pooling operations ensure that small spatial shifts or translations in the input data do not significantly affect the extracted features. In contrast, for a feature extractor without such a downsizing, the translational invariance property is limited, relying solely on convolution operations due to shared weights and kernel shifts. This limitation may explain why some experiments we performed on misaligned datasets[7] showed unsatisfying results, given that translation invariance property is the basis for CNN to handle desynchronization [CDP17; Tim19].

[7]Such experiments are not reported in the present chapter.

Projector Size. Since downsizing is performed entirely by the projector, we studied the impact of its size (*i.e.*, the number of neurons in the unique dense layer that composes it) on our DCCA attack. The results of these experiments are depicted in Figure 4.14. We observed that, although the projector size does not have a direct impact on the performance of the best attacks, it can strongly impact their stability. By choosing a large projector size, we limited the drastic bottleneck constraint and improved the overall performance of the DCCA models. However, considering a large-scale projector brings some architectural constraints. First of all, it significantly increases the number of model parameters, due to the fully dense connections with the feature extractor output, which is extremely high (*i.e.* 200 000 dimensions in our experiments[8]). This increases the memory and computational costs of training.[9] Furthermore, DCCA loss requires costly operations such as matrix inversions, which cannot be efficiently computed on such a high dimension.

As a result, these limitations underline the importance of striking a balance between the feature extractor that expands dimensionality and the preservation of information in the projector to handle the practical constraints imposed by available computing resources.

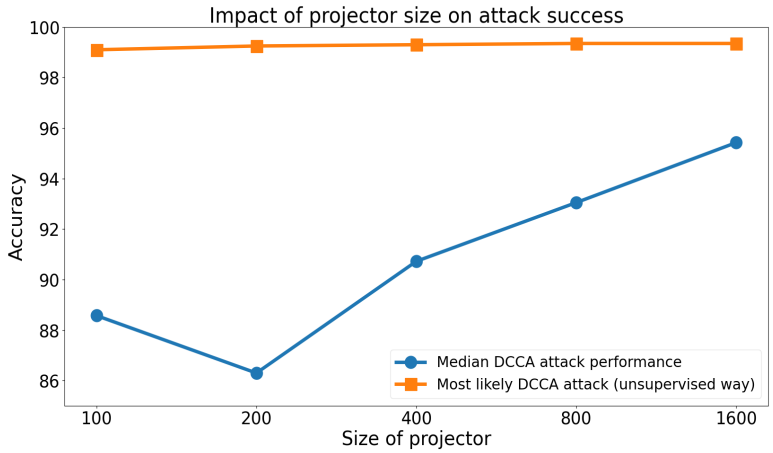


Figure 4.14: Impact of projector size (number of neurons) on the DCCA attack.

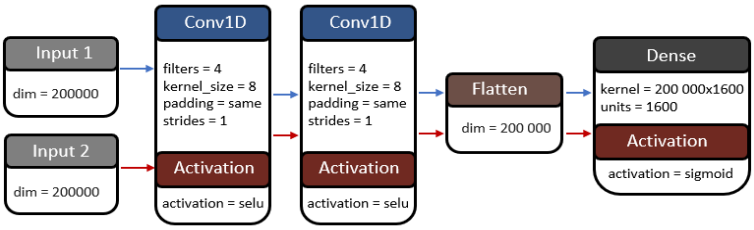


Figure 4.15: Suitable DCCA model for the protected RSA dataset.

Suitable DCCA Model. As a result of our experimental campaign on the protected RSA dataset, we identified a suitable DCCA architecture (depicted in Figure 4.15) consisting of

[8]The input modular operation traces consist of 50 000 points and all convolution layers contain 4 filters, thus the feature extractor always keeps a dimensionality of 200 000 up to its output.

[9]With our computational resources, we could not exceed a projector size of 1600.

a two-layer convolutional feature extractor (each layer using 4 filters of size 8) followed by a dense projector of 1600 neurons. This architecture delivered strong attack performance while noticeably reducing their instability.

4.4.5 Take-Away Messages

In this section, we propose a non-profiling horizontal collision attack using an unsupervised deep learning method called Deep Canonical Correlation Analysis. In this approach, we propose to use an unsupervised siamese neural network to maximize the correlation between pairs of modular operation traces through canonical correlation analysis, projecting them into a highly correlated low-dimensional latent space that is more suitable for identifying operand collisions.

Several experimental results, on simulated traces and a protected RSA implementation with up-to-date countermeasures, show how our proposal outperformed state-of-the-art attacks despite being simpler to implement. Notably, DCCA simplifies the trace preprocessing, making it possible to attack high-dimensional noisy traces without the need to select the PoI. Our approach only requires that the exponentiation traces are properly cut into modular operation traces and realigned.

However, DCCA suffers from some limitations. The major one lies in its loss function, which lacks a parameter to regularize the correlation objective. As a result, DCCA can over-correlate latent representations, making them so similar that it becomes impossible to distinguish differences in correlation between pairs of modular operation traces containing collisions and those that do not. In deep learning, this phenomenon, where learned representations become overly similar, or even identical, across inputs, is called *collapse* [Jin+22] and remains an open issue for the DCCA model [He+24]. This collapse issue tends to make DCCA unstable, rendering its ability to converge towards latent representations useful for collision attacks highly dependent on the random initialization of its weights.

DCCA Weight Initialization Framework. To address this instability, we proposed a framework based on a weight initialization search strategy. In this way, the evaluator must perform several attacks with different starting weights in order to identify an attack with a good success rate. As a result, the practical implementation of our attack may involve several steps, which we summarize below.

- *Step 1: Correlation on DCCA model latent space.* The first step consists in training several DCCA models and producing for each of them the highly correlated latent representations of all pairs of modular operation traces for which we wish to detect the presence of collisions. Then, for each pair of latent representations, compute their correlation using the Pearson correlation $\rho(F(\vec{X}_1, \boldsymbol{\theta}), F(\vec{X}_2, \boldsymbol{\theta}))$.

- *Step 2: Collision threshold identification.* The second step seeks to identify the collision / no-collision threshold on the correlation coefficients, using a clustering algorithm, *e.g.* K-means clustering.

- *Step 3: Rejecting unlikely attacks.* Assuming that the secret exponent is roughly balanced, an attack that produces a significant cluster imbalance in step 2 is unlikely. The third step aims to reject those unlikely attacks caused by some outliers that may bias the clustering algorithm in the collision/no-collision threshold identification. For this purpose, we propose the use of a statistical hypothesis binomial test to assess cluster imbalance. Attacks that lead to a clustering result that is rejected by the binomial test, are rejected.

- *Step 4: Ranking likely attacks.* The final step consists in ranking all the attacks that has passed the binomial test in order to identify the best among them. For this purpose, we propose the use of the silhouette score [Rou87], a widely used clustering metric in the machine learning community. This metric evaluates how points in the same cluster are similar to each other, and how different they are from points in other clusters. It assumes that a good clustering solution encompasses compact and well-separated clusters.

Another limitation of DCCA is its inability to handle high-dimensional latent spaces due to the computational cost of some operations, such as matrix inversion required to obtain the correlation matrix $\boldsymbol{T}$ used in Equation 4.8. In this way, to handle high-dimensional traces, DCCA requires a drastic downsizing (*i.e.* $o \ll d$) to estimate canonical correlations in the low-dimensional latent space.[10] Moreover, implementing such a downsizing in the DCCA model is challenging. Notably, we observed that integrating downsizing in the feature extractor tends to promote collapse, which increases instability. Consequently, we recommend to keep the spatial dimensionality in the feature extractor and to only perform it in the dense projector. In this context, we empirically observed that increasing the size of the projector improved the stability of the attack by reducing collapse. However, the use of such a large-scale projector considerably increases the model complexity due to the fully dense connections with the feature extractor output, whose dimensionality is extremely high. In this way, to thoroughly handle the instability issue, the evaluator must carefully design the DCCA model according to its computational resources, striking a balance between the complexity of the feature extractor and the one of the projector.

In addition, the architectural constraints imposed on DCCA, and notably the difficulty of integrating downsizing into the feature extractor, limit the model's ability to learn translational invariant representations [Jad+15; MMD20]. As a result, DCCA struggles to deal with misaligned traces, limiting its application in evaluations where realignment methods are either impractical or not satisfying.

4.5 Barlow Twins Collision Correlation Analysis

In the previous section we introduced DCCA, which can handle high-dimensional traces without prior PoI selection. While DCCA mitigates some limitations of non-profiling horizontal collision attacks, it still faces practical issues in certain evaluation contexts. In particular, attack efficiency is highly sensitive to random weight initialization, resulting in insta-

[10] CCA computation is an open issue in the literature [She+23].

bility, and DCCA performs poorly when traces are misaligned. In this section we go beyond these limitations by introducing BTCCA, a novel attack that tailors the Barlow Twins learning paradigm to the context of collision attacks. BTCCA improves stability and reduces the need for prior realignment pre-processing, thereby simplifying the attack workflow and making the approach more robust in realistic evaluation settings.

4.5.1 BTCCA Learning Paradigm

In this section, we provide a detailed description of the Barlow Twins loss used to train the BTCCA models, describing how it extracts highly correlated latent representations and explaining the intuition behind its design.

Barlow Twins is a self-supervised learning approach originally introduced in computer vision by Zbontar *et al.* [Zbo+21]. Its principle is to process two artificially augmented views of the same input with a siamese model and to maximize the correlation between the resulting latent representations while reducing redundancy across their dimensions, thus encouraging diverse and informative representations.

Let $\boldsymbol{Z_1} = \Big(F(\vec{x}_{1,1}, \boldsymbol{\theta}), \cdots, F(\vec{x}_{1,n}, \boldsymbol{\theta})\Big)$ and $\boldsymbol{Z_2} = \Big(F(\vec{x}_{2,1}, \boldsymbol{\theta}), \cdots, F(\vec{x}_{2,n}, \boldsymbol{\theta})\Big) \in \mathbb{R}^{n \times o}$ be the latent representation matrices for a mini-batch of n data and a latent dimension of size o. These matrices are first mean-centered along the mini-batch dimension and then used to compute a cross-correlation matrix $\boldsymbol{C} \in \mathbb{R}^{o \times o}$ between them.

$$\boldsymbol{C}[i,j] = \frac{\sum_{b=1}^{n} \boldsymbol{Z_{1,b}}[i] \boldsymbol{Z_{2,b}}[j]}{\sqrt{\sum_{b=1}^{n} (\boldsymbol{Z_{1,b}}[i])^2} \sqrt{\sum_{b=1}^{n} (\boldsymbol{Z_{2,b}}[j])^2}}, \tag{4.13}$$

with b indexing batch samples and i, j indexing the dimensions of the latent representations (*i.e.*, $1 \leq i, j \leq o$). The Barlow Twins loss function L_{BT}, defined in Equation 4.14, aims to push $\boldsymbol{C}$ toward an identity matrix. This enforces strong correlations with diagonal terms close to 1, while simultaneously promoting redundancy reduction (*i.e.*, decorrelation) across dimensions by pushing the off-diagonal terms toward 0.

$$L_{BT} = \sum_{i=1}^{n} (1 - \boldsymbol{C}[i,i])^2 + \lambda \sum_{i=1 | i \neq j}^{n} \boldsymbol{C}[i,j]^2. \tag{4.14}$$

The regularization hyperparameter λ in Equation 4.14 aims to control the balance between the correlation term and the decorrelation term. To determine an appropriate value for λ, Zbontar *et al.* [Zbo+21] suggest to use a grid search. In their seminal work, a λ value of $5e^{-3}$ was found to be effective.

Mini-Batch-Oriented Variant. In our contribution, we reformulate this principle by applying it to the mini-batch dimension (as depicted in Figure 4.16). Concretely, the correlation matrix is computed along the mini-batch dimension (*i.e.*, $\boldsymbol{C} \in \mathbb{R}^{n \times n}$). By proposing this mini-batch-oriented variant we define a learning objective that is better suited to the problem of collision attacks. Indeed, we implicitly inject some a priori knowledge while

maintaining an unsupervised approach. More precisely, the model is encouraged to correlate representations from consecutive pairs of modular operation traces, which are supposed to potentially share an operand (*i.e.*, pairs of the same color in Figure 4.16), while decorrelating those from non-consecutive pairs, which cannot share an operand (*i.e.*, pairs of different colors in Figure 4.16).

In this way, this method offers a flexible solution to the collapse problem encountered with DCCA. Indeed, for consecutive pairs that share an operand, the model is strongly encouraged to exploit collision information in order to maintain a high correlation while respecting the mini-batch decorrelation constraint. Conversely, for consecutive pairs that do not share an operand, it becomes harder for the model to correlate them while satisfying this constraint, thus naturally limiting the risk of undesirable over-correlation.

Furthermore, although the traditional Barlow Twins [Zbo+21] is theoretically capable of processing high-dimensional data, as its loss function is based on a simple matrix product, it requires the storage of the correlation matrix in GPU memory, whose size increases quadratically with the dimension of the latent space, becoming rapidly too computationally expensive for the evaluator. Thus, by reformulating the loss function to operate on the mini-batch dimension, we remove this hardware constraint and enable the use of extremely high-dimensional representations without restrictions.

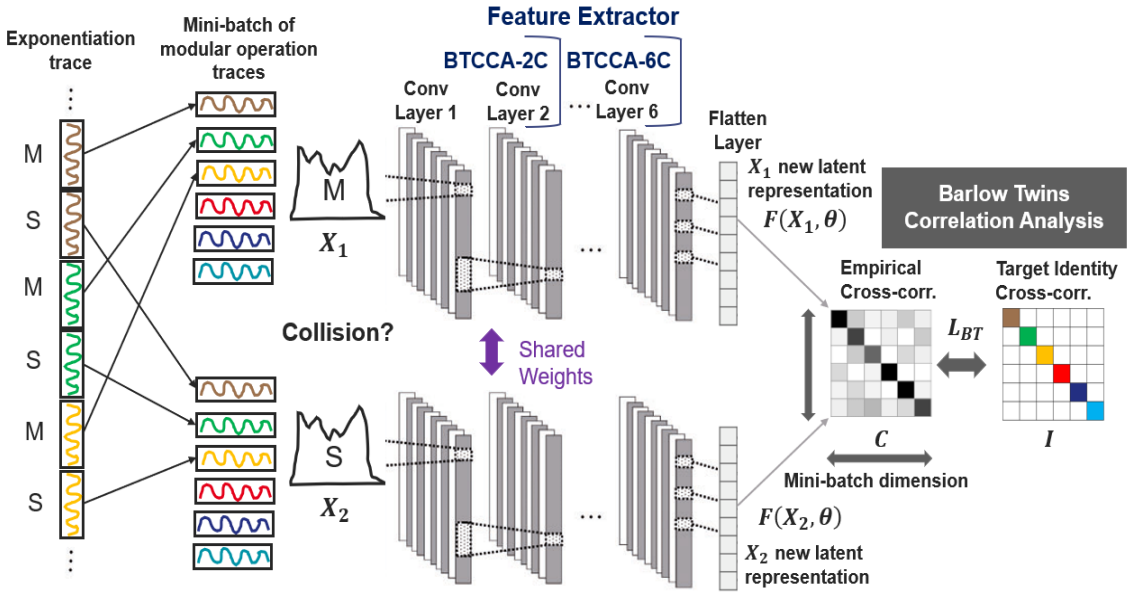


Figure 4.16: Barlow twins collision correlation analysis.

4.5.2 Experiments Results on Protected RSA Realigned Traces

This section presents an experimental exploration of BTCCA on realigned traces from the protected RSA dataset described in Section 4.2.2, assessing its relevance in comparison with DCCA.

Experiments Settings. For this experimental campaign, we randomly selected $10\,000$ pairs of modular operation traces (multiplication/squaring) for training the BTCCA models (without using label knowledge), as well as $2\,000$ distinct pairs for assessing the performance of our collision attacks. In accordance with the procedure described in Section 4.2.2, each modular operation trace was subsampled, yielding traces of $50\,000$ time points. This subsampling allows us to stay in a high dimension while limiting the computational complexity of the experiments.

As a baseline, we use the suitable DCCA model identified in Section 4.4.4 (depicted in Figure 4.15). Before training the DCCA model, in accordance with Section 4.4.4, the modular operation traces were pre-processed so that all samples within a mini-batch were normalized between 0 and 1. However, this normalization step was omitted for BTCCA, since the Barlow Twins loss function inherently includes a form of mini-batch normalization (*i.e.*, mean-centering along the mini-batch dimension).

Both BTCCA and DCCA correlation objectives were optimized using the RMSProp optimizer, with a learning rate of 0.0001 and a mini-batch size of 100. For all architectures, convolutional layers used SeLU activation function, while dense layers used Sigmoid activation function. The BTCCA models were trained with a λ value of 0.005, as recommended in [Zbo+21]. To evaluate the impact of training time, both DCCA and BTCCA models were trained from 5 to 120 epochs. Additionally, each model was trained 100 times to assess the effect of the random weight initialization on stability and performance.

All experiments are implemented in Python using Keras library with a TensorFlow 2.8 backend and are run on a workstation equipped with 32GB RAM and an NVIDIA GTX 4080 SUPER with 16GB memory.

First Step. As a first step, we experimented with the DCCA model baseline by replacing its loss function with that of Barlow Twins. This yielded poor results, highlighting an important insight: hyperparameters optimized for DCCA are not necessarily suitable for BTCCA. Indeed, the loss functions of Barlow Twins and DCCA work differently from each other. A key distinction lies in the fact that BTCCA could benefits from high-dimensional latent spaces. Hence, instead of extending the projector, which would significantly increase the complexity of our model, we chose to drop it and rely solely on the convolutional feature extractor, which makes dimensionality increase.[11] In this way, we propose to directly compute the correlation loss function on the flattened layer output of the feature extractor, as depicted in Figure 4.16.

Feature Extractor. We investigated the influence of feature extractor complexity on the performance of BTCCA. For this purpose, we experimented with several BTCCA models featuring a variable number of convolutional layers in the feature extractor block, ranging from 2 to 6.[12] For the sake of completeness, all BTCCA models used in the following experiments (depicted in Figure 4.17) share the same convolutional configuration, with each layer composed of 4 filters of size 8. Therefore, the 2-layer BTCCA model (*i.e.* BTCCA-

[11] The input modular operation traces consist of 50 000 points and all convolution layers used in our experiments contain 4 filters, thus the feature extractor always keeps a dimensionality of 200 000 up to its output

[12] By dropping the projector, the model's complexity is reduced, allowing the use of a more complex feature extractor.

2C) employs the same feature extractor as the baseline DCCA model (*i.e.* DCCA-2C). Both are similar, differing only in their loss functions and the lack of a projector in the BTCCA model.

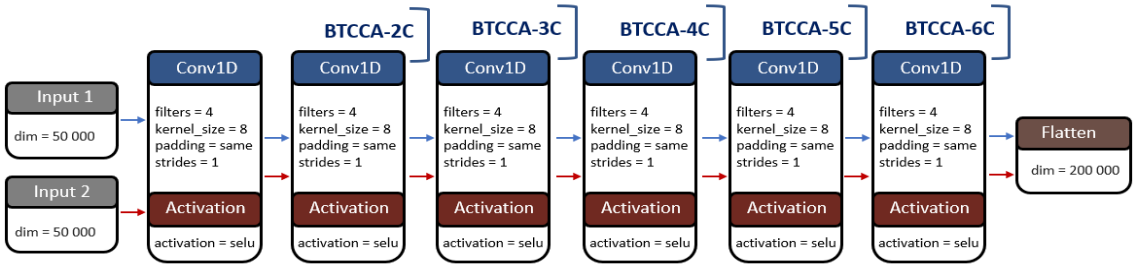
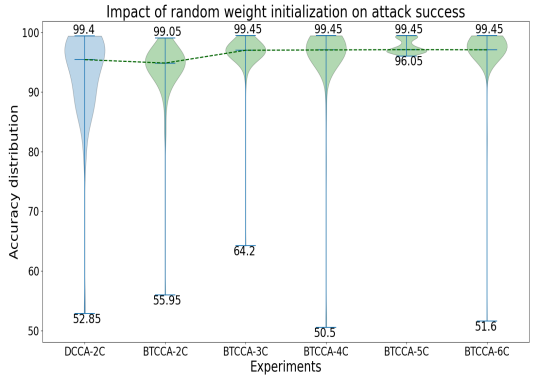
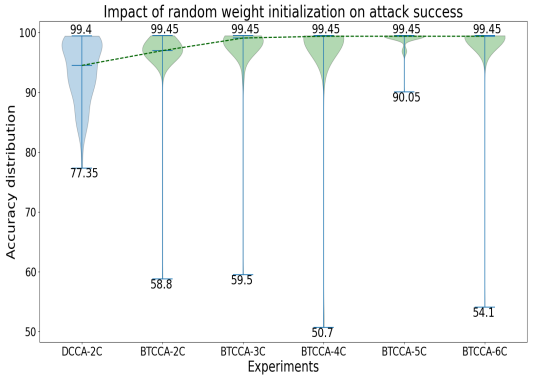


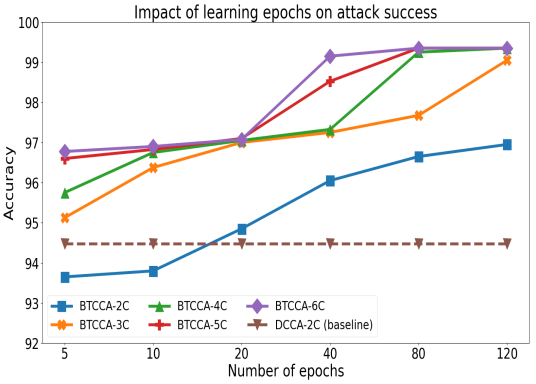
Figure 4.17: BTCCA-XC models.



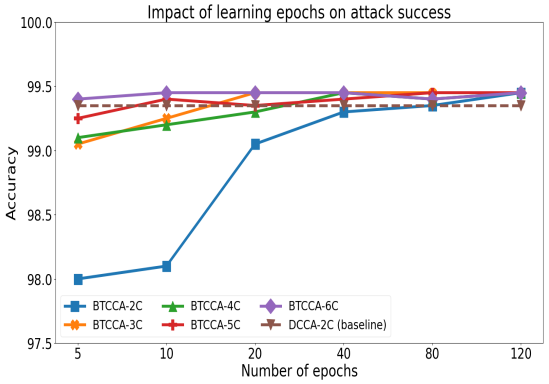
(a) Results for 20 training epochs.



(b) Results for 120 training epochs.



(c) Median of attacks.



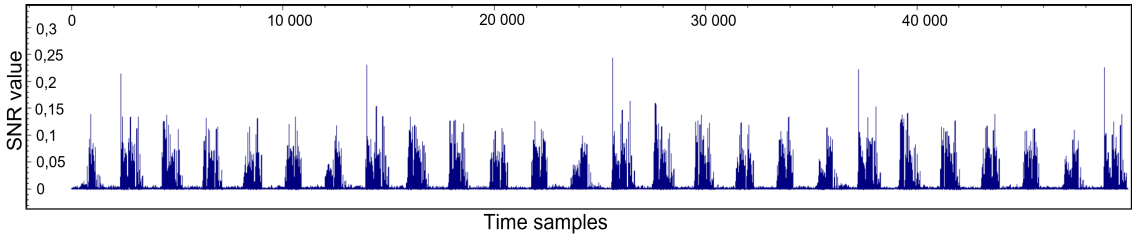
(d) Best attack.

Figure 4.18: Experiment results on realigned traces for different BTCCA feature extractors.

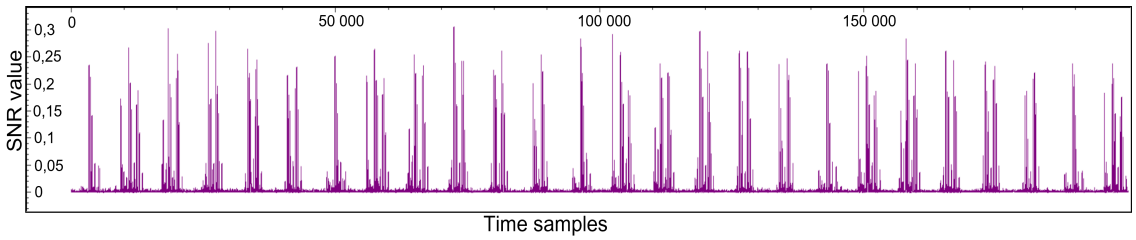
These results, summarized in Figure 4.18, enabled us to assess both attack effectiveness and stability. To this end, we report the median attack performance (Figure 4.18c) and the best attack performance (Figure 4.18d), corresponding to the best model identified in an unsupervised way using the weight initialization search framework described in Section 4.4.5. The latter reflects the practical capability an evaluator might have by applying that initialization search in a real evaluation. Firstly, by comparing the performance of BTCCA-2C

with that of DCCA-2C, we observed that both methods achieved comparable results when the weight initialization search strategy was applied (as depicted in Figure 4.18d). However, BTCCA-2C required a longer training time (at least 40 epochs) to reach the same attack performance as DCCA-2C. This can be explained by the lack of a projector in BTCCA-2C, which forces the model to rely solely on the feature extractor to learn and structure meaningful latent representations for the collision attack. As a result, convergence is slower when using a simple feature extractor. Nevertheless, increasing the feature extractor's complexity allows BTCCA to learn better representations in a faster way, effectively compensating for the lack of a projector. Consequently, BTCCA achieves similar, or even slightly better, attack performance compared to DCCA in the early learning stages.

It is worth noting that the mini-batch decorrelation constraint, combined with the absence of any form of downsizing, promotes stable convergence by preventing collapse. This enables the BTCCA models to learn latent representations efficient for collision attacks. As a result, BTCCA exhibits high attack stability, whereas DCCA shows no improvement in stability across training epochs (as depicted in Figure 4.18c). In particular, BTCCA-6C achieves a median attack success above 99% after 40 epochs, while BTCCA-(5-4-3)C reach similar stability after 120 epochs. In this way, BTCCA proved to be less sensitive to random weight initialization, especially for longer training, reducing the need for a weight initialization search.



(a) SNR on protected RSA realigned traces.



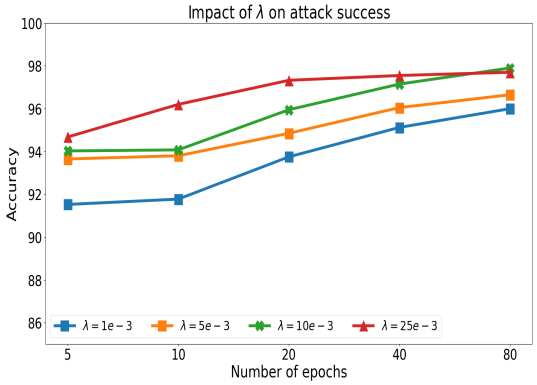
(b) SNR on BTCCA latent representation of the protected RSA realigned traces.

Figure 4.19: SNRs of operand collision leakage computed with the time-sample wise absolute difference between the modular operation traces of multiplications and their respective consecutive squares. Here the labelling is correct, obtained with the perfect knowledge of the colliding and non-colliding consecutive modular operations.

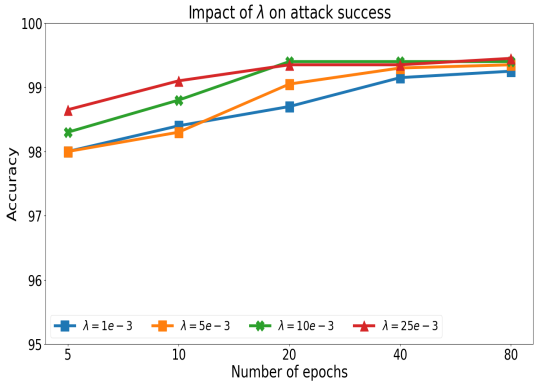
Among other aspects, our experiments demonstrate the effectiveness of the dimensionality expansion approach. In the absence of a global feature aggregator (*e.g.* a projector), this approach both preserves and amplifies collision leakage in latent representations, thanks to various convolutional transformations that maintain the locality of the leakage (as depicted in Figure 4.19).

Sensitivity to λ. Then, we investigated the sensitivity of BTCCA to the λ hyperparameter, which controls the trade-off between correlation maximization and mini-batch decorrelation. For this experiment, we focused on two models: the least complex one (*i.e.* BTCCA-2C) and the most complex one (*i.e.* BTCCA-6C).

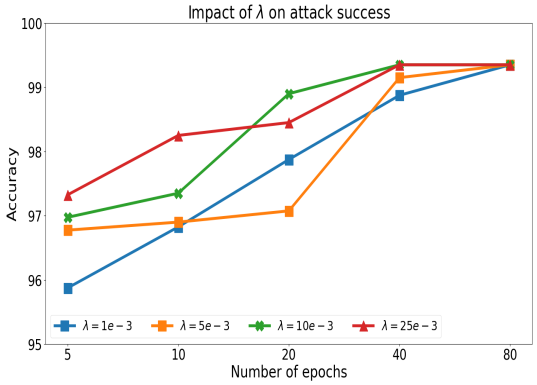
The results are summarized in Figure 4.20. Firstly, we observed that this hyperparameter affects the convergence of BTCCA models. In particular, for BTCCA-2C, which uses a simple feature extractor, increasing the values of λ, *i.e*, giving more importance to the mini-batch decorrelation constraint, improves attack stability in the early stages of training, as depicted in Figure 4.20a. This suggests that mini-batch decorrelation not only mitigates collapse but also helps the model form latent representations that are effective for collision attacks. However, this benefit tends to diminish with prolonged training.



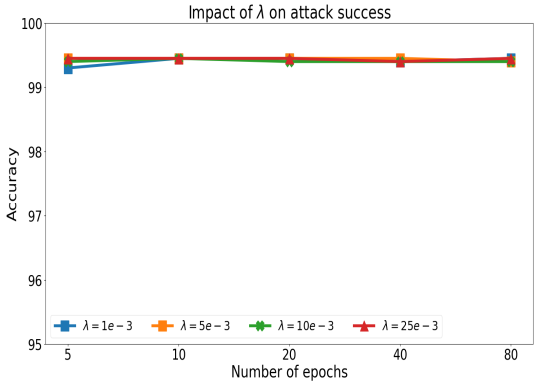
(a) BTCCA-2C median of attacks.



(b) BTCCA-2C best attack.



(c) BTCCA-6C median of attacks.



(d) BTCCA-6C best attack.

Figure 4.20: Experiment results on realigned traces for different λ values.

In analyzing the performance of the best attack using the weight initialization search strategy (depicted in Figure 4.20b), we observed a similar trend: a higher value of λ improves initial attack performance, but beyond approximately 80 epochs, overall performance becomes roughly similar, regardless of the value of λ.

Finally, these findings are also confirmed for BTCCA-6C (as depicted in Figure 4.20c and Figure 4.20d). This suggests that, in these experiments on realigned traces, the λ hy-

perparameter mainly affects the convergence speed of the BTCCA model towards latent representations favorable for collision attacks, regardless of the complexity of the feature extractor.

4.5.3 Experiments Results on Protected RSA Misaligned Traces

This section presents an experimental exploration of BTCCA on misaligned traces from the protected RSA dataset described in Section 4.2.2, assessing its effectiveness in the most pessimistic scenario for the evaluator, featuring a non-profiling attack without any preprocessing, nor PoI selection or trace realignment.

Experiments Settings. The experiments were conducted using settings similar to those in Section 4.5.2, but we focused only on the most complex BTCCA model from previous experiments, *i.e.*, BTCCA-6C, considering that a sophisticated feature extractor is required to handle misaligned traces. The model was also slightly adjusted to improve its translational invariance properties. For this purpose, the convolutional window was extended to a size of 33, a choice motivated by its satisfying performance in the supervised experiments reported in Section 4.3.2. We also implemented gradual downsizing using strided convolution layers, shifting the filters by 2 units every two layers (*i.e*, for layers 2, 4, and 6). In this way, we address the downsizing in a fully convolutional architecture with learned filters. The resulting BTCCA model used in these experiments is depicted in Figure 4.21.

As reported in Section 4.4.4, integrating downsizing into the feature extractor is challenging, as it often leads to latent collapse. However, when dealing with misaligned traces, such downsizing is essential to ensure that small spatial shifts or translations in the input do not affect the extracted features. This paradox introduces a major constraint: downsizing must be carefully balanced to improve translation invariance while avoiding collapse. We recall that this constraint explains why DCCA struggles to handle misaligned traces, as it lacks a regularization mechanism to mitigate the collapse.

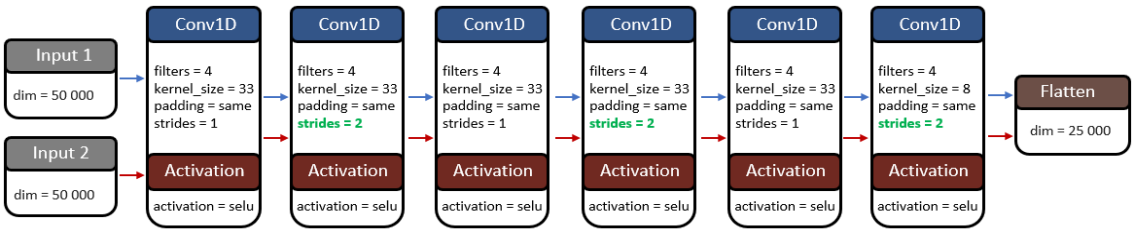
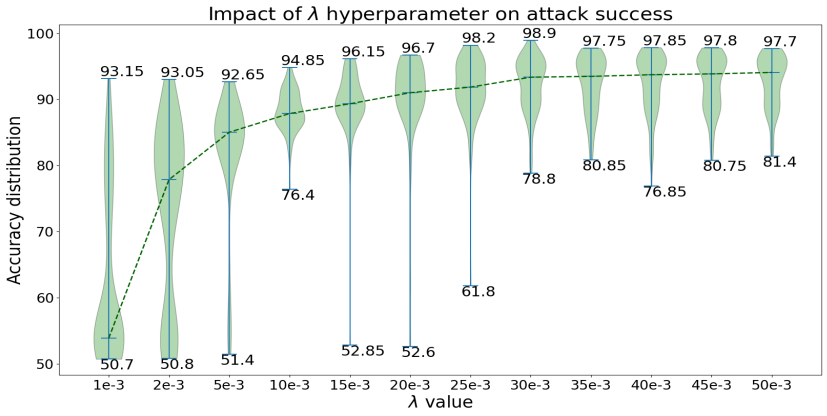
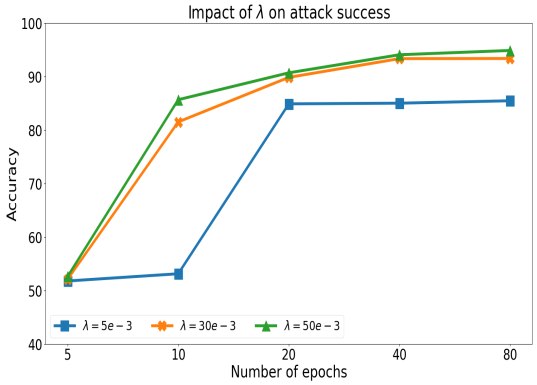


Figure 4.21: BTCCA-6C for protected RSA misaligend traces.

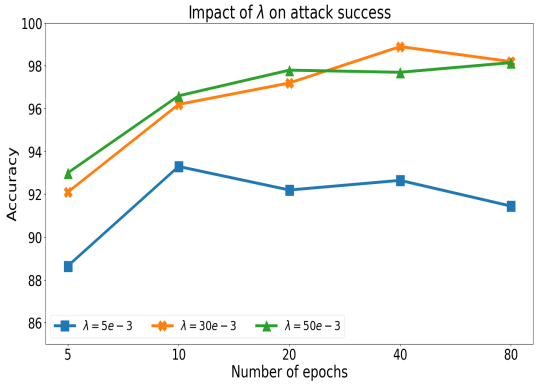
Collapse Mitigation. To address this issue with BTCCA, we performed a thorough grid search on the λ hyperparameter to select the most appropriate rate of mini-batch decorrelation for balancing the collapse caused by the downsizing in the feature extractor. For this experimental campaign, the BTCCA models were trained for 40 epochs, and each model was trained 100 times to assess the effect of random weight initialization on both stability and performance.



(a) BTCCA λ grid search on misaligned traces.



(b) Median of attacks.

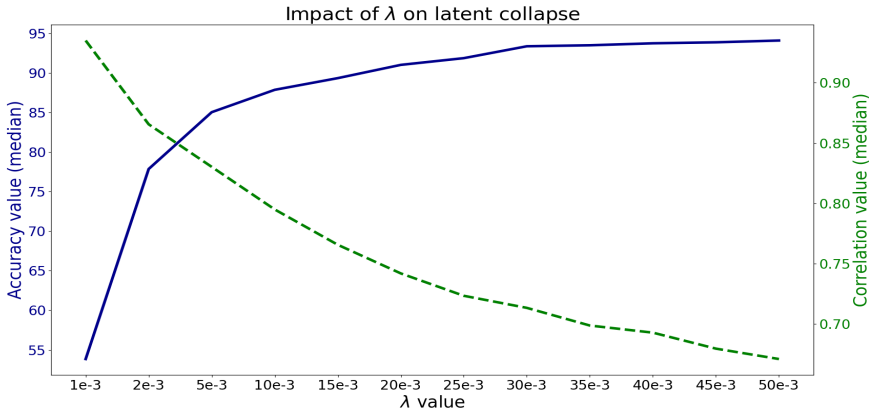


(c) Best attack.

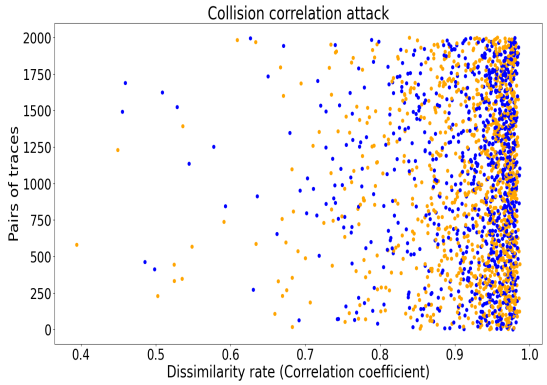
Figure 4.22: BTCCA results on misaligned traces for different λ values.

The results are summarized in Figure 4.22a. We observed a significant impact of the λ hyperparameter on both attack performance and stability. As expected, increasing the importance of the mini-batch decorrelation better mitigates collapse, thereby improving both the median and best attack performance. Notably, performance gradually improves up to a value of $\lambda = 30e^{-3}$, beyond which no further improvement was observed.

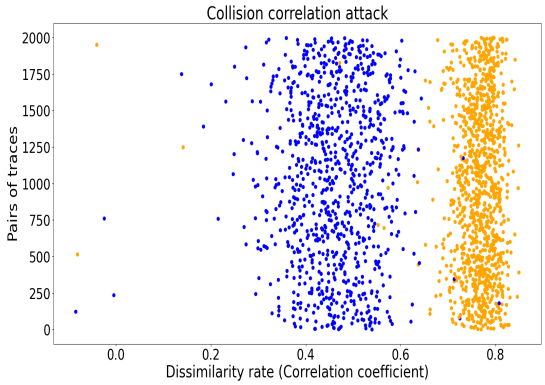
The close relationship between the correlation rate of latent representations and the success of collision attacks is depicted in Figure 4.23, highlighting both the impact of collapse and the relevance of the mini-batch decorrelation constraint in this experiment. Indeed, although misaligned traces are theoretically more difficult to correlate in a common latent space than synchronized ones, simply applying downsizing tended to over-correlate the latent representations, as depicted in Figure 4.23a for lower λ values. In this case, most latent representations exhibit correlations close to 1, making it impossible to distinguish pairs of traces with and without collisions (as depicted in Figure 4.23b). However, increasing the rate of mini-batch decorrelation by using higher values of λ inverses this trend, improving collision detection and attack success. An example of such a successful attack thanks to a properly correlated latent space is depicted in Figure 4.23c.



(a) Attack success and correlation (median values).



(b) Overly correlation with $\lambda = 1e^{-3}$.



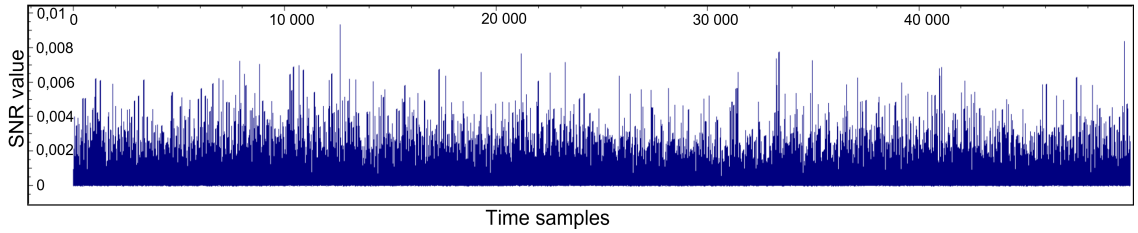
(c) Suitable correlation with $\lambda = 30e^{-3}$.

Figure 4.23: Impact of λ on latent collapse and attack success. The correlation coefficients are colored with their ground-truth labels. Blue (resp. orange) values represent pairs of modular operation traces without collision (resp. with collision).

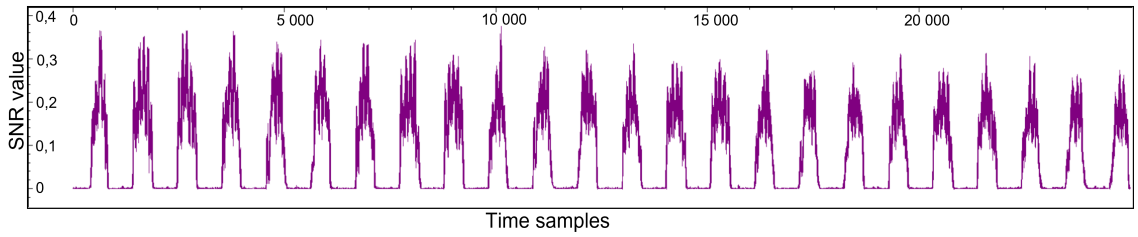
λ across epochs. As the mini-batch decorrelation constraint proved essential for handling misaligned traces, we conducted a detailed study of the λ hyperparameter in this context. For this purpose, we selected three λ values, *i.e.*, $5e^{-3}$, $30e^{-3}$, and $50e^{-3}$, and observed their impact on BTCCA model convergence across learning epochs. This analysis, summarized in Figure 4.22b and Figure 4.22c, reveals that a small decorrelation rate leads to less efficient BTCCA models, regardless of the learning epoch, affecting both stability and attack performance. This deeper analysis also revealed that the best BTCCA configuration for misaligned traces was achieved in our experiment at 40 learning epochs with $\lambda = 30e^{-3}$, reaching an accuracy close to 99%. In comparison, DCCA and other state-of-the-art collision attacks remain totally ineffective on misaligned traces.

Interestingly, we computed the collision leakage SNR on the BTCCA latent representations and observed a leakage pattern similar to that obtained on the realigned traces (as depicted in Figure 4.24b), which was originally blurred in the misaligned raw traces (as depicted in Figure 4.24a). This observation suggests that BTCCA implicitly realigns the leakage within its latent representations, explaining why collision attacks are effective on

them.



(a) SNR on protected RSA misaligned traces.



(b) SNR on BTCCA latent representation of the protected RSA misaligned traces.

Figure 4.24: SNRs of operand collision leakage computed with the time-sample wise absolute difference between the modular operation traces of multiplications and their respective consecutive squares. Here the labelling is correct, obtained with the perfect knowledge of the colliding and non-colliding consecutive modular operations.

As a result, to successfully attack misaligned traces, the evaluator must balance the use of downsizing, which improves translation invariance, with the mini-batch decorrelation rate, which prevents collapse. Since this balance is not perfect, the attack exhibits significant instability due to its sensitivity to random weight initialization. Consequently, for misaligned traces, the weight initialization search strategy remains mandatory.

4.5.4 Take-Away Messages

In this section, we introduce BTCCA, a novel unsupervised method that tailors the Barlow Twins learning paradigm for non-profiling collision attacks.

Specifically, we tailored the loss function to the collision attack problem by integrating a mini-batch decorrelation constraint and designing it to work efficiently in very high-dimensional latent representations. These two properties together prevent collapse and give the evaluator greater flexibility in architectural design.

This flexibility allows BTCCA to be deployed in various architectural configurations. For instance, in our experiments on realigned traces, we used BTCCA models without any form of downsizing, which demonstrated strong attack stability and reduced the need for a weight initialization search strategy, thus saving precious time for time-constrained evaluation contexts. Conversely, when downsizing is required, *e.g.*, to design more translation-invariant models in the case of misaligned traces, the mini-batch decorrelation constraint enables to balance the collapse induced by this operation. In this context, BTCCA proved

particularly effective, achieving accuracy close to 99%, whereas other state-of-the-art non-profiling collision attacks were completely ineffective on misaligned traces. Although some instability remains in this scenario, the weight-initialization framework remains effective.

In summary, BTCCA effectively addresses the two main limitations of DCCA. It brings a solution to the problem of instability encountered on realigned traces and opens up new possibilities for handling misaligned traces in non-profiling horizontal collision attacks. To the best of our knowledge, BTCCA is the first non-profiling horizontal collision attack that requires neither prior PoI selection nor trace realignment, offering a simple yet powerful addition to the evaluator's toolbox.

4.6 Evaluation Report

In this section, to illustrate the relevance of the proposed attacks, we assess the impact of remaining complexity for successful key recovery. Notably, to assess the feasibility of remaining attacks, we consider the European SOG-IS scheme which considers a maximum brute-force complexity of around 2^{100} operations to be practical [SI+23].[13] Two different remaining complexity metrics are considered in our analysis: the naive complexity (C_{NC}) and the 2^n complexity (C_{2^n}). Considering these two metrics provides a valuable comparison of different evaluation strategies and offers a broader perspective on the security of the targeted implementation. Notably, in our analysis, as already explained in Section 1.4.1, we treat the 2^n-complexity as a lower bound and the naive complexity as an upper bound for attack quotation. In the following, an attack that can be performed with a 2^n complexity (*i.e.* $C_{2^n} \leq 100$) is called a 2^n-attack, while an attack that can be performed with a naive complexity (*i.e.* $C_{NC} \leq 100$) is called a naive attack.

To compare our proposed approaches on real dataset, we followed the same baselines as in simulation experiments. Thus, we carried out several collision correlation attacks using the K-means algorithm to identify the collision/no-collision threshold, across different trace representations of state-of-the-art pre-processing techniques: direct correlation on raw modular operation traces, correlation on PCA-reduced traces, correlation on Big Mac [Wal01] pre-processed traces.[14]

Table 4.3 summarizes our results over 5 secret exponents. For BTCCA and DCCA, we used the weight initialization framework[15] (described in Section 4.4.5) across all models trained in our experiments to identify, in an unsupervised way, the best performing models among them for the attacks. First, we note that the profiling attacks reported in Section 4.3.2 succeed in achieving both the 2^n-attack and the naive attack on realigned and misaligned traces. This demonstrates that the implementation's security can be compromised even without any pre-processing, apart from cutting of modular operation traces. Moreover,

[13] Note that the notion of remaining complexity is independent of the invocation time of the targeted implementation, as well as the computational power available to the attacker.

[14] Note that, we do not compare our results with horizontal clustering methods, as the attack paths are different from collision attacks.

[15] The latter reflects the actual capability an attacker might have by exploiting the framework in an actual attack scenario.

even in a non-profiling scenario, we observed that the dataset is particularly vulnerable to horizontal attacks, as a simple collision correlation attack on realigned modular operation traces, without any pre-processing beyond realignment, allows the recovery of more than 92% of the secret exponent bits. Nevertheless, this is not enough to claim a feasible attack with naive complexity.

Table 4.3: Attack evaluation for 1088 bits exponent (average over 5 secret exponents). Green (resp. red) values are considered as practicable complexity with respect to the SOG-IS criteria (resp. unpracticable).

Attack	Acc.	C_{NC}	C_{2^n}
Realigned traces			
Raw traces	92.31	428	84
PCA	96.21	258	42
Big Mac	58.62	1068	451
DCCA	99.33	68	8
BTCCA	99.52	54	6
Supervised SNN	99.85	20	2
Misaligned traces			
DCCA	51.85	1090	524
BTCCA	98.85	102	13
Supervised SNN	99.35	68	8

It is noteworthy that the second best approach on simulation experiments, namely the Big Mac attack, did not work on this dataset despite the absence of countermeasures against horizontal attacks. In fact, Big Mac pre-processing has even led to a loss of information and a drastic drop of performance compared to the collision attack on raw traces. This highlights once again that, despite the applicability of Big Mac pre-processing has been confirmed by simulation experiments, it still needs to be assessed with a real device [Car+19; Sai+22]. It is worth noting that Big Mac pre-processing relies heavily on the assumption that the evaluator is able to extract LIM multiplier leakage from a raw trace. Indeed, even in our simulation experiments, we observed the strong impact of non-informative points on Big Mac pre-processing. Moreover, this pre-processing is not effective if noise cannot be sufficiently suppressed by the segment averaging. Under these conditions, we experimentally found that collision detection with Big Mac pre-processing was not successful in our real trace environment.

On realigned traces, we observed that a collision correlation attack performed on a linear dimensionality reduction by PCA constitutes the most representative baseline. Therefore, we thoroughly investigated this approach by varying the number of principal components from 100 to 1600[16], by steps of 100. Since none of these alternative number of principal components yielded better performance, we ensured a fair comparison with the DCCA and BTCCA results, where a significant improvement in the remaining complexity was observed.

[16] corresponding to the size of the latent space of DCCA

Notably, we observed that only deep learning-based attacks succeed in achieving remaining naive attacks in a realistic scenario for potential adversaries, making them a major security flaw. Indeed, we recall that according to the criteria established by the French National Security Agency, during evaluations carried out as part of the French scheme, if the complexity of completing the attack is equal to, or less than, 2^{70} invocations of the implementation, the cost of the effort required to recover the residual information is deemed negligible [AWG+19]. In this context, DCCA and BTCCA achieve performances that meet these criteria, underlining the seriousness of the threat they pose to the targeted implementation previously considered robust against unsupervised attacks.

BTCCA stands out for its ability to effectively handle misaligned traces, which remains a major issue for state-of-the-art attacks, including DCCA.[17] Indeed, BTCCA makes it possible to practically perform a 2^n-attack while also demonstrating strong potential to achieve a feasible naive attack without prior realignment, or point of interest selection[18]. Indeed, reaching a value of 102 for C_{NC} suggests that with an effective key enumeration strategy, the attack could probably become practical even in a realistic scenario. Moreover, it also suggests that, with further hyperparameter tuning, it may have been possible to achieve such naive attack. Therefore, we believe that the practicability of BTCCA can turn potential vulnerabilities into exploitable ones, leading to a reassessment of the targeted system's security and raising the need to implement dedicated countermeasures against horizontal collision attacks.

Indeed, classical exponent masking countermeasure in asymmetric implementations prevent vertical attacks and in most cases make supervised attacks impractical. As DCCA and BTCCA are unsupervised and use a single exponentiation trace, this kind of masking is automatically ineffective. Although various countermeasures against horizontal side-channel attacks aiming to implement masking or shuffling at the modular operation level (*i.e* in the underlying long integer multiplication algorithm) have been proposed in the literature [Bau+13], they are generally not implemented in practice as non-profiling horizontal attacks are often considered too difficult to implement. A thorough analysis of the security and efficiency of such countermeasures and/or the design of new ones are open avenues for further research.

4.7 Applicability to Other Cryptosystems

In this chapter, we presented an analysis of a standard RSA implementation based on the binary square and multiply always exponentiation. Because our attacks operates on a single exponentiation trace and requires no message or modulus knowledge, the methods and tools we propose are broadly applicable to other implementations. Therefore, RSA-CRT exponentiation and exponentiation schemes that use random or ephemeral exponents, such as DSA or Diffie–Hellman, are also vulnerable to the proposed attack methodologies. In addition, our proposed approaches can be naively extended to elliptic-curve implementations

[17] We only report the results of deep learning-based attacks for misaligned traces, as traditional attacks are not assumed to be practicable in this context.

[18] BTCCA only requires preliminary cutting of the exponentiation trace into modular operation traces.

that rely on analogous algorithms (*i.e.*, double and add always).

We also believe that the proposed approaches could be generalized to other cryptographic contexts involving horizontal collision vulnerabilities, although its adaptability may require adjustments to align with the specific properties of the targeted implementation. For instance, in atomic square and multiply implementations the right-hand operand collides on each multiplication operation because it corresponds to the processed message [Cla+12]. With regard to post quantum cryptography, recent works such as those presented in [Gro+23; Bit+24] targeting syndrome decoding of the Classic McEliece algorithm involves the detection of operand collisions in computational loops. Hence, we affirm that the ability to identify and exploit operand collisions of our approaches may be exploited in the context of PQC to improve the effectiveness of such attacks. However, the application may not be trivial and could be an interesting direction for future research.

4.8 Conclusion

In this chapter, we propose new methods for conducting horizontal collision attacks by redefining collision detection as a deep learning verification task processed through siamese networks. Following the work of [Car+19], which represents a profiling and highly optimistic scenario for the evaluator, we gradually reduce the profiling and pre-processing steps involved to provide methods that can be implemented where both time and resources are limited. Particular attention is given to critical scenarios lacking profiling capabilities, in which we introduce unsupervised loss functions specifically tailored to collision attacks. We further describe how to train unsupervised siamese networks to project pairs of modular operation traces into a highly correlated latent space, facilitating the detection of operand collisions without the need for prior PoI selection or trace realignment.

Our contributions are supported by a broad experimental exploration on a protected RSA implementation running on a defensive arithmetic coprocessor with up-to-date countermeasures. Following the SOG-IS and the French National Security Agency security guidelines, the improvement in remaining complexity provided by our attacks lead to reassessment of the targeted system's security even in a non-profiling context, raising the need to implement dedicated countermeasures against horizontal attacks.

Simple to implement, the proposed methods represent a valuable addition to the evaluator's toolbox, complementing existing approaches. Finally, we believe that the ideas and tools presented in this chapter can be adapted to other cryptographic contexts with collision vulnerabilities, subject to adjustments to take into account the specificities of the targeted implementations.

Chapter 5 The Last Chapter

General Conclusion

You only grow by coming to the end of something and by beginning something else.

– John Irving

Resume

In this thesis, we addressed several current issues related to neural network-based side-channel attacks, investigating diverse deep learning methodologies and adapting them to address practical evaluation constraints, as well as proposing novel attack strategies. In this regard, this chapter concludes the main contributions of this manuscript along two lines of research: (i) the problem of neural network hyperparameterization in profiling vertical attacks, (ii) the potential of siamese neural networks for performing both profiling and non-profiling horizontal collision attacks. For each of these two aspects, we report the main findings and discuss tracks for future research stemming from the results obtained.

5.1 Contributions

In this manuscript, we investigated several approaches for improving neural network-based side-channel evaluation.

In Chapter 3, we addressed the problem of hyperparameterization of neural networks for profiling attacks, a major challenge in time-constrained evaluation contexts. Indeed, the evaluator cannot afford to train hundreds or thousands of models to identify a suitable architecture, notably in the case of attacks on protected symmetric implementations, where training often requires millions of traces and is rapidly becoming a costly endeavor. To address this issue, we explored the use of ensemble learning, a methodology that seeks to combine several partially hyperparameterized models rather than seeking for a single perfectly optimized one. Following previous research already introduced in the side-channel context that explored bagging-based aggregation, we investigated a more advanced approach called stacking aggregation. This method relies on training a meta-model that learns how to combine the predictions of the several models composing the ensemble in order to get the most out of them. Our experimental study on several AES datasets showed that stacking has the potential to significantly enhance attack performance and consistently outperform bagging. We observed that this aggregation method is better suited to deal with performance disparities among the models within the ensemble, as the meta-model learns to weight their contributions and extract useful and complementary information from them. In this way, even ensembles composed of models that are not well-performing or not so well diversified in their predictions can yield effective results, making the choice of models to include in the ensemble less critical for an evaluator.

As a result, our experimental analysis showed that stacking ensemble can turn incomplete exploration of hyperparameters into a well-performing architecture, offering attack performance similar to that of rigorously hyperparameterized models. This approach therefore offers a simple and cost-effective solution to ease the implementation of neural network attacks in a profiling evaluation context. The obtained results of this contribution were published at CARDIS 2023 [Lla+23].

In Chapter 4, our work turned towards horizontal collision attacks targeting a protected RSA implementation with up-to-date countermeasures. We investigated several deep learning methodologies by redefining the operand collision detection between pairs of modular operation traces as a verification task that can be processed using siamese neural networks. First, we described how to train such a model using supervised contrastive learning in such a way as to implement a profiling horizontal collision attack that efficiently operates on high-dimensional, noisy, possibly misaligned traces, relieving the evaluator of the need to perform realignment and PoI selection pre-processing. Next, we paid particular attention to critical evaluation scenarios lacking profiling capabilities. For this purpose, we bridged traditional non-profiling collision attacks based on cross-correlation and siamese networks, which have the ability to automatically extract common characteristics from their inputs. In this respect, we introduced in the side-channel context two methods for training unsupervised siamese networks to pre-process pairs of modular operation traces, projecting them into highly correlated latent representations, better suited for operand collision detection.

The first investigated method, namely DCCA, performs this projection by applying a canonical correlation analysis on latent representations and has resulted in a successful fully unsupervised collision attack onto high-dimensional traces without PoI selection pre-processing, outperforming existing state-of-the-art non-profiling collision approaches. Nevertheless, the lack of regularization of the unsupervised correlation objective implies that DCCA relies on proper weight initialization to converge to latent representations that can be used for collision detection, while the method struggles with misaligned traces. In practice, DCCA requires an empirical weight initialization search strategy and realignment pre-processing to operate efficiently. To mitigate this, we equipped DCCA with a weight-initialization search framework that runs multiple random restarts and selects the most promising attacks in an unsupervised way, enabling reliable and complete attacks even if many of them may fail individually. The obtained results of this contribution were published at CiC 2025 [Lla+25].

The second investigated method, namely BTCCA, tailors the Barlow Twins self-supervised learning paradigm by integrating a mini-batch decorrelation constraint, thereby defining an unsupervised learning objective that is better suited to the problem of collision attacks. As a result, BTCCA simplifies the attack workflow and addresses the instability issues observed with DCCA on realigned traces, saving valuable time in time-constrained evaluation contexts while also opening new possibilities for handling misaligned traces. Although some instability remains in this latter scenario, BTCCA achieved a successful unsupervised collision attack without requiring PoI selection or realignment pre-processing, whereas other state-of-the-art non-profiling collision attacks proved completely ineffective on misaligned traces. A publication based on the results of this contribution is currently ongoing and has been accepted for TCHES, Volume 2026, Issue 2.

Together, the outcomes of the different methods proposed in this chapter highlight the relevance of siamese networks as a versatile and powerful tool for horizontal collision attacks, making them a valuable addition to the evaluator's toolbox and complementing existing approaches.

5.2 Perspectives

From the contributions presented in this thesis, several perspectives naturally emerged, opening up promising directions for future research and potential improvements in deep learning-based side-channel attacks.

The work conducted on ensemble learning for profiling vertical attacks has enabled us to identify several promising avenues for future research. For instance, in our experimental campaign, we only used a MLP as a meta-model, meaning that the way the ensemble predictions were concatenated had no particular influence on its learning process. However, we believe that using a different type of meta-model, such as a CNN, could be of interest to take advantage of a class-wise concatenation, placing the ensemble predictions for each class next to each other. The sliding-window mechanism of CNNs could then exploit this class-wise organization to capture relationships per class among the ensemble predictions, potentially leading to more effective combinations of them. In addition, the use of deep

ensemble learning in the context of side-channel attacks has attracted growing interest in recent years. However, although many studies have focused on parallel ensembles, such as bagging or stacking, to our knowledge, the sequential approach of deep boosting ensembles has not yet been explored. The latter is a promising perspective, as it could help to build more robust models in contexts where the quality of traces, *e.g.*, affected by high noise or misalignment, makes neural network learning difficult.

In the work conducted on the use of siamese networks for horizontal collision attacks we presented an analysis of a standard RSA implementation based on the binary square and multiply always exponentiation. We affirm that the ability to identify and exploit operand collisions of the proposed methods may be exploited as a tools to raise efficiency of attacks against all cryptographic implementations vulnerable to collision-detection-based side-channel attacks, including ECC or some PQC. However, the application may not be trivial and could be an interesting direction for future research. We also believe that the fusion method in latent representation of the siamese model could be of interest for multi-probe attacks, to pre-process the trace captured by different probes during the same operation. By projecting the traces from each probe into a common latent representation, this could amplify the shared leakage while attenuating the probe-specific noise.

In general, we are convinced of the value of further exploring deep learning techniques in the context of non-profiling horizontal side-channel attacks. In particular, self-supervised learning [Gui+24] is currently a very popular area of research in the deep learning community, leading to the development of numerous methods for learning useful representations from the data itself. Such approaches have proven their worth in various domains, including image [Alb22], natural language [Bae+22], and speech processing [Zai+22]. Notably, in the specific context of speech processing, these methods capitalize on decades of expertise in signal processing. Therefore, we believe that a thoughtful transfer of these methods could be of great benefit for improving the efficiency and reliability of non-profiling horizontal side-channel attacks.

Final Words

To all the brave readers who have made it this far, I hope you have enjoyed the journey and found this work useful. May this thesis provide you an overview of the state of the art in deep learning–based side-channel attacks, and above all, inspire you to explore further and push beyond the current boundaries.

As Master Yoda wisely said, “Do or do not. There is no try.”
This thesis is done.

Bibliography

[AGF23] Rabin Yu Acharya, Fatemeh Ganji, and Domenic Forte. "Information Theory-based Evolution of Neural Networks for Side-channel Analysis". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2023.1 (2023), pp. 401–437. DOI: 10.46586/TCHES.V2023.I1.401-437. URL: https://doi.org/10.46586/tches.v2023.i1.401-437.

[Alb22] Saleh Albelwi. "Survey on Self-Supervised Learning: Auxiliary Pretext Tasks and Contrastive Learning Methods in Imaging". In: *Entropy* 24.4 (2022), p. 551. DOI: 10.3390/E24040551. URL: https://doi.org/10.3390/e24040551.

[And+13] Galen Andrew et al. "Deep Canonical Correlation Analysis". In: *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*. Vol. 28. JMLR Workshop and Conference Proceedings. JMLR.org, 2013, pp. 1247–1255. URL: http://proceedings.mlr.press/v28/andrew13.html.

[AP95] Kamal M Ali and Michael J Pazzani. "On the link between error correlation and error reduction in decision tree ensembles". In: (1995).

[Arb+13] Olatz Arbelaitz et al. "An extensive comparative study of cluster validity indices". In: *Pattern Recognit.* 46.1 (2013), pp. 243–256. DOI: 10.1016/J.PATCOG.2012.07.021. URL: https://doi.org/10.1016/j.patcog.2012.07.021.

[AT19] Sunil Annareddy and Srikanth Tammina. "A comparative study of deep learning methods for spam detection". In: *2019 third international conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC)*. IEEE. 2019, pp. 66–72.

[AWG+19] ANSSI-Working-Group et al. *ANSSI-CC-NOTE-23-v1.0: DECISION SUR LA SECURITE ALGORITHMIQUE RESIDUELLE(REMAINING STRENGTH)*. 2019. URL: https://cyber.gouv.fr/sites/default/files/2022-08/anssi-cc-note-23-remaining-strength_v1.0[1].pdf.

[Aya+20] Riadh Ayachi et al. "Strided convolution instead of max pooling for memory efficiency of convolutional neural networks". In: *Proceedings of the 8th International Conference on Sciences of Electronics, Technologies of Information and Telecommunications (SETIT'18), Vol. 1*. Springer. 2020, pp. 234–243.

[BA23] Sana Boussam and Ninon Calleja Albillos. "Keep It Unsupervised: Horizontal Attacks Meet Simple Classifiers". In: *Smart Card Research and Advanced Applications - 22nd International Conference, CARDIS 2023, Amsterdam, The Netherlands, November 14-16, 2023, Revised Selected Papers*. Ed. by Shivam Bhasin

and Thomas Roche. Vol. 14530. Lecture Notes in Computer Science. Springer, 2023, pp. 213–234. DOI: 10.1007/978-3-031-54409-5_11. URL: https://doi.org/10.1007/978-3-031-54409-5_11.

[Bae+22] Alexei Baevski et al. "data2vec: A General Framework for Self-supervised Learning in Speech, Vision and Language". In: *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*. Ed. by Kamalika Chaudhuri et al. Vol. 162. Proceedings of Machine Learning Research. PMLR, 2022, pp. 1298–1312. URL: https://proceedings.mlr.press/v162/baevski22a.html.

[Bal+17] Josep Balasch et al. "Consolidating Inner Product Masking". In: *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I*. Ed. by Tsuyoshi Takagi and Thomas Peyrin. Vol. 10624. Lecture Notes in Computer Science. Springer, 2017, pp. 724–754. DOI: 10.1007/978-3-319-70694-8_25. URL: https://doi.org/10.1007/978-3-319-70694-8_25.

[Bar+22] Alessandro Barenghi et al. "Profiled side channel attacks against the RSA cryptosystem using neural networks". In: *J. Inf. Secur. Appl.* 66 (2022), p. 103122. DOI: 10.1016/J.JISA.2022.103122. URL: https://doi.org/10.1016/j.jisa.2022.103122.

[Bau+13] Aurélie Bauer et al. "Horizontal and Vertical Side-Channel Attacks against Secure RSA Implementations". In: *Topics in Cryptology - CT-RSA 2013 - The Cryptographers' Track at the RSA Conference 2013, San Francisco,CA, USA, February 25-March 1, 2013. Proceedings*. Ed. by Ed Dawson. Vol. 7779. Lecture Notes in Computer Science. Springer, 2013, pp. 1–17. DOI: 10.1007/978-3-642-36095-4_1. URL: https://doi.org/10.1007/978-3-642-36095-4_1.

[BCO04] Eric Brier, Christophe Clavier, and Francis Olivier. "Correlation Power Analysis with a Leakage Model". In: *Cryptographic Hardware and Embedded Systems - CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11-13, 2004. Proceedings*. Ed. by Marc Joye and Jean-Jacques Quisquater. Vol. 3156. Lecture Notes in Computer Science. Springer, 2004, pp. 16–29. DOI: 10.1007/978-3-540-28632-5_2. URL: https://doi.org/10.1007/978-3-540-28632-5_2.

[Ben+20] Ryad Benadjila et al. "Deep learning for side-channel analysis and introduction to ASCAD database". In: *J. Cryptogr. Eng.* 10.2 (2020), pp. 163–188. DOI: 10.1007/S13389-019-00220-8. URL: https://doi.org/10.1007/s13389-019-00220-8.

[BH21] Daehyeon Bae and JaeCheol Ha. “Performance Metric for Differential Deep Learning Analysis”. In: *J. Internet Serv. Inf. Secur.* 11.2 (2021), pp. 22–33. DOI: 10.22667/JISIS.2021.05.31.022. URL: https://doi.org/10.22667/JISIS.2021.05.31.022.

[Bis95] Christopher M Bishop. *Neural networks for pattern recognition*. Oxford university press, 1995.

[Bit+24] Sebastian Bitzer et al. “How to Lose Some Weight - A Practical Template Syndrome Decoding Attack”. In: *IACR Cryptol. ePrint Arch.* (2024), p. 621. URL: https://eprint.iacr.org/2024/621.

[Bou+25] Sana Boussam et al. “Optimal Dimensionality Reduction using Conditional Variational AutoEncoder”. In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2025.3 (2025), pp. 164–211. DOI: 10.46586/TCHES.V2025.I3.164-211. URL: https://doi.org/10.46586/tches.v2025.i3.164-211.

[Bre01] Leo Breiman. “Random Forests”. In: *Mach. Learn.* 45.1 (2001), pp. 5–32. DOI: 10.1023/A:1010933404324. URL: https://doi.org/10.1023/A:1010933404324.

[Bre96] Leo Breiman. “Bagging Predictors”. In: *Mach. Learn.* 24.2 (1996), pp. 123–140. DOI: 10.1007/BF00058655. URL: https://doi.org/10.1007/BF00058655.

[Bur+24] Elie Bursztein et al. “Generalized Power Attacks against Crypto Hardware using Long-Range Deep Learning”. In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2024.3 (2024), pp. 472–499. DOI: 10.46586/TCHES.V2024.I3.472-499. URL: https://doi.org/10.46586/tches.v2024.i3.472-499.

[Cag18] Eleonora Cagli. “Feature Extraction for Side-Channel Attacks. (Extraction de caractéristiques pour les attaques par canaux auxiliaires)”. PhD thesis. Sorbonne University, France, 2018. URL: https://tel.archives-ouvertes.fr/tel-02494260.

[Cao+22] Pei Cao et al. “Improving Deep Learning Based Second-Order Side-Channel Analysis With Bilinear CNN”. In: *IEEE Trans. Inf. Forensics Secur.* 17 (2022), pp. 3863–3876. DOI: 10.1109/TIFS.2022.3216959. URL: https://doi.org/10.1109/TIFS.2022.3216959.

[Car+18] Mathilde Caron et al. “Deep Clustering for Unsupervised Learning of Visual Features”. In: *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XIV*. Ed. by Vittorio Ferrari et al. Vol. 11218. Lecture Notes in Computer Science. Springer, 2018, pp. 139–156. DOI: 10.1007/978-3-030-01264-9_9. URL: https://doi.org/10.1007/978-3-030-01264-9_9.

[Car+19] Mathieu Carbone et al. “Deep Learning to Evaluate Secure RSA Implementations”. In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2019.2 (2019), pp. 132–161. DOI: 10.13154/TCHES.V2019.I2.132-161. URL: https://doi.org/10.13154/tches.v2019.i2.132-161.

[CCJ04] Benoît Chevallier-Mames, Mathieu Ciet, and Marc Joye. “Low-Cost Solutions for Preventing Simple Side-Channel Analysis: Side-Channel Atomicity”. In: *IEEE Trans. Computers* 53.6 (2004), pp. 760–768. DOI: 10.1109/TC.2004.13. URL: https://doi.org/10.1109/TC.2004.13.

[CDP17] Eleonora Cagli, Cécile Dumas, and Emmanuel Prouff. “Convolutional Neural Networks with Data Augmentation Against Jitter-Based Countermeasures - Profiling Attacks Without Pre-processing”. In: *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*. Ed. by Wieland Fischer and Naofumi Homma. Vol. 10529. Lecture Notes in Computer Science. Springer, 2017, pp. 45–68. DOI: 10.1007/978-3-319-66787-4_3. URL: https://doi.org/10.1007/978-3-319-66787-4_3.

[CH21] Xinlei Chen and Kaiming He. “Exploring Simple Siamese Representation Learning”. In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*. Computer Vision Foundation / IEEE, 2021, pp. 15750–15758. DOI: 10.1109/CVPR46437.2021.01549. URL: https://openaccess.thecvf.com/content/CVPR2021/html/Chen_Exploring_Simple_Siamese_Representation_Learning_CVPR_2021_paper.html.

[CHL05] Sumit Chopra, Raia Hadsell, and Yann LeCun. “Learning a Similarity Metric Discriminatively, with Application to Face Verification”. In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2005), 20-26 June 2005, San Diego, CA, USA*. IEEE Computer Society, 2005, pp. 539–546. DOI: 10.1109/CVPR.2005.202. URL: https://doi.org/10.1109/CVPR.2005.202.

[CK09] Jean-Sébastien Coron and Ilya Kizhvatov. “An Efficient Method for Random Delay Generation in Embedded Software”. In: *Cryptographic Hardware and Embedded Systems - CHES 2009, 11th International Workshop, Lausanne, Switzerland, September 6-9, 2009, Proceedings*. Ed. by Christophe Clavier and Kris Gaj. Vol. 5747. Lecture Notes in Computer Science. Springer, 2009, pp. 156–170. DOI: 10.1007/978-3-642-04138-9_12. URL: https://doi.org/10.1007/978-3-642-04138-9_12.

[Cla+10] Christophe Clavier et al. “Horizontal Correlation Analysis on Exponentiation”. In: *Information and Communications Security - 12th International Conference, ICICS 2010, Barcelona, Spain, December 15-17, 2010. Proceedings*. Ed.

by Miguel Soriano, Sihan Qing, and Javier López. Vol. 6476. Lecture Notes in Computer Science. Springer, 2010, pp. 46–61. DOI: 10.1007/978-3-642-17650-0_5. URL: https://doi.org/10.1007/978-3-642-17650-0_5.

[Cla+12] Christophe Clavier et al. "ROSETTA for Single Trace Analysis". In: *Progress in Cryptology - INDOCRYPT 2012, 13th International Conference on Cryptology in India, Kolkata, India, December 9-12, 2012. Proceedings*. Ed. by Steven D. Galbraith and Mridul Nandi. Vol. 7668. Lecture Notes in Computer Science. Springer, 2012, pp. 140–155. DOI: 10.1007/978-3-642-34931-7_9. URL: https://doi.org/10.1007/978-3-642-34931-7_9.

[COM23] Gauthier Cler, Sébastien Ordas, and Philippe Maurine. "Bernoulli at the Root of Horizontal Side Channel Attacks". In: *Smart Card Research and Advanced Applications - 22nd International Conference, CARDIS 2023, Amsterdam, The Netherlands, November 14-16, 2023, Revised Selected Papers*. Ed. by Shivam Bhasin and Thomas Roche. Vol. 14530. Lecture Notes in Computer Science. Springer, 2023, pp. 107–126. DOI: 10.1007/978-3-031-54409-5_6. URL: https://doi.org/10.1007/978-3-031-54409-5_6.

[COM25] Gauthier Cler, Sébastien Ordas, and Philippe Maurine. "Improving Leakage Exploitability in Horizontal Side Channel Attacks Through Anomaly Mitigation with Unsupervised Neural Networks". In: *Constructive Approaches for Security Analysis and Design of Embedded Systems - First International Conference, CASCADE 2025, Saint-Etienne, France, April 2-4, 2025, Proceedings*. Ed. by Matthieu Rivain and Pascal Sasdrich. Vol. 15952. Lecture Notes in Computer Science. Springer, 2025, pp. 554–579. DOI: 10.1007/978-3-032-01405-4_23. URL: https://doi.org/10.1007/978-3-032-01405-4_23.

[Com90] Paul G. Comba. "Exponentiation Cryptosystems on the IBM PC". In: *IBM Syst. J.* 29.4 (1990), pp. 526–538. DOI: 10.1147/SJ.294.0526. URL: https://doi.org/10.1147/sj.294.0526.

[Cor99] Jean-Sébastien Coron. "Resistance against Differential Power Analysis for Elliptic Curve Cryptosystems". In: *Cryptographic Hardware and Embedded Systems, First International Workshop, CHES'99, Worcester, MA, USA, August 12-13, 1999, Proceedings*. Ed. by Çetin Kaya Koç and Christof Paar. Vol. 1717. Lecture Notes in Computer Science. Springer, 1999, pp. 292–302. DOI: 10.1007/3-540-48059-5_25. URL: https://doi.org/10.1007/3-540-48059-5_25.

[CRR02] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. "Template Attacks". In: *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International*

Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers. Ed. by Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar. Vol. 2523. Lecture Notes in Computer Science. Springer, 2002, pp. 13–28. DOI: 10.1007/3-540-36400-5_3. URL: https://doi.org/10.1007/3-540-36400-5_3.

[CUH16] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. "Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)". In: *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2016. URL: http://arxiv.org/abs/1511.07289.

[Dah+13] George E. Dahl et al. "Large-scale malware classification using random projections and neural networks". In: *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2013, Vancouver, BC, Canada, May 26-31, 2013*. IEEE, 2013, pp. 3422–3426. DOI: 10.1109/ICASSP.2013.6638293. URL: https://doi.org/10.1109/ICASSP.2013.6638293.

[DBDM03] Tijl De Bie and Bart De Moor. "On the regularization of canonical correlation analysis". In: *Int. Sympos. ICA and BSS* (2003), pp. 785–790.

[Des+20] Gabriel Destouet et al. "Wavelet Scattering Transform and Ensemble Methods for Side-Channel Analysis". In: *Constructive Side-Channel Analysis and Secure Design - 11th International Workshop, COSADE 2020, Lugano, Switzerland, April 1-3, 2020, Revised Selected Papers*. Ed. by Guido Marco Bertoni and Francesco Regazzoni. Vol. 12244. Lecture Notes in Computer Science. Springer, 2020, pp. 71–89. DOI: 10.1007/978-3-030-68773-1_4. URL: https://doi.org/10.1007/978-3-030-68773-1_4.

[DHD24] Ngoc-Tuan Do, Van-Phuc Hoang, and Van-Sang Doan. "A novel non-profiled side channel attack based on multi-output regression neural network". In: *J. Cryptogr. Eng.* 14.3 (2024), pp. 427–439. DOI: 10.1007/S13389-023-00314-4. URL: https://doi.org/10.1007/s13389-023-00314-4.

[DHS10] John C. Duchi, Elad Hazan, and Yoram Singer. "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization". In: *COLT 2010 - The 23rd Conference on Learning Theory, Haifa, Israel, June 27-29, 2010*. Ed. by Adam Tauman Kalai and Mehryar Mohri. Omnipress, 2010, pp. 257–269. URL: http://colt2010.haifa.il.ibm.com/papers/COLT2010proceedings.pdf\#page=265.

[Din+25] Yaoling Ding et al. "An Efficient Ensemble Framework to Assist Profiled Side-Channel Analysis by Machine Learning". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2025).

[Dwo+01] Morris J Dworkin et al. "Advanced encryption standard (AES)". In: (2001).

[FS95] Yoav Freund and Robert E. Schapire. "A decision-theoretic generalization of on-line learning and an application to boosting". In: *Computational Learning Theory, Second European Conference, EuroCOLT '95, Barcelona, Spain, March 13-15, 1995, Proceedings*. Ed. by Paul M. B. Vitányi. Vol. 904. Lecture Notes in Computer Science. Springer, 1995, pp. 23–37. DOI: 10.1007/3-540-59119-2_166. URL: https://doi.org/10.1007/3-540-59119-2_166.

[Fum+10] Guillaume Fumaroli et al. "Affine Masking against Higher-Order Side Channel Analysis". In: *Selected Areas in Cryptography - 17th International Workshop, SAC 2010, Waterloo, Ontario, Canada, August 12-13, 2010, Revised Selected Papers*. Ed. by Alex Biryukov, Guang Gong, and Douglas R. Stinson. Vol. 6544. Lecture Notes in Computer Science. Springer, 2010, pp. 262–280. DOI: 10.1007/978-3-642-19574-7_18. URL: https://doi.org/10.1007/978-3-642-19574-7_18.

[FV03] Pierre-Alain Fouque and Frédéric Valette. "The Doubling Attack - *Why Upwards Is Better than Downwards*". In: *Cryptographic Hardware and Embedded Systems - CHES 2003, 5th International Workshop, Cologne, Germany, September 8-10, 2003, Proceedings*. Ed. by Colin D. Walter, Çetin Kaya Koç, and Christof Paar. Vol. 2779. Lecture Notes in Computer Science. Springer, 2003, pp. 269–280. DOI: 10.1007/978-3-540-45238-6_22. URL: https://doi.org/10.1007/978-3-540-45238-6_22.

[Gao+20] Quanxue Gao et al. "Cross-Modal Subspace Clustering via Deep Canonical Correlation Analysis". In: *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*. AAAI Press, 2020, pp. 3938–3945. DOI: 10.1609/AAAI.V34I04.5808. URL: https://doi.org/10.1609/aaai.v34i04.5808.

[Gao+21] Feng Gao et al. "Leveraging ensemble learning for side channel analysis on masked AES". In: *2021 7th International Conference on Computer and Communications (ICCC)*. IEEE. 2021, pp. 267–271.

[GBB06] Ursula Gonzales-Barron and Francis Butler. "A comparison of seven thresholding techniques with the K-means clustering algorithm for measurement of bread-crumb features by digital image analysis". In: *Journal of food engineering* 74.2 (2006), pp. 268–278.

[GGJR+11] Benjamin Jun Gilbert Goodwill, Josh Jaffe, Pankaj Rohatgi, et al. "A testing methodology for side-channel resistance validation". In: *NIST non-invasive attack testing workshop*. Vol. 7. 2011, pp. 115–136.

[GMO01] Karine Gandolfi, Christophe Mourtel, and Francis Olivier. "Electromagnetic Analysis: Concrete Results". In: *Cryptographic Hardware and Embedded Systems - CHES 2001, Third International Workshop, Paris, France, May 14-16, 2001, Proceedings*. Ed. by Çetin Kaya Koç, David Naccache, and Christof Paar. Vol. 2162. Lecture Notes in Computer Science. Springer, 2001, pp. 251–261. DOI: 10.1007/3-540-44709-1_21. URL: https://doi.org/10.1007/3-540-44709-1_21.

[Goo+14] Ian J. Goodfellow et al. "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*. Ed. by Zoubin Ghahramani et al. 2014, pp. 2672–2680. URL: https://proceedings.neurips.cc/paper/2014/hash/5ca3e9b122\f61f8f06494c97b1afccf3-Abstract.html.

[GP99] Louis Goubin and Jacques Patarin. "DES and Differential Power Analysis (The "Duplication" Method)". In: *Cryptographic Hardware and Embedded Systems, First International Workshop, CHES'99, Worcester, MA, USA, August 12-13, 1999, Proceedings*. Ed. by Çetin Kaya Koç and Christof Paar. Vol. 1717. Lecture Notes in Computer Science. Springer, 1999, pp. 158–172. DOI: 10.1007/3-540-48059-5_15. URL: https://doi.org/10.1007/3-540-48059-5_15.

[Gro+23] Vincent Grosso et al. "Punctured Syndrome Decoding Problem - Efficient Side-Channel Attacks Against Classic McEliece". In: *Constructive Side-Channel Analysis and Secure Design - 14th International Workshop, COSADE 2023, Munich, Germany, April 3-4, 2023, Proceedings*. Ed. by Elif Bilge Kavun and Michael Pehl. Vol. 13979. Lecture Notes in Computer Science. Springer, 2023, pp. 170–192. DOI: 10.1007/978-3-031-29497-6_9. URL: https://doi.org/10.1007/978-3-031-29497-6_9.

[Gui+24] Jie Gui et al. "A Survey on Self-Supervised Learning: Algorithms, Applications, and Future Trends". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 46.12 (2024), pp. 9052–9071. DOI: 10.1109/TPAMI.2024.3415112. URL: https://doi.org/10.1109/TPAMI.2024.3415112.

[Haj+22] Suvadeep Hajra et al. "TransNet: Shift Invariant Transformer Network for Side Channel Analysis". In: *Progress in Cryptology - AFRICACRYPT 2022: 13th International Conference on Cryptology in Africa, AFRICACRYPT 2022, Fes, Morocco, July 18-20, 2022, Proceedings*. Ed. by Lejla Batina and Joan Daemen. Vol. 13503. Lecture Notes in Computer Science. Springer Nature Switzerland, 2022, pp. 371–396. DOI: 10.1007/978-3-031-17433-9_16. URL: https://doi.org/10.1007/978-3-031-17433-9_16.

[Han+18] Bo Han et al. "Co-teaching: Robust training of deep neural networks with extremely noisy labels". In: *Advances in Neural Information Pro-*

cessing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada. Ed. by Samy Bengio et al. 2018, pp. 8536–8546. URL: https://proceedings.neurips.cc/paper/2018/hash/a19744e268754fb0148b017647355b7b-Abstract.html.

[Haq+22] Imran Ul Haq et al. "Feature fusion and Ensemble learning-based CNN model for mammographic image classification". In: *J. King Saud Univ. Comput. Inf. Sci.* 34.6 Part B (2022), pp. 3310–3318. DOI: 10.1016/J.JKSUCI.2022.03.023. URL: https://doi.org/10.1016/j.jksuci.2022.03.023.

[HCM24] Suvadeep Hajra, Siddhartha Chowdhury, and Debdeep Mukhopadhyay. "EstraNet: An Efficient Shift-Invariant Transformer Network for Side-Channel Analysis". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2024.1 (2024), pp. 336–374. DOI: 10.46586/TCHES.V2024.I1.336-374. URL: https://doi.org/10.46586/tches.v2024.i1.336-374.

[He+15] Kaiming He et al. "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification". In: *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*. IEEE Computer Society, 2015, pp. 1026–1034. DOI: 10.1109/ICCV.2015.123. URL: https://doi.org/10.1109/ICCV.2015.123.

[He+16] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. IEEE Computer Society, 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90. URL: https://doi.org/10.1109/CVPR.2016.90.

[He+24] Junlin He et al. "Preventing Model Collapse in Deep Canonical Correlation Analysis by Noise Regularization". In: *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*. Ed. by Amir Globersons et al. 2024. URL: http://papers.nips.cc/paper_files/paper/2024/hash/9626a58529367967b71c17c9b2db72f1-Abstract-Conference.html.

[Her+17] Shawn Hershey et al. "CNN architectures for large-scale audio classification". In: *2017 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2017, New Orleans, LA, USA, March 5-9, 2017*. IEEE, 2017, pp. 131–135. DOI: 10.1109/ICASSP.2017.7952132. URL: https://doi.org/10.1109/ICASSP.2017.7952132.

[Hey+13] Johann Heyszl et al. “Clustering Algorithms for Non-profiled Single-Execution Attacks on Exponentiations”. In: *Smart Card Research and Advanced Applications - 12th International Conference, CARDIS 2013, Berlin, Germany, November 27-29, 2013. Revised Selected Papers*. Ed. by Aurélien Francillon and Pankaj Rohatgi. Vol. 8419. Lecture Notes in Computer Science. Springer, 2013, pp. 79–93. DOI: 10.1007/978-3-319-08302-5_6. URL: https://doi.org/10.1007/978-3-319-08302-5_6.

[HG16] Dan Hendrycks and Kevin Gimpel. “Bridging Nonlinearities and Stochastic Regularizers with Gaussian Error Linear Units”. In: *CoRR* abs/1606.08415 (2016). arXiv: 1606.08415. URL: http://arxiv.org/abs/1606.08415.

[HKT15] Neil Hanley, HeeSeok Kim, and Michael Tunstall. “Exploiting Collisions in Addition Chain-Based Exponentiation Algorithms Using a Single Trace”. In: *Topics in Cryptology - CT-RSA 2015, The Cryptographer’s Track at the RSA Conference 2015, San Francisco, CA, USA, April 20-24, 2015. Proceedings*. Ed. by Kaisa Nyberg. Vol. 9048. Lecture Notes in Computer Science. Springer, 2015, pp. 431–448. DOI: 10.1007/978-3-319-16715-2_23. URL: https://doi.org/10.1007/978-3-319-16715-2_23.

[HMS13] Daniel Hernández-Lobato, Gonzalo Martínez-Muñoz, and Alberto Suárez. “How large should ensembles of classifiers be?” In: *Pattern Recognit.* 46.5 (2013), pp. 1323–1336. DOI: 10.1016/J.PATCOG.2012.10.021. URL: https://doi.org/10.1016/j.patcog.2012.10.021.

[Hom+08] Naofumi Homma et al. “Collision-Based Power Analysis of Modular Exponentiation Using Chosen-Message Pairs”. In: *Cryptographic Hardware and Embedded Systems - CHES 2008, 10th International Workshop, Washington, D.C., USA, August 10-13, 2008. Proceedings*. Ed. by Elisabeth Oswald and Pankaj Rohatgi. Vol. 5154. Lecture Notes in Computer Science. Springer, 2008, pp. 15–29. DOI: 10.1007/978-3-540-85053-3_2. URL: https://doi.org/10.1007/978-3-540-85053-3_2.

[HRA24] Faisal Hameed, Sumesh Manjunath Ramesh, and Hoda Alkhzaimi. “Improved Hybrid Bagging Resampling Framework for Deep Learning-Based Side-Channel Analysis”. In: *Comput.* 13.8 (2024), p. 210. DOI: 10.3390/COMPUTERS13080210. URL: https://doi.org/10.3390/computers13080210.

[HS90] Lars Kai Hansen and Peter Salamon. “Neural Network Ensembles”. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 12.10 (1990), pp. 993–1001. DOI: 10.1109/34.58871. URL: https://doi.org/10.1109/34.58871.

[HSS12] Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. “Neural networks for machine learning lecture 6a overview of mini-batch gradient descent”. In: *Cited on* 14.8 (2012), p. 2.

[IS15] Sergey Ioffe and Christian Szegedy. “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”. In: *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*. Ed. by Francis R. Bach and David M. Blei. Vol. 37. JMLR Workshop and Conference Proceedings. JMLR.org, 2015, pp. 448–456. URL: http://proceedings.mlr.press/v37/ioffe15.html.

[Ito+23] Akira Ito et al. “Formal Analysis of Non-profiled Deep-learning Based Side-channel Attacks”. In: *IACR Cryptol. ePrint Arch.* (2023), p. 1563. URL: https://eprint.iacr.org/2023/1563.

[IUH25] Akira Ito, Rei Ueno, and Naofumi Homma. “Perceived Information Revisited II Information-Theoretical Analysis of Deep-Learning Based Side-Channel Attacks”. In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2025.1 (2025), pp. 450–474. DOI: 10.46586/TCHES.V2025.I1.450-474. URL: https://doi.org/10.46586/tches.v2025.i1.450-474.

[Jad+15] Max Jaderberg et al. “Spatial Transformer Networks”. In: *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*. Ed. by Corinna Cortes et al. 2015, pp. 2017–2025. URL: https://proceedings.neurips.cc/paper/2015/hash/33ceb07bf4eeb3da587e26_8d663aba1a-Abstract.html.

[Jav+16] Ahmad Y. Javaid et al. “A Deep Learning Approach for Network Intrusion Detection System”. In: *EAI Endorsed Trans. Security Safety* 3.9 (2016), e2. DOI: 10.4108/EAI.3-12-2015.2262516. URL: https://doi.org/10.4108/eai.3-12-2015.2262516.

[Jin+22] Li Jing et al. “Understanding Dimensional Collapse in Contrastive Self-supervised Learning”. In: *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL: https://openreview.net/forum?id=YevsQ05DEN7.

[JY02] Marc Joye and Sung-Ming Yen. “The Montgomery Powering Ladder”. In: *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*. Ed. by Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar. Vol. 2523. Lecture Notes in Computer Science. Springer, 2002, pp. 291–302. DOI: 10.1007/3-540-36400-5_22. URL: https://doi.org/10.1007/3-540-36400-5_22.

[Kar+14] Andrej Karpathy et al. “Large-Scale Video Classification with Convolutional Neural Networks”. In: *2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, June 23-28, 2014*. IEEE Computer Society, 2014, pp. 1725–1732. DOI: 10.1109/CVPR.2014.223. URL: https://doi.org/10.1109/CVPR.2014.223.

[Kar63] Anatolii Karatsuba. "Multiplication of multidigit numbers on automata". In: *Soviet physics doklady*. Vol. 7. 1963, pp. 595–596.

[KB15] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2015. URL: http://arxiv.org/abs/1412.6980.

[Ket71] Jon R Kettenring. "Canonical analysis of several sets of variables". In: *Biometrika* 58.3 (1971), pp. 433–451.

[KH91] Anders Krogh and John A. Hertz. "A Simple Weight Decay Can Improve Generalization". In: *Advances in Neural Information Processing Systems 4, [NIPS Conference, Denver, Colorado, USA, December 2-5, 1991]*. Ed. by John E. Moody, Stephen Jose Hanson, and Richard Lippmann. Morgan Kaufmann, 1991, pp. 950–957. URL: http://papers.nips.cc/paper/563-a-simple-weight-decay-can-improve-generalization.

[KHK22] Donggeun Kwon, Seokhie Hong, and Heeseok Kim. "Optimizing Implementations of Non-Profiled Deep Learning-Based Side-Channel Attacks". In: *IEEE Access* 10 (2022), pp. 5957–5967. DOI: 10.1109/ACCESS.2022.3140446. URL: https://doi.org/10.1109/ACCESS.2022.3140446.

[Kho+20] Prannay Khosla et al. "Supervised Contrastive Learning". In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*. Ed. by Hugo Larochelle et al. 2020. URL: https://proceedings.neurips.cc/paper/2020/hash/d89a66c7c80a29b1bdbab0f2a1a94af8-Abstract.html.

[Kim+19] Jaehun Kim et al. "Make Some Noise. Unleashing the Power of Convolutional Neural Networks for Profiled Side-channel Analysis". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2019.3 (2019), pp. 148–179. DOI: 10.13154/TCHES.V2019.I3.148-179. URL: https://doi.org/10.13154/tches.v2019.i3.148-179.

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. "Differential Power Analysis". In: *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*. Ed. by Michael J. Wiener. Vol. 1666. Lecture Notes in Computer Science. Springer, 1999, pp. 388–397. DOI: 10.1007/3-540-48405-1_25. URL: https://doi.org/10.1007/3-540-48405-1_25.

[Kla+17] Günter Klambauer et al. “Self-Normalizing Neural Networks”. In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. Ed. by Isabelle Guyon et al. 2017, pp. 971–980. URL: https://proceedings.neurips.cc/paper/2017/hash/5d44ee6f2c3f71b73125876103c8f6c4-Abstract.html.

[Kne+23] Karlo Knezevic et al. “NeuroSCA: Evolving Activation Functions for Side-Channel Analysis”. In: *IEEE Access* 11 (2023), pp. 284–299. DOI: 10.1109/ACCESS.2022.3232064. URL: https://doi.org/10.1109/ACCESS.2022.3232064.

[Koc96] Paul C. Kocher. “Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems”. In: *Advances in Cryptology - CRYPTO ’96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings*. Ed. by Neal Koblitz. Vol. 1109. Lecture Notes in Computer Science. Santa Barbara, California: Springer, 1996, pp. 104–113. DOI: 10.1007/3-540-68697-5_9. URL: https://doi.org/10.1007/3-540-68697-5_9.

[Kra91] Mark A Kramer. “Nonlinear principal component analysis using autoassociative neural networks”. In: *AIChE journal* 37.2 (1991), pp. 233–243.

[Kur+23] Kunihiro Kuroda et al. “Practical aspects on non-profiled deep-learning side-channel attacks against AES software implementation with two types of masking countermeasures including RSM”. In: *J. Cryptogr. Eng.* 13.4 (2023), pp. 427–442. DOI: 10.1007/S13389-023-00312-6. URL: https://doi.org/10.1007/s13389-023-00312-6.

[KZS+15] Gregory Koch, Richard Zemel, Ruslan Salakhutdinov, et al. “Siamese neural networks for one-shot image recognition”. In: *ICML deep learning workshop*. Vol. 2. 1. Lille. 2015, pp. 1–30.

[LB98] Yann LeCun and Yoshua Bengio. “Convolutional networks for images, speech, and time series”. In: *The handbook of brain theory and neural networks* (1998).

[LF00] Pei Ling Lai and Colin Fyfe. “Kernel and Nonlinear Canonical Correlation Analysis”. In: *Int. J. Neural Syst.* 10.5 (2000), pp. 365–377. DOI: 10.1142/S012906570000034X. URL: https://doi.org/10.1142/S012906570000034X.

[LHK22] Nayeon Lee, Seokhie Hong, and Heeseok Kim. “Single-Trace Attack Using One-Shot Learning With Siamese Network in Non-Profiled Setting”. In: *IEEE Access* 10 (2022), pp. 60778–60789. DOI: 10.1109/ACCESS.2022.3180742. URL: https://doi.org/10.1109/ACCESS.2022.3180742.

[Lie+16] Shan Sung Liew et al. “Gender classification: a convolutional neural network approach”. In: *Turkish Journal of Electrical Engineering and Computer Sciences* 24.3 (2016), pp. 1248–1264.

[Liu+19] Wei Liu et al. “Multimodal Emotion Recognition Using Deep Canonical Correlation Analysis”. In: *CoRR* abs/1908.05349 (2019). arXiv: 1908.05349. URL: http://arxiv.org/abs/1908.05349.

[Lla+23] Dorian Llavata et al. “Deep Stacking Ensemble Learning Applied to Profiling Side-Channel Attacks”. In: *Smart Card Research and Advanced Applications - 22nd International Conference, CARDIS 2023, Amsterdam, The Netherlands, November 14-16, 2023, Revised Selected Papers*. Ed. by Shivam Bhasin and Thomas Roche. Vol. 14530. Lecture Notes in Computer Science. Springer, 2023, pp. 235–255. DOI: 10.1007/978-3-031-54409-5_12. URL: https://doi.org/10.1007/978-3-031-54409-5_12.

[Lla+25] Dorian Llavata et al. “Unsupervised Horizontal Attacks against Public-Key Primitives with DCCA: - From Deep Canonical Correlation Analysis to Deep Collision Correlation Attacks -”. In: *IACR Commun. Cryptol.* 2.1 (2025), p. 32. DOI: 10.62056/AH5W7TA5V. URL: https://doi.org/10.62056/ah5w7ta5v.

[LLO24] Di Li, Lang Li, and Yu Ou. “Side-channel analysis based on Siamese neural network”. In: *J. Supercomput.* 80.4 (2024), pp. 4423–4450. DOI: 10.1007/S11227-023-05631-3. URL: https://doi.org/10.1007/s11227-023-05631-3.

[LY09] Dongju Liu and Jian Yu. “Otsu Method and K-means”. In: *9th International Conference on Hybrid Intelligent Systems (HIS 2009), August 12-14, 2009, Shenyang, China*. Ed. by Ge Yu et al. IEEE Computer Society, 2009, pp. 344–349. DOI: 10.1109/HIS.2009.74. URL: https://doi.org/10.1109/HIS.2009.74.

[MB89] Nelson Morgan and Hervé Bourlard. “Generalization and Parameter Estimation in Feedforward Netws: Some Experiments”. In: *Advances in Neural Information Processing Systems 2, [NIPS Conference, Denver, Colorado, USA, November 27-30, 1989]*. Ed. by David S. Touretzky. Morgan Kaufmann, 1989, pp. 630–637. URL: http://papers.nips.cc/paper/275-generalization-and-parameter-estimation-in-feedforward-nets-some-experiments.

[MCM83] RS Michalski, JG Carbonell, and TM Mitchell. “Machine learning: An artificial intelligence approach”. In: (1983).

[MG19] Samaneh Mahdavifar and Ali A. Ghorbani. “Application of deep learning to cybersecurity: A survey”. In: *Neurocomputing* 347 (2019), pp. 149–176. DOI: 10.1016/J.NEUCOM.2019.02.056. URL: https://doi.org/10.1016/j.neucom.2019.02.056.

[MH24] Gabrielle De Micheli and Nadia Heninger. “Survey: Recovering cryptographic keys from partial information, by example”. In: *IACR Commun. Cryptol.* 1.1 (2024), p. 28. DOI: 10.62056/AHJBKSDJA. URL: https://doi.org/10.62056/ahjbksdja.

[MHN+13] Andrew L Maas, Awni Y Hannun, Andrew Y Ng, et al. “Rectifier nonlinearities improve neural network acoustic models”. In: *Proc. icml.* Vol. 30. 1. Atlanta, GA. 2013, p. 3.

[Mil85] Victor S. Miller. “Use of Elliptic Curves in Cryptography”. In: *Advances in Cryptology - CRYPTO ’85, Santa Barbara, California, USA, August 18-22, 1985, Proceedings*. Ed. by Hugh C. Williams. Vol. 218. Lecture Notes in Computer Science. Springer, 1985, pp. 417–426. DOI: 10.1007/3-540-39799-X_31. URL: https://doi.org/10.1007/3-540-39799-X_31.

[Min+23] Shervin Minaee et al. “Biometrics recognition using deep learning: a survey”. In: *Artif. Intell. Rev.* 56.8 (2023), pp. 8647–8695. DOI: 10.1007/S10462-022-10237-X. URL: https://doi.org/10.1007/s10462-022-10237-x.

[MK22] Ammar Mohammed and Rania Kora. “An effective ensemble deep learning framework for text classification”. In: *J. King Saud Univ. Comput. Inf. Sci.* 34.10 Part A (2022), pp. 8825–8837. DOI: 10.1016/J.JKSUCI.2021.11.001. URL: https://doi.org/10.1016/j.jksuci.2021.11.001.

[MK23] Ammar Mohammed and Rania Kora. “A comprehensive review on ensemble deep learning: Opportunities and challenges”. In: *J. King Saud Univ. Comput. Inf. Sci.* 35.2 (2023), pp. 757–774. DOI: 10.1016/J.JKSUCI.2023.01.014. URL: https://doi.org/10.1016/j.jksuci.2023.01.014.

[MM79] Nick Martin and Hermine Maes. “Multivariate analysis”. In: *London, UK: Academic* (1979).

[MMD20] Coenraad Mouton, Johannes C. Myburgh, and Marelie H. Davel. “Stride and Translation Invariance in CNNs”. In: *Artificial Intelligence Research - First Southern African Conference for AI Research, SACAIR 2020, Muldersdrift, South Africa, February 22-26, 2021, Proceedings*. Ed. by Aurona J. Gerber. Vol. 1342. Communications in Computer and Information Science. Springer, 2020, pp. 267–281. DOI: 10.1007/978-3-030-66151-9_17. URL: https://doi.org/10.1007/978-3-030-66151-9_17.

[MME10] Amir Moradi, Oliver Mischke, and Thomas Eisenbarth. “Correlation-Enhanced Power Analysis Collision Attack”. In: *Cryptographic Hardware and Embedded Systems, CHES 2010, 12th International Workshop, Santa Barbara,*

CA, USA, August 17-20, 2010. Proceedings. Ed. by Stefan Mangard and François-Xavier Standaert. Vol. 6225. Lecture Notes in Computer Science. Springer, 2010, pp. 125–139. DOI: 10.1007/978-3-642-15031-9_9. URL: https://doi.org/10.1007/978-3-642-15031-9_9.

[Mon85] Peter L Montgomery. "Modular multiplication without trial division". In: *Mathematics of computation* 44.170 (1985), pp. 519–521.

[Moo+02] Simon W. Moore et al. "Improving Smart Card Security Using Self-Timed Circuits". In: *8th International Symposium on Advanced Research in Asynchronous Circuits and Systems (ASYNC 2002), 9-11 April 2002, Manchester, UK*. IEEE Computer Society, 2002, pp. 211–218. DOI: 10.1109/ASYNC.2002.1000311. URL: https://doi.org/10.1109/ASYNC.2002.1000311.

[MOP07] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power analysis attacks - revealing the secrets of smart cards*. Springer, 2007. ISBN: 978-0-387-30857-9.

[MPP16] Houssem Maghrebi, Thibault Portigliatti, and Emmanuel Prouff. "Breaking Cryptographic Implementations Using Deep Learning Techniques". In: *Security, Privacy, and Applied Cryptography Engineering - 6th International Conference, SPACE 2016, Hyderabad, India, December 14-18, 2016, Proceedings*. Ed. by Claude Carlet, M. Anwar Hasan, and Vishal Saraswat. Vol. 10076. Lecture Notes in Computer Science. Springer, 2016, pp. 3–26. DOI: 10.1007/978-3-319-49445-6_1. URL: https://doi.org/10.1007/978-3-319-49445-6_1.

[MTM14] Rami Mustafa A. Mohammad, Fadi A. Thabtah, and Lee McCluskey. "Predicting phishing websites based on self-structuring neural network". In: *Neural Comput. Appl.* 25.2 (2014), pp. 443–458. DOI: 10.1007/S00521-013-1490-Z. URL: https://doi.org/10.1007/s00521-013-1490-z.

[Muk+19] Sudipto Mukherjee et al. "ClusterGAN: Latent Space Clustering in Generative Adversarial Networks". In: *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*. AAAI Press, 2019, pp. 4610–4617. DOI: 10.1609/AAAI.V33I01.33014610. URL: https://doi.org/10.1609/aaai.v33i01.33014610.

[Muk+22] Naila Mukhtar et al. "Fake It Till You Make It: Data Augmentation Using Generative Adversarial Networks for All the Crypto You Need on Small Devices". In: *Topics in Cryptology - CT-RSA 2022 - Cryptographers' Track at the RSA Conference 2022, Virtual Event, March 1-2, 2022, Proceedings*. Ed. by Steven D. Galbraith. Vol. 13161. Lecture Notes in Computer Science. Springer, 2022, pp. 297–

321. DOI: 10.1007/978-3-030-95312-6_13. URL: https://doi.org/10.1007/978-3-030-95312-6_13.

[MWM21] Thorben Moos, Felix Wegener, and Amir Moradi. "DL-LA: Deep Learning Leakage Assessment A modern roadmap for SCA evaluations". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2021.3 (2021), pp. 552–598. DOI: 10.46586/TCHES.V2021.I3.552-598. URL: https://doi.org/10.46586/tches.v2021.i3.552-598.

[MY24] Zhao Minghui and Trevor Yap. "Avenger Ensemble: Genetic Algorithm-Driven Ensemble Selection for Deep Learning-based Side-Channel Analysis". In: *Cryptology ePrint Archive* (2024).

[Nan+20] Loris Nanni et al. "Ensemble of convolutional neural networks to improve animal audio classification". In: *EURASIP J. Audio Speech Music. Process.* 2020.1 (2020), p. 8. DOI: 10.1186/S13636-020-00175-3. URL: https://doi.org/10.1186/s13636-020-00175-3.

[NBS22] Meenal V. Narkhede, Prashant P. Bartakke, and Mukul S. Sutaone. "A review on weight initialization strategies for neural networks". In: *Artif. Intell. Rev.* 55.1 (2022), pp. 291–322. DOI: 10.1007/S10462-021-10033-Z. URL: https://doi.org/10.1007/s10462-021-10033-z.

[NC17] Erick Nascimento and Lukasz Chmielewski. "Applying Horizontal Clustering Side-Channel Attacks on Embedded ECC Implementations". In: *Smart Card Research and Advanced Applications - 16th International Conference, CARDIS 2017, Lugano, Switzerland, November 13-15, 2017, Revised Selected Papers*. Ed. by Thomas Eisenbarth and Yannick Teglia. Vol. 10728. Lecture Notes in Computer Science. Springer, 2017, pp. 213–231. DOI: 10.1007/978-3-319-75208-2_13. URL: https://doi.org/10.1007/978-3-319-75208-2_13.

[NF95] Beat E Neuenschwander and Bernard D Flury. "Common canonical variates". In: *Biometrika* 82.3 (1995), pp. 553–560.

[Ngu+22] Thanh Nguyen et al. "Deep learning for deepfakes creation and detection: A survey". In: *Comput. Vis. Image Underst.* 223 (2022), p. 103525. DOI: 10.1016/J.CVIU.2022.103525. URL: https://doi.org/10.1016/j.cviu.2022.103525.

[NH10] Vinod Nair and Geoffrey E. Hinton. "Rectified Linear Units Improve Restricted Boltzmann Machines". In: *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*. Ed. by Johannes Fürnkranz and Thorsten Joachims. Omnipress, 2010, pp. 807–814. URL: https://icml.cc/Conferences/2010/papers/432.pdf.

[OM99] David W. Opitz and Richard Maclin. "Popular Ensemble Methods: An Empirical Study". In: *J. Artif. Intell. Res.* 11 (1999), pp. 169–198. DOI: 10.1613/JAIR.614. URL: https://doi.org/10.1613/jair.614.

[PC15] Guilherme Perin and Lukasz Chmielewski. "A Semi-Parametric Approach for Side-Channel Attacks on Protected RSA Implementations". In: *Smart Card Research and Advanced Applications - 14th International Conference, CARDIS 2015, Bochum, Germany, November 4-6, 2015. Revised Selected Papers*. Ed. by Naofumi Homma and Marcel Medwed. Vol. 9514. Lecture Notes in Computer Science. Springer, 2015, pp. 34–53. DOI: 10.1007/978-3-319-31271-2_3. URL: https://doi.org/10.1007/978-3-319-31271-2_3.

[PCP20] Guilherme Perin, Lukasz Chmielewski, and Stjepan Picek. "Strength in Numbers: Improving Generalization with Ensembles in Machine Learning-based Profiled Side-channel Analysis". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2020.4 (2020), pp. 337–364. DOI: 10.13154/TCHES.V2020.I4.337-364. URL: https://doi.org/10.13154/tches.v2020.i4.337-364.

[Per+14] Guilherme Perin et al. "Attacking Randomized Exponentiations Using Unsupervised Learning". In: *Constructive Side-Channel Analysis and Secure Design - 5th International Workshop, COSADE 2014, Paris, France, April 13-15, 2014. Revised Selected Papers*. Ed. by Emmanuel Prouff. Vol. 8622. Lecture Notes in Computer Science. Springer, 2014, pp. 144–160. DOI: 10.1007/978-3-319-10175-0_11. URL: https://doi.org/10.1007/978-3-319-10175-0_11.

[Per+21] Guilherme Perin et al. "Keep it Unsupervised: Horizontal Attacks Meet Deep Learning". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2021.1 (2021), pp. 343–372. DOI: 10.46586/TCHES.V2021.I1.343-372. URL: https://doi.org/10.46586/tches.v2021.i1.343-372.

[Pic+19] Stjepan Picek et al. "The Curse of Class Imbalance and Conflicting Metrics with Machine Learning for Side-channel Evaluations". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2019.1 (2019), pp. 209–237. DOI: 10.13154/TCHES.V2019.I1.209-237. URL: https://doi.org/10.13154/tches.v2019.i1.209-237.

[PSG16] Romain Poussier, François-Xavier Standaert, and Vincent Grosso. "Simple Key Enumeration (and Rank Estimation) Using Histograms: An Integrated Approach". In: *Cryptographic Hardware and Embedded Systems - CHES 2016 - 18th International Conference, Santa Barbara, CA, USA, August 17-19, 2016, Proceedings*. Ed. by Benedikt Gierlichs and Axel Y. Poschmann. Vol. 9813. Lecture Notes in Computer Science. Springer, 2016, pp. 61–81. DOI: 10.1007/978-3-662-53140-2_4. URL: https://doi.org/10.1007/978-3-662-53140-2_4.

[QLL18] Jie-Lin Qiu, Wei Liu, and Bao-Liang Lu. "Multi-view Emotion Recognition Using Deep Canonical Correlation Analysis". In: *Neural Information Processing*

- *25th International Conference, ICONIP 2018, Siem Reap, Cambodia, December 13-16, 2018, Proceedings, Part V*. Ed. by Long Cheng, Andrew Chi-Sing Leung, and Seiichi Ozawa. Vol. 11305. Lecture Notes in Computer Science. Springer, 2018, pp. 221–231. DOI: 10.1007/978-3-030-04221-9_20. URL: https://doi.org/10.1007/978-3-030-04221-9_20.

[RAD20] Keyvan Ramezanpour, Paul Ampadu, and William Diehl. "SCAUL: Power Side-Channel Analysis With Unsupervised Learning". In: *IEEE Trans. Computers* 69.11 (2020), pp. 1626–1638. DOI: 10.1109/TC.2020.3013196. URL: https://doi.org/10.1109/TC.2020.3013196.

[Ren+25] Yazhou Ren et al. "Deep Clustering: A Comprehensive Survey". In: *IEEE Trans. Neural Networks Learn. Syst.* 36.4 (2025), pp. 5858–5878. DOI: 10.1109/TNNLS.2024.3403155. URL: https://doi.org/10.1109/TNNLS.2024.3403155.

[RHW85] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. *Learning internal representations by error propagation*. Tech. rep. 1985.

[RHW86] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. "Learning representations by back-propagating errors". In: *nature* 323.6088 (1986), pp. 533–536.

[Rij+21] Jorai Rijsdijk et al. "Reinforcement Learning for Hyperparameter Tuning in Deep Learning-based Side-channel Analysis". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2021.3 (2021), pp. 677–707. DOI: 10.46586/TCHES.V2021.I3.677-707. URL: https://doi.org/10.46586/tches.v2021.i3.677-707.

[RMH17] Ciara Rafferty, Máire McLoone, and Neil Hanley. "Evaluation of Large Integer Multiplication Methods on Hardware". In: *IEEE Trans. Computers* 66.8 (2017), pp. 1369–1382. DOI: 10.1109/TC.2017.2677426. URL: https://doi.org/10.1109/TC.2017.2677426.

[Rob+20] Damien Robissout et al. "Online Performance Evaluation of Deep Learning Networks for Profiled Side-Channel Analysis". In: *Constructive Side-Channel Analysis and Secure Design - 11th International Workshop, COSADE 2020, Lugano, Switzerland, April 1-3, 2020, Revised Selected Papers*. Ed. by Guido Marco Bertoni and Francesco Regazzoni. Vol. 12244. Lecture Notes in Computer Science. Springer, 2020, pp. 200–218. DOI: 10.1007/978-3-030-68773-1_10. URL: https://doi.org/10.1007/978-3-030-68773-1_10.

[Rob+21] Damien Robissout et al. "Improving Deep Learning Networks for Profiled Side-channel Analysis Using Performance Improvement Techniques". In: *ACM J. Emerg. Technol. Comput. Syst.* 17.3 (2021), 41:1–41:30. DOI: 10.1145/3453162. URL: https://doi.org/10.1145/3453162.

[Ros58] Frank Rosenblatt. "The perceptron: a probabilistic model for information storage and organization in the brain." In: *Psychological review* 65.6 (1958), p. 386.

[Rou+25] Nathan Rousselot et al. "Scoop: An Optimization Algorithm for Profiling Attacks against Higher-Order Masking". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2025.3 (2025), pp. 56–80. DOI: 10.46586/TCHES.V2025.I3.56-80. URL: https://doi.org/10.46586/tches.v2025.i3.56-80.

[Rou87] Peter J Rousseeuw. "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis". In: *Journal of computational and applied mathematics* 20 (1987), pp. 53–65.

[Roy+24] Sayak Saha Roy et al. "From Chatbots to Phishbots?: Phishing Scam Generation in Commercial Large Language Models". In: *IEEE Symposium on Security and Privacy, SP 2024, San Francisco, CA, USA, May 19-23, 2024*. IEEE, 2024, pp. 36–54. DOI: 10.1109/SP54263.2024.00182. URL: https://doi.org/10.1109/SP54263.2024.00182.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems". In: *Commun. ACM* 21.2 (1978), pp. 120–126. DOI: 10.1145/359340.359342. URL: https://doi.org/10.1145/359340.359342.

[RW17] Waseem Rawat and Zenghui Wang. "Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review". In: *Neural Comput.* 29.9 (2017), pp. 2352–2449. DOI: 10.1162/NECO_A_00990. URL: https://doi.org/10.1162/neco_a_00990.

[Sai+22] Kotaro Saito et al. "One Truth Prevails: A Deep-learning Based Single-Trace Power Analysis on RSA-CRT with Windowed Exponentiation". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2022.4 (2022), pp. 490–526. DOI: 10.46586/TCHES.V2022.I4.490-526. URL: https://doi.org/10.46586/tches.v2022.i4.490-526.

[Sam59] Arthur L Samuel. "Machine learning". In: *The Technology Review* 62.1 (1959), pp. 42–45.

[She+23] Xiang-Jun Shen et al. "Extraordinarily Time-and Memory-Efficient Large-Scale Canonical Correlation Analysis in Fourier Domain: From Shallow to Deep". In: *IEEE Transactions on Neural Networks and Learning Systems* (2023).

[Shi+22] Seiya Shimada et al. "Deep Learning-Based Side-Channel Attacks against Software-Implemented RSA using Binary Exponentiation with Dummy Multiplication". In: *Proceedings of the 4th International Symposium on Advanced Technologies and Applications in the Internet of Things (ATAIT 2022), Ibaraki and Virtual, Japan, August 24-26, 2022*. Ed. by Hiroki Nishikawa and Xiangbo Kong. Vol. 3198. CEUR Workshop Proceedings. CEUR-WS.org, 2022, pp. 75–84. URL: https://ceur-ws.org/Vol-3198/paper10.pdf.

[Sho97] Peter W. Shor. “Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer”. In: *SIAM J. Comput.* 26.5 (1997), pp. 1484–1509. DOI: 10.1137/S0097539795293172. URL: https://doi.org/10.1137/S0097539795293172.

[SI+23] SOG-IS et al. *SOG-is crypto evaluation scheme. Agreed cryptographic mechanisms.* 2023. URL: https://www.sogis.eu/documents/cc/crypto/SOGIS-Agreed-Cryptographic-Mechanisms-1.3.pdf.

[SLP05] Werner Schindler, Kerstin Lemke, and Christof Paar. “A Stochastic Model for Differential Side Channel Cryptanalysis”. In: *Cryptographic Hardware and Embedded Systems - CHES 2005, 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005, Proceedings*. Ed. by Josyula R. Rao and Berk Sunar. Vol. 3659. Lecture Notes in Computer Science. Springer, 2005, pp. 30–46. DOI: 10.1007/11545262_3. URL: https://doi.org/10.1007/11545262_3.

[SM15] Tobias Schneider and Amir Moradi. “Leakage Assessment Methodology - A Clear Roadmap for Side-Channel Evaluations”. In: *Cryptographic Hardware and Embedded Systems - CHES 2015 - 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings*. Ed. by Tim Güneysu and Helena Handschuh. Vol. 9293. Lecture Notes in Computer Science. Springer, 2015, pp. 495–513. DOI: 10.1007/978-3-662-48324-4_25. URL: https://doi.org/10.1007/978-3-662-48324-4_25.

[SM23] Marvin Staib and Amir Moradi. “Deep Learning Side-Channel Collision Attack”. In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2023.3 (2023), pp. 422–444. DOI: 10.46586/TCHES.V2023.I3.422-444. URL: https://doi.org/10.46586/tches.v2023.i3.422-444.

[SMY09] François-Xavier Standaert, Tal Malkin, and Moti Yung. “A Unified Framework for the Analysis of Side-Channel Key Recovery Attacks”. In: *Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings*. Ed. by Antoine Joux. Vol. 5479. Lecture Notes in Computer Science. Springer, 2009, pp. 443–461. DOI: 10.1007/978-3-642-01001-9_26. URL: https://doi.org/10.1007/978-3-642-01001-9_26.

[Spe+15] Robert Specht et al. “Improving Non-profiled Attacks on Exponentiations Based on Clustering and Extracting Leakage from Multi-channel High-Resolution EM Measurements”. In: *Constructive Side-Channel Analysis and Secure Design - 6th International Workshop, COSADE 2015, Berlin, Germany, April 13-14, 2015. Revised Selected Papers*. Ed. by Stefan Mangard and Axel Y. Poschmann. Vol. 9064. Lecture Notes in Computer Science. Springer, 2015,

pp. 3–19. DOI: 10.1007/978-3-319-21476-4_1. URL: https://doi.org/10.1007/978-3-319-21476-4_1.

[Sri+14] Nitish Srivastava et al. "Dropout: a simple way to prevent neural networks from overfitting". In: *J. Mach. Learn. Res.* 15.1 (2014), pp. 1929–1958. DOI: 10.5555/2627435.2670313. URL: https://dl.acm.org/doi/10.5555/2627435.2670313.

[SSS15] Takeshi Sugawara, Daisuke Suzuki, and Minoru Saeki. "Two Operands of Multipliers in Side-Channel Attack". In: *Constructive Side-Channel Analysis and Secure Design - 6th International Workshop, COSADE 2015, Berlin, Germany, April 13-14, 2015. Revised Selected Papers*. Ed. by Stefan Mangard and Axel Y. Poschmann. Vol. 9064. Lecture Notes in Computer Science. Springer, 2015, pp. 64–78. DOI: 10.1007/978-3-319-21476-4_5. URL: https://doi.org/10.1007/978-3-319-21476-4_5.

[Sun+20] Zhongkai Sun et al. "Learning Relationships between Text, Audio, and Video via Deep Canonical Correlation for Multimodal Language Analysis". In: *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*. AAAI Press, 2020, pp. 8992–8999. DOI: 10.1609/AAAI.V34I05.6431. URL: https://doi.org/10.1609/aaai.v34i05.6431.

[Tan+23] M. Tanveer et al. "Ensemble deep learning in speech signal tasks: A review". In: *Neurocomputing* 550 (2023), p. 126436. DOI: 10.1016/J.NEUCOM.2023.126436. URL: https://doi.org/10.1016/j.neucom.2023.126436.

[Tim19] Benjamin Timon. "Non-Profiled Deep Learning-based Side-Channel attacks with Sensitivity Analysis". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2019.2 (2019), pp. 107–131. DOI: 10.13154/TCHES.V2019.I2.107-131. URL: https://doi.org/10.13154/tches.v2019.i2.107-131.

[Vas+17] Ashish Vaswani et al. "Attention is All you Need". In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. Ed. by Isabelle Guyon et al. 2017, pp. 5998–6008. URL: https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

[Wal01] Colin D. Walter. "Sliding Windows Succumbs to Big Mac Attack". In: *Cryptographic Hardware and Embedded Systems - CHES 2001, Third International Workshop, Paris, France, May 14-16, 2001, Proceedings*. Ed. by Çetin Kaya Koç, David Naccache, and Christof Paar. Vol. 2162. Lecture Notes in Computer Sci-

ence. Springer, 2001, pp. 286–299. DOI: 10.1007/3-540-44709-1_24. URL: https://doi.org/10.1007/3-540-44709-1_24.

[Wol92] David H. Wolpert. "Stacked generalization". In: *Neural Networks* 5.2 (1992), pp. 241–259. DOI: 10.1016/S0893-6080(05)80023-1. URL: https://doi.org/10.1016/S0893-6080(05)80023-1.

[Won+16] Sebastien C. Wong et al. "Understanding Data Augmentation for Classification: When to Warp?" In: *2016 International Conference on Digital Image Computing: Techniques and Applications, DICTA 2016, Gold Coast, Australia, November 30 - December 2, 2016*. IEEE, 2016, pp. 1–6. DOI: 10.1109/DICTA.2016.7797091. URL: https://doi.org/10.1109/DICTA.2016.7797091.

[Wou+20] Lennert Wouters et al. "Revisiting a Methodology for Efficient CNN Architectures in Profiling Attacks". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2020.3 (2020), pp. 147–168. DOI: 10.13154/TCHES.V2020.I3.147-168. URL: https://doi.org/10.13154/tches.v2020.i3.147-168.

[WP20] Lichao Wu and Stjepan Picek. "Remove Some Noise: On Pre-processing of Side-channel Measurements with Autoencoders". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2020.4 (2020), pp. 389–415. DOI: 10.13154/TCHES.V2020.I4.389-415. URL: https://doi.org/10.13154/tches.v2020.i4.389-415.

[WPP24] Lichao Wu, Guilherme Perin, and Stjepan Picek. "I Choose You: Automated Hyperparameter Tuning for Deep Learning-Based Side-Channel Analysis". In: *IEEE Trans. Emerg. Top. Comput.* 12.2 (2024), pp. 546–557. DOI: 10.1109/TETC.2022.3218372. URL: https://doi.org/10.1109/TETC.2022.3218372.

[WT23] Yaru Wang and Ming Tang. "A survey of side-channel leakage assessment". In: *Electronics* 12.16 (2023), p. 3461.

[WWB11] Jasper G. J. van Woudenberg, Marc F. Witteman, and Bram Bakker. "Improving Differential Power Analysis by Elastic Alignment". In: *Topics in Cryptology - CT-RSA 2011 - The Cryptographers' Track at the RSA Conference 2011, San Francisco, CA, USA, February 14-18, 2011. Proceedings*. Ed. by Aggelos Kiayias. Vol. 6558. Lecture Notes in Computer Science. Springer, 2011, pp. 104–119. DOI: 10.1007/978-3-642-19074-2_8. URL: https://doi.org/10.1007/978-3-642-19074-2_8.

[WWM11] Marc F. Witteman, Jasper G. J. van Woudenberg, and Federico Menarini. "Defeating RSA Multiply-Always and Message Blinding Countermeasures". In: *Topics in Cryptology - CT-RSA 2011 - The Cryptographers' Track at the RSA Conference 2011, San Francisco, CA, USA, February 14-18, 2011. Proceedings*. Ed. by Aggelos Kiayias. Vol. 6558. Lecture Notes in Computer Science. Springer,

2011, pp. 77–88. DOI: 10.1007/978-3-642-19074-2_6. URL: https://doi.org/10.1007/978-3-642-19074-2_6.

[XGF16] Junyuan Xie, Ross B. Girshick, and Ali Farhadi. "Unsupervised Deep Embedding for Clustering Analysis". In: *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*. Ed. by Maria-Florina Balcan and Kilian Q. Weinberger. Vol. 48. JMLR Workshop and Conference Proceedings. JMLR.org, 2016, pp. 478–487. URL: http://proceedings.mlr.press/v48/xieb16.html.

[Yan+17] Bo Yang et al. "Towards K-means-friendly Spaces: Simultaneous Deep Learning and Clustering". In: *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, 2017, pp. 3861–3870. URL: http://proceedings.mlr.press/v70/yang17b.html.

[YLO25] Zhicheng Yin, Lang Li, and Yu Ou. "A Novel Stacked Network Method for Enhancing the Performance of Side-Channel Attacks." In: *Computers, Materials & Continua* 83.1 (2025).

[YM15] Fei Yan and Krystian Mikolajczyk. "Deep correlation for matching images and text". In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*. IEEE Computer Society, 2015, pp. 3441–3450. DOI: 10.1109/CVPR.2015.7298966. URL: https://doi.org/10.1109/CVPR.2015.7298966.

[Zai+20a] Gabriel Zaid et al. "Methodology for Efficient CNN Architectures in Profiling Attacks". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2020.1 (2020), pp. 1–36. DOI: 10.13154/TCHES.V2020.I1.1-36. URL: https://doi.org/10.13154/tches.v2020.i1.1-36.

[Zai+20b] Gabriel Zaid et al. "Understanding Methodology for Efficient CNN Architectures in Profiling Attacks". In: *IACR Cryptol. ePrint Arch.* (2020), p. 757. URL: https://eprint.iacr.org/2020/757.

[Zai+21a] Gabriel Zaid et al. "Efficiency through Diversity in Ensemble Models applied to Side-Channel Attacks - A Case Study on Public-Key Algorithms -". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2021.3 (2021), pp. 60–96. DOI: 10.46586/TCHES.V2021.I3.60-96. URL: https://doi.org/10.46586/tches.v2021.i3.60-96.

[Zai+21b] Gabriel Zaid et al. "Ranking Loss: Maximizing the Success Rate in Deep Learning Side-Channel Analysis". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2021.1 (2021), pp. 25–55. DOI: 10.46586/TCHES.V2021.I1.25-55. URL: https://doi.org/10.46586/tches.v2021.i1.25-55.

[Zai+22] Salah Zaiem et al. “Pretext Tasks Selection for Multitask Self-Supervised Audio Representation Learning”. In: *IEEE J. Sel. Top. Signal Process.* 16.6 (2022), pp. 1439–1453. DOI: 10.1109/JSTSP.2022.3195430. URL: https://doi.org/10.1109/JSTSP.2022.3195430.

[Zai+23] Gabriel Zaid et al. “Conditional Variational AutoEncoder based on Stochastic Attacks”. In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2023.2 (2023), pp. 310–357. DOI: 10.46586/TCHES.V2023.I2.310-357. URL: https://doi.org/10.46586/tches.v2023.i2.310-357.

[Zbo+21] Jure Zbontar et al. “Barlow Twins: Self-Supervised Learning via Redundancy Reduction”. In: *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 2021, pp. 12310–12320. URL: http://proceedings.mlr.press/v139/zbontar21a.html.

[Zei12] Matthew D. Zeiler. “ADADELTA: An Adaptive Learning Rate Method”. In: *CoRR* abs/1212.5701 (2012). arXiv: 1212.5701. URL: http://arxiv.org/abs/1212.5701.

[Zhe+19] Jiewan Zheng et al. “Deep Ensemble Machine for Video Classification”. In: *IEEE Trans. Neural Networks Learn. Syst.* 30.2 (2019), pp. 553–565. DOI: 10.1109/TNNLS.2018.2844464. URL: https://doi.org/10.1109/TNNLS.2018.2844464.

[ZL25] Junfan Zhu and Jiqiang Lu. “Leading Degree: A Metric for Model Performance Evaluation and Hyperparameter Tuning in Deep Learning-Based Side-Channel Analysis”. In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2025.2 (2025), pp. 333–361. DOI: 10.46586/TCHES.V2025.I2.333-361. URL: https://doi.org/10.46586/tches.v2025.i2.333-361.