THESIS TO OBTAIN THE DEGREE OF DOCTOR OF THE UNIVERSITY OF MONTPELLIER

In Information Security

Doctoral School: Information, Structure and Systems sciences (I2S)

Research Unit : LIRMM

Horizontal Side Channel Attacks on Noisy Traces

Presented by Gauthier CLER

On July 4, 2024

Under the supervision of:
Philippe MAURINE

In front of the jury composed of:

Christophe CLAVIER, Professor, Université de Limoges	Reviewer
Guénaël RENAULT, Associate Researcher, ANSSI, GRACE/INRIA	Reviewer
Eleonora CAGLI, Engineer, CEA LETI	Examiner
Lionel TORRES, Professor, Université Montpellier	Examiner, President of the Jury
Lionel AGULHON, Engineer, SERMA Safety & Security	Examiner
Philippe MAURINE, Associate Professor, Université Montpellier	Supervisor

UNIVERSITÉ DE MONTPELLIER

Acknowledgements

A travers ces quelques lignes, j'aimerais exprimer ma gratitude envers toutes les personnes qui m'ont accompagné et qui m'ont aidé de près ou de loin à la réalisation de ces travaux.

En premier lieu, j'aimerais remercier mon directeur de thèse Philippe Maurine, pour son encadrement tout au long de cette thèse. La pertinence de ses conseils, sa patience ainsi que sa bienveillance ont été cruciale à la réalisation de ces travaux.

Un grand merci également à Michaël Grand et Sébastien Ordas pour leur accompagnement au sein du laboratoire SERMA Safety & Security. Merci pour leur disponibilité ainsi que la pertinence de leurs retours.

Je suis également très reconnaissant envers Christophe Clavier et Guénaël Renault en acceptant d'évaluer cette thèse. Je remercie également Eleonora Cagli, Lionel Torres, et Lionel Agulhon pour avoir accepté de participer à ce jury. Un remerciement particulier à ce dernier pour m'avoir donné l'opportunité de réaliser cette thèse.

Je tiens aussi à exprimer ma gratitude envers toutes les personnes du CESTI, qui m'ont à un moment ou un autre aidé, que ce soit techniquement ou non. Je pense notamment à Pierre, Alexia, Corentin, Antoine, Adel, Gwenhael, Anatole, ...

De plus, je remercie les membres de la SERMA Trail Team : Alexis, Thibaud (a.k.a. le Tigre), Léo et Benoit, pour les sorties et courses, passées et à venir (et également de s'être dit que participer au Marathon du Montcalm était une bonne idée).

Un grand merci également à mes amis qui n'étaient pas présents au quotidien mais qui ont partagé à mes côtés ces trois années de thèse, notamment Valentin, Clément, Charles et Anas.

Je remercie enfin chaleureusement l'ensemble de ma famille, et en particulier mes parents, mon frère et ma grand-mère pour leur amour ainsi que leur soutien inconditionnel durant ces trois années et lors de toute la durée de mes études.

마지막으로 제가 사랑하는 한국인 예비신부와 그녀의 가족들에게도 감사를 표합니다. 멀리서도 늘 따뜻한 마음으로 응원해 주신 한국 가족들의 아낌없는 사랑과 격려는 저에게 큰 힘이 되어 잘 마무리할 수 있었습니다. 특히 장모님의 사랑이 듬뿍 담긴 김치와 한국 음식은 저에게 큰 힘이 되었습니다. 다시 한번 진심으로 감사합니다.

À mon grand-père

Contents

Contents v

Introduction 1

1 State of the Art 3
1.1 Security of embedded devices . . . 3
1.2 Asymmetric cryptography . . . 4
1.2.1 RSA cryptosystem . . . 5
1.2.2 Elliptic Curve Cryptography . . . 7
1.3 Side Channel Attacks . . . 8
1.3.1 Initial attacks . . . 9
1.3.2 Countermeasures . . . 11
1.3.3 Montgomery Ladder . . . 11
1.3.4 Correlation Attacks . . . 12
1.3.5 Profiled Attacks . . . 12
1.3.6 Towards Single Trace Attacks . . . 13
1.3.7 Horizontal Attacks . . . 13
1.3.7.1 Horizontal Collision Attacks . . . 13
1.3.7.2 Clustering based Attacks . . . 14
1.4 Conclusion . . . 17

2 Limits of Clustering Horizontal Attacks 19
2.1 Introduction . . . 19
2.2 Preliminaries . . . 20
2.2.1 Notations . . . 20
2.2.2 Considered targets . . . 20
2.2.2.1 MbedTLS . . . 20
2.2.2.2 Cswap pointer & arith . . . 20
2.3 Effectiveness of common horizontal attacks . . . 21
2.3.1 Univariate PoI selection . . . 22
2.3.1.1 Supervised selection . . . 22
2.3.1.2 Unsupervised selection . . . 23
2.3.2 Gaussian Mixture Model . . . 25
2.3.2.1 Linear separability . . . 26
2.3.2.2 Unsupervised T-test for PoI selection . . . 27
2.3.2.3 $\mathcal{T}$-value relevance for PoI selection . . . 29
2.3.2.4 Partial conclusion . . . 31
2.3.3 Impact of outliers and saturation on leakage . . . 32
2.3.3.1 Outlier detection . . . 32

2.3.3.2 Outlier or not . . . 32
2.3.3.3 Outliers in considered datasets . . . 33
2.3.3.4 Impact of outliers and saturated values on the clustering . . . 35
2.3.4 Compression based features extraction . . . 36
2.3.4.1 PCA as dimension reduction . . . 37
2.3.4.2 Quantify lost information . . . 37
2.3.4.3 Impact of noise . . . 38
2.3.4.4 PCA on considered datasets . . . 39
2.3.4.5 Conclusion and limits . . . 43
2.3.5 Deep Learning based bits correction . . . 43
2.4 Conclusion and Problematic . . . 44

3 Neural Network Approach **47**
3.1 Introduction . . . 47
3.2 Deep Learning . . . 48
3.2.1 Network topology . . . 48
3.2.2 Activation functions . . . 49
3.2.3 Optimization algorithm . . . 49
3.2.4 Learning paradigms . . . 50
3.3 Autoencoders . . . 51
3.3.1 Network architecture . . . 51
3.3.2 Autoencoder as an alternative to PCA . . . 51
3.3.2.1 Correlation between projected dimensions . . . 53
3.3.2.2 Data transformation and information loss . . . 54
3.3.2.3 Clustering and projections . . . 56
3.3.2.4 Alternative architectures . . . 58
3.3.2.5 Conclusion and limits . . . 59
3.3.2.6 Perspectives . . . 59
3.3.3 Robust Autoencoders for outliers mitigation . . . 61
3.3.3.1 Parametrization . . . 61
3.3.3.2 Impact on leakage . . . 62
3.3.3.3 Outliers mitigation capability . . . 64
3.3.3.4 Conclusion . . . 66
3.4 Generative Adversarial Networks . . . 68
3.4.1 GAN architecture . . . 68
3.4.2 CycleGAN . . . 69
3.4.3 Driven outliers mitigation . . . 70
3.4.3.1 Network Parametrization . . . 72
3.4.3.2 Outliers mitigation performance . . . 72
3.4.3.3 Impact on leakage . . . 73
3.4.3.4 Impact on distributions . . . 74
3.4.3.5 Updated univariate PoI identification . . . 75
3.4.4 Conclusion and Limits . . . 77
3.5 Self Organizing Maps . . . 78
3.5.1 Architecture definition . . . 78
3.5.2 Optimization procedure . . . 79
3.5.3 Alternative for horizontal attacks . . . 80
3.5.3.1 Application on high dimensional data . . . 80

3.5.3.2 Application after features extraction . . . 82
3.5.4 Conclusion and limits . . . 83
3.6 Global Conclusion . . . 84

4 Statistical Machine Learning Approach **85**
4.1 Introduction . . . 85
4.2 Univariate PoI Selection . . . 85
4.2.1 The ideal leakage model . . . 86
4.2.1.1 Gaussian mixture distribution . . . 86
4.2.1.2 Parameters estimation . . . 86
4.2.1.3 Limits of the T-test . . . 87
4.2.1.4 The probability of confusion . . . 88
4.2.2 Application for horizontal attacks . . . 90
4.2.2.1 Classes repartition . . . 90
4.2.2.2 Means difference . . . 91
4.2.2.3 Variance disparity . . . 91
4.2.2.4 Additional noise components . . . 92
4.2.2.5 Proposed PoI selection . . . 93
4.2.2.6 Neighbors correlation analysis . . . 94
4.2.2.7 Choosing the number of PoI for the attack . . . 96
4.2.2.8 Results . . . 97
4.2.3 Application on cleaned patterns . . . 100
4.2.3.1 Mixture analysis . . . 101
4.2.3.2 Using the same PoI . . . 102
4.2.4 Adaptive mixture analysis . . . 104
4.2.4.1 Model goodness of fit . . . 104
4.2.4.2 Lowest number of components . . . 105
4.2.4.3 Application on cleaned patterns . . . 107
4.2.4.4 Limits . . . 109
4.3 Conclusion . . . 109

5 Conclusion & Perspectives **111**
5.1 Contributions . . . 111
5.2 Perspectives . . . 111
5.2.1 Horizontal attacks . . . 112
5.2.2 Deep Learning . . . 112
5.3 Final Words . . . 112

A Notes **I**

B List of acronyms **III**

C Glossary **V**

List of Figures **VII**

List of Tables **XI**

Introduction

Side channel attacks on embedded devices, taking advantage of side channels such as power consumption or electromagnetic emanation constitute a real threat to the security and confidentiality of sensitive digital information. For instance, the way the implementation of cryptographic routines is made can sometimes be exploited. Using statistical tools, it is then possible to recover the secret information using one or several acquired traces. To protect sensitive content, chip manufacturers implemented countermeasures to prevent such attacks. Still, some attack vectors are still possible, often designed as horizontal attacks. These attacks can be applied even a single unique trace is available, thus making proposed countermeasures ineffective.

In this work, focus is made on the improvement of such horizontal attacks, and more precisely, clustering based horizontal attacks. While several contributions have been made to improve these attacks, their applicability often remains hard in practice. Indeed, in noise environments, especially with the presence of additional behaviors, saturation or outliers, proposed tools are not always adapted, and it becomes hard to successfully carry such attacks.

In parallel, the rise of neural networks in the domain of side channel attacks has showed promising results in the context of profiled attacks. While not directly applied to the case of unsupervised horizontal attacks, several neural architectures, able to learn with alternative objectives in a complete unsupervised way, could be seen as relevant candidate for improving the success rate of horizontal attacks. Several architectures are thus studied in this work to see if they can provide some benefits over traditionally used methods and be applicable in noisy environments.

Additionally, we propose an alternative approach for performing univariate Points of Interest (PoI) analysis based on statistical tools and probabilities. In fact, state of the art used methods often fails to apply in noisy context where additional behavior are observed. Furthermore, a novel metric for quantifying the exploitability of selected PoI is introduced as an alternative to currently use T-test statistic.

To this aim, this work is organized as follows in five different chapters.

The first chapter gives an overview of the current state of the art. It describes the context of embedded device security, asymmetric cryptography and their exploitation through side channel attacks. Then, the evolution of horizontal side channel attacks is then presented.

The second chapter studies the currently used clustering approaches for performing horizontal side channel attacks. The limits of dimensionality reduction methods for patterns compressions and the procedures for univariate PoI selection are emphasized.

In the third chapter, several neural network architectures are considered as candidates for improving the success of horizontal attacks. Firstly, autoencoder based networks are considered as an alternative for the Principal Component Analysis (PCA) projection. Then, other architectures are studied for improving the quality of the patterns, by mitigating out-

liers or saturation.

In the fourth chapter, a new framework for the selection of univariate PoI is proposed. It is shown that statistical analysis of the points distribution can be relevant for selecting more relevant PoI in the analysis process. Besides, a new metric for the quantification of the leakage quality of a PoI is introduced and compared to currently used metrics.

Finally, a global retrospective of the presented contributions of this work is done. Additionally, axes of improvements for future research are proposed.

Chapter 1

State of the Art

Abstract

In this first Chapter, context regarding the security of embedded devices and information is described. A brief overview of the asymmetric cryptography is then presented, followed by a presentation of the history and state of the art of side channel attacks. Focus is made on unsupervised attacks, especially clustering based horizontal attacks, being the main study of this work.

1.1 Security of embedded devices

In a world of growing technologies, the amount of information that transit and is manipulated by systems is abundant. Digital information can be found in various domains such as Banking, Telecommunications, Online services and more.

Information security (also known as InfoSec) is fundamental in the domain of digital systems to guarantee the protection of sensitive information. Several criteria define the prerequisite of Information Security and can be formulated as five main pillars:

- **Confidentiality**: The divulgation of information to unauthorized individuals must be prevented. This is ensured by securing information through encryption schemes, access controls and user permissions.
- **Integrity** Data integrity ensures the completeness and accuracy of information during its lifecycle. For example, hash functions are a way to assess data non alteration.
- **Availability** Information should be made available at any time when needed. Any applied processing has to be reversible. This also requires suitable and reliable hardware over time.
- **Authentication** Only the authorized party must be able to access addressed information. Identity of the sender or receiver shall be established in communications. Many authentication methods exist for different needs (passwords, certificates, biometrics...).
- **Non-repudiation** One has to make sure that authorship of content is not disputable. This is often synonym of digital signature systems to authenticate authors of documents or systems.

The security of embedded systems is a primary concern for chip and hardware manufacturers. Undoubtedly, all provided need to comply with presented five pillars rules to guarantee information security. In highly sensitive contexts, such as banking (credits cards)

or digital identification documents (passports, ID cards, health insurance cards...), robust routines should be implemented to make sure of the non divulgation of sensitive information.

While the security is systematically based on proven resistant cryptographic schemes, their embedding in electronic devices gives rise to additional problems that can often be a source of possible exploitation. These additional problems typically take the form of a wide variety of attacks, exploiting hardware weaknesses to recover sensitive information.

For example, fault injection attacks consists in inducing physical perturbations (using clock glitch, laser, electromagnetic, ...) on a circuit in a way to extract secret information, bypass a security mechanism or dump source code. A non-invasive alternative known as Side Channel Attacks (SCA) involves monitoring channels like the device power consumption or its electromagnetic emanation to extract secret data. Explanation on SCA is detailed in Section 1.3.

1.2 Asymmetric cryptography

Asymmetric cryptography (also known as Public-key cryptography) is a field of cryptography introduced by Diffie and Hellman in 1976 [DH76]. It is used for providing secure communications through the usage of digital keys pair, composed of a public key k_{pub} and its associated private key k_{priv}. The public key is used for information encryption and the private key for decryption. The principle of asymmetric cryptography is often illustrated by taking in example two individuals A and B wishing to communicate securely, with their respective key pairs $k_A = (A_{pub}, A_{priv})$ and $k_B = (B_{pub}, B_{priv})$. These key pairs are generally generated by choosing a large randomized number. Asymmetric cryptography is used in several varied applications:

- Encryption/Decryption: Anyone in possession of the public key is able to encrypt messages, but only the owner the private key can decrypt them. This ensures a secure communication between pairs.

- Key exchange (Diffie-Hellman): This is usually used to compute shared secrets that can be used, for instance, as a symmetric cipher key.

- Signature/Verification: Instead of encrypting messages, this is used to sign them. This allows the message recipient to verify (using the key) that it comes from the expected sender.

Figure 1.1 illustrates the three aforementioned applications.

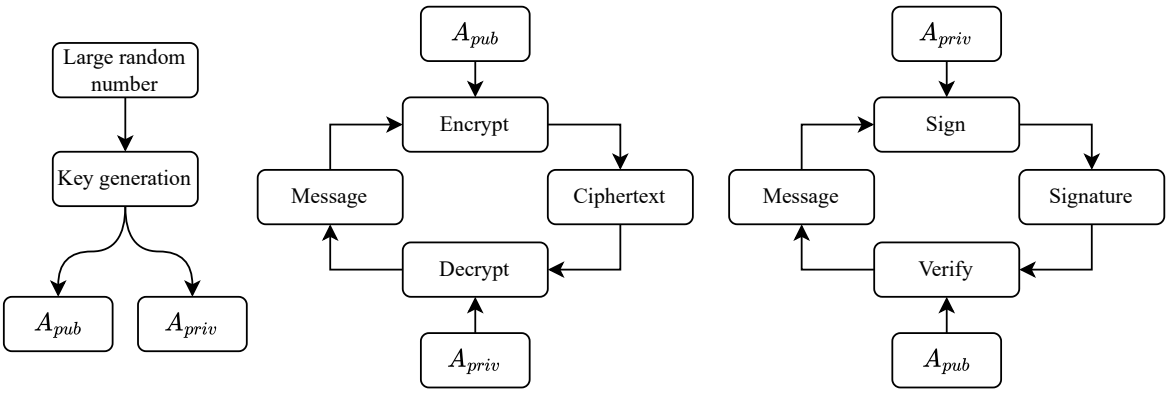

Figure 1.1: Illustration of key generation (left) encryption/decryption (middle) and signature process (right) in asymmetric cryptography.

The security and reliability of asymmetric cryptography is based on trapdoor one-way functions. It relies on the fact that some arbitrary functions, f can be efficiently computed in polynomial time. But their inverse f^{-1} is hard to compute unless the trapdoor is provided. This allows to distribute the public key k_{pub} to anyone without any risk that k_{priv} can be guessed. This is not possible for example in the case of symmetric cryptography (say Advanced Encryption Standard (AES) for example), where the divulgation of the encryption key to an unwanted party would compromise the security of the communication.

Today, the two main public key algorithms widely used for cryptographic tasks are the RSA cryptosystem and Elliptic Curves based cryptography. They are recalled in the next sections.

1.2.1 RSA cryptosystem

Introduced in 1978, Rivest-Shamir-Adleman (RSA) is the most used public key algorithm [RSA78]. Its hardness relies on the Integer Factorization problem, that is, the difficulty in factoring large integers. Given a base b, an exponent e and a modulus n, the modular exponentiation is given by:

$$c = b^e \mod n \tag{1.1}$$

with $0 \leq c < n$. Indeed, modular exponentiation is easy to compute, even on very large numbers, but the inverse is not. Knowing b, c and n, it is hard to find e. To this day, no efficient method (excluding quantum computing [Sho97]) has been found to factor integers in polynomial time. In RSA, modular exponentiation is used for the key generation, message encryption and decryption.

Key generation In order to generate the pair (k_{pub}, k_{priv}), two secret prime numbers p and q, and small e are chosen such that:

$$\gcd(e, \phi(n)) = 1 \tag{1.2}$$

where $n = pq$ is the modulus and $\phi(n) = (p-1)(q-1)$ is the Euler totient. The private exponent d can be deduced by computing the modular inverse of $e \mod \phi(n)$ such that:

$$ed = 1 \mod \phi(n) \tag{1.3}$$

This can be done by applying Extended Euclidean algorithm. Public key $k_{pub} = (n, e)$ can be transmitted and used by anybody to encrypt messages but only the owner of the private key $k_{priv} = d$, kept secret would be able to decrypt messages.

Encryption The encryption of a message is straightforward and can be achieved by computing the ciphertext c:

$$c = m^e \bmod n \tag{1.4}$$

using public key (n, e) and plaintext m satisfying $0 \leq m < N$. This ciphertext c is then transmitted to the other party. Without knowing the private key k_{priv}, it is impossible to recover m.

Decryption The receiver in possession of the private key k_{priv}, can apply the same operation to retrieve the plaintext from the ciphertext c

$$m = c^d \bmod n \tag{1.5}$$

While the modular exponentiation is easy and fast to compute on relatively small length integers, it can become ineffective on long integers, necessary to ensure high security levels. For instance a minimum of 1024, 2048 or 4096 bits for the modulus size is often preconized for security standard. Instead, the modular exponentiation of c^d can be represented as a series of multiplications of c:

$$m \equiv \prod_{i=1}^{d} c \mod n \tag{1.6}$$

Following this fashion, implementations have been proposed for improving the performance and efficiency of modular exponentiation. In practical implementations, especially on embedded devices, binary exponentiation algorithm is often considered. This consists in converting e or d to binary notation $d = \sum_{i=0}^{\ell-1} = b_i 2^i$ and process each bit individually. Depending on the bit value, one or several operations are applied. This is known as the Square and Multiply exponentiation method, which procedure is described on Algorithm 1. Left to right binary exponentiation implementation treats bits sequentially starting from the Most Significant Bit (MSB) to the Least Significant Bit (LSB), while the right to left implementation does the opposite.

Algorithm 1 Left to right binary exponentiation

Require: $c, d = (d_{\ell-1}, \ldots, d_0)_2, n$
Ensure: $m = c^d \mod n$
  $m \leftarrow 1$
  for $i = \ell - 1$ to 0 **do**
    $m \leftarrow m^2 \mod n$ ▷ Squaring operation
    if $d_i = 1$ **then**
      $m \leftarrow m \times c \mod n$ ▷ Multiply operation
    end if
  end for
  return m

As embedded devices are subject to constraints such as limited computational power or energy saving as well as optimized processing time, implementations variants such as the Chinese Remainder Theorem (CRT) are often used for faster private key operations,

considered as the bottleneck since d is large. It allows to divide the modular exponentiation into two smaller sub-operations, resulting in a speed up factor of up to 4 times. The private exponent d is instead split into d_p and d_q.

$$\begin{aligned} d_p &= d \bmod p - 1 \\ d_q &= d \bmod q - 1 \\ q_{\text{inv}} &= q^{-1} \bmod p \end{aligned} \tag{1.7}$$

Therefore, modular exponentiation is performed by first producing m_p and m_q:

$$\begin{aligned} m_p &= c^{d_p} \bmod p \\ m_q &= c^{d_q} \bmod q \end{aligned} \tag{1.8}$$

Then, calculate h using the CRT and combine results to obtain m:

$$\begin{aligned} h &= q_{\text{inv}}(m_p - m_q) \bmod p \\ m &= m_p + hq \bmod pq \end{aligned} \tag{1.9}$$

As said before, the security hardness of RSA relies on the Integer Factorization Problem and thus can't be exploited. Still, it will be studied in the next sections, how an attacker can take advantage of specific implementations of modular exponentiation (or scalar multiplication) to recover d in auxiliary ways.

1.2.2 Elliptic Curve Cryptography

Elliptic Curve Cryptography (ECC) denotes the usage of elliptic curves for public cryptography. Compared to RSA, it allows smaller key for equivalent security levels. An elliptic curve E over finite field $\mathbb{F}_p$ (where p is a prime number) is given by:

$$y^2 \equiv x^3 + ax + b \pmod{p} \tag{1.10}$$

where $a, b \in \mathbb{F}_p$ and $p \notin \{2, 3\}$. Its security relies on the Elliptic Curve Discrete Logarithm Problem (ECDLP). Given two points P and Q in $E(\mathbb{F}_p)$, it is easy to compute $Q = kP$, but it is hard to find back k by knowing only P and Q. Elliptic curves are widely used in cryptosystems such as keys generations or digital signatures.

In an analogous way with RSA cryptosystem, Elliptic curve keys generation is defined over domains parameters (p, a, b, G, n, H) where prime p is the size of the finite field, a and b are coefficients of the curve, G is the base point to generate subgroup, and n and h are order and respectively cofactor of the subgroup.

Private key k_{priv} is formed by the scalar d, chosen by generating a random integer in $\{1, \ldots, n-1\}$. Associated public key k_{pub} is given by $H = dG$. If d and G are known, it is straightforward to find H. But the opposite is hard, if H and G are known, it is not possible to recover d because of the Discrete Logarithm Problem.

In the same way as before, scalar multiplication can be computed effectively by using d binary representation $d = (d_0, \ldots, d_{\ell-1})_2$ with l the bit length of d. This is known as Double and Add scalar multiplication implementation (see Algorithm 2).

Algorithm 2 Left to right Double and Add scalar multiplication

Require: $P, d = (d_{\ell-1}, \ldots, d_0)_2, p$
Ensure: $Q = dP \mod p$
  $Q \leftarrow P$
  for $i = \ell - 1$ to 0 **do**
    $Q \leftarrow 2Q \mod p$ ▷ Double operation
    if $d_i = 1$ **then**
      $Q \leftarrow Q + P \mod p$ ▷ Add operation
    end if
  end for
  return Q

Overall, ECC based implementations are sensitive to the same attack vectors as RSA if binary scalar multiplication are targeted, since it follows the same processing as the binary modular exponentiation. In the next section, we present how Algorithms 1 and 2 or their derivates can be exploited using techniques that take advantage of side channels on which an attacker can listen to deduce the binary values of d.

1.3 Side Channel Attacks

SCA is a non-invasive (or semi invasive in some cases) family of attacks that can be carried out on embedded devices. It aims to recover sensitive information manipulated by the targeted devices by monitoring and performing analysis on some side channels such as their power consumption, their electromagnetic emanations, their execution time, etc. Several types of SCA exist, depending on the level of access to the targeted device family. For instance, carried attacks are said to be supervised when it is possible to perform a profiling phase on a clone device, in a way to build a leakage model. This model is then applied on the specific target device to extract the enclosed secret. On the other side, when it is not possible to apply profiling beforehand, thus attacking directly the target device, SCA are known as unsupervised. In this case, the attacks are performed without having insights about the underlying leakage model. Even in this case, different types of attacks apply or not depending on some knowledge the adversary has about the targeted implementations, on the access to plaintext that can be manipulated, different kind of countermeasures, etc. Figure 1.2 shows an example of side channel scenario on an embedded device.

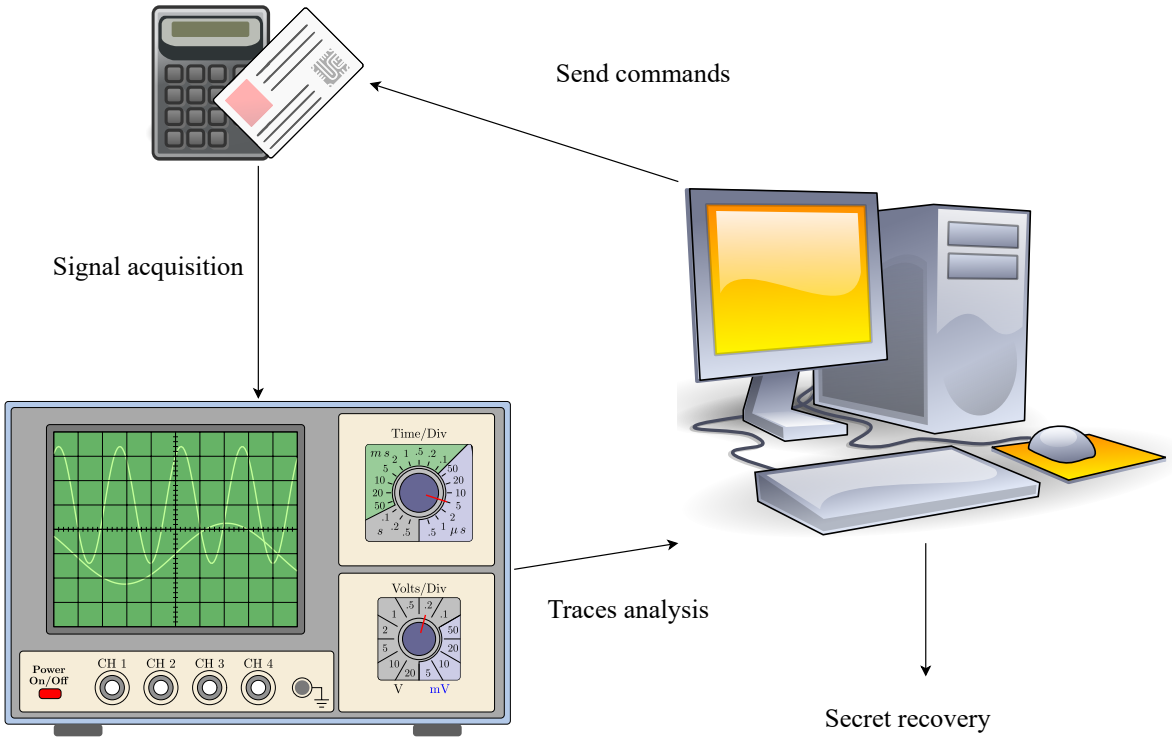

Figure 1.2: Simplified schema of a side channel analysis.

1.3.1 Initial attacks

In 1996, the first SCA was introduced by Kocher. It is known as the Simple Power Analysis (SPA) [Koc96]. By visual inspection of the power consumption during the computation of the modular exponentiation, it is possible to deduce the value of each manipulated bit based on its pattern length. Indeed, according to Algorithm 1, if the assessed bit $d_i = 1$, a Multiply operation is added, resulting in a longer pattern (and possibly a difference in current consumption if a different function is used for the multiplication computation). This different behavior can be exploited through a timing analysis, which simply consists of discriminating patterns based on their length to recover private exponent d. Figure 1.3 illustrates an example of SPA by visual inspection on a simulated power trace.

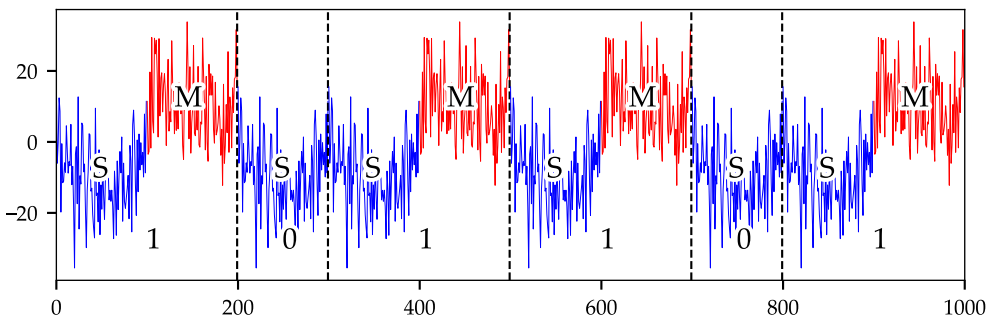

Figure 1.3: SPA by visual inspection on Square and Multiply binary exponentiation (Algorithm 1). Recovered private exponent is $d = 101101$.

This attack can be applied to any algorithm following a non-regular modular exponentiation implementation, even those based on CRT. Still, a simple countermeasure has been

proposed to mitigate the applicability of such an attack. It is indeed possible to remove the conditional branching from the Square and Multiply (SM) exponentiation (or Double and Add (DA) multiplication) by adding a supplementary dummy operation if $d_i = 0$ [Koc96]. This results in regular time processing of each bit, making the SPA inapplicable, at the cost of a non-negligible impact on performance due to dummy multiplications. This implementation variant is widely known as the Square and Multiply Always exponentiation (or Double and Add Always multiplication for ECC). Algorithm 3 describes the implementation.

Algorithm 3 Left to right Square Multiply Always

Require: $c, d = (d_{\ell-1}, \ldots, d_0)_2, n$
Ensure: $m = c^d \mod n$
  $m \leftarrow 1$
  for $i = \ell - 1$ to 0 **do**
    $R_0 \leftarrow m^2 \mod n$ ▷ Squaring operation
    $R_1 \leftarrow R_0 \times c \mod n$ ▷ Multiply operation
    $m = R_{d_i}$
  end for
  return m

The next year, Kocher *et al.* introduced the Differential Power Analysis (DPA) [KJJ99], an improvement over SPA using statistical analysis. The attacker makes guesses on the manipulated bit for every pattern, then the difference of averaged traces of the two sets (a bit of the manipulated message or output value for example) is performed. If the resulting curve differs in some point, it emphasizes a difference thus the presence of leakage. This approach feasibility relies on the fact that some values of the targeted algorithm intermediate are observable and can be used to make correlation with observed signal. The result on a DPA example on simulated data can be observed on Figure 1.4.

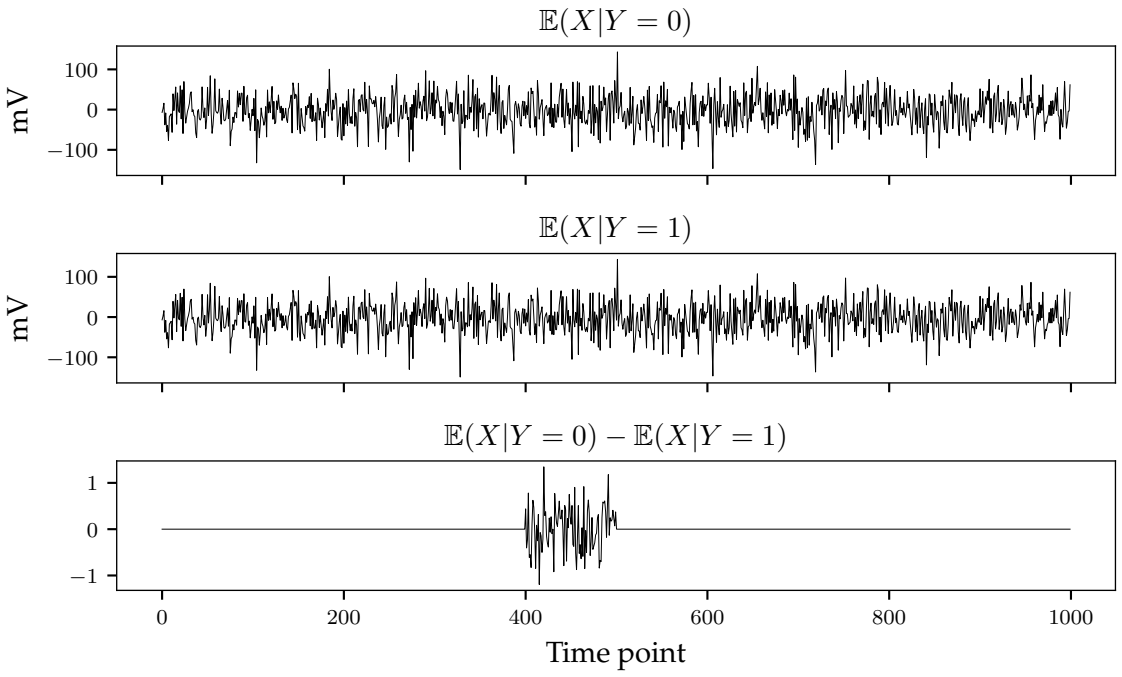

Figure 1.4: Illustration of the DPA. Upper and middle plots show averaged traces with $Y = 0$ and $Y = 1$ respectively. Lower plot shows the difference.

1.3.2 Countermeasures

The same year, Coron proposed several countermeasures to defeat DPA [Cor99] applied to scalar multiplication. In that context, let us remind the attacker tries to recover d in the case where $Q_i = dP_i$ with various values of known P_i.

The first countermeasure consists in randomization of the private scalar. Generate a random number k and define $d' = d + k.\#E$ to be used as a substitute, where $\#E$ denotes curve number of points and k is a random scalar. Instead, result of the multiplication is given by $Q = d'P$. Since d' is different at each new execution, DPA cannot be applied.

The second proposed countermeasure consists in blinding the point P. Instead of computing $Q = dP$, scalar multiplication is done by multiplying $P' = P + R$ such that $Q' = dP'$, where R is a random point. Then, Q is obtained through subtraction $Q = Q' - dR$. Because of this, an attacker cannot use P anymore as a leverage to recover d.

Eventually, the third countermeasure considered the randomized projective coordinates. When performing point addition or doubling, projective coordinates can be used to speedup computation when inversion cost is higher than field multiplications. Projective coordinates of an affine point $P = (x, y)$ are given by the triplet (X, Y, Z) such that:

$$x = \frac{X}{Z} \quad y = \frac{Y}{Z} \tag{1.11}$$

Coron proposed to randomize the projective coordinates of P for each scalar multiplication execution. This can be done by multiplying each coordinate with random λ such that $(X', Y', Z') = (\lambda X, \lambda Y, \lambda Z)$. Thus, P is no longer exploitable in projective coordinates representation.

1.3.3 Montgomery Ladder

At the same period, alternative modular exponentiation or scalar multiplication implementations were proposed. Initially introduced by Montgomery [Mon87] in a way to improve performance of scalar multiplication in Elliptic curves, Montgomery Ladder has been applied by authors of [JY03] as a countermeasure to prevent attacks such as [Wal01] or timing attacks [Koc96]. The modular exponentiation implementation becomes now fully regular and has the same number of operations as a Square and Multiply Always implementation. Algorithm 4 describes the exponentiation implementation using Montgomery Ladder.

Algorithm 4 Left to Right Montgomery Ladder

Require: c, $d = (d_{\ell-1}, \ldots, d_0)_2$, n
Ensure: $m = c^d \mod n$
  $R_0 = 1$
  $R_1 = c$
  for $i = \ell - 1$ to 0 **do**
    if $d_i = 0$ **then**
      $R_1 = R_0 \times R_1 \mod n$
      $R_0 = R_0 \times R_0 \mod n$
    else
      $R_0 = R_0 \times R_1 \mod n$
      $R_1 = R_1 \times R_1 \mod n$
    end if
  end for
  return R_0

1.3.4 Correlation Attacks

Despite the fact that Coron's countermeasures render multiple traces exploitation inapplicable in some contexts, several attacks have still been proposed, often applied to symmetric ciphers.

In 2004, Brier *et al.* introduced the Correlation Power Analysis (CPA) [BCO04]. In this attack, a Hamming distance leakage model is considered, where the power consumption can be represented as:

$$W = H(D \oplus R) + b \tag{1.12}$$

with D the manipulated binary word, R its reference state, and b the additional noise of the power consumption. A linear relationship is assumed between the consumption and $H(D \oplus R)$. This can then be used for secret inference by analyzing the correlation between the power consumption and candidates $H(D \oplus R)$. In Brier *et al.* work, the Pearson's correlation factor is used. The candidate with the highest correlation value represents the most probable guess for the manipulated value. This attack has been and is still widely used for performing non profiled SCA.

1.3.5 Profiled Attacks

From 2003 with Chari *et al.* work [CRR03], SCA domain have seen the raise of new types of attacks, known as profiled side channel attacks. These attacks assume the availability to an open device, a perfect copy of the target device, on which the attacker has partial (or full in best conditions) control to better characterize the leakage and mitigate noise. During the profiling phase, the attacker collects several traces for a variety of different keys/messages to create an as exhaustive as possible model. Then, during the matching phase, learned model is applied on the target device (with fixed unknown key for example) to recover the sensitive information. However, attention should be paid to practical details. Indeed, in order for profiled attacks to work efficiently, the same hardware setups and operating conditions have to be used on both targets during traces acquisition phases.

In Chari *et al.* Template Attacks (TA) [CRR03], this consists in creating a template for each possible values of unknown key bits. This is done by first acquiring a large number of samples per possible key value, in order to build averaged trace for every possible value. Then, points of interests are identified by selecting samples points with highest pairwise differences among all means. This step, commonly designed as leakage assessment procedure to characterize relevant points in the obtained trace, has been improved by using other metrics in works such as [BDGN14, GJJR, BCD+] or [GBTP08, CLM20]. Then, one can compute the noise covariance matrix using the identified PoI, to get a template for each value. Eventually, during the attack phase on the target device, the attacker compares acquired trace to each template, and the one with the highest probability should indicate the secret key.

A more recent alternative for profiled side channel attacks is Deep Learning Side Channel Attacks (DLSCA). Instead of building covariance matrix templates, it is possible to train a neural network to achieve the same task. During the profiling phase (or training phase), the network learns to associate each trace with its key value using a forward/backward propagation optimization algorithm. Then during the matching phase (or prediction phase), acquisitions from the target device are presented to the trained network which produces a prediction probability on the key value. Knowing the pair input trace/key value as a training set, this can be seen as a neural network classification problem. To build such neural architecture, Multilayer Perceptrons (MLP) or Convolutional layers are commonly used.

While neural network based methods display competitive results (or even superior) than Template Attacks, it is important to note nevertheless, that (unlike TA), DLSCA is highly parametric, and its performance relies on optimal parameters tuning, which is often necessary where leakage is low or attack complexity high (in the presence of masking, jitter, or noise for example).

1.3.6 Towards Single Trace Attacks

Because on the complexity of some devices, but also on the variety configurations and possible contexts, one could almost associate an attack vector for each context. In worst case scenario, where performing a profiling is not possible because of the deployment of countermeasures such as blinding or masking [Cor99], some attacks are still applicable other than the SPA (that often fails to apply on regular implementations). Traditionally, these types of attacks are called Horizontal, Single trace or One Shot attacks. Often applied to asymmetric implementations, they try to recover private exponent or scalar using a unique exponentiation or scalar multiplication trace. They are described in more details in the following section.

1.3.7 Horizontal Attacks

Following the proposal of the aforementioned countermeasures, in particular the message blinding and the exponent/scalar randomization, all previously existing attacks turned to be inefficient. This constraint ultimately reduces the scope of possible actions for horizontal attacks. As a result, the only remaining way to exploit the leakage consists in extracting the secret from a single exponentiation or scalar multiplication trace.

Horizontal single trace attacks can be divided in two distinct families of attacks. The first one uses collision between trace patterns while the second one involves clustering algorithms to discriminate operations.

1.3.7.1 Horizontal Collision Attacks

In 2001, Walter introduced the first collision attack in the context of SCA, known as the Big Mac collision attack [Wal01]. This attack tackles Long-Integer Multiplication (LIM) implementations. Instead of considering as a pattern the part of trace associated with the processing of each bit d_i, it can be subdivided into smaller patterns, each one representing an atomic segment of the single-precision multiplication. A template can be built using precomputation or first exponentiation patterns and the Euclidean distance between the template and each long-integer multiplication pattern calculated. If the distance exceed a defined threshold, the operation is a squaring and a multiplication otherwise. Initially applied to Sliding window implementation [Koc95], the Big Mac attack also works on the classical Square and Multiply exponentiation, as long as the algorithm implementation is not regular. It should be noticed, that its efficiency significantly depends on the ability to identify, cut and align inner multiplications.

In 2010, Clavier *et al.* proposed the Horizontal Correlation Attack (HCA) [CFG+10]. It assumes the knowledge of the input message m. Instead of using the Euclidean distance as in Big Mac attack [Wal01], the Pearson correlation coefficient is employed. This attack can be applied on any regular exponentiation methods like Square and Multiply Always or Montgomery Ladder implementation. While this attack is effective against exponent blinding countermeasure, it can be prevented by message randomization.

The ROSETTA from Clavier *et al.* [CFG+12] is an improvement over the Horizontal Correlation Attack [CFG+10] proposed in 2012. No knowledge of m nor any manipulated data is needed. It can thus be applied even in the presence of blinded message or exponent. To distinguish Squaring from Multiplications, they perform a Triangular trace analysis. This consists in representing the LIM in base $b = 2^t$ to build the inner multiplication matrix $\mathbf{M}$. Then, upper and lower triangles of $\mathbf{M}$ are compared, if they match, the operation is a Square, a Multiply otherwise. To that aim, two distinguishers are considered, either the Euclidean distance or the Pearson correlation coefficient.

The same year, Witteman *et al.* proposed the Cross Correlation attack [WvWM11]. Their method tries to discriminate dummy multiply from real multiply in Square and Multiply Always implementation. This is done by considering that a square operation performed after a dummy multiplication will use one of the operands from that dummy multiplication, and this can be used to reveal hidden zero bits if their correlation is high, using Pearson correlation factor. However, the downside of this approach lies in the fact that several traces are needed, and thus is not applicable in the presence of exponent blinding. Nevertheless, this attack is generalized by Hanley *et al.* in 2015 [HKT15] to be applicable using a single trace on Montgomery Ladder implementations.

In 2014, authors from [BJPW14] combine [Wal01] with correlation applied on several ECC atomic implementations with input randomization. This allows to dismiss several elliptic curves implementation countermeasures.

Overall, while the collision attacks are still applicable and relevant to this day, they require a low level of noise to be efficient and are often implementation dependent.

1.3.7.2 Clustering based Attacks

Clustering algorithms are another type of distinguisher that can be used for unsupervised horizontal attacks. They can be seen as an alternative to collision/correlation based attacks.

In 2013, Heyszl *et al.* proposed in [HIM+13] to use the k-means multidimensional clustering algorithm [Llo82] as an unsupervised distinguisher for attacking an FPGA ECC implementation. They consider that in a two classes problem, all measurements are affected by the same normal noise. Therefore, samples of classes c_i such that $i \in \{a, b\}$ should be distributed around means μ_a and μ_b. This model can be exploited using a clustering algorithm to discriminate c_a from c_b based on their difference of means on all temporal samples. Still, they emphasize that the exploitation capability is highly dependent of the EM probe position that is to say on the leakage level. Furthermore, this is applicable only in cases where the number of points per pattern is reasonably low. In a clustering context, as the number of dimensions p increases, a given distance metric d turns to be less precise and thus the discrimination of extreme points becomes less meaningful:

$$\lim_{p \to \infty} \frac{d_{max} - d_{min}}{d_{min}} = 0 \tag{1.13}$$

In [SHKS15], authors decided to consider three mentioned improvements suggested in [HIM+13]. The first one consists in using Expectation Maximization (EM) [DLR77] instead of k-means clustering. They show that both approaches lead to almost same results. On the other hand, they used the Principal Component Analysis (PCA) [Jol02] for patterns dimensionality reduction instead of applying clustering on the raw patterns. As stated, only a small proportion of acquired patterns is believed to contain exploitable leakage information. Still, PCA resulting principal components do not necessarily convey leakage information

since the data variation is used as a score function. Indeed, as observed in [SHKS15] work as well as other applications of PCA in profiled SCA [BNSQ03, APSQ06, BHvW12, MBvLM12, SA08], it is often necessary to discard the few first components in the presence of high levels of noise, depending on the attacked target and signal quality. Eventually, they make use of multiple EM probes, but no improvement was noticed compared to the results obtained with a single probe.

An alternative approach has been considered in [PITM14]. Instead of trying to exploit multivariate leakage, where high levels of noise can affect the analysis, authors perform univariate analysis to select most relevant points, i.e. PoI. According to their approach, a set of patterns is considered as a matrix $\mathbf{D}$ of size $m \times p$ with m the number of patterns and p the number of samples per pattern. The number of points p should be equal for every pattern to be able to build $\mathbf{D}$. The proposed selection methodology can then be divided into the following steps:

This first step consists in applying univariate clustering (1 dimensional) to every temporal sample (every column) of $\mathbf{D}$. This yields two clusters c_1 and c_2, with associated empirical means μ_1, μ_2 and variances σ_1^2, σ_2^2.

The means are then used during the second step in order to compute the Difference of Means (DoM) $|\mu_1 - \mu_2|$ between identified cluster on each dimension. It allows to quantify the separability of the classes on each temporal point. One can then select the u points with highest DoM as the most relevant PoI. This approach can be seen as an attempt to perform leakage assessment in an unsupervised context, whereas compared to supervised one, true labels are unknown and thus estimated through points difference of means. The number of chosen PoI u is done by visual inspection.

Each of the u selected PoI u are then re-classified in the third step using the Fuzzy k-means algorithm. Unlike the k-means which performs hard assignations $P(c_i|x_j) \in \{0, 1\}$, this is relaxed by performing soft assignments such that $P(c_i|x_j) \in [0, 1]$ where $\sum_{i=1}^{k} P(c_i|x_j) = 1$. This process yields new estimates of classes as a matrix $\mathbf{L}$ of size $m \times u \times 2$ where $\ell_{i,j,k} \in [0, 1]$.

In the last step, author proposed three different methods for performing the exponent recovery. The first one consists in performing a majority voting on the obtained classes of each sample across the u PoI. The predicted bit value for the ith pattern is given by equation 1.14.

$$d_i = \begin{cases} 1, & \text{if } \left(\sum_{j=1}^{u} \operatorname{argmax}_k \ell_{i,j,:}\right) \geq \frac{u}{2} \\ 0, & \text{otherwise} \end{cases} \tag{1.14}$$

The second approach for exponent recovery is based on the Probability Density Function (p.d.f). Assuming that c_1 and c_2 samples points follow normal distributions, the likelihood (joint probability) of each sample $P(x_j, c_i)$ (for each PoI) is computed by plugging estimated classes and variances from Fuzzy clustering as normal distributions parameters (see [PITM14] for details). Then, probabilities are averaged into S and exponent $d_i = 1$ if $S \geq 0.5$, otherwise $d_i = 0$.

The last approach is an extension of the p.d.f approach, using a Bayesian classifier. The posterior probability is $P(c_i|x_j)$, computed from the prior $P(c_i)$ and joint probabilities for each PoI using Bayes formula.

Overall, minimal differences are observed in the resulting bit recovery rate among considered exponent recovery methods on the considered target. Table 1.1 summarizes the described method from [PITM14] work.

In 2016, Perin and Chmielewski improved previously introduced method and provide

1. On each temporal point $\forall i \in \{0, n-1\}$:
1.1. Apply clustering s.t. $\mu_1, \mu_2 = k$-means$(\mathcal{D}_{:,i})$
1.2. Compute Difference of Means DoM $= \lvert\mu_1 - \mu_2\rvert$.
2. On first u PoI with highest DoM $\forall i \in \{0, u-1\}$:
2.1. Apply Fuzzy clustering $\ell =$ Fuzzy$(\mathcal{D}_{:,i})$.
3. Recover exponent from predicted labels ℓ:
3.a. Majority Voting on the u predictions.
3.b. p.d.f from cluster results.
3.c. Bayesian classifier.

Table 1.1: Summary of [PITM14] methodology.

extended results by comparing several clustering algorithms, leakage assessment methods and statistical classifiers [PC16]. They also demonstrate that the analysis can be carried out on more than one trace even in presence of blinding countermeasures. Indeed, assuming the global underlying leakage model is defined as mixture of Gaussian distributions, the order of modular operations in each exponentiation is irrelevant. Furthermore, they also introduced a ratio for the leakage assessment metric, to quantify how observed repartition deviate from the expected one. Indeed, unbalanced classes can be often obtained depending on the data distribution, in previous work from Perin *et al.* [PITM14], since clustering assessment has no prior on classes repartition but aims only at optimizing clusters compactness. In the end, results show that the Fuzzy k-means clustering, associated with Sum Of Squared $\mathcal{T}$-values (SOST) metric and Log likelihood classifier yield slightly better bit recovery rate, but overall, all combinations show similar results.

The next year, in [NC17], Nascimento and Chmielewski applied [PC16] framework in protected ECC implementations. They proposed several improvements over previous framework. First, they used multiple dimensions for the clustering step. In addition, they applied the Davies-Bouldin index clustering quality metric [DB79] to measure the compactness of resulting clusters. As clustering parameters are usually initialized randomly, several runs can lead to different clusters outputs. DB index is therefore used to find the initialization giving the most compact clusters. Finally, they pointed out possible issues and bias encountered in the presence of outliers values. They identify said outliers on each temporal feature using the IQR method and replace them with median value.

In 2020, Perin *et al.* proposed a bit error correction framework based on neural networks [PCBP20]. They stated that given a base prediction a network is able to generalize over those predictions so that to correct some prediction errors and thus to improve the bit recovery rate. They also show that the use of data augmentation and regularization techniques has a positive impact on classification accuracy.

The method they proposed consists in two distinct parts. First, it should be observed that the core of the method remains similar to that described in [PITM14], but using improvement proposed in [PC16] and [NC17]. Indeed, it starts by labeling all patterns using the enhanced method proposed in [PITM14]. Then, using obtained (estimated) labels, they performed a supervised learning by splitting their dataset in sub-sets. The first one is used for the learning phase while the second one for prediction phase. They then reassigned labels from prediction labels. This process is repeated several times, after a reshuffling of the dataset at each time. While in itself, this process of learning is supervised, the framework remains unsupervised as no knowledge about labels is needed to carry out the optimization step. The authors show that using this approach, from a base bit recovery rate of about 52%

on two targets after applying methodology described in [PITM14, PC16, NC17] methodology, they were able to iteratively improve the success rate towards 100% correctly identified bits values. Still, in order for this bit correction method to work, initially selected PoI should carry a sufficient amount of leakage information to be exploitable. Otherwise, the network would only generalize over noise. Furthermore, as the architecture used is large and, significant hyper-parameters tuning is needed, which can be view as incompatible with the context of unsupervised attack. Furthermore, in 2024, Boussam and Calleja [BA24] showed that the proposed architecture from Perin *et al.* 2020 work [PCBP20] can be reduced to a smaller and simpler architecture built from a single perceptron, assuming correct initial selection of PoI is done. Additionally, an alternative approach for leakage extraction is proposed through the analysis of the PCA projection components, based on Cagli *et al.* work [CDP16].

1.4 Conclusion

This chapter presented an overview of the problematic of information security in embedded devices.

We briefly described how asymmetric cryptography is built and on what it relies on to ensure the security of communications.

We gave an overview of side channel attacks history and possible attack scenarios, from best cases scenarios, allowing the attacker to carry out profiling on open devices, to most constrained contexts in the presence of countermeasures, leading to the study of horizontal attacks.

As explained, the presence of countermeasures such as blinding or exponent/scalar randomization cause traditional attacks using several traces to become inapplicable. Horizontal attacks become therefore the only possible attack vector.

This work focus on clustering based PoI selection and attacks, assumed to be more applicable in high noise environments. Still, the applicability of these methods can often fail or not be applicable to more complex targets.

In the next chapter, an extensive study is carried out on several targets to assess the relevance of these methods in different complexity and noise level contexts.

Chapter 2

Limits of Clustering Horizontal Attacks

Abstract

This chapter provides a status of horizontal attacks in a real life environment. It is shown that state of the art proposed methods often fail to apply to noisy environment. Theoretical and experimental studies on real targets are performed to emphasize the limits of such methods. Finally, the impact of outliers and saturation on leakage and its exploitation is demonstrated.

2.1 Introduction

According to the state of the art, several approaches and methodologies have been proposed to perform horizontal side channel attacks. In this work, we pay attention to two prevalent approaches for selecting relevant leakage features from acquired traces, namely univariate features selection, from Perin *et al.*'s work and improvements [PITM14, PC16, NC17, PCBP20] as well features compression using dimensionality reduction approach [SHKS15]. The rest of this chapter is divided in three main parts.

The first part introduces notations that will be used throughout this thesis. It also describes the considered hardware targets and their implementations. Three datasets with different behavior, leakage and noise levels are chosen, in a goal to get a broader landscape of the performance of studied methods on different test cases.

The second part examines two approaches to perform Points of Interest (PoI) selection for horizontal attacks: a univariate selection method and a multivariate compression technique based on feature extraction. The study includes experimental analyses on both simulated and real traces obtained from actual measurements on several targets.

The final section summarizes the results of the studied cases and highlights the limitations of the currently most used methods. This aids in identifying the perspectives and areas for improvement for the following chapters.

The results and conclusions of this chapter will serve as a reference and performance baseline to demonstrate the validity of the proposed contributions in Chapter 3 and 4 of this thesis.

2.2 Preliminaries

2.2.1 Notations

Random variables will be denoted as uppercase letters X and their realization following their distribution will be written as lowercase letters x.

A probability over a discrete variable will be denoted as $P(x)$ while a probability over a continuous variable will be written $p(x)$. Marginal probability is noted as $P(X = x)$ or $P(x)$, $P(Y|X)$ is the conditional probability of Y given X and $P(X, Y)$ their joint probability.

Fields notations will use blackboard bold symbols such that the field of real numbers will be denoted as $\mathbb{R}$, and the finite field of integers of order q as $\mathbb{Z}_q$.

A matrix is defined with bold letter $\mathbf{A}$. Element at position i, j of $\mathbf{A}$ is noted $\mathbf{A}_{i,j}$. Matrix i^{th} row is given by $\mathbf{A}_{i,:}$ and its ith column by $\mathbf{A}_{:,i}$. The transpose of $\mathbf{A}$ is denoted $\mathbf{A}^\mathsf{T}$.

2.2.2 Considered targets

This section describes the targets considered and their implementations used in this work.

2.2.2.1 MbedTLS

This target considers the RSA sliding window software implementation from MbedTLS library. As the window size chosen to conduct our work is equal to 1, it reduces to a SM implementation, in which the same function is used to compute the square and multiply operations. The binary exponentiation is processed from the MSB to the LSB. Since the MSB is always equal to 1 (thus a square operation followed by a multiply operation), the fourth pattern of the exponentiation is chosen, where it will result in a square operation if the second bit value is equal to zero, or a multiply operation otherwise.

The device on which is executed the considered RSA software implementation is an LPC-55S69 microcontroller from NXP which is designed around a Cortex M-33 core. It is mounted on a NewAE Chipwhisperer CW308 UFO attack board. Acquired traces result from the global power consumption measurement with a Tektronik CT6 current probe, coupled with a low-noise amplifier. The oscilloscope used for the traces acquisition is a Rohde & Schwarz RTO 1044 configured with a 5GS/s sampling rate.

The processing steps are done following [DLO+19] work. The obtained dataset thus consists in a collection of 80000 patterns, each one with a random key bit. A sample pattern of MbedTLS dataset can be observed on Figure 2.1. Each individual pattern consists in 4100 points and can be subdivided into 32 smaller peaks, corresponding to the computation of each limb in the execution of the `mbedtls_mpi_core_montmul` function. For the rest of the study, this dataset will be referred as $\mathcal{D}_{\text{mbedtls}}$.

2.2.2.2 Cswap pointer & arith

This target has been proposed by authors from [NC17] and then used in [PCBP20]. They targeted the Curve25519 curve from μNaCl library [HS] with a 128 bits security level. More specifically, two software Elliptic Curve Scalar Multiplication (ECSM) implementations, denoted as *pointer* and *arith*, and both based on a Montgomery Ladder approach, where considered in these works. The only difference between both implementations lies in the way the conditional swapping is performed. In the *arith* implementation, conditional *swap* is done by exchanging actual coordinates values. However, in the *pointer* implementation, swapping is performed between pointers instead of values themselves, resulting in an eight time faster

execution. As the authors mentioned, the μNaCl implementation does not feature any countermeasure against SCA. Nevertheless, the computation is done by randomizing the scalar at each execution, making resulting traces unique. The used hardware is a STM32F4 microcontroller with 32-bit ARM-M4 CPU running at 168MHz. The acquisition is performed using a Langer RF-U 2.5-2 probe, a Lecroy Waverunner 8254M oscilloscope with a 2.5GS/s sampling rate and an analogue low pass filter at 250MHz. The datasets are built from the collection of measurements of 300 complete scalar multiplications, each being divided in 255 patterns. This results in sets of 76500 patterns representing each the processing of a single bit of the private scalar s. In the rest of this work, references to *pointer* and *arith* datasets are denoted by $\mathcal{D}_{\text{pointer}}$ and $\mathcal{D}_{\text{arith}}$ respectively. Figure 2.1 displays a randomly selected pattern (in black) along with the average pattern (in red) for both datasets. Since, arith implementation is slower, the resulting size of patterns is eight times bigger than that of pointer patterns. One can notice these two datasets exhibit a non-negligible amount of saturated (truncated values) as this can be observed on the random patterns as on the averaged patterns. This will be developed in further analysis in the next sections.

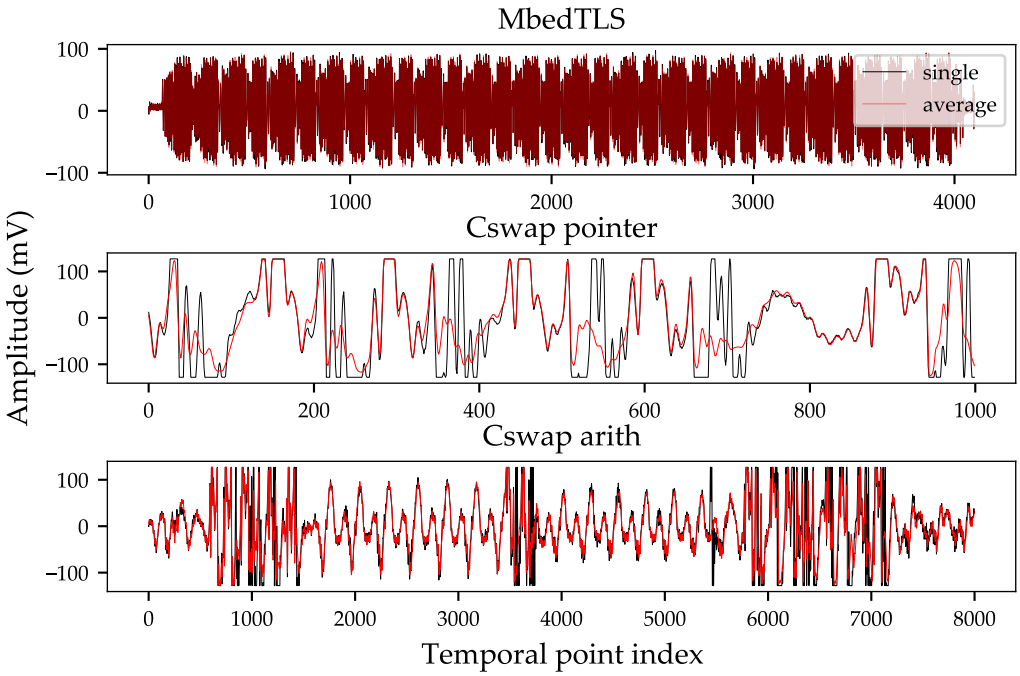

Figure 2.1: A randomly selected pattern and the average pattern for the three considered datasets

2.3 Effectiveness of common horizontal attacks

Several factors can affect the effectiveness of horizontal attacks and PoI selection methods.

To gain insight into the importance of the effect of each factor on the effectiveness of leakage extraction methods, we considered two approaches that are very common in the context of horizontal SCA. First, a univariate selection method based on the T-test, which can be used in both supervised and unsupervised modes. Secondly, a multivariate method based on dimensionality reduction. These methods are first studied theoretically and then applied to synthetic and real traces, the latter coming from the three datasets mentioned

above.

2.3.1 Univariate PoI selection

When performing data analysis, it is mandatory to distinct relevant from non-relevant features. This can be done through several ways. One approach, called univariate feature analysis, considers each univariate feature independently, unlike the multivariate approach, which considers statistical relationships between features. This univariate method is usually called univariate PoI selection in the SCA context.

2.3.1.1 Supervised selection

In SCA, univariate PoI selection is typically used during the profiling phase of supervised attacks. Its application yields a set of PoI enclosing the leakage related to the secret or target classes. The most commonly used metric for univariate PoI selection when the number of target classes is equal to $k = 2$ is the Welch T-test statistic [WEL47]. This approach can be generalized to problems with $k > 2$ by using variance decomposition by label, which can be done with methods such as the ANOVA [Fis19] or the NICV [BDGN14]. However, as considered targets for this work only present two classes problems, focus is made on the Welch's T-test statistic, denoted as $\mathcal{T}$ and computed as follows:

$$\mathcal{T} = \frac{\overline{X_1} - \overline{X_2}}{\sqrt{\frac{s_1^2}{N_1} + \frac{s_2^2}{N_2}}} \tag{2.1}$$

where $\overline{X_i}$ denotes the ith class empirical mean, s_i^2 its empirical variance and N_i the number of elements in the class.

This statistic, which follows a Student's distribution, is generally used to decide whether two samples have the same mean. In the context of horizontal attacks, it is often used as a score metric of separability between the two classes using a clustering algorithm, and as a measure of the leakage strength. Consequently, in a profiling phase, an attacker would therefore choose points characterized by high $\mathcal{T}$-values as PoI to build an attack model. Since ground truth labels are known beforehand, this statistic can easily be used to accurately identify the most relevant points.

In order to characterize the exploitability of previously introduced datasets, the T-test has been applied on each temporal feature $\mathcal{D}_{:,i}$ independently. This procedure is also known as the Test Vector Leakage Assessment (TVLA) [GJJR]. Figure 2.2 shows the $\mathcal{T}$ values obtained for each time point and for each considered target.

At first glance, one can notice different leakage manifestations on these three datasets. On MbedTLS, observed absolute $\mathcal{T}$-values, often greater than 200, give evidence of a highly exploitable leakage evenly distributed on the pattern. On the other hand on the Cswap pointer set, most of the leakage manifestation is very localized between time points 600 and 650. Finally, the leakage on Cswap arith is more diffused through the patterns and seems less important with $\mathcal{T}$-values often less than 100, with two main leakages zones between time points 1500 to 3300 and 3800 to 5500. According to the μNaCl implementation, these two zones seem to correspond to the arithmetic swap of (X_p, X_q) and (Z_p, Z_q) respectively (see implementation for details).

Overall, the leakage levels seem satisfactory for exploitation in a supervised context, given that a profiling phase is possible since true labels are known. Thus, Figure 2.2 can be

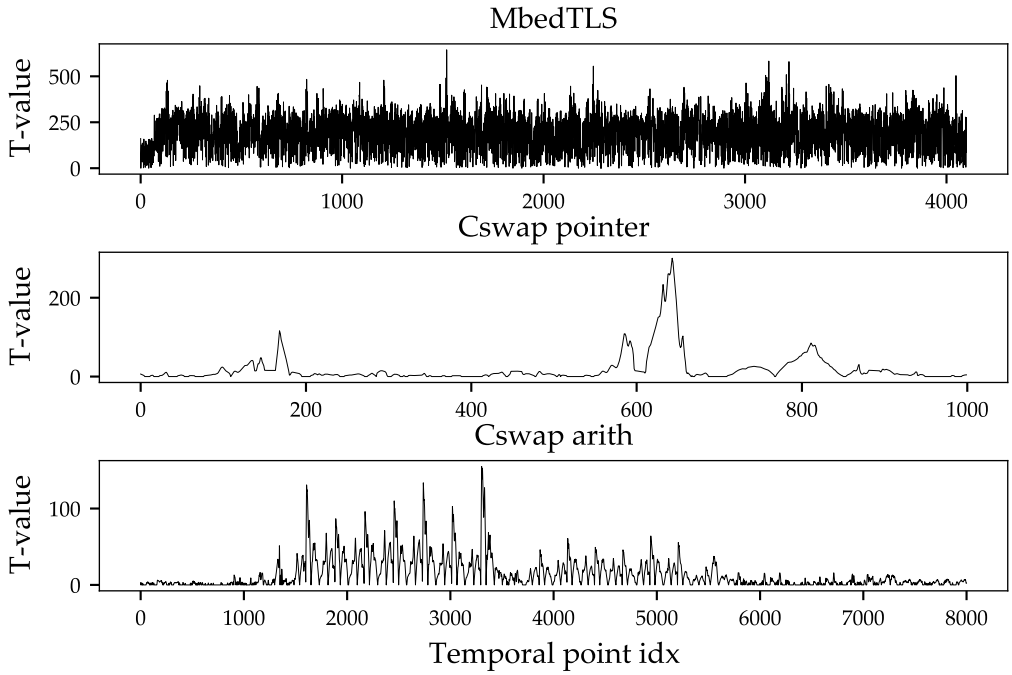

Figure 2.2: Welch's statistic values obtained for the three datasets considered

seen as a baseline or upper bound for the leakage analysis performed in supervised context, considering the T-test statistic metric. It will be used as a reference for the rest of this work. Still, it should be noted that $\mathcal{T}$-values cannot be taken as a safe and absolute indicator of the separability of two classes since not all aspects of the leakage are involved in the calculation of this statistic. Defining a dedicated metric thus seems to be a good way to enhance the PoI selection in the context of horizontal attacks. This is elaborated later in this work.

2.3.1.2 Unsupervised selection

However, in an unsupervised attack scenario, namely unsupervised horizontal attacks, access to true labels is not available. It is thus not possible to apply Welch's T-test to characterize which univariate time point carry leakage information.

Still, it is possible to get some insights of the exploitability of each time point. As previously presented, the proposed method from Perin *et al.* work [PITM14] aims to identify candidate leakage PoI by performing univariate clustering on each time point and then analyze the difference of means between the resulting two clusters. Usually, the k-means algorithm [Mac67] or variants (Fuzzy k-means [JCB84], Expectation Maximization [DLR77]) are used. The k-means algorithm can be performed as follows. First, define an initial set of centroids

$$\mu = (\mu_1, \ldots, \mu_k), \mu_i \in \mathbb{R}$$

and a set of clusters assignment

$$c = (c_1, \ldots, c_n), c_i \in \mathbb{Z}_k$$

For the initialization, centroids are usually randomly assigned. Alternatively, in [PITM14], authors choose the minimum and maximum values of observed data as default centroids values. Then, μ and c are updated alternately through coordinate descent:

1. Assign each sample x_i to its nearest centroid based on Euclidean distance:

$$c_i = \underset{j=1,\dots,k}{\operatorname{argmin}} ||x_i - \mu_j||^2 \tag{2.2}$$

2. Update centroids as the mean of their assigned samples:

$$\mu_j = \frac{\sum_{i=1}^{n} 1\{c_i = j\} x_i}{\sum_{i=1}^{n} 1\{c_i = j\}} \tag{2.3}$$

This two steps are repeated until no change in assignment is observed between two cycles, i.e. the algorithm converged. Overall, k-means aims at optimizing the following global objective (also known as the k-means loss), considering the Euclidean distance is used:

$$\mathcal{L} = \sum_{i=1}^{n} \sum_{j=1}^{k} 1\{c_i = j\} ||x_i - \mu_j||^2 \tag{2.4}$$

Equation 2.4 try to minimize the intra-clusters sum of squares, or equivalently maximize the inter-clusters sum of squares. In other words, k-means aims to find a partitioning in k classes that gives the most compact and separated clusters. It is also important to note that as Equation 2.4 is non-convex and since k-means results depend on the centroid's initialization, it will only find a local optimum.

After the application of the aforedescribed clustering step, it is required to compute the $\mathcal{T}$-value (Equation 2.1) for each time point of the patterns to obtain a separability or exploitability score for the attack. In addition, a ratio parameter can be used to weight each $\mathcal{T}$-value, according to [PC16]. Indeed, k-means optimization has no prior on the awaited clusters repartitions, and thus is not guaranteed to produce balanced clusters as expected when the secret is randomized. Therefore, it is suggested to multiply the obtained $\mathcal{T}$-values by the ratio coefficient (see Equation 2.6) to give higher importance to time points where observed partition is close to the awaited one.

$$r_i = \frac{\min(N_i, l_i)}{\max(N_i, l_i)} \tag{2.5}$$

where N_i is the number of elements observed in the ith class and l_i the assumed number of awaited elements. For this work, $l_1 = l_2 = \frac{|\mathcal{D}|}{2}$, with $|\mathcal{D}|$ the cardinal of $\mathcal{D}$. The updated $\mathcal{T}$-values considering ratio parameter are then given by:

$$\mathcal{T}_{ratio} = r_1 r_2 \frac{\bar{X_1} - \bar{X_2}}{\sqrt{\frac{s_1^2}{N_1} + \frac{s_2^2}{N_2}}} \tag{2.6}$$

From there, it is interesting to compare the behavior of the $\mathcal{T}$-values in a supervised way with those obtained following the above described unsupervised approach. Note that for the supervised approach, since labels are known, Equation 2.6 reduces to Equation 2.1. First, let us observe the behavior on $\mathcal{D}_{mbedtls}$ dataset on Figure 2.3. For readability purpose, supervised $\mathcal{T}$-values are reported from Figure 2.2.

As noted above, $\mathcal{D}_{mbedtls}$ has relatively high leakage. It can be observed that most of the relevant PoI obtained with the supervised approach are the same with the unsupervised approach. Overall, the application of unsupervised method allows to successfully identify most of the leaking PoI. Furthermore, we can observe on Figure 2.4, the distributions of

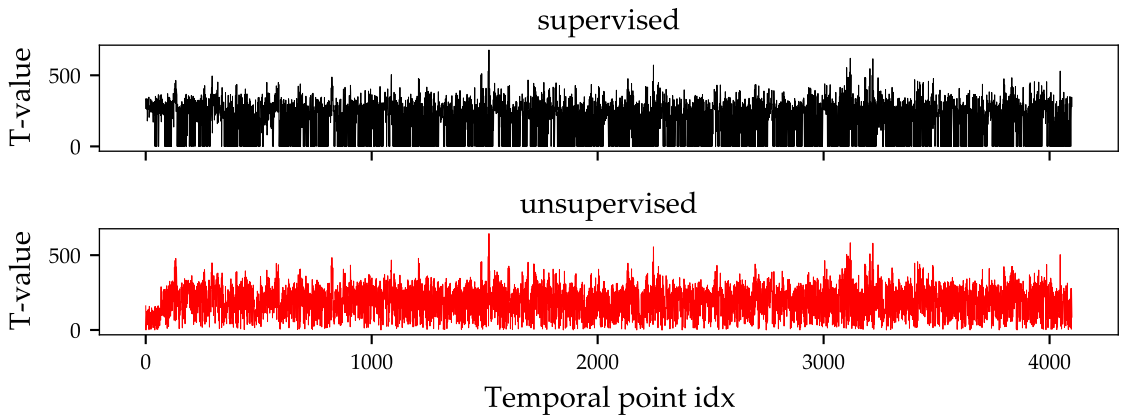

Figure 2.3: Comparison of supervised $\mathcal{T}$-values with the unsupervised $\mathcal{T}$-values obtained after clustering of each time point for $\mathcal{D}_{mbedtls}$ dataset.

the four time points with highest $\mathcal{T}$-values which are the same on both unsupervised and supervised approaches.

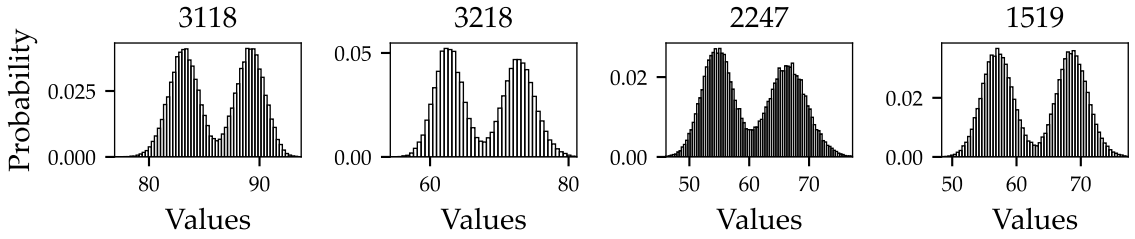

Figure 2.4: Histograms of the four most leaking points (and their index) according to $\mathcal{T}$-values obtained after applying the unsupervised approach of Perin *et al.* to the $\mathcal{D}_{mbedtls}$ dataset. The results are the same with supervised approach.

One can notice that observation of global distribution of these time points depicts two Gaussian distributions, with enough separation, making the discrimination between the two behavior from the two classes almost straightforward. This confirms the exploitability of $\mathcal{D}_{mbedtls}$ dataset in the context of horizontal attacks. The application of method proposed by Perin *et al.* [PITM14] and [PC16] is relevant for identifying the most prevalent PoI in the context of a high leaking target dataset.

2.3.2 Gaussian Mixture Model

Let us model the observed distributions of Figure 2.4. In a univariate context with two classes, namely regular modular exponentiation or scalar multiplication implementations, the two possible values of the secret manipulated bits of d are expected to express on two specific values in PoI. In addition, because the measurement noise is assumed to be Gaussian, these specific values become two Gaussian distributions. The underlying leakage model behind the application of Welch' test can then be formulated as a Gaussian mixture distribution, which p.d.f is given by:

$$p(x) = \pi_1 \mathcal{N}(x|\mu_1, \sigma_1^2) + \pi_2 \mathcal{N}(x|\mu_2, \sigma_2^2) \tag{2.7}$$

where each component p.d.f is a Gaussian distribution with parameters $\theta_i = \{\mu_i, \sigma_i^2, \pi_i\}$, denoting distribution mean and variance, as well as the repartition weight coefficient in the mixture. A leaking point distribution can be then be formulated as a mixture of two components, well separated (with high difference of means) and low variances, or in other words, a minimum intra-class variance and a maximum inter-class variance. On the contrary, a non-leaking point should exhibit an unimodal global distribution with completely overlapped components, making them indistinguishable.

2.3.2.1 Linear separability

Figure 2.5 displays the $\mathcal{T}$-values resulting from the application of Welch's test to synthetic mixtures of Gaussian distributions with varying means and variances. It allows to describe the awaited distribution of a perfect leaking point in a univariate context:

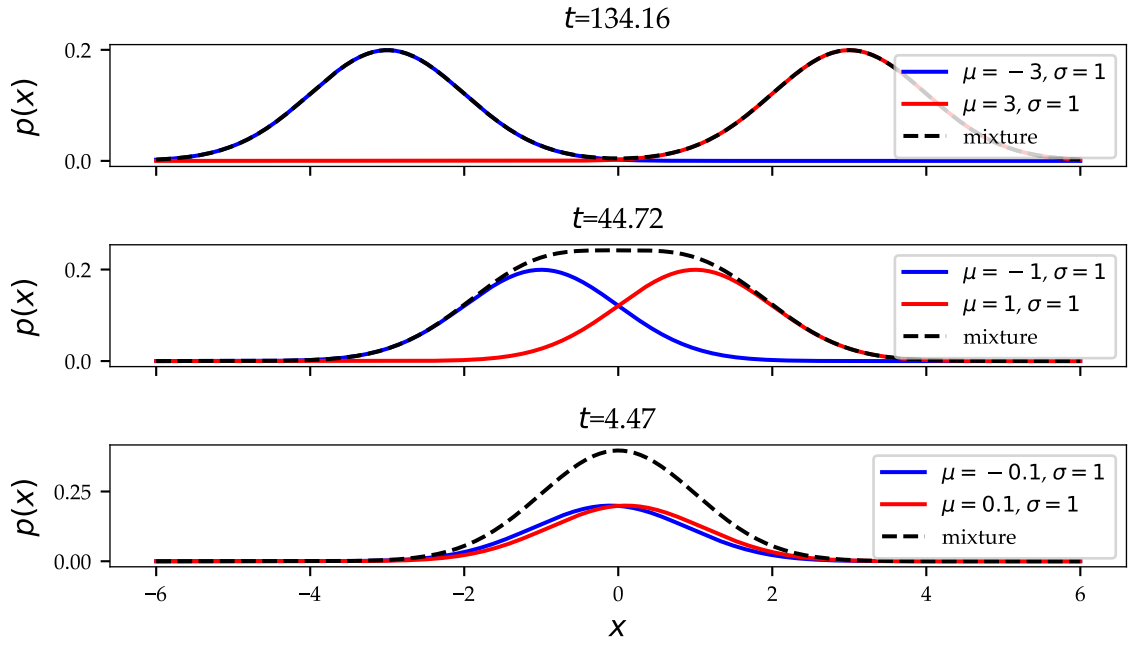

Figure 2.5: Illustration of simulated univariate leakages. Top figure displays a high leaking point, characterized by high difference of means. Bottom figure displays completely overlapping components, resulting in non-exploitable point. Middle figure shows two different behavior but not enough separated to results in a bimodal mixture. Note that bimodality arises only when $|\mu_1 - \mu_2| > 2 \times \min(\sigma_1, \sigma_2)$ [Beh70].

Definition 1. *A time point is considered to be perfectly distributed if all samples belonging to each of its components are linearly separable at one point.*

Indeed, if a linear classifier can perfectly separate all observed samples for a given time point, with a class repartition matching the expected one, there subsists no ownership uncertainty. Most clustering algorithms can be seen as linear classifier in the case of univariate clustering.

Still, it is important to note that a high $\mathcal{T}$-value does not necessarily translate into perfect linear separability. Rather, it indicates a high means difference with respect to the involved component variances, providing evidence that the underlying classes are highly discernible.

Consequently, as long as $\mu_1 \neq \mu_2$, the decision boundary becomes more and more discriminative, as components variances σ_1^2 and σ_2^2 approach zero. This results in tightly packed clusters and making it easier to separate the components. Above the threshold where linear separability is reached, higher $\mathcal{T}$-values do not result in higher leakage exploitability nor better classes discrimination.

2.3.2.2 Unsupervised T-test for PoI selection

With this in mind, let us now compare the results of unsupervised T-test (by applying clustering) with the supervised one on Cswap datasets. First, let us observe the results for $\mathcal{D}_{pointer}$ shown on Figure 2.6.

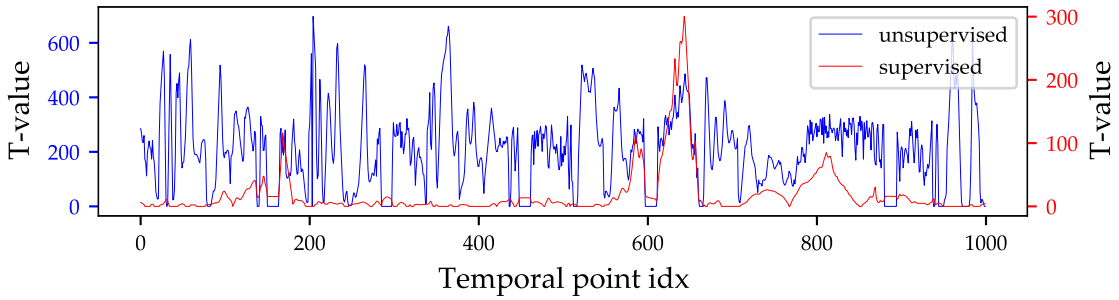

Figure 2.6: Comparison of supervised vs unsupervised $\mathcal{T}$-values (after univariate clustering on each time point) for $\mathcal{D}_{pointer}$ dataset.

For readability, both results are stacked on the same plot. It can be seen that most prevalent PoI found on the unsupervised approach are not leaking according to the supervised approach. Indeed, while some high values are observed on the blue curve between time points 610 and 670, their $\mathcal{T}$-values remains lower when compared to other points that don't carry significant leakage according to the red curve baseline.

In this situation, if an adversary had decided to exploit this dataset in an unsupervised manner, namely by considering the 10 points with the highest $\mathcal{T}$-values as PoI, he would not have recovered d because these PoI do not carry enough information, or an information which is hardly exploited using a clustering algorithm to make classes discriminable.

To get more insight about this behavior and understand why false positive points with such high $\mathcal{T}$-values appear, let us observe their distribution on Figure 2.7. The top row shows the distributions of the four most leaking points according to an unsupervised analysis, while the bottom row shows the same results for supervised analysis. At first glance, it is easy to notice that top found PoI for $\mathcal{D}_{pointer}$ are distributed differently compared to the previously studied dataset, namely $\mathcal{D}_{mbedtls}$. Indeed, the reported distributions show a high percentage of saturated values, especially at time points 364 and 365. Herein, we refer as saturated values a value which is outside of the observable range due to the occurrence of outliers or a poorly set caliber of the oscilloscope used to acquire the waveforms. Since all datasets are sampled at 8 bits, which corresponds to a range of 256 values between -128 and 127, saturated values are observations equal to -128 or 127 and result most of the time in a loss or deformation of information.

On the other hand, most of the leakage points in the bottom row of this figure, identified using a supervised approach, appear to be bimodal Gaussian mixtures, as expected from the aforementioned intuitive univariate leakage model.

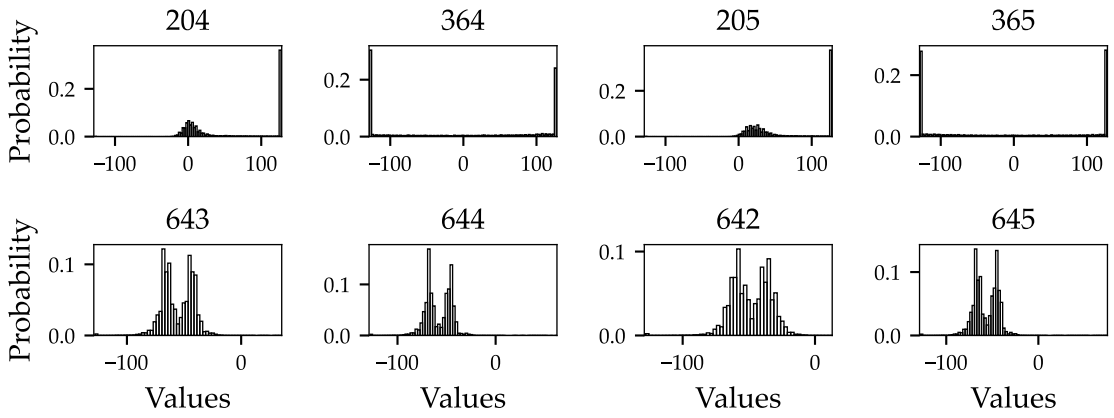

Figure 2.7: Histogram of the four most leaking points and their index according to $\mathcal{T}$-values on unsupervised (top row) and supervised (bottom row) approaches for $\mathcal{D}_{pointer}$.

The same behavior can be observed for $\mathcal{D}_{arith}$ dataset in Figure 2.8 which shows the same results as those in Figure 2.6. Indeed, most of the prominent peaks obtained with the

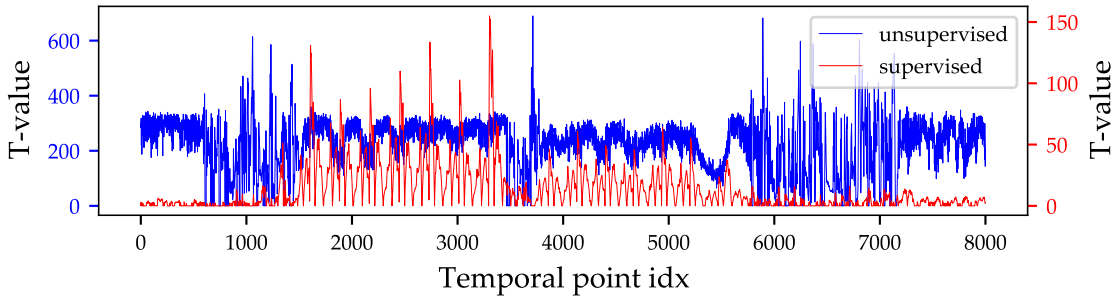

Figure 2.8: Comparison of supervised vs unsupervised T-test (after clustering) T-test values for $\mathcal{D}_{arith}$.

unsupervised approach do not match those obtained with the supervised approach (in red). Moreover, the highest unsupervised $\mathcal{T}$-values are about 4 times larger than the highest supervised ones. Observation of some point distributions suggests that this could be explained by the very small variances of the clusters. To support this intuition, Figure 2.9 shows the distributions of such PoI obtained with using unsupervised (top row) and supervised (bottom row) methods. A shown, identified PoI using the clustering method (unsupervised approach) display a high amount of saturated points. As a result, the two clusters obtained have means such that $\mu_1 \approx -128$ and $\mu_2 \approx 127$, and thus a very large difference of means. Furthermore, since almost all values are either equal to extreme values -128 or 127, the variances of the two clusters are very close to zero. All in all, this results in very high $\mathcal{T}$-values.

On the other hand, most of the leakage points obtained by applying a supervised T-test describe almost Gaussian distributions. This emphasizes that leakage on timestamps of $\mathcal{D}_{arith}$ set is less exploitable than that of $\mathcal{D}_{pointer}$. This intuition is supported by the $\mathcal{T}$-values. Indeed, $\mathcal{D}_{arith}$ most leaking PoI has a supervised $\mathcal{T}$-value of about 150, while for $\mathcal{D}_{pointer}$ this value is about 300.

Overall, the above analysis shows that, unlike $\mathcal{D}_{mbedtls}$, $\mathcal{D}_{pointer}$ and $\mathcal{D}_{arith}$ are strongly affected by saturated values, making the use of the T-test metric in this context imprecise and

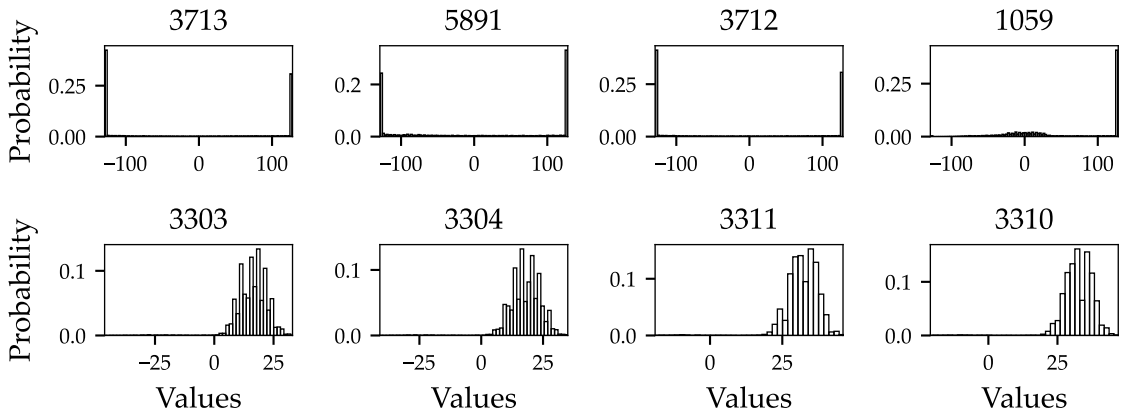

Figure 2.9: Histogram of the four most leaking points of $\mathcal{D}_{arith}$ (and their index) according to $\mathcal{T}$-values for the unsupervised (top row) and supervised (bottom row) approaches.

unsafe. Indeed, it is not possible to estimate properly the means, involved in the computation of the $\mathcal{T}$-value, of the underlying distributions. Indeed, on $\mathcal{D}_{pointer}$ and $\mathcal{D}_{arith}$, saturated values can represent up to about 30% and 13% of observed values respectively. This can be caused by errors in the setup of the acquisition equipment, some additional behavior, or side effects from other sources. Note that $\mathcal{D}_{mbedtls}$ does not contain any saturated values. The problem of saturated values will be developed in the next sections.

2.3.2.3 $\mathcal{T}$-value relevance for PoI selection

To elaborate on the T-test relevance as a PoI selection metric, multidimensional clustering was applied on the most relevant PoI according to the T-test statistic. To characterize the percentage of correctly recovered bits after clustering, we defined the Bit Recovery Rate (BRR) as the maximum percentage of bit successfully recovered from a given prediction:

$$\text{BRR} = \max_{m} \frac{\sum_{i}^{n} 1\{y_i = m(c_i)\}}{n} \times 100 \tag{2.8}$$

where c_i is the predicted value of ith pattern and y_i its true label, m is a mapping function ranging on all possible one-to-one mappings. The BRR metric is herein used for study purposes only, since these true labels are not available in an unsupervised attack context. Figure 2.10 reports the BRR values by selecting the p PoI with the highest $\mathcal{T}$-values for the three datasets. These results were obtained by running the attack 100 times using 255 patterns for $\mathcal{D}_{pointer}$, $\mathcal{D}_{arith}$ and 2048 patterns for $\mathcal{D}_{mbedtls}$. As expected, it is possible to recover almost 100 % of the bits for $\mathcal{D}_{mbedtls}$ and the results between the supervised and unsupervised approaches are really close since most of the selected PoI are the same, as highlighted in Figure 2.4.

On the other hand, $\mathcal{D}_{pointer}$ and $\mathcal{D}_{arith}$ sets shows an average BRR of about 52 % overall in unsupervised context. This corroborates with results reported by authors of [PCBP20] (before application of their framework). In the supervised context, considering the advantage of knowing labels, both sets depicts more satisfactory BRR. For $\mathcal{D}_{pointer}$, selected points clustering results in a BRR of about 98%, but we can observe a slope after 50 PoI, as the quality of accumulated PoI decreases. For $\mathcal{D}_{arith}$, the opposite behavior is seen, the BRR is lower at first (about 80%) but increases towards an average of 96% after aggregating enough

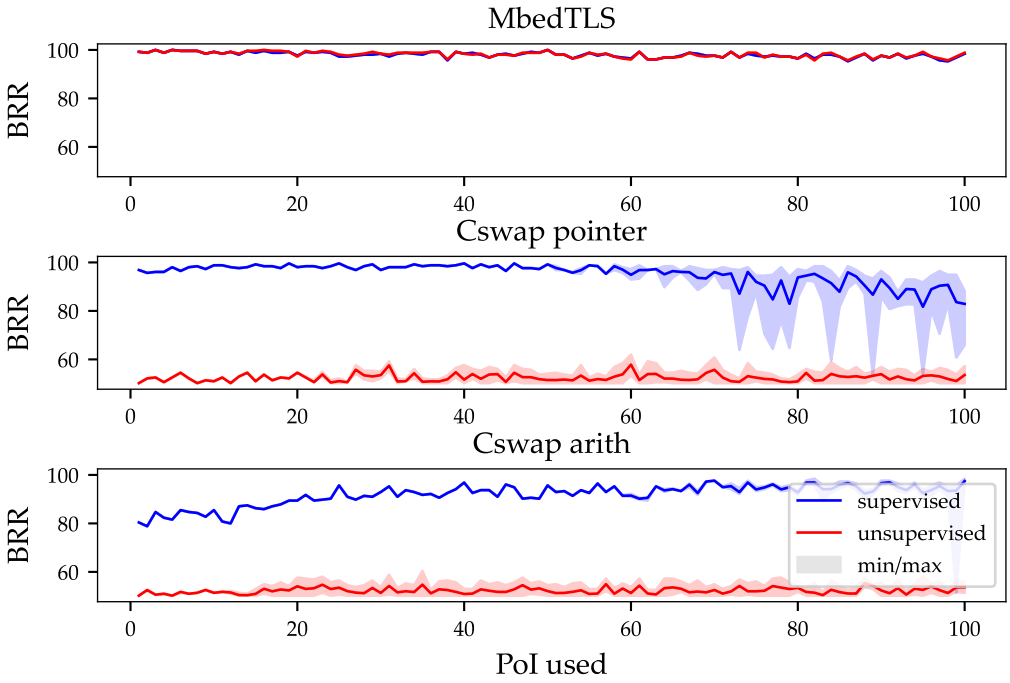

Figure 2.10: BRR.

PoI. This increase suggests that the $\mathcal{T}$-value is not a perfect metric to assess the separability of the underlying distributions.

Finally, to better understand the limitations of using the T-test for PoI selection, we applied it to synthetic data. All synthetic data sets considered were built from an univariate mixture of two normal distribution having same weighting (or mixing) parameters, opposite means and equal standard deviations. Parameters are listed on Table 2.1. For each mixture,

Parameter	Value
μ_1	$[0.1, 5]$
μ_2	$-\mu_1$
$\sigma_1 = \sigma_2$	$[1, 5]$
$\pi_1 = \pi_2$	0.5

Table 2.1: Mixture parameters for T-test experiment

the following experiment consisted in computing the $\mathcal{T}$-value both in a supervised (Equation 2.1) and in an unsupervised context. The Figure 2.11 allows comparing $\mathcal{T}$-values obtained on both contexts. It reports the results with on the right the $\mathcal{T}$-values obtained with the supervised approach, considered as the ground truth, and on the left, those obtained with the unsupervised approach. Comparing the two maps indicates that the unsupervised approach provides overestimated $\mathcal{T}$-values for small values of $|\mu|$, regardless of the value of σ, and for moderate values of $|\mu|$ and large values of σ, i.e. for all pairs ($|\mu|$,σ) below the first bisector. Indeed, if the observed global distribution tends to be unimodal, the clustering will produce separated clusters, since k-means try to optimize Equation 2.4 and does not converge to singularity clustering. As a result, since obtained clusters are well separated, the variances of each cluster are lower than that of the underlying true distributions (mixture components) and result in the overestimation of $\mathcal{T}$-values. Furthermore, instead of ownership probabilities of class c_i given by $P(c_i|x) \in [0, 1]$, hard clustering reduces to

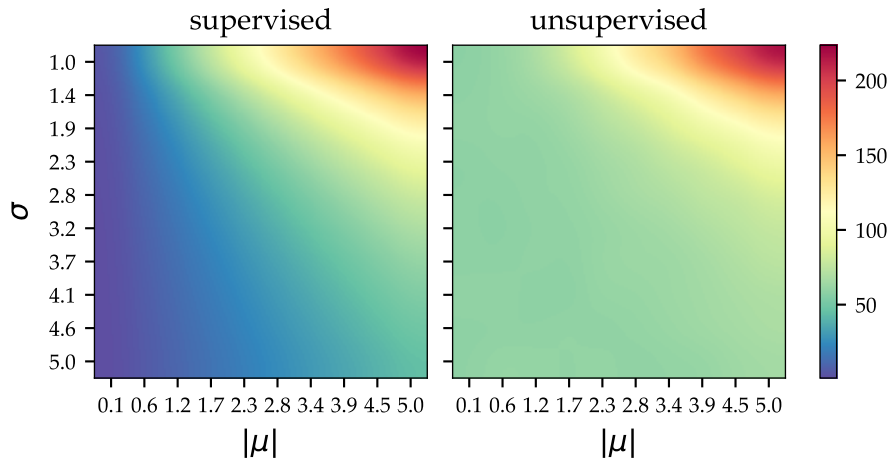

Figure 2.11: Comparison of $\mathcal{T}$-values obtained in supervised manner (left) and in unsupervised manner (right) on simulated mixtures.

$P(c_i|x) \in \{0, 1\}$, producing smaller variances and thus overestimated $\mathcal{T}$-values.

To further support this statement, the same experiment is carried out again, but instead of calculating $\mathcal{T}$-values, the maximum observed standard deviation $\max(\sigma_1, \sigma_2)$ between the two obtained clusters has been computed after applying the k-means clustering algorithm on the mixture. Results are depicted on Figure 2.12. As expected, standard deviation are

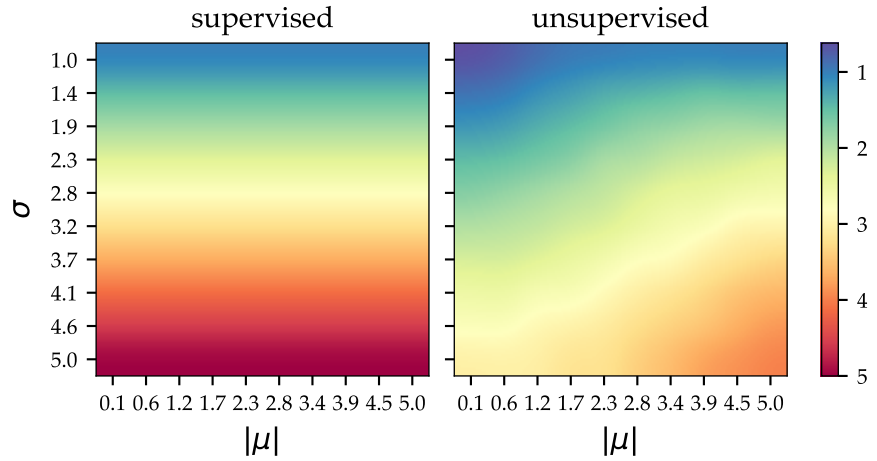

Figure 2.12: Heatmap of estimated $\max(\sigma_1, \sigma_2)$. Left: supervised approach ($\max(\sigma_1, \sigma_2) = \sigma_1 = \sigma_2$). Right: unsupervised approach with σ values estimated from after classification of values by the k-means algorithm.

underestimated as μ decreases and σ increases i.e. as soon as the mixtures become closer to an unimodal distribution.

2.3.2.4 Partial conclusion

These results highlight some limitations of using the T-test and k-means simultaneously to select relevant PoI in a univariate analysis. Indeed, if it is efficient at detecting strong leakages (characterized by well separated components in the mixtures like those found in MbedTLS), it appears to be insufficient as soon as the mixture components overlap sufficiently to form a distribution close to be unimodal which is typical situation in traces with

low leakages (like Cswap arith). Furthermore, T-test metric considers hard class assignment, but an uncertainty exists when there is no linear separability as seen previously.

2.3.3 Impact of outliers and saturation on leakage

We also considered the problem of additional components in observed mixture. For components identification, as explained before, clustering is applied to search for $k = 2$ components. Indeed, it is assumed that since the target process can be divided in two behaviors, the resulting observation should be composed of only two components. In reality, this is not always the case. If the behavior of these components is distributed differently with main components, we can define these values as outliers.

Definition 2. *An outlier is a realization that differs significantly from other ones to the point that it is not part of the same generation process.*

2.3.3.1 Outlier detection

If one considers the Inter-Quantile Range (IQR) (also used in [NC17] work) as an outlier identification method, a sample value x is then identified as an outlier if it is outside the interquartile range:

$$[Q_1 - 1.5 \times IQR, Q_3 + 1.5 \times IQR] \tag{2.9}$$

where $IQR = Q_3 - Q_1$. This approach allows to quantify the volume of abnormal observed behavior in a distribution that could impact either the PoI selection method or the multidimensional clustering.

It is important to notice that the topic of outliers is closely related to that of saturated points, which are extreme points falling out of the measurement range with considered equipments and applied settings. In the context of SCA, this corresponds to elements taking values -128 or 127 (since traces are discretized on 8 bits). If the amount of saturated values on an empirical distribution exceeds a certain threshold, these points cannot be considered anymore as outliers as their population is too large to be seen as an anomalous behavior. They should rather be viewed as the result of the truncation of the real underlying distribution.

2.3.3.2 Outlier or not

We examined the relationship between outliers (in terms of IQR) and saturated values with synthetic data. To that aim, we drew samples from a univariate mixture distribution such that $\pi_1 = \pi_2 = \frac{1}{2}$, $\mu_1 = -2, \mu_2 = -\mu_1$, and $\sigma = 1$. Then, we progressively add saturated values by randomly replacing some values from drawn samples with a defined extreme value (10 or -10), until the percentage of saturation reached 100%. We did this either on both sides of the distribution or on a single side. During this process, we estimated the rate of observed outliers for each saturation level.

The results from this experiment are shown on Figure 2.13. As expected, as the percentage of saturated values increases, the percentage of detected outliers first increases as well. However, when saturated values are added on both sides of the distribution, the resulting distribution is considered to be outlier free as soon as the percentage of saturated value rises above 25%. When saturated values are added on a single side of the distribution, the situation is slightly different. Indeed, when the percentage of added saturated values rises above 75%, values of initial mixture become themselves outliers regarding Equation 2.9.

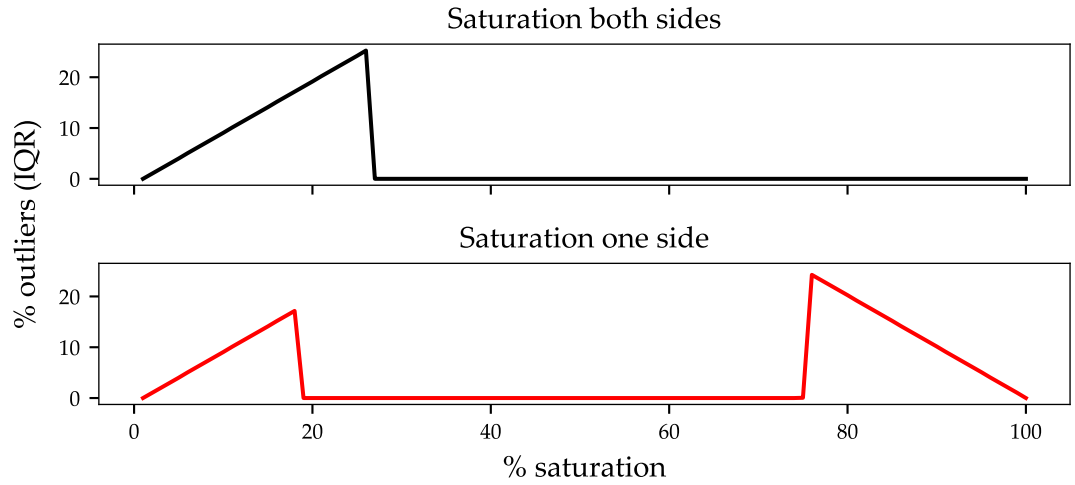

Figure 2.13: Evolution of the percentage of identified outliers using the IQR method when adding saturation on both sides of the distribution (top) or on a single side (bottom).

This is illustrated in Figure 2.14 by the three histograms of three distributions observed during the analysis process. According to Equation 2.9, the two left ones feature 25% and 18% of outliers while for the distribution on the right, values of initial mixture are considered themselves outliers. This shows that in the case of an over-saturated distribution, the

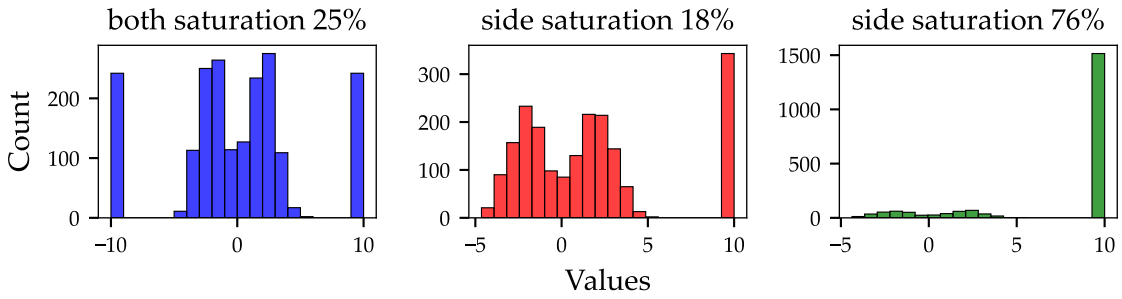

Figure 2.14: Distribution of points.

true measurement values are characterized as outliers by the IQR method instead of the saturated values.

2.3.3.3 Outliers in considered datasets

Despite the limitations of the criterion of Equation 2.9, the latter has been applied to estimate the rate of outliers in all the considered datasets. The results are shown in Figure 2.15. They support the previous observations about the limited quality of traces from $\mathcal{D}_{pointer}$ and $\mathcal{D}_{arith}$ sets. Indeed, only a reduced fraction of points incorporates less than 5% of outliers, unlike $\mathcal{D}_{mbedtls}$.

Because the criterion of Equation 2.9 fails to distinguish correct values from outliers as soon as the percentage of outliers becomes high, we also computed the percentage of saturated values for all points on all datasets. The results are shown in Figure 2.16. If $\mathcal{D}_{arith}$ features some points with more than 5% of saturated values (either -128 or 127), most of the

points of $\mathcal{D}_{pointer}$ are characterized by a rate of saturated values higher than 25%. Regarding $\mathcal{D}_{mbedtls}$, this dataset is saturated value free.

As expected due to the limitations of Equation 2.9, we can observe, especially in $\mathcal{D}_{pointer}$, points with a low percentage of outliers, but a percentage of saturated values close to 100%. From Figure 2.15 and 2.16, we were able to count the number of points with percentages of outliers and saturated values both greater than 5%. It gave the results reported in Table 2.2.

	$\mathcal{D}_{mbedtls}$	$\mathcal{D}_{pointer}$	$\mathcal{D}_{arith}$
Number of points per pattern	4100	1000	8000
Outliers (IQR)	0.79%	4.32%	5.15%
Outliers (IQR) per PoI > 5%	0%	27.5%	34.04%
Saturated	0%	30.27%	12.88%
Saturated per PoI > 5%	0%	53.8%	29.15%
Outliers OR Saturated	0.79%	33.37%	16.51%
Outliers OR Saturated per PoI > 5%	0%	65.20%	46.35%

Table 2.2: Percentage of outliers and saturated values on each considered dataset.

From this table it is clear that the quality of $\mathcal{D}_{pointer}$ is very limited. One must expect to get only a few PoI. The situation of $\mathcal{D}_{arith}$ is slightly different. Indeed, there are only two portions of the traces with really high percentages of saturated values and outliers. Figure 2.15 shows the outlier percentage per time point for all datasets. These results corroborate

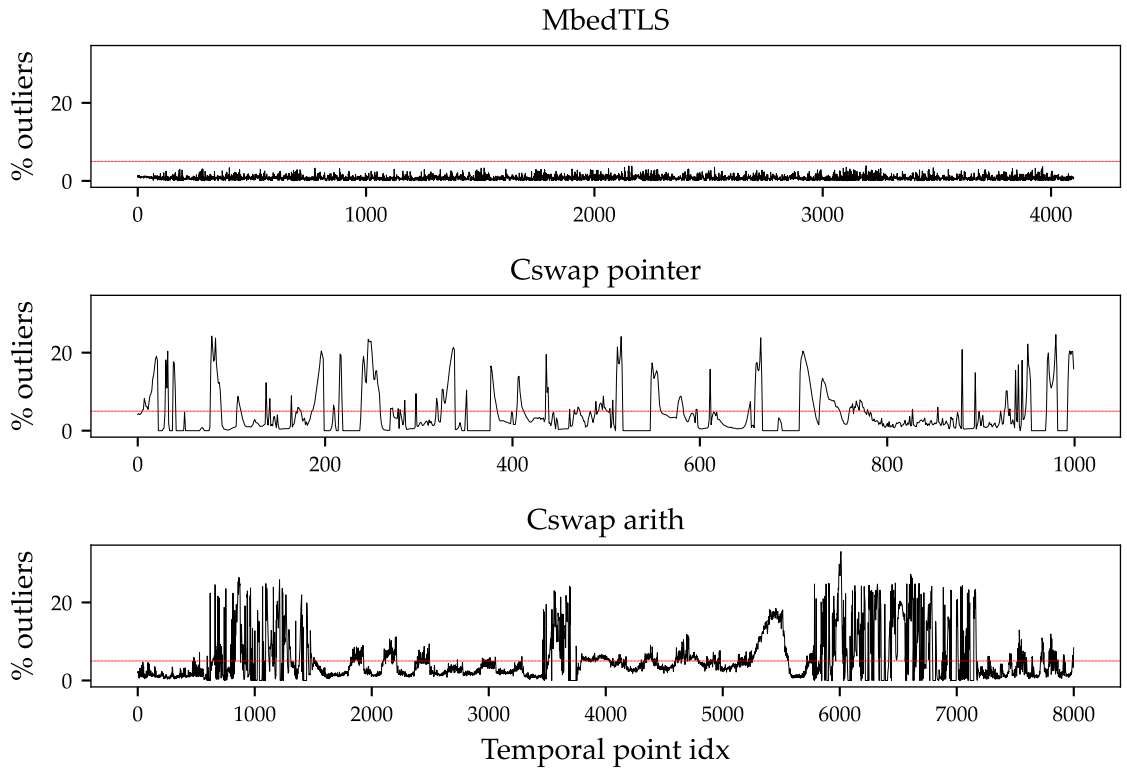

Figure 2.15: Percentage of outliers values per time points for $\mathcal{D}_{mbedtls}$ (top), $\mathcal{D}_{pointer}$ (middle) and $\mathcal{D}_{arith}$ (bottom) according to Equation 2.9.

previous observations about the quality of traces from $\mathcal{D}_{pointer}$ and $\mathcal{D}_{arith}$ sets. In the same way, Figure 2.16 displays percentage of saturation points per time point. One can notice that

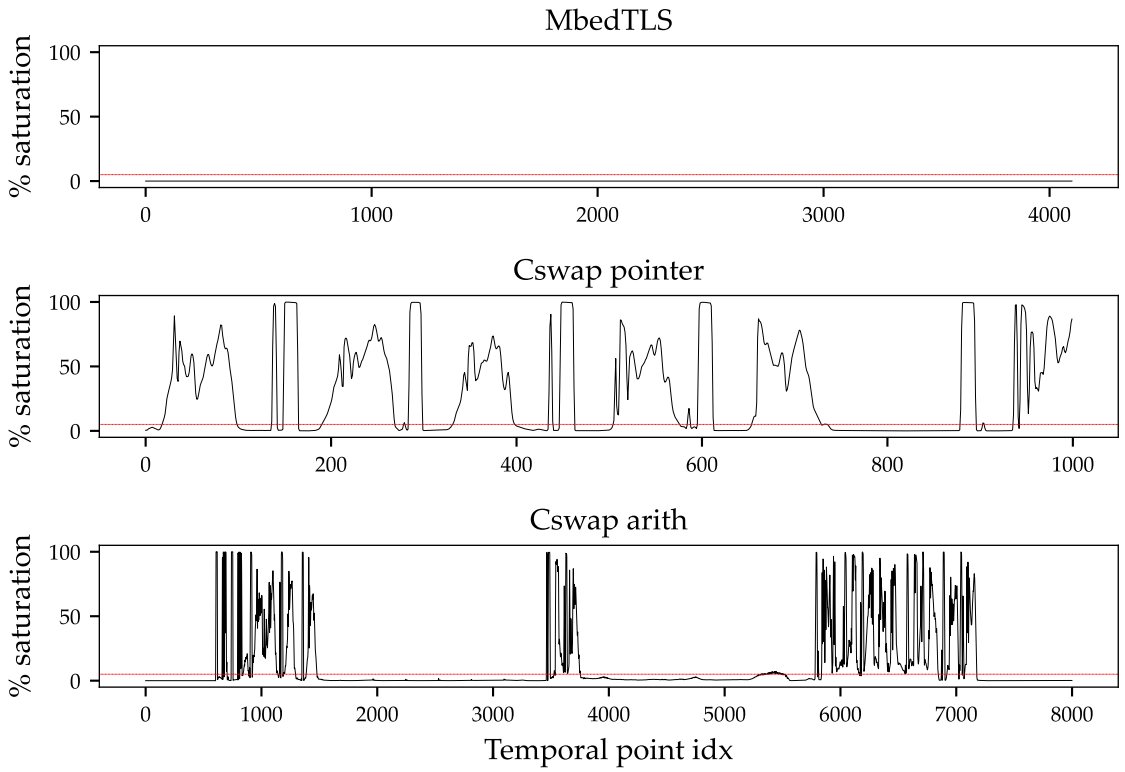

Figure 2.16: Percentage of saturation values per time points for $\mathcal{D}_{mbedtls}$ (top), $\mathcal{D}_{pointer}$ (middle) and $\mathcal{D}_{arith}$ (bottom).

$\mathcal{D}_{mbedtls}$ is free of saturated values, as it has been correctly sampled during the acquisition process. That is not the case for the two other studied sets. If we calculate the percentage of time points where outliers or saturated values represents more than 5% of samples (reported on Table 2.2), we can see that $\mathcal{D}_{pointer}$ and $\mathcal{D}_{arith}$ have 65.20% and 46.35% respectively of their time points that display abnormal behavior. This cannot be ignored in the context of an analysis to select relevant PoI.

2.3.3.4 Impact of outliers and saturated values on the clustering

We took a look at how outliers and saturated values affect the clustering process. Indeed, the presence of abnormal values has a direct impact on the estimation of clusters since outliers influence the calculation of means. To that aim, simulation process was used to generate synthetic data to be partitioned using the k-means clustering. As in previous experiments, points were sampled from a univariate mixture of two normal distributions, well separated and highly discriminable. Samples from this distribution can be observed on the left plot in Figure 2.17. It also shows the clusters obtained using the k-means algorithm with $k = 2$. Unsurprisingly, it is trivial task to separate the two components of the mixture.

However, when adding outliers, this is not the case anymore. To get a better idea on the effect of outliers on the cluster results with the k-means, outliers were generated by progressively replacing random points with extreme values (in this experiment, we chose $o = 10$). The middle and right plots of Figure 2.17 shows that as the percentage of introduced outliers reaches 10%, one of the two denied clusters is formed from the fusion of the two classes, while the other cluster only gather outliers. The same is observed at 50% of outliers replaced values. To corroborate this result, observe Figure 2.18. The left part shows the BRR

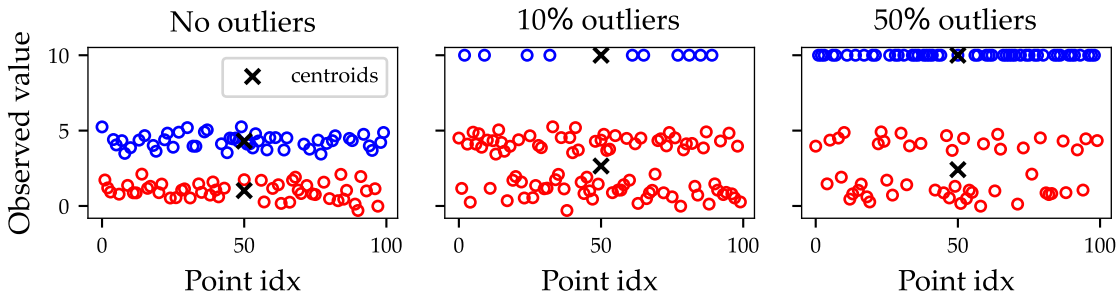

Figure 2.17: Visualization of clustering partitions obtained for different percentages of added outliers.

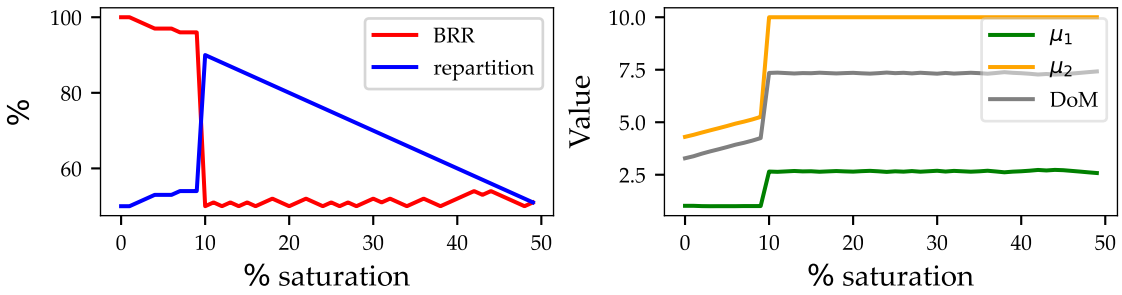

Figure 2.18: Metrics evolution.

(in red) obtained from the clustering process and the associated class repartition (in blue, sampled from $\pi_1 = \pi_2 = 0.5$). It appears that only 10% of outliers are sufficient to induce a cluster shift and making discrimination of the two classes not possible anymore. The right plot on Figure 2.18 which shows the centroids values and the DoM $|\mu_1 - \mu_2|$ confirms this shift.

While this behavior can be partially mitigated by moving outliers (if identified) toward the median of non outliers points (as pointed out by Perin *et al.* in [PC16]), this is only possible when the amount of outliers is relatively low. Moreover, doing this has the effect of degrading the separability capability of mixture components, as the resulting global distribution would tend toward an unimodal Gaussian distribution when facing samples with a high percentage of outliers or saturated values.

In the end, dealing with outliers and saturated data can be a tricky task without compromising the exploitability of PoI. When high saturation levels are identified, it is sometimes wiser to discard these points, whether in the context of a univariate PoI selection or that of a multivariate analysis. Quantify the quality of a dataset before further analysis is therefore always a mandatory task to ensure correct applicability of subsequent steps.

2.3.4 Compression based features extraction

As noted in the previous chapter, compression-based methods are alternative approaches widely used to extract relevant features in a completely unsupervised context. These methods can be applied either for data visualization or for exploiting high dimensional patterns.

2.3.4.1 PCA as dimension reduction

A very common machine learning algorithm used for dimensionality reduction tasks is the well known PCA [Pea01, Hot33]. The main idea behind the PCA is to reduce the dimensionality of an initial set of data to a smaller representation that would retain the maximum possible amount of variation. The resulting projection then consists of independent orthogonal principal components.

This method is sometimes used after univariate leakage assessment in the case of supervised SCA, but it is more prevalent in unsupervised situations. Indeed, an attacker without insight of the exact location of the leakage (if it exists) would benefit from dimensionality reduction. Since the acquired traces (and resulting patterns) are usually long in length, the application of clustering methods such as k-means is not possible because of phenomena that arise in high dimensional spaces (namely the Curse of Dimensionality [BCC57, Bel61]). Thus, the PCA can be seen as a relevant alternative to univariate PoI selection. PCA assumes that the directions of the projected axes with the largest variances are expected to contain the dynamics of interest. In addition, another advantage in using the PCA method for dimensionality reduction is the fact that it is entirely non-parametric and deterministic.

PCA dimensionality reduction process can be achieved through Eigen decomposition of the covariance matrix or using Singular Value Decomposition (SVD). The latter one is often preferred to compute PCA as it provides better numerical stability than computing covariance matrix for high-dimensional spaces. [1]

Consider an input matrix $\mathbf{A} \in \mathbb{R}^{n \times p}$ with n the number of observations and p the number of observed features. The SVD of input $\mathbf{A}$ is its decomposition into the product of three matrices:

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathsf{T}} \tag{2.10}$$

where $\mathbf{U} \in \mathbb{R}^{n \times n}$ is an orthogonal matrix which columns represent left-singular vectors, $\mathbf{\Sigma} \in \mathbb{R}^{n \times p}$ is a diagonal matrix of singular values $\lambda_1, \ldots, \lambda_p$ with $\lambda_1 \geq \lambda_2 \geq \ldots \geq 0$ and $\mathbf{V}^{\mathsf{T}} \in \mathbb{R}^{p \times p}$ is a transpose orthogonal matrix where columns represent right-singular vectors, that is the principal components or axes.

To apply dimensionality reduction, projection to subspace can be done by computing the dot product between the input matrix and the truncated transpose of the right singular vector:

$$\mathbf{A}_k = \mathbf{A}\mathbf{V}_k^{\mathsf{T}} \tag{2.11}$$

This provides an approximation matrix of rank k. In terms of PCA, this corresponds to the projection into the k ranked principal axes with the highest variances.

After this process, resulting low rank matrix can be used for visualization (considering the first two or three dimensions) or for further analysis.

2.3.4.2 Quantify lost information

It is important to be able to quantify the information loss during the compression process of the PCA. The Explained Variance (EV) ratio is the percentage of the total variance enclosed in the first k selected components. It is calculated from the singular values of the decompo-

[1] Eigen decomposition method requires the computation of the covariance matrix and is limited to square matrices, resulting in a high computation cost of $\mathcal{O}(p^2 n + p^3)$, while SVD can be applied to non-square matrices and provides a lower complexity of $\mathcal{O}(\min(p^2 n, n^2 p))$.

sition. The EV of the k first components is given by:

$$\mathrm{EV}_{1:k} = \frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{p} \lambda_i} \tag{2.12}$$

Similarly, the single EV of the ith component is simply $\frac{\lambda_i}{\sum_{i=1}^{p} \lambda_i}$. Thus, in the process of the PCA compression, if one wants to retain 90% of the variance of the input data, k must be chosen such that $\mathrm{EV}_{1:k} \geq 0.9$. The reconstruction error can also be considered to assess how much information is lost when considering the first k components to represent the entire data. It is computed by inverse transform $\mathbf{A}' = \mathbf{A}_{1:k}\mathbf{U}_{1:k}^{\mathsf{T}}$. The projection error between $\mathbf{A}$ and $\mathbf{A}'$ can then be calculated using the Euclidean distance.

This allows to define a threshold on the number of components k to keep based on the acceptable information loss or variance representation percentage.

2.3.4.3 Impact of noise

To assess the relevance of PCA on high-dimensional data sets, we characterized the effect of noise on the generated projection. In the previous section, we saw that leakage can be very localized in time depending on the target under consideration. Therefore, we simulated the behavior of a leaking sample among several informationless samples. The leaking sample was generated according to the univariate leakage model from equation 2.7, i.e. a bimodal mixture with parameters described in table 2.3.

Parameter	Value
μ_1	-5
μ_2	5
$\sigma_1 = \sigma_2$	1
$\pi_1 = \pi_2$	0.5

Table 2.3: Mixture parameters for PCA experiment

Then, noise dimensions were added, each one sampled from a univariate Gaussian distribution $\mathcal{N}(\mu_{noise}, \sigma^2_{noise})$ in the same range as the previously generated mixture such that:

$$\begin{aligned} \mu_{noise} &= \sum_{i=1}^{k} \pi_i \mu_i \\ \sigma^2_{noise} &= \sum_{i=1}^{k} \pi_i \sigma_i^2 + \sum_{i=1}^{k} \pi_i (\mu_i - \mu_{noise})^2 \end{aligned} \tag{2.13}$$

Details on the calculation of Equation 2.13 can be found in the Appendix A. This procedure has ensured that the generated noise samples cannot be easily separated from the actual leakage. Indeed, in a SCA context, the observed noise samples are usually produced by the same physical process as the leakage.

The experiment was then performed as follows. We gradually increased the number of noise dimensions, p_{noise}, from 1 to 20. For each value of p_{noise}, the synthetic traces were compressed using the PCA from $p_{noise} + 1$ dimensions to 2 dimensions. Figure 2.19 shows the results of the PCA projection on the first two principal components for $p_{noise} = 1, 2, 3$ dimensions. We can see that once more than one noise dimension is added, PCA is unable to successfully project the leakage into a subspace without also including some noise. This behavior was expected since we have defined the noise to be in the same variation range

as the generated leakage. To ensure leakage extraction, leakage variation should be greater than ambient noise or other undesirable behavior.

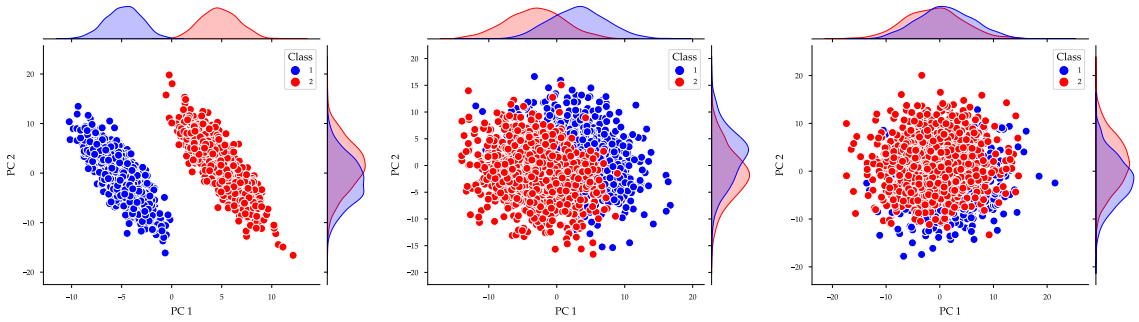

Figure 2.19: Visualization of the first two components obtained by decomposition in principal components of the synthetic datasets with $p_{noise} = 1$ (left), $p_{noise} = 2$ (middle) and $p_{noise} = 3$ additional noise dimensions.

This emphasizes that in a SCA context, where the dimensions of the patterns are on the order of a few hundred to several thousand (or even more), PCA application becomes impractical, especially when the signal-to-noise ratio is low.

2.3.4.4 PCA on considered datasets

To confirm the above conclusion derived from the synthetic data, we applied the PCA compression to the three datasets considered. The first two components obtained by the PCA are shown in Figure 2.20. It appears that the PCA successfully projects the leakage contained in

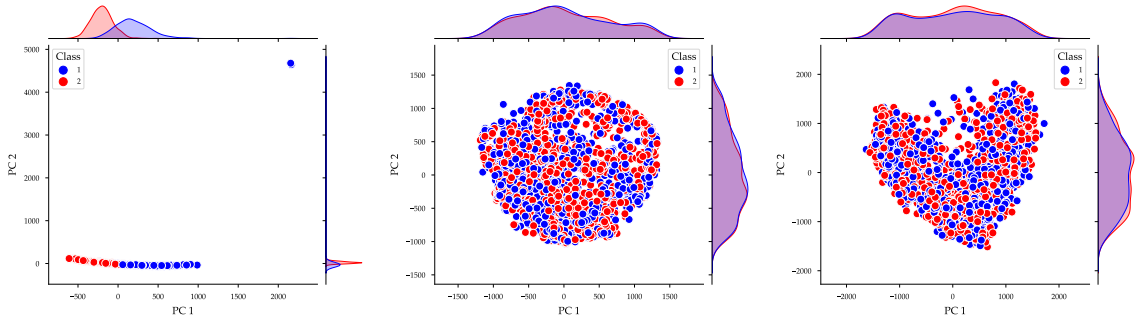

Figure 2.20: Visualization of the first two components obtained by decomposition in principal components of $\mathcal{D}_{mbedtls}$ (left), $\mathcal{D}_{pointer}$(middle) and $\mathcal{D}_{arith}$ (right). For readability purpose, only 2000 points are displayed, sampled uniformly from all patterns. Nevertheless, the PCA is applied on the entire datasets.

$\mathcal{D}_{mbedtls}$ into the first two components. Nevertheless, we noticed a degradation in the quality of the obtained mixtures compared to the univariate PoI selection. On the other hand, PCA fails at extracting relevant leakage enclosed in $\mathcal{D}_{pointer}$ and $\mathcal{D}_{arith}$. Additional inspection of the two first components has shown that there are complex mixtures of Gaussian distributions with almost complete overlap of the two classes.

To get more insight about non visualized additional principal components, we can observe the explained variance ratio of each individual component on Figure 2.21.

Surprisingly, on $\mathcal{D}_{mbedtls}$ dataset, most of the variance is explained through the first two components (more than 70% on the first one), meaning that most of the data can be repre-

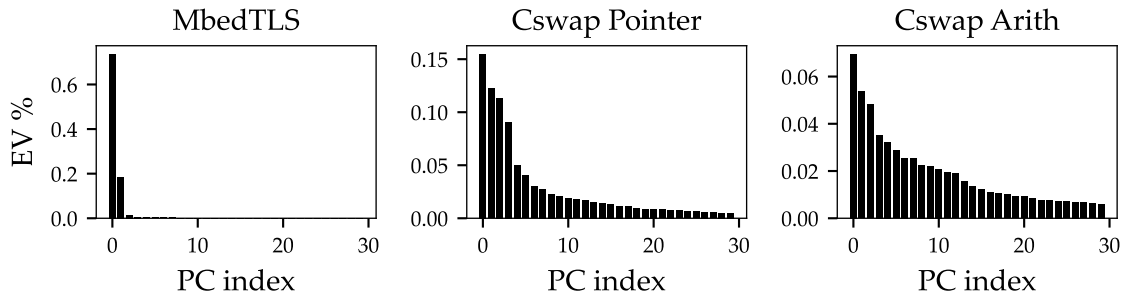

Figure 2.21: Explained Variance ratios of the first thirty components for the considered datasets.

sented through these component, which also implies a high correlation between time points of the original dataset. This is not the case for $\mathcal{D}_{pointer}$ and $\mathcal{D}_{arith}$ where the first Principal Component (PC) accounts for only 15% and 7% of the total variance, respectively.

In addition, on Figure 2.22, we observe the right singular vectors, i.e. the principal axes in the feature space. It allows to see the weight and variance contribution of each time point

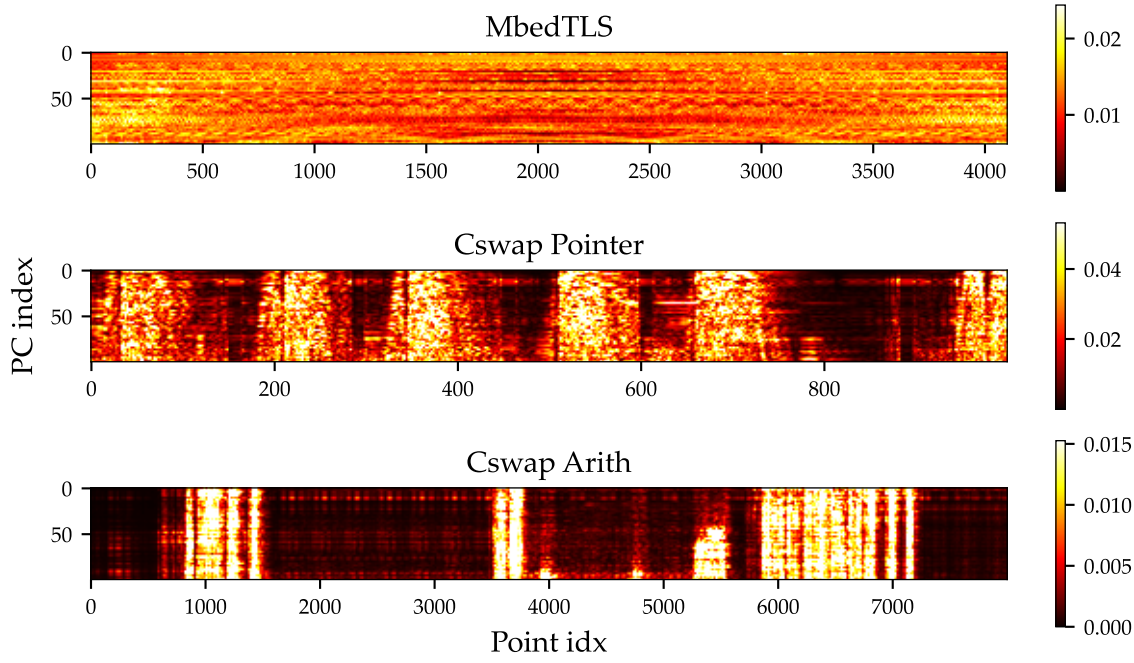

Figure 2.22: Heatmap of the contribution of each temporal point from the input to the first 100 principal components extracted from the PCA.

from the input to the principal components.

For $\mathcal{D}_{pointer}$ and $\mathcal{D}_{arith}$, this is evidence that the first 100 principal components do not contain any relevant information about the leakage. Indeed, if the time intervals with the largest variance contributions (the highest values) are compared to leakage points indices from Figure 2.2, it can be seen that no leakage is represented by these principal components. This confirms that if the leakage regarding the value of d is not prominent in the global variance, it would not be extracted and projected into the first component by the PCA. The PCA is therefore only able to extract information about the leakage of d when applied to

$\mathcal{D}_{mbedtls}$. In addition, the ELV proposed in [CDP16] did not provide better insight about the optimal selection of components in a univariate context, as illustrated on Figure 2.23. While

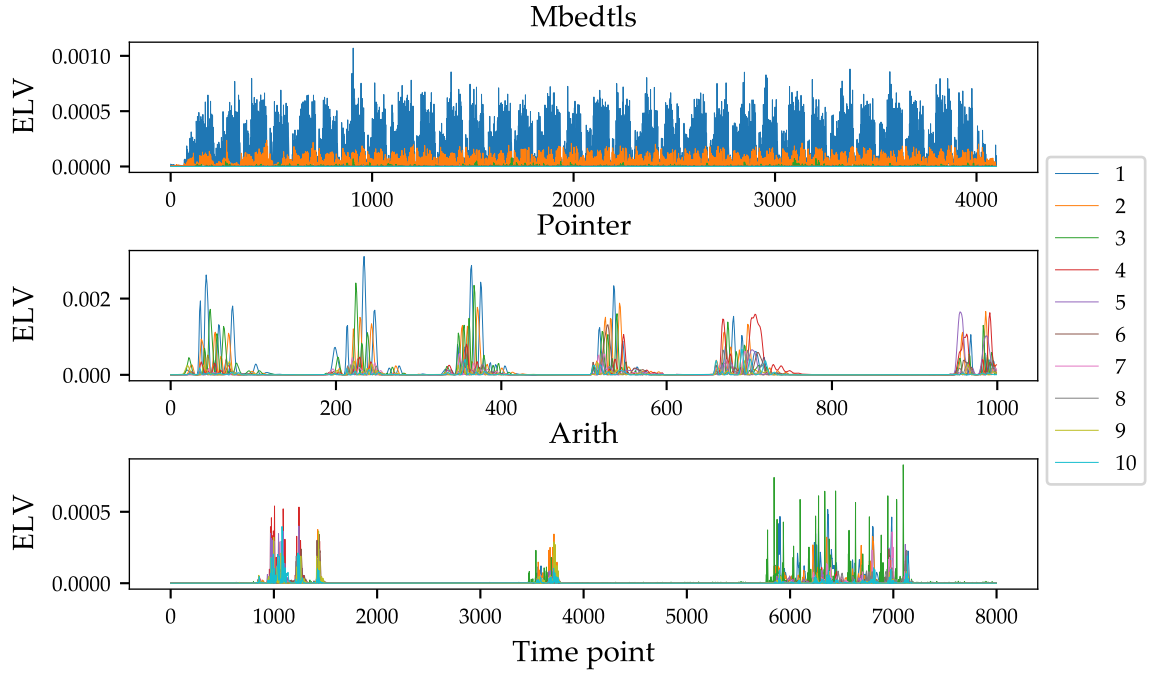

Figure 2.23: Produced ELV for each time point on the first 10 principal components extracted from the PCA.

authors of [BA24] proposed to use the cumulative ELV to determine which principal component reaches its maximum ELV with the minimum of time points and successfully applied this method to the case of the Pointer dataset, it however fails for the Arith and MbedTLS datasets, yielding BRR close to 50%. For reference, Figure 2.24 displays the obtained cumulative ELV for the three considered dataset, where the red highlight curve indicates the selected component in this case.

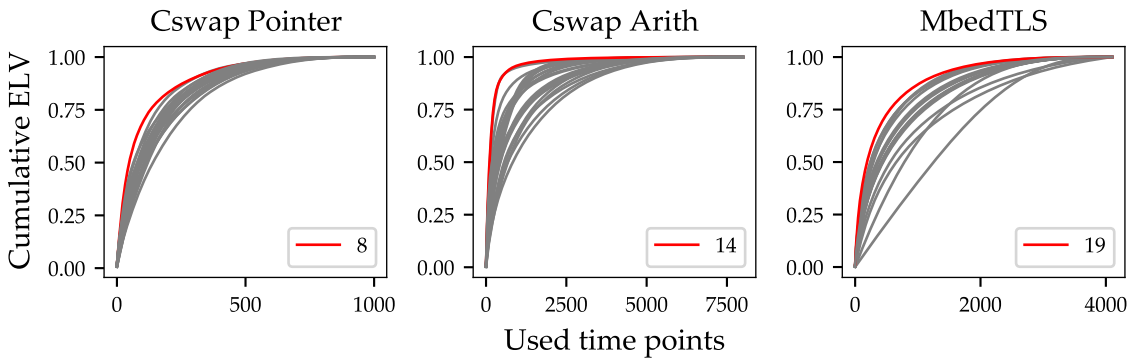

Figure 2.24: Cumulative ELV for each time point on the first 20 principal components extracted from the PCA.

In order to observe how the leakage is distributed among the principal components, we applied a supervised T-test on each of them, independently. The resulting $\mathcal{T}$-values for the three datasets are shown on Figure 2.25. If the leakage is expressed through the variance,

leakages would then appear on the first components. Figure 2.25 shows the resulting $\mathcal{T}$-values on the three datasets. For $\mathcal{D}_{mbedtls}$, high $\mathcal{T}$-values are observed only on the first three

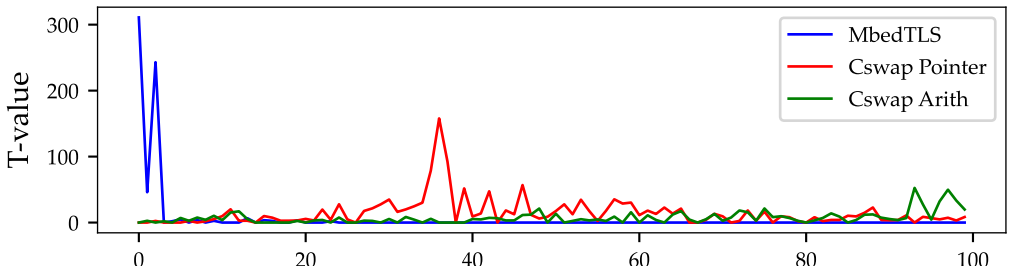

Figure 2.25: T-test on each PC

components: the leakage explains most of the global variance.

However, the obtained $\mathcal{T}$-values are lower than those previously observed during the univariate PoI selection as depicted on Figure 2.3, implying that the PCA projection degrades the leakage quality. For $\mathcal{D}_{pointer}$ set, the manifestation of class leakage only occurs after the first twenty principal components and is distributed over several components. This indicates that there exists no linear correlation between these leakages points in the original dataset, since the resulting principal components of PCA should be linearly independent. For $\mathcal{D}_{arith}$, almost no leakage is detected, except from some peaks near the 100th component. This confirms presence of high level of noise in the dataset and the difficulty of separating classes compared to other sets.

Finally, we wanted to evaluate the quality of the partitioning through clustering on the PCA projection. In an attack scenario, an attacker would typically choose the first k principal components to apply clustering on. In this aim, we apply k-means algorithm by considering the first $k \in \{1, \ldots, 50\}$ PCs. After each clustering, we computed the resulting BRR from the multidimensional clustering assignment. Table 2.4 reports the obtained BRR on the three datasets. For all datasets, it seems that the clustering produces almost the same result what-

	$\mathcal{D}_{mbedtls}$	$\mathcal{D}_{pointer}$	$\mathcal{D}_{arith}$
$k = 2$	86.57%	50.53%	50.14%
$k = 3$	86.73%	50.53%	50.09%
$k = 5$	86.99%	50.35%	50.12%
$k = 10$	87.00%	50.37%	50.14%
$k = 30$	87.05%	50.59%	50.12%
$k = 40$	87.04%	50.57%	50.14%
$k = 50$	87.04%	50.58%	50.15%

Table 2.4: BRR obtained after apply clustering on k first PC.

ever the number of components chosen is. Thus, if the leakage is not contained into the first PCs, selecting a larger number of components does not result in better clustering. Overall, the reported BRR values are lower than those obtained after a univariate PoI selection, as shown in Figure 2.10. An alternative selection method can be to discard the first principal component, if it is assumed to be informationless. The same experiment as above as thus been repeated while omitting the first component. On $\mathcal{D}_{pointer}$ and $\mathcal{D}_{arith}$ datasets, this removal did not have any effect regardless of the value of k. On the other hand, removing the first component on $\mathcal{D}_{mbedtls}$ completely eliminated the leaked information, as the obtained

BRR dropped from more than 80% down to 50%, whatever the value of k considered.

2.3.4.5 Conclusion and limits

While the PCA is a straightforward non-parametric method for features extraction, it is highly dependent on the data quality. It means that the PCA can be sensitive to noise and other behaviors described in the previous sections. This directly affects the quality of the resulting PCA projection and the usefulness of these principal components for class discrimination.

Furthermore, even when trying to use metrics to gain some insight of the variance represented by components or the impact of originals points on each component, it is always assumed that useful information is expected to be contained in the global variance.

2.3.5 Deep Learning based bits correction

As introduced in the previous chapter, the authors of [PCBP20] introduced a deep learning based framework in the context of single traces attacks. Rather than a modification of previously proposed PoI selection methods, this framework can be seen as a bit correction procedure after the prior PoI identification and labeling steps. Consequently, the way the PoI selection is performed will still strongly influence the results of the framework. In fact, since the proposed method implies supervised learning, generalization by learning with incorrect pre-labeled patterns is unlikely to converge towards a better bit prediction.

Extensive analysis of the framework is not carried out in this work as it is considered out of the scope of study. Indeed, this step is posterior to PoI selection and clustering analysis, and used only for improving the bit recovery rate. Still, we report results from the application of this methodology on three considered datasets.

For better insight, let us detail the proposed framework behavior. Consider a starting set D built as a set of patterns T and their actual bit predictions Y from previous steps $D = \{T_i, Y_i\}$ for each pattern $i \in [1, N]$ with N the total number of patterns for all executions. From there, the framework can be divided in four steps.

1. D is split into two subsets D_1 and D_2 of identical sizes, representing the initial training subset and test subset respectively.

2. A neural network is trained to predict Y_1 from T_1 during the training phase. Then during the evaluation phase, the trained network is used to predict values for unseen data T_2. These values are then used to relabel Y_2.

3. The previous step is applied in the opposite direction. That is, a neural network is trained to learn Y_2 from T_2 and then predict using T_1 to relabel Y_1

4. Finally, resulting subsets D_1 and D_2 are combined back to D and then shuffled.

These four steps are applied 30 times in [PCBP20]. It is important to note that for each network training at each step, a new untrained network is used.

Authors of [PCBP20] reported that application of [PITM14] method with [PC16, NC17] improvements yields an average bit recovery rate of about 52% for $\mathcal{D}_{pointer}$ and $\mathcal{D}_{arith}$ datasets (Results can be seen on [PCBP20]). No information is given concerning the number of PoI considered for the scalar recovery step prior to the optimization with the deep learning

framework. However, we were not able to reproduce the same results (100% BRR after optimization) using the same network architecture and methodology for the initial scalar recovery with a starting BRR of about 51%. For comparison, initial attack on $\mathcal{D}_{mbedtls}$ set gives an average BRR of about 95% on the 50 first selected PoI. Using the same network architecture as for $\mathcal{D}_{pointer}$, this allows to recover the remaining bits until 100%.

While Perin *et al.* approach for improving the bit recovery rate after the initial attack can be relevant in some contexts, it is too complex and dependent of the initial predictions. Recently, a new contribution showed that the considered architecture in [PCBP20] is oversized for the treated problem and can be reduced to a single perceptron classifier [BA24]. Moreover, the tweaking of hyperparameters is not possible in an unsupervised attack context and no metric is provided to assess the quality of the framework optimization. Finally, this approach assumes a sufficient information leakage present in the selected PoI, thus assuming an efficient PoI selection, that is not always the case when highly noised datasets are considered.

2.4 Conclusion and Problematic

In this chapter, we studied several approaches currently used in the state of the art for selecting relevant features in the context of unsupervised horizontal attacks.

In the univariate analysis, we studied the methods used in supervised approaches to see how they can be adapted in an unsupervised context. Experiments showed the limitations of the used metrics, such as the T-test statistic when used in unsupervised conditions. Furthermore, we characterized the impact of noise and outliers on the leakage and on the clustering process.

In multivariate feature extraction, we showed the limits of used methods for dimensionality reduction in the presence of noisy data. This also implies the quality of targeted datasets to ensure the best possible leakage extraction.

Within context, emphasizing the limits of actual methods for performing unsupervised single traces attacks, the global problematic of this work can be formulated as follows.

First, the selection of relevant features from obtained patterns is prevalent. Indeed, as presented before, state of the art currently considers either patterns compression approach using dimensionality reduction methods such as the PCA, or univariate PoI selection approach, mostly based on [PITM14] work and improvements.

The second category is about the quality of the signal. Due to several reasons, the quality of the performed acquisitions can vary and emphasize noise, saturation, outliers or other behavior that can directly affect the analysis quality and success. The goal here is to study if some network architecture can be fit to mitigate this issue and thus have a positive impact on the analysis results.

Finally, it is also question to study neural network as an alternative to clustering distinguisher. Indeed, while the success of the attack mainly depends on the signal quality and PoI selection phase, an adapted distinguisher is mandatory for efficient bit recovery stage.

While the contribution of [PCBP20] is a relevant extension of [PITM14] work and other improved methods such as [PC16, NC17], it still relies on a sufficiently correct PoI identification during the first phase. The goal of this work is to propose a more robust methodology based on the three aforementioned key that would define the success of horizontal attacks. In this work, this is done through two different approaches.

The first one consists in the study of neural based architectures that could improve the

performance results obtained with state-of-the-art methods. One would like to know if it exists some neural network architecture, able to be trained in a completely unsupervised manner, which could be an alternative choice over currently used methods. Furthermore, the performance of the identified alternatives should be measured and compared to traditional methods to ensure their relevance in the context of horizontal attacks.

The second approach instead consider statistical testing using traditional machine learning methods. Based on previous contribution, it aims at redefining the underlying leakage model in a statistical way, taking into account noise, saturation and other additional effects.

Chapter 3

Neural Network Approach

Abstract

This chapter studies the application of unsupervised neural network architecture to tackle the two most prominent issues of horizontal side channel attacks (ensure exploitable data quality and provide correct leakage extraction). The usage of autoencoders are firstly considered as an alternative for patterns compression. Then, two other architectures are applied for enhancing the quality of input patterns. Finally, an alternative approach is studied for compression and visualization purposes.

3.1 Introduction

This chapter is dedicated to the evaluation of some unsupervised neural networks based methods for improving the performance and success capability of horizontal attacks. Indeed, the rise of deep learning and neural networks in various domains has showed to provide improved performances over some traditionally used methods. Even in the field of SCA, neural networks architectures are now almost systematically used for performing profiled attacks. Thus, we would like to consider neural networks in the context of unsupervised horizontal attacks. Since horizontal attacks are carried out in an unsupervised manner, architectures or training mechanisms used in profiled attacks cannot be applied in these attack contexts.

To that aims, this Chapter is organized as follows.

In Section 2, we give an overview of the deep learning domain, the optimization problem, components for building architecture and the different learning paradigms.

Section 3 explores the usage of neural architectures for compression (namely the autoencoders) as an alternative to the traditional PCA widely used in the context of SCA. It also addresses the problem of outliers in the data and how they can be mitigated using unsupervised learning.

Then, other approaches based on generative models are considered as an alternative to compression models for the task of outliers mitigation.

Finally, we explore the use of unsupervised neural networks for compression, classification and visualization as an alternative to classical clustering algorithms.

In the end, a conclusion is drawn about the potential and limits of neural based solutions for the improvement of unsupervised horizontal attacks.

3.2 Deep Learning

Deep learning is a branch of machine learning based on the use of artificial neural networks for optimization problems. The principle of artificial neural networks is inspired from information processing in biological systems. Unlike machine learning which consists in statistical algorithms and necessitate feature engineering to exploit the data it is fed, neural networks are able to extract high level features directly from the raw input. Neural networks usage can be found in various applications in many different domains such as image processing, robotics, finance, medical diagnosis... The fact that they are able to approximate any function make them powerful in various discipline. Especially, several architectures coming from the image processing domain have been successfully applied to SCA tasks [PSK+18, MDP19, KPH+19, CCC+19, ZBHV20, BPS+20].

In this section, we recall the main principles of neural network optimization problems and commonly used architectures.

3.2.1 Network topology

Most neural architectures can be built from the common network building blocks that we describe hereafter.

Multilayer Perceptron MLP is the fundamental block for building complex neural networks architectures. They consist in a series of fully connected layers from a finite number of nodes and associated weights $W^{(i)}$ for each layer. During the training forwarding pass, the input data is fed through subsequent connected layers. Each single layer computation can be represented as a linear combination of the input (or the results from the previous layer) with the weights, followed by an activation function. The produced final output is the result from the last fully connected layer.

Convolutional Layer Convolutional Neural Network (CNN) is a network architecture widely used in image processing tasks. Its relevance arises in cases when fully connected architectures are impractical to apply as the size of the input is too big. CNN can be seen as a learnable process for feature engineering. It consists in a sequence of filters (with associated weights) used for features extraction. More precisely, a convolution layer is defined by a kernel (or filter) of defined size, which slides along the input matrix to produce a new feature map. An example of a one dimensional convolution is illustrated on Figure 3.1. Ad-

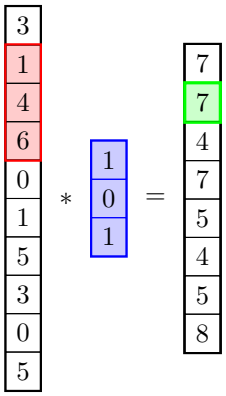

Figure 3.1: Example of a one dimensional convolution.

ditionally, pooling layers are added for dimensionality reduction by combining the previous layers output. Convolutional blocks are flattened at the end of the feature extracting process to be fed to fully connected layers. When applied to SCA, most commonly used CNN are often based on the VGG architecture, which is widely used for image processing tasks [SZ15].

3.2.2 Activation functions

In order to solve non-trivial problems, activation functions (in their non-linear form) should be applied between each layer. The use of activation functions, when they are non linear, gives to neural networks the potential to become universal function approximators [Cyb89, Hor91].

However, they must satisfy several properties. They should be monotonic functions to preserve direction changes between the input and output. In addition, they must be continuously differentiable to allow back propagation using gradient-based methods. Furthermore, to avoid problems such as exploding or vanishing gradient in deep architectures [BJZP20], activations functions whose derivatives are less prone to saturation are often sought. Table 3.1 lists the most commonly used activations functions. It should be noted that the usage

Name	Formula	Range
Identity	x	$(-\infty, \infty)$
Sigmoid	$\frac{1}{1+e^{-x}}$	(0, 1)
Tanh	$\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$	$(-1, 1)$
ReLU [NH10]	$\begin{cases} 0 & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases}$	$[0, \infty)$
ELU [CUH16]	$\begin{cases} \alpha(e^{x}-1) & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases}$	$(-\alpha, \infty)$

Table 3.1: Activation functions

of the identity function as activation function has the effect of reducing a multilayer network to a single linear layer and its capability to a linear classifier. Figure 3.2 shows the behavior of the activation functions over the interval $(-10, 10)$. The choice of the activation functions thus depends on the input range and desired output distribution. In this work, ELU and Tanh activations are mostly used, for hidden layers activations and output scaling respectively.

3.2.3 Optimization algorithm

The optimization of neural networks is carried out by adjusting the model weights until a convergence criterion is reached. This is done through alternating two phases.

The forward-propagation step consists in feeding the input data through the network. The obtained output from linear combination of neuron weights with input values plus a non linear activation allows the generation of an error (or loss), based on a defined objective.

Then, during the back-propagation step, the gradient of the loss function with respect to the model weights is calculated, and weights updated to minimize the error in the next

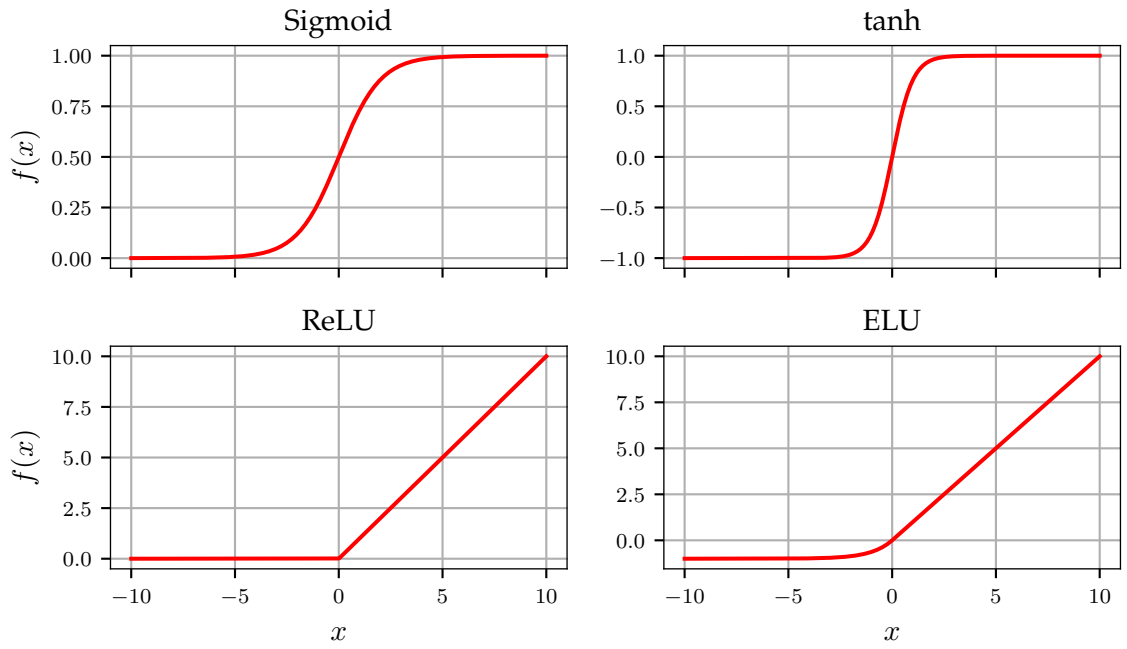

Figure 3.2: Illustration of main activation functions

cycle. The rate at which weights are updated based on the gradient direction is defined by the learning rate step.

The data is usually processed in batches of defined size, and the network weights updated at each batch. A complete cycle, i.e. the processing of all batches, is called a training epoch.

Note that other optimization methods other than error learning exist such as competitive learning, where candidates compete to represent a portion of the input data.

3.2.4 Learning paradigms

To fully understand the problematic of the so-called optimization problem, we can enumerate the three main learning paradigms on which neural networks learning paradigms are based and the problem they address.

Supervised Learning In supervised learning, the available data usually consists in the input X and output pairs n elements $\{(x_i, y_i)\}_{i=1}^{n}$. X corresponds to the observed data while Y is the ground truth that the network try to predict from the observation of X. After training convergence, the inference step is used to provide the trained model with unseen data. Analogous to the context of SCA, the learning phase is equivalent to the profiling phase, while the prediction phase is the matching or attack phase.

Unsupervised Learning In opposition to supervised learning, when the data consists only in n values $\{x_i\}_{i=1}^{n}$, only unsupervised learning can be performed. Since no ground truth is provided, it is necessary to quantify the model error using an alternative metric. Depending on the context, various objective can be used. Note that self-supervised learning is a subset of unsupervised learning, where part of the input can be used as supervision [HCL06, JT19].

Reinforcement Learning Unlike other learning paradigms, reinforcement learning does not rely on the presence of static input (and possibly output) data. Instead, it can be modeled as a Markov decision process $(\mathcal{S}, \mathcal{A}, \mathcal{P}_a, \mathcal{R}_a)$, where each term respectively denotes the state of the environment, the state of the actions, the set of transition probabilities between states and the associated rewards. The optimization is performed by an agent interacting with its dynamic environment and taking actions in order to maximize its reward [KLM96].

3.3 Autoencoders

In this section, we study the application of autoencoder as candidates for two distinct tasks. First, they are compared to the Principal Component Analysis and display their added value. Second, autoencoders are also considered in a way to mitigate outliers or saturated values present in the data.

The term autoencoder denotes a neural network architecture that is trained to learn the identity function. It has first been proposed in 1991 by Kramer *et al.* as a non-linear generalization of the PCA [Kra91]. Autoencoders are applied in many domains such as digital image processing, medical, telecommunications, etc.

Some works have shown that autoencoders (in their non-linear form) provide some benefits over the PCA, especially to extract non-linear relationships [HS06, BCV14, GBC16].

3.3.1 Network architecture

Autoencoders are built from the union of two siblings networks, usually referred as the encoder and the decoder. The encoder network $\mathcal{E}_\phi : \mathbb{R}^m \to \mathbb{R}^p$, maps the input x to a latent space representation $z = \mathcal{E}_\phi(x)$ (also traditionally denoted as the code or bottleneck). If $p < m$, this is known as an undercomplete autoencoder and is usually used for compression purposes. The decoder $\mathcal{F}_\theta : \mathbb{R}^p \to \mathbb{R}^m$ learns the inverse mapping from z back to x, known as the reconstruction $x' = \mathcal{F}_\theta(z)$. The objective of an autoencoder is to learn to represent the input data in a lower dimensional space. The loss of the network is given by the reconstruction error between x and its resulting reconstruction x'. In a general way, it can be defined as:

$$\mathcal{L}(\theta, \phi) = \mathbb{E}_x \left[d(x, \mathcal{F}_\theta(\mathcal{E}_\phi(x))) \right] \tag{3.1}$$

where d is an arbitrary distance metric. The autoencoder optimization aims at finding network parameters θ, ϕ that minimize the loss $\operatorname{argmin}_{\theta,\phi} \mathcal{L}(\theta, \phi)$. This optimization phase can be carried out by gradient descent. Several types of architecture and variation of autoencoders exist for different usages. In its simplest version, the autoencoder can be formed from a single hidden layer in the encoder and decoder:

$$\mathcal{E}_\phi(x) = \sigma(W^{(1)}x + b^{(1)}) \tag{3.2}$$

$$\mathcal{F}_\theta(z) = \sigma(W^{(2)}z + b^{(2)}) \tag{3.3}$$

If the chosen activation σ is the identity function, this is known as a linear autoencoder. Figure 3.3 shows the example of an autoencoder architecture using fully connected layers.

3.3.2 Autoencoder as an alternative to PCA

The linear autoencoder architecture is closely related to the inner workings of the PCA. Indeed, the linear autoencoder weights span the same subspace as the principal components

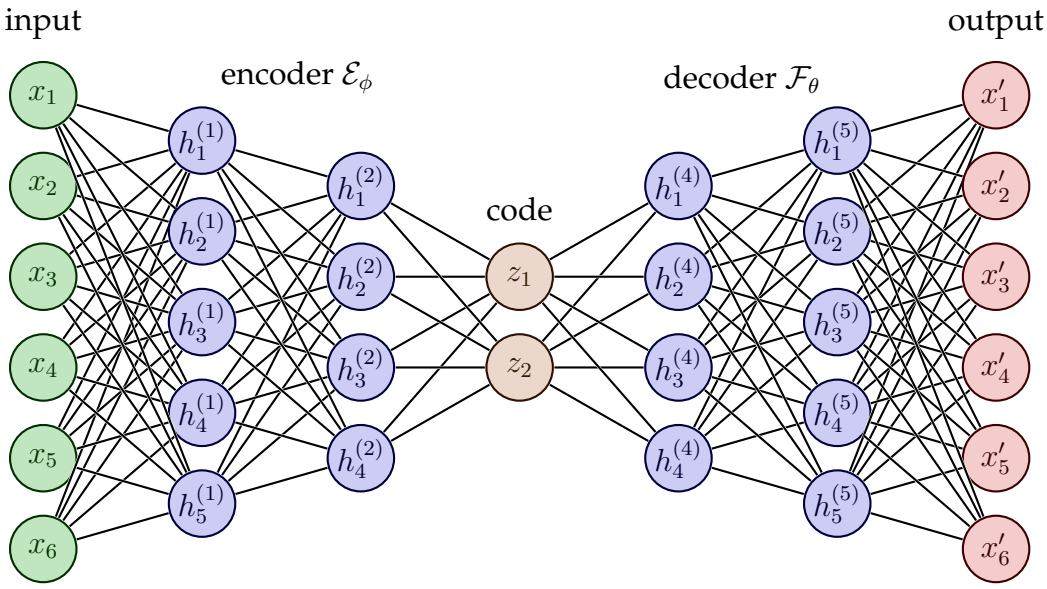

Figure 3.3: Autoencoder architecture illustration using fully connected layers.

from the PCA [Pla18]. However, they exhibit some differences. First, while PCA produces orthogonal and thus independent components, this is not the case for the autoencoder, where bottleneck features can be correlated. Moreover, PCA components are sorted in descending order of variance. This means that choosing the k first dimensions of the resulting projection ensures reliable solution for representing the variability of the input data. This is not the case for the autoencoder, since there is no notion of ranking between features. However, these differences do not affect the compression capability of either method, considering the same code size is used.

On the other hand, the addition of non-linear activation functions has the effect to break the linearity [JJHG00]. This allows learning non-linear transformations. A non-linear autoencoder provides better reconstruction during the learning, as it is capable of non-linear reconstruction of the feature space. It can also extract non-linear dependencies between features. In cases where there exists non-linearity relationships between input features, a non-linear autoencoder would be able to encode more information using fewer dimensions.

Thus, the idea of this section is to investigate in practice if the use of a non-linear autoencoder provides some benefits over the PCA or a linear autoencoder. This will be done by analyzing the quality and exploitability of resulting projections from the two considered methods, as well as the characterization of information and leakage loss in the process.

The autoencoder network architecture we considered for this experiment was defined as follows. Both the encoder and decoder were built from a sequence of three fully connected layers. The size of each layer in the encoder was set to the half that of the preceding. The reverse process was used to design the decoder to get an output equal to the size of the input of the decoder. If the considered autoencoder was linear, the activation function of each layer was simply the identity function. Otherwise, we considered in this work the use of the ELU function as non-linear activation. The layers weights were initialized using the Xavier uniform distribution [GB10] and the biases were set to zero. The Mean Squared Error (MSE) was chosen as the distance metric d, for the computation of the reconstruction error (from Equation 3.1). The network gradient optimization was done using the Adam algorithm [KB17] with a learning rate set to $\alpha = 10^{-4}$. Different code sizes $p \in \{1, 5, 10, 15, \ldots, 100\}$ have been considered to compare both the compression behavior and the efficiency between the

autoencoder, the linear autoencoder and the PCA. While it is straightforward to select the k most important components in the case of the PCA (see Equation 2.11), the autoencoder has to be trained again using a new network with a different code size each time.

3.3.2.1 Correlation between projected dimensions

As said before, obtained dimensions (or components) from the application of the autoencoder (linear or not) are not orthogonal. This means that they can be correlated in some ways. Figure 3.4 shows, for the three datasets considered, the pairwise Pearson correlation for each code dimension / component pair for a considered code size $p = 100$. As expected,

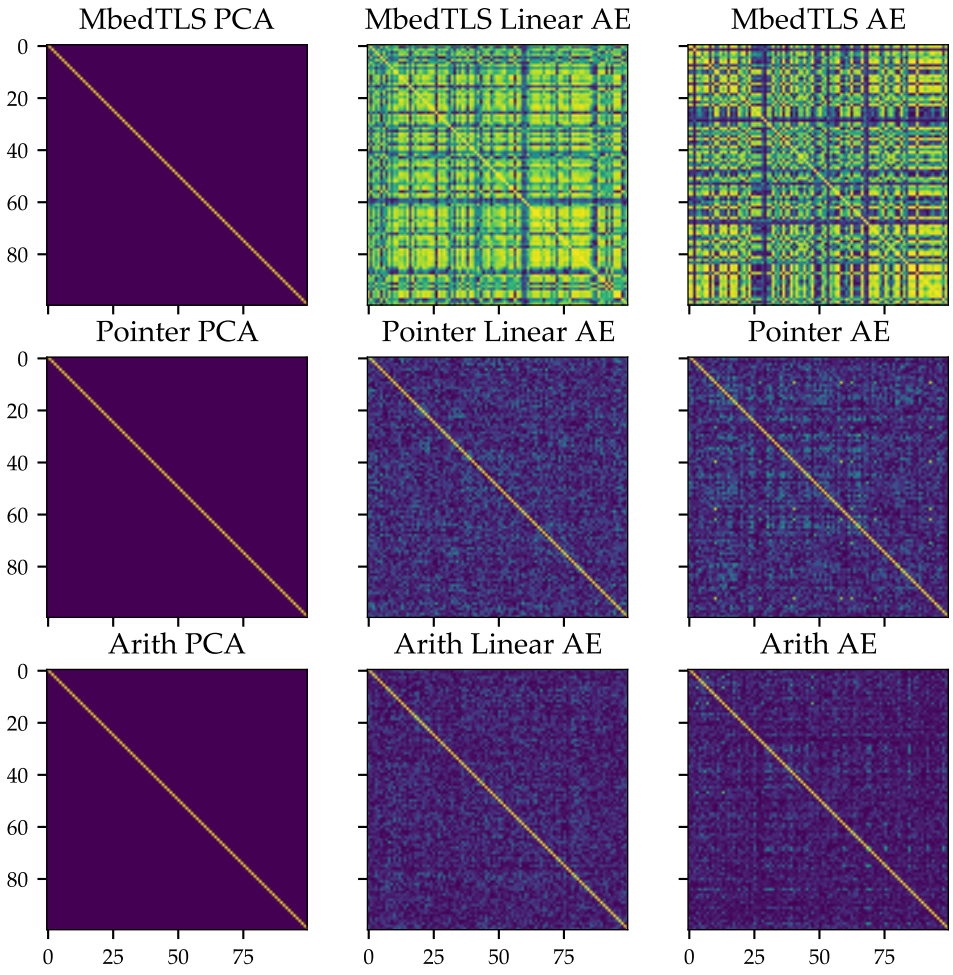

Figure 3.4: Pairwise Pearson correlation

the principal components found by PCA are completely independent of each other, regardless of the data set considered. In comparison, those found by feeding a linear or a non-linear autoencoder with $\mathcal{D}_{mbedtls}$ dataset are strongly correlated. The same behavior is observed for the other two data sets, but the correlation values are weaker. As a result, if an adversary were to perform a leakage assessment on each component / dimension in a supervised manner, he could observe its presence on many of them, especially on $\mathcal{D}_{mbedtls}$. The fact that, for

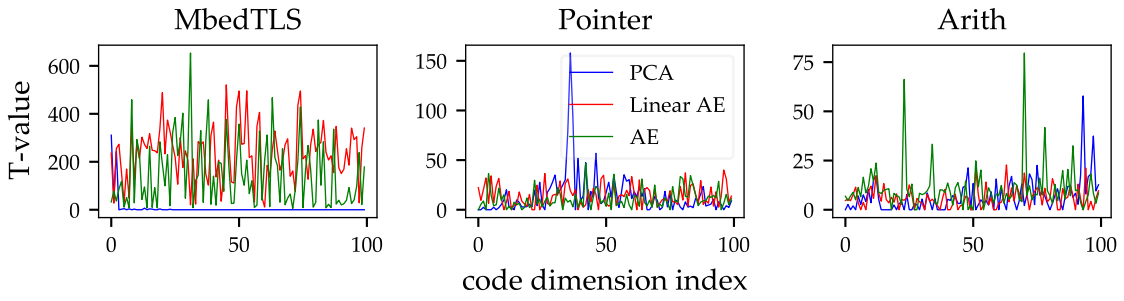

Figure 3.5: $\mathcal{T}$-values obtained in a supervised manner for each dimension of the projected subspace with using either a linear autoencoder, a non-linear autoencoder or a PCA. From left to right the results for $\mathcal{D}_{mbedtls}$, $\mathcal{D}_{pointer}$ and $\mathcal{D}_{arith}$.

$\mathcal{D}_{mbedtls}$, the leakage is not bound to a few components, as it is in the PCA projection, seems to give an interesting distribution of the leakage on each dimension and even higher leakage values. For comparison, it allows recovery of the same leakage magnitudes ($\mathcal{T}$-values) as supervised univariate leakage analysis (see Figure 2.2).

This suggests that the use of the generated projection in a clustering process should be done without dimension selection, assuming that leakage is diffused across many dimensions. Conversely, this implies that the selection of some dimension among all, has not a real impact on the clustering performed during an attack.

However, this observation does not hold for $\mathcal{D}_{pointer}$ and $\mathcal{D}_{arith}$. In the end, both linear and non-linear autoencoders do not seem to provide a more usable representation for conducting an attack than the true measures.

3.3.2.2 Data transformation and information loss

Several approaches can be used to quantify the information loss resulting from the compression process. One way, as described in the previous chapter, can be to compute the reconstruction loss between the true input pattern and their reconstructions. This is what the optimization of the autoencoder is based on. However, this only reflects the global deviation of values between input and output and has no direct relation with the underlying leakage. For example, an encoder/decoder that always produces the average pattern among all patterns used for training would in fact only retains a reduced and really specific part of the information that does not include the leakage.

If we consider that we are in a study context (not a real attack case), where associated patterns labels are available, we can quantify the information that is lost in the process by applying a leakage assessment (TVLA) on the reconstructed patterns. The difference between the initial and reconstructed leakage describe the leakage loss. One could think that it has a close relationship with the reconstruction error, since that if it is close to zero, meaning if there is almost no changes on the global distribution of time points, the univariate leakage should show no degradation.

However, this is not the case. For example, consider a leakage time point with the presence of leakage, describing a bimodal mixture distribution of two components. If the reconstruction displays the same distribution, but all points are now shuffled, the reconstruction error for this point would be equal to 0, despite the fact that the leakage completely disappeared. As a result, while this metric is relevant for assessing whether the network

converged during his training, this cannot be used as a way of quantifying leakage conservation.

A more general and robust way of quantifying the information contained in the code about the value of the labels is the mutual information, the value of which does not depend on any assumption about the link between the data and the labels. The mutual information between two random variable X and Z can be formulated as the Kullback-Leibler divergence [Kul97] between their joint distribution and the outer product of their marginal distributions:

$$I(X,Z) = D_{KL}\left(P(X,Z)||P(X)\otimes P(Z)\right) \tag{3.4}$$

where the Kullback-Leibler divergence is given by $D_{KL}(P||Q) = \sum_{x\in X} P(x)\log\frac{P(x)}{Q(x)}$. However, traditional methods for estimating the Mutual information (using histograms [DV99], kernel-density estimators [MRL95], maximum-likelihood estimators [SSSK08], ...) are prone to the curse of dimensionality. This is due to the fact that it requires the estimation of the joint and marginal density that becomes challenging on high-dimensional data.

An alternative approach would be to calculate the Mutual Information (MI) between each input point and each code point in a univariate manner. As comparing individual components across approaches is not relevant since autoencoder components are not ranked by importance unlike the PCA, we want to compare the global information loss or gain on the whole code result by changing the code size on both approaches. Additionally, such an approach does not provide a complete estimation of the total information and remains difficult to interpret.

Instead, in order to characterize the mutual information for the pair $\{(z_i, y_i)\}$ in a multivariate way, we used the MINE framework [BBR+18]. Mutual Information Neural Estimator (MINE) uses a neural network with gradient descent optimization to provide a lower bound estimation of the mutual information that is linearly scalable to high-dimensional data. It has been previously applied to SCA and has shown to be a relevant method for high dimensional mutual information estimation [CLM20]. The neural mutual information is given by:

$$I_\theta(X,Z) = \sup_{\theta\in\Theta} \mathbb{E}_{P(X,Z)}[T_\theta] - \log(\mathbb{E}_{P(X)\otimes P(Z)}[e^{T_\theta}]) \tag{3.5}$$

where T_θ is a family of functions which is parameterized by a neural network. It defines the best lower-bound estimation of the mutual information such that:

$$I(X,Z) \geq I_\theta(X,Z) \tag{3.6}$$

Using this approach, after obtaining resulting code projections using the three methods considered (PCA, linear autoencoder, non-linear autoencoder), we calculated the MI between known labels and their code. This was repeated several times with different code sizes. The architecture of the neural network T_θ was composed of two hidden layers of 10 neurons, followed by a non-linear activation (ELU). The output consisted of a single neuron (scalar) representing the MI estimation. The optimization procedure was the following. First, b sample pairs were drawn from the joint distribution $\{(z_i, y_i)\}_{i=1}^b \sim P(Z,Y)$ and from the marginal distribution $\{\overline{z}_i\}_{i=1}^b \sim P(Z)$. Then, the lower bound $\mathcal{V}(\theta)$ from Equation 3.5 was computed using pairs (z_i, y_i) for the left term and pairs $(\overline{z}_i, y_i)$ for the right term. After this, the gradients were computed and network parameters $\theta \leftarrow \theta + \nabla_\theta\mathcal{V}(\theta)$ updated. This was repeated until convergence of the MI estimation.

Figure 3.6 shows, for the three datasets considered, the obtained MI after the convergence of MINE on compressed patterns with different code sizes. The results on $\mathcal{D}_{mbedtls}$ are

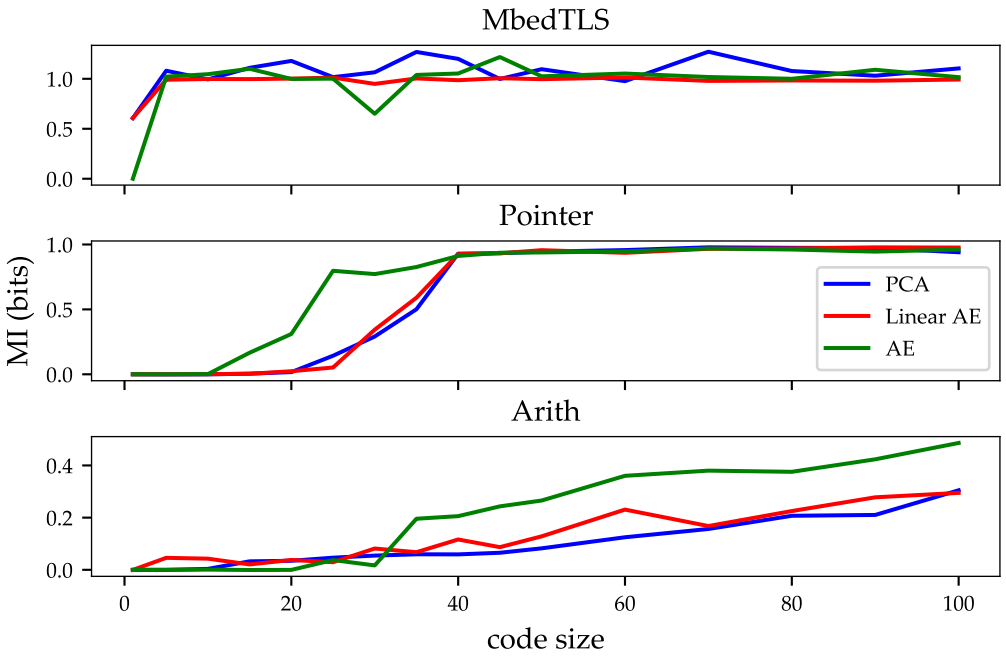

Figure 3.6: Mutual Information between the code and labels for different code sizes.

similar regardless of the method used. This emphasizes that the leakage is only linear for this dataset. Indeed, the non-linear autoencoder, which is capable of extracting non-linear manifolds, does not provide better leakage preservation, regardless of the code size chosen.

Otherwise, it can be observed that the usage of the non-linear autoencoder has benefits on $\mathcal{D}_{pointer}$ and $\mathcal{D}_{arith}$. For the former, this allows to retain more information with a smaller code size. For the latter, increasing the code size leads to more information encoding, resulting in higher information conservation. In fact, with the PCA or the linear autoencoder with code size $p = 100$, the MI obtained between the labels and the projections is about 0.3 bits. Such an MI value is already achieved with the non-linear autoencoder with a code size of only 50. This also emphasizes the presence of non-linear correlations among time points.

This shows several advantages. First, the usage of an autoencoder with non-linear activations does not have negative impact in the information retention during the compression, since its capacity is higher than linear autoencoders. Secondly, the ability to represent more information using fewer dimensions is relevant for the clustering phase. This will be discussed in the next section.

Overall, as expected, on all targets, no significant difference is observed between the PCA and the linear autoencoder, meaning that both methods can be used interchangeably. Still in practice, the use of the PCA should be preferred over the linear autoencoder, since its transformation is non parametric and deterministic, and components sorted in a known importance order.

3.3.2.3 Clustering and projections

If PCA and autoencoder can be used to reduce the dimension of patterns when performing clustering based horizontal attacks, one can wonder how this pre-processing alters the efficiency of the clustering step of such attacks. Despite some approaches consider some heuristics or metrics such as explained variance in the context of PCA, in practice, the number of dimensions or PoI chosen for the clustering step of horizontal attacks is often arbitrary.

However, when dealing with autoencoders, there is no apparent reason in filtering the obtained code to select only some dimensions. Indeed, the chosen code size is meant to be exploited in further analysis as it is, and there is no notion of ranking among the dimensions produced. In addition, dimensions can be correlated and thus information may be diffused, unlike in the PCA. Consequently, clustering should be applied directly on the entire code.

We applied the Fuzzy k-means on the projections provided by both the PCA and the two autoencoder architectures. The averaged BRR obtained are plotted in Figure 3.7 with respect to p, which represents the code size when considering the autoencoder or the number of components retained when dealing with the PCA.

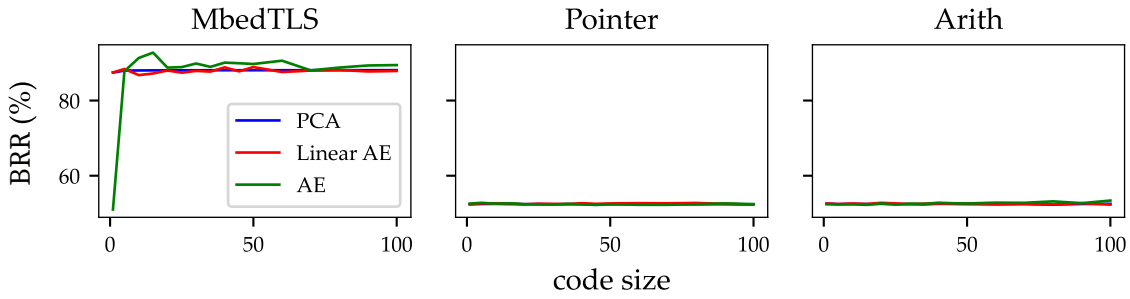

Figure 3.7: Averaged BRR vs p obtained, for the three datasets considered, by applying the Fuzzy k-means to the projections /codes provided by the PCA, linear autoencoder and non-linear autoencoder.

On $\mathcal{D}_{pointer}$ and $\mathcal{D}_{arith}$, no method was able to provide a BRR higher than a random classification, regardless of the value of p. This means that given the current input patterns, no projection is found that makes the leakage distinguishable by the Fuzzy k-means clustering. The opposite results were obtained for $\mathcal{D}_{mbedtls}$. Regardless of the value of p considered, the PCA and autoencoder (linear or non-linear) easily separated patterns according to the secret exponent bit, except for the $p = 1$ when using the non-linear autoencoder. It is unclear why the autoencoder in its non-linear form was not able to present a relevant projection when considering a code size of $p = 1$. We could not find a clear explanation for this behavior.

For visualization purposes, a t-SNE dimensionality reduction [vdMH08] was applied to each resulting code / projection. The t-SNE consists in a non-linear embedding, that allows to high dimensional data to be represented in a two or three dimensional space. The resulting projection cannot however be used for clustering purposes, since its projection does not preserve distance between points from the input space. Still, it can be useful to observe the modeling of similar objects into lower dimensional space, that is, if the probability densities of leakage classes at the output of the PCA, linear autoencoder and non-linear ones are the same with the two-dimensional projection of the t-SNE.

If we can observe separated clusters through the projection, this would mean that the method used for the dimensionality reduction is able to successfully extract the leakage (or a part of it) contained in original patterns of size m to the reduced code of dimension p. It would also mean that the use of the Fuzzy k-means was able to provide correct partitioning and is thus adapted for clustering the output of non-linear dimensional reduced subspaces. However, if this is not the case, this would imply that information about the leakage has been lost during the projection.

Figure 3.8 shows the resulting projections into two dimensional space using t-SNE for code sizes $p \in \{5, 20, 50\}$, with coloring added using known labels to identify scalar or expo-

nent bit values. The application of the t-SNE on the $\mathcal{D}_{mbedtls}$ dataset (left column) shows that the projection is able to separate both classes. Regardless of the value of p, and method used, it clearly captures the leakage into well separated parts of the two dimensional space. This explains the successfully clustering using the Fuzzy k-means algorithm and the obtained high BRR.

But for $\mathcal{D}_{pointer}$ (middle column) and $\mathcal{D}_{arith}$ (right column), two behaviors are also projected but not related to the leakage value for all p code size. Whatever the chosen code size or considered dimensionality reduction method, no projection results in exploitable clustering analysis. Since we are sure that there is some leakage in these datasets, we can think that other and stronger behaviors have driven the projections toward hardly exploitable results from a SCA point of view. As a result, the BRR obtained are very close from 50%.

This suggests that in cases of low leakage levels, the presence of outliers or highly saturated data, autoencoders, even if they are able to catch non-linear relationship between input dimensions unlike the PCA, were unable to provide relevant representations of the data in a SCA context. This emphasizes that outliers and saturated data are a major obstacle to provide successful leakage extraction in multivariate analysis by following a dimension reduction approach, as it is the case for univariate PoI selection methods.

3.3.2.4 Alternative architectures

Other variations of the classical autoencoder architecture have also been considered for dimensionality reduction. However, no satisfactory results have been obtained from these alternative candidates when applied on noisy datasets. For the sake of completeness, we briefly describe these architectures hereafter.

Variational Autoencoder Introduced by Kingma *et al.* in 2013, variational autoencoder is a probabilistic generative model [KW22]. Although it shares a similar architecture with classical autoencoders (which can be seen as a non-linear extension of the PCA), it differs in its mathematical formulation. Indeed, the generative encoder is able to output samples following a Gaussian distribution for each code dimension, this is done through sampling using two additional layers, representing the mean and variance of said code dimension. In some applications, the variational autoencoder have showed to be superior in extraction capability and generalization compared to non-probabilistic autoencoders. However, in our case study, it did not provide better results than classical autoencoder architectures. Furthermore, it requires additional parameters tuning, such as a regularization parameter used to balance the loss terms (see [KW22] for details), it is thus hard to apply in an unsupervised horizontal context.

Gaussian Mixture Variational Autoencoder The Gaussian Mixture Variational Autoencoder (GMVAE) is an extension of the VAE, where each code dimension is sampled from a Gaussian mixture instead of a single Gaussian distribution [DMG+17]. Furthermore, obtained values from the sampling of the categorical latent variable can be seen as a replacement to the clustering step, thus the GMVAE allows obtaining classes assignment during the optimization. This however results in a more complex architecture, which is hard to optimize especially in an unsupervised context. Indeed, the proposed lower bound objective implies four terms and three regularization parameters, making the network optimization and convergence cumbersome. Even when applied on simple cases, this produced low clustering accuracy rates and inconsistent results. This makes this considered architecture, while

interesting in its formulation, inapplicable to the problem of leakage extraction and clustering in a complete unsupervised context.

3.3.2.5 Conclusion and limits

This section explored the use of autoencoder based architectures for dimensionality reduction purposes, as an alternative to the traditional PCA method. While they showed some benefits for better information retention using smaller code sizes in the presence of non-linear relationships, it did not result in better compression for leakage exploitation. As seen, the linear autoencoder provided the same projection capability as the PCA, but the non orthogonality of subspace dimensions and correlation among them did not provide additional benefits on the considered datasets. The ability of spanning a wider subspace by adding non-linear activations in the autoencoder did not allow better leakage extraction.

Moreover, considered variants such as the variational autoencoder and GMVAE cannot be considered as applicable alternative in the context of unsupervised horizontal SCA, since their optimization procedure and architecture is too complex and depends on additional parameters. Overall, unsupervised or self-supervised neural network optimization and parameters tuning is often difficult in practice, as the ground truth is not available.

Overall, the investigated methods failed to apply on highly saturated data. It is always assumed a sufficiently high signal to noise ratio to be able to extract relevant leakage from unrelated noise. In the context of side-channel attacks, especially in the presence of high noise, leakage is often found in a small portion of the total data. Thus, compression from high-dimensional input data will often inevitably fail if the leakage or relevant information is not expressed in sufficient quantity or quality.

3.3.2.6 Perspectives

These results lead to the formulation of two statements regarding solutions for leakage exploitability. First, since extraction or compression is highly dependent on the data quality, it would be relevant to see if outliers or saturation mitigation, in order to enhance the input patterns quality, would have a benefit on the leakage extraction capability. Secondly, in cases where the input data is highly noisy and not exploitable through multivariate dimensionality reductions, it may benefit from univariate PoI selection instead, considering adapted methods for these cases are applied. These two axes will be treated in the subsequent sections of this chapter and Chapter 4.

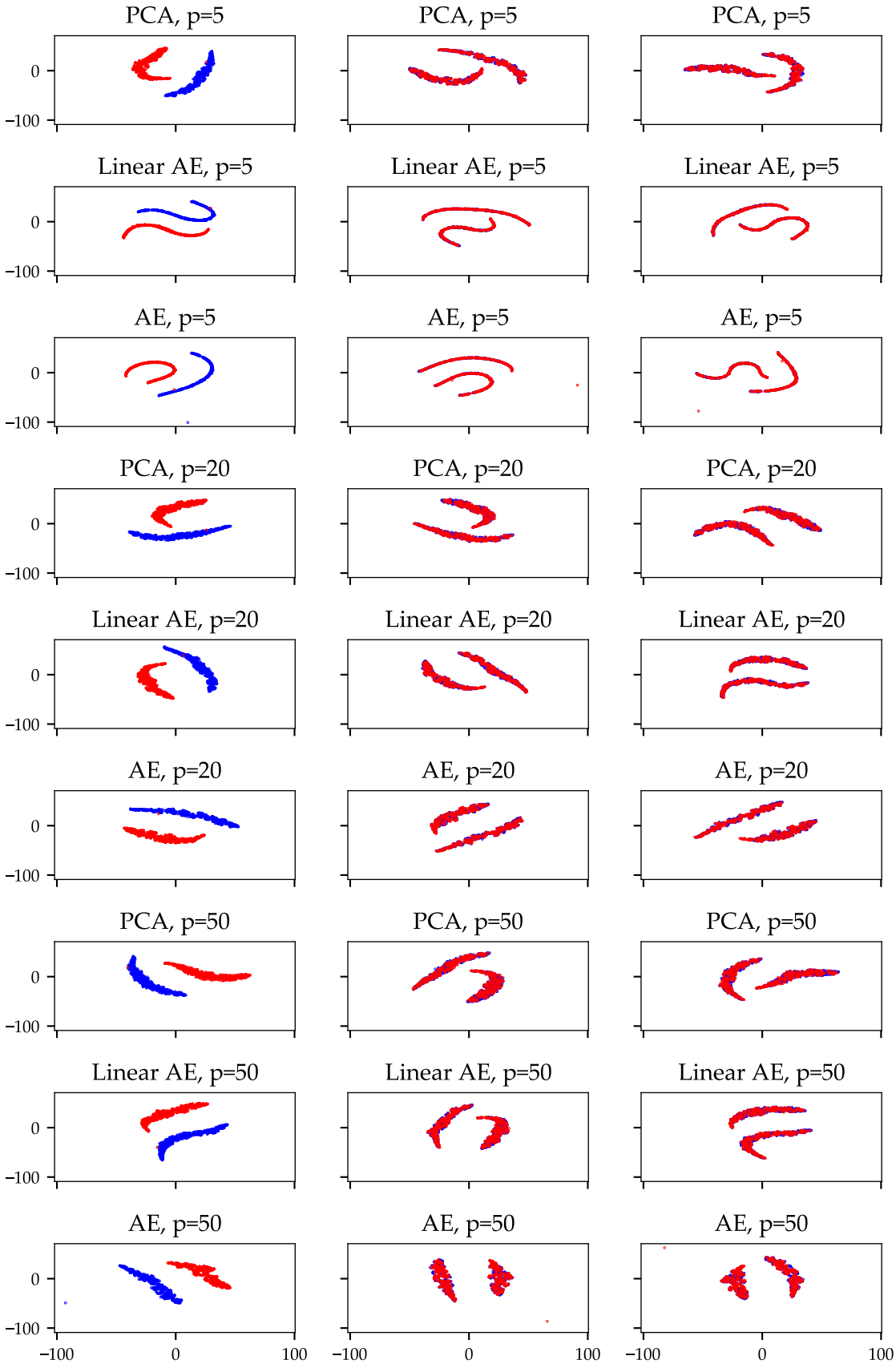

Figure 3.8: t-SNE projections into a two-dimensional space of 1000 patterns extracted from the code. Left column: $\mathcal{D}_{mbedtls}$. Middle column: $\mathcal{D}_{pointer}$. Right column: $\mathcal{D}_{arith}$.

3.3.3 Robust Autoencoders for outliers mitigation

As stated in the previous Chapter, a major challenge in the presence of outliers or highly saturated data is to be able to efficiently mitigate them so they do not affect the leakage identification and exploitation. Some neural network architecture able to train in a completely unsupervised manner can be considered as relevant candidates for this task. For example, the robust autoencoder architecture [KYG+22], which is based on the robust PCA principle [CLMW09] and aims to identify and remove outliers from the input data. It can be seen as non-linear generalization of the robust PCA. As mentioned above, allowing the learning of non-linear relationships between features can result in better low-space representation using fewer dimensions and can extract additional information. This method assumes that the input data matrix $\mathbf{X}$ can be represented as the decomposition:

$$\mathbf{X} = \mathbf{L} + \mathbf{S} \tag{3.7}$$

where $\mathbf{L}$ is the clean matrix and $\mathbf{S}$ the extracted noise or outliers sparse matrix. The objective of the autoencoder is to achieve this decomposition by training a neural network, whose loss is given by:

$$\mathcal{L}(\theta, \phi) = ||\mathbf{L} - \mathcal{F}_\theta(\mathcal{E}_\phi(\mathbf{L}))||_2 + \tau||\mathbf{S}||_1 \tag{3.8}$$

where τ is tuning parameter balancing both terms. A large τ value induces high sparsity of $\mathbf{S}$. The loss function from Equation 3.8 is decomposable as the sum of two terms which cannot be optimized as a single block. Instead, optimization must be performed alternatively on each sub-objective while keeping the other one fixed. This can be done using the Alternating Direction Method of Multipliers (ADMM) optimization algorithm. The left term of Equation 3.8, which is the reconstruction error (as seen in the previous section), can be optimized through back-propagation using gradient descent. The right term can use the proximal algorithm to optimize elements of $\mathbf{S}$ on the ℓ_1 norm. The proximal operator of the ℓ_1 norm is given by the soft threshold of each matrix element:

$$\mathcal{S}_r[x] = \text{sign}(x)\max(|x| - \tau, 0) \tag{3.9}$$

where τ is the coefficient parameter that controls the regularization and importance of $||\mathbf{S}||_1$. The global optimization procedure is given in Algorithm 5. More details about the implementation can be found in [ZP17, KYG+22]. The convergence of the optimization is reached when $\mathbf{X} = \mathbf{L} + \mathbf{S}$, or no change are observed in $\mathbf{L}$ and $\mathbf{S}$. At the end of the optimization, we obtain $\mathbf{L}$ as a cleaned version of matrix $\mathbf{X}$, and $\mathbf{S}$ containing outliers and saturations values. The process of identifying outliers or saturated values is not supervised, in the sense that no insight is given in the optimization procedure about how should behave an abnormal point compared to the rest of the data.

The idea of this approach is to see if the application of this method can have a positive effect on the leakage quality, and the percentage of outliers. We applied this methodology only on $\mathcal{D}_{pointer}$ and $\mathcal{D}_{arith}$ sets, as $\mathcal{D}_{mbedtls}$ is free of outliers and saturated values and therefore would not benefit from application of the robust autoencoder.

3.3.3.1 Parametrization

The robust autoencoder is parameterized by the choice of the autoencoder architecture and the regularization parameter τ. We adopted the same architecture as for the classical autoencoder considered in the previous section, that is, three fully connected hidden layers of

Algorithm 5 Robust autoencoder optimization procedure

Require: $\mathbf{X}, \tau, \epsilon$
Ensure: $\mathbf{X} = \mathbf{L} + \mathbf{S}$
  $\mathbf{L} \leftarrow 0$
  $\mathbf{S} \leftarrow 0$
  $\mathbf{X}^* \leftarrow \mathbf{X}$
  repeat
    $\mathbf{L} \leftarrow \mathbf{X} - \mathbf{S}$
    $\theta, \phi = \text{backprop}(\mathbf{L}, \mathcal{F}_\theta, \mathcal{E}_\phi)$ ▷ optimize left term of Equation 3.8 using gradient descent
    $\mathbf{L} \leftarrow \mathcal{F}_\theta(\mathcal{E}_\phi(\mathbf{L}))$
    $\mathbf{S} \leftarrow \mathbf{X} - \mathbf{L}$
    $\mathbf{S} \leftarrow \text{prox}(\mathbf{S}, \tau)$ ▷ update $\mathbf{S}$ by minimizing $\tau||\mathbf{S}||_1$ using Equation 3.9
    $\epsilon_1 \leftarrow \frac{||\mathbf{X}-\mathbf{L}-\mathbf{S}||_2}{||\mathbf{X}||_2}$
    $\epsilon_2 \leftarrow \frac{||\mathbf{X}^*-\mathbf{L}-\mathbf{S}||_2}{||\mathbf{X}||_2}$
    $\mathbf{X}^* \leftarrow \mathbf{L} + \mathbf{S}$
  until $\epsilon_1 \leq \epsilon$ or $\epsilon_2 \leq \epsilon$
  return m

decreasing size and ELU non-linear activations. Note that CNN were also been considered for this experiment but did not provide noticeably different outcomes, and thus will not be detailed here. The regularization parameter τ is given by:

$$\tau = \frac{4\lambda||\mathbf{X}||_1}{m \times n} \tag{3.10}$$

where λ is a chosen parameter, and m and n dimensions of matrix $\mathbf{X}$ (see details on [CLMW09]). However, in order to obtain good results, appropriate λ value must be chosen. Several methods have been proposed to deduce the value of λ, but this requires heavy parameter search and results in heavy computation complexity. For this work, we chose $\lambda_{\text{pointer}} = 10$ and $\lambda_{\text{arith}} = 100$ values such that the resulting τ value approximates the global percentage of identified outliers on $\mathbf{X}$ (about 4% for $\mathcal{D}_{pointer}$ and 5% for $\mathcal{D}_{arith}$). This gives an order of magnitude of the expected sparsity of $\mathbf{S}$.

The optimization and approximation of Equation 3.7 was done by applying the procedure from Algorithm 5 for 100 loops, or until a stopping criterion threshold was reached ($\epsilon_1, \epsilon_2 \leq \epsilon$), with $\epsilon = 10^{-4}$. Additionally, we set the learning rate for the network optimization $\alpha = 10^{-4}$ and used the Adam optimizer.

3.3.3.2 Impact on leakage

After the optimization procedure (which consists in the neural network training reconstruction of updated $\mathbf{L}$, as well as the minimization of ℓ_1 on $\mathbf{S}$), this process yields the cleaned matrix $\mathbf{L}$ and the noise matrix $\mathbf{S}$. To ensure leakage conservation in $\mathbf{L}$ and correct outliers extraction in $\mathbf{S}$, we computed the global mutual information between each matrix and associated ground truth labels. This estimation was done by applying the MINE framework. The resulting estimates are listed in Table 3.2. On $\mathcal{D}_{pointer}$, $I_\theta(\mathbf{L}, \mathbf{Y}) < I_\theta(\mathbf{X}, \mathbf{Y})$, thus a slight degradation of the MI is observed through the compression. For $\mathcal{D}_{arith}$, the opposite behavior is observed, i.e., the mitigation of some outliers or saturation values allowed to obtain better MI values thus to recover some information. As expected, for both datasets, the measured MI between outliers matrix $\mathbf{S}$ and labels $I_\theta(\mathbf{S}, \mathbf{Y})$ was equal to zero, meaning that no

Dataset	$I_\theta(\mathbf{X}, \mathbf{Y})$	$I_\theta(\mathbf{L}, \mathbf{Y})$	$I_\theta(\mathbf{S}, \mathbf{Y})$
$\mathcal{D}_{pointer}$	0.98	0.91	0
$\mathcal{D}_{arith}$	0.88	0.97	0

Table 3.2: Mutual information estimates (using MINE) of each matrix and the ground truth labels vector $\mathbf{Y}$.

information regarding the leakage was transferred to $\mathbf{S}$.

To gain some insight about the effect of the Robust Autoencoder (RAE) on the univariate leakage contained in the patterns, we produce Figure 3.9 and 3.10. They show the $\mathcal{T}$-values and the BRR respectively, obtained before (in blue) and after (in red) the application of the RAE. The green curves, plotted with the y-axis reversed, show the differences between the

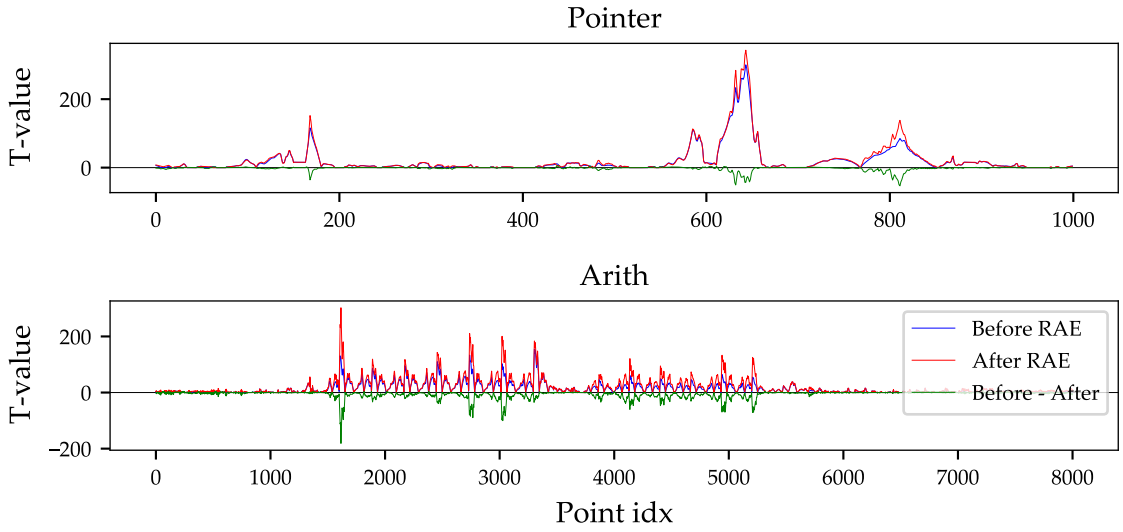

Figure 3.9: $\mathcal{T}$-test comparison before (blue) and after (red) application of the RAE, and the difference (green).

$\mathcal{T}$-values and the BRR before and after the application of the RAE. Thus peaks below the x-axis represent improvements on the considered metric after the application of the RAE, while the peaks above represent a degradation.

The application of the RAE shows positive effects on the leakage levels of both considered datasets. Furthermore, as stated in previous Chapter, there is no linear relationship between the $\mathcal{T}$-value and associated BRR for a given time point. To better illustrate the advantages of RAE, we have plotted on the first (resp. second) row of Figure 3.11 the $\mathcal{T}$-values (resp. BRR) obtained after applying the RAE (i.e. considering $\mathbf{L}$) versus the values obtained on raw patterns (i.e. considering $\mathbf{X}$). Values observed above of the diagonal line translate to improvement of the considered metric, the opposite transcribe a degradation in the RAE optimization, and values along the line mean no changes are observed. Most of the values obtained on $\mathbf{L}$ are above the first bisector, demonstrating that the applied RAE has effectively cleaned the data as the observed leakage increased. Some $\mathcal{T}$-values obtained for $\mathcal{D}_{arith}$ were even triple. The same is displayed using BRR on the bottom row of Figure 3.11. For both datasets, many BRR values have shifted from 0.5 to more than 0.7 demonstrating once again the very significant effect of outliers and saturated values on the leakage.

On all cases, it is straightforward to see that the mitigation procedure has a beneficial

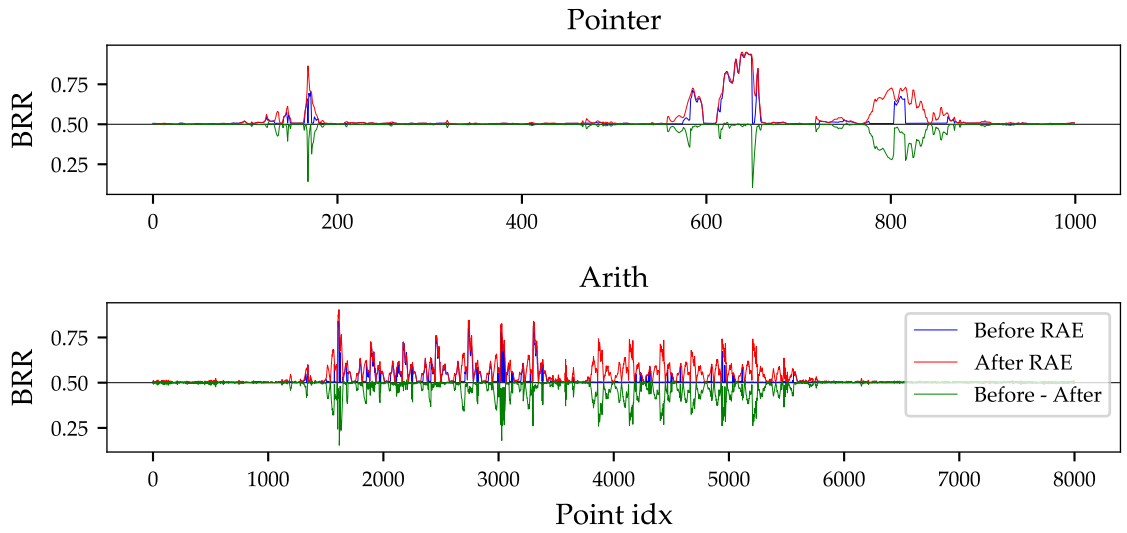

Figure 3.10: BRR values before (blue) and after (red) application of the RAE, and the difference (green).

effect on the measured $\mathcal{T}$-values and BRR scores. No major degradation is observed, and many points displaying BRR equal to 0.5 before the optimization are now better clustered after the removal of saturated elements, resulting in higher BRR values. Finally, Figure 3.12 compares the percentage of time points with BRR values greater than a threshold. It is clear that the RAE increases the BRR obtained for a really high percentage of time points for both datasets. Of course, the increase in BRR disappears progressively as the threshold gets closer to 1. All in all, one could think of systematically using the RAE as a preprocessing tool to mitigate outliers. In our context, this should make it easier to identify PoI when using previously presented methods (e.g. Perin *et al.* with ratio [PC16]).

3.3.3.3 Outliers mitigation capability

This section compares the outliers changes in the application of the RAE. It aims at assessing how the process of outliers mitigation affects the identified outliers and saturation levels on the input patterns (see Chapter 2), globally and locally. This is possible by searching for non-zero elements indicating the presence of outliers in the matrix $\mathbf{S}$ provided by the RAE to construct the matrix $\mathbf{R}$, which elements are defined as follows:

$$r_{i,j} = \begin{cases} 1 & \text{if } |s_{i,j}| > 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.11}$$

From this, to compute the percentage of correctly identified outliers by comparing the matrix $\mathbf{R}$ with the matrix $\mathbf{O}$, which is computed in a similar way as $\mathbf{R}$ but by applying the IQR criterion to each column of the matrix $\mathbf{X}$:

$$\text{outliers score} = \frac{1}{n \times m} \sum_{i=1}^{n} \sum_{j=1}^{m} [r_{i,j} = o_{i,j}] \tag{3.12}$$

Table 3.3 reports the percentage of outliers and saturated values computed on the input matrix $\mathbf{X}$ and after the robust autoencoder optimization on the cleaned matrix $\mathbf{L}$. We report

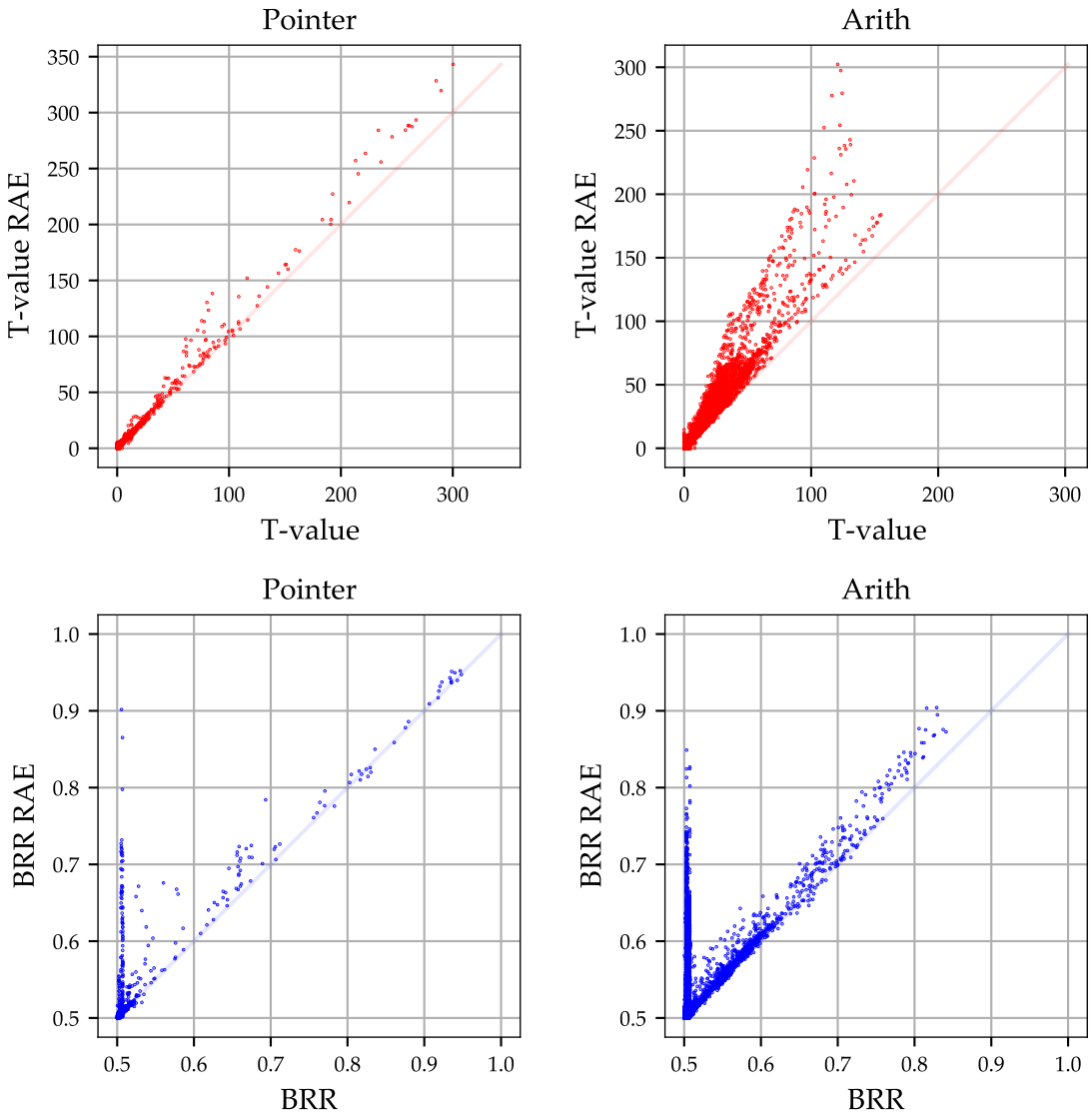

Figure 3.11: Scatter plot of the BRR and $\mathcal{T}$-test values for each time point before and after application of the RAE.

for $\mathcal{D}_{pointer}$ and $\mathcal{D}_{arith}$ previously identified values from table 2.2 to allow better comparison. As reported, RAE correctly identifies more than 80% of the values flagged as outliers by the IQR criterion. However, it only partially mitigates the outliers. In fact, the process has a minor impact on the mitigation of said outliers. On $\mathcal{D}_{arith}$, it allows removing only a small fraction of outliers globally. A slight degradation is observed on $\mathcal{D}_{pointer}$.

Regarding the saturated values, their amount is drastically reduced for both datasets (reaching almost 0% on $\mathcal{D}_{arith}$), as well as the number of time points with more than 5% of saturated values.

These results confirm the relevance of the chosen methodology for the mitigation of the saturation values, considering that the process has no negative side effect on the observed leakage levels, but allows to obtain better and more usable point distributions.

In addition to these results, Figure 3.13 shows the percentage of outliers per time point, before and after the application of the RAE optimization procedure. Negative results on the green plot translate a reduction of the outliers percentage.

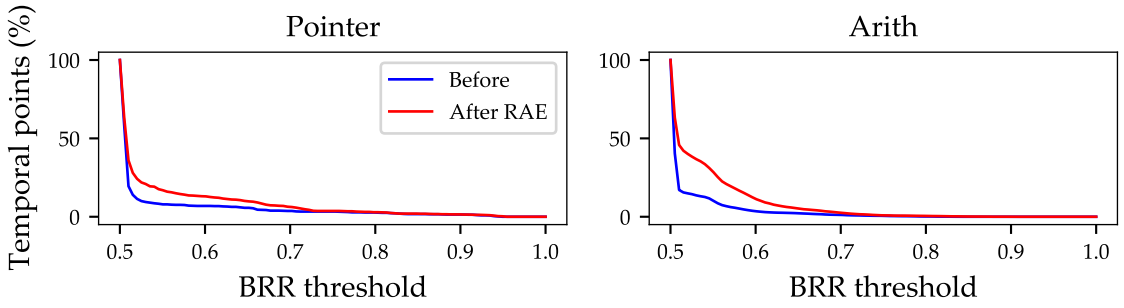

Figure 3.12: Percentage of time points with a BRR higher than a given threshold obtained after and before applying the RAE.

	$\mathcal{D}_{pointer}$		$\mathcal{D}_{arith}$	
	X	L	X	L
Overall number of outliers (IQR)	**4.32%**	4.63%	5.15%	**4.67%**
% of points with >5% of outliers	**27.5%**	29.00%	34.45%	**27.35%**
Overall number of saturated values (SV)	30.27%	**1.19%**	12.88%	**0.01%**
% of points with >5% of saturated values	53.80%	**3.40%**	29.15%	**0.07%**
Overall number of outliers and SV	33.39%	**5.82%**	16.54%	**4.68%**
% of points with >5% of outliers and SV	65.20%	**34.50%**	46.74%	**27.98%**
Correctly identified outliers (IQR)	83.65%		87.62%	

Table 3.3: Comparison of the percentage of outliers and saturated values before and after the RAE optimization. Best obtained results are highlighted with bold font.

As expected from the previous table, the advantages are reduced for $\mathcal{D}_{pointer}$, but much more significant for $\mathcal{D}_{arith}$, especially in the zones where the leakage is located (ranges of index $\{1600, \ldots, 3300\}$ and $\{3900, \ldots, 5200\}$). To gain more insight into the RAE effect on saturated values, we have plotted the traces consisting of the minimum and maximum of each time point in $\mathbf{X}$ and $\mathbf{L}$ in Figure 3.14. The contrast between the two plots is striking. Most of the time points of $\mathbf{X}$ have minimum and maximum values equal to -1 and 1 respectively. This is no longer the case when looking at the cleaned patterns of $\mathbf{L}$. While some time points remain saturated after applying the RAE most of the saturated values are removed after application of the RAE. These results emphasize on the pertinence of the RAE as a relevant candidate for unsupervised saturation mitigation.

3.3.3.4 Conclusion

In the previous sections, experiments have shown that robust autoencoders are able to correctly factorize $\mathbf{X}$ into $\mathbf{L}+\mathbf{S}$ to decompose the input matrix into a clean matrix and an outlier matrix. This allowed to significantly mitigate the saturated values present in the data and as a result to better detect the leakage (increase of $\mathcal{T}$-values). In the attack context, this leads to obtain better clustering results (BRR values).

This is all the more interesting as the method is completely unsupervised. Only an appropriate λ value is required, and it has been shown that choosing λ according to the percentage of outliers identified using the IQR criterion can be a good starting point. No extensive autoencoder architecture or hyper-parameters tuning has been required, as a simple fully con-

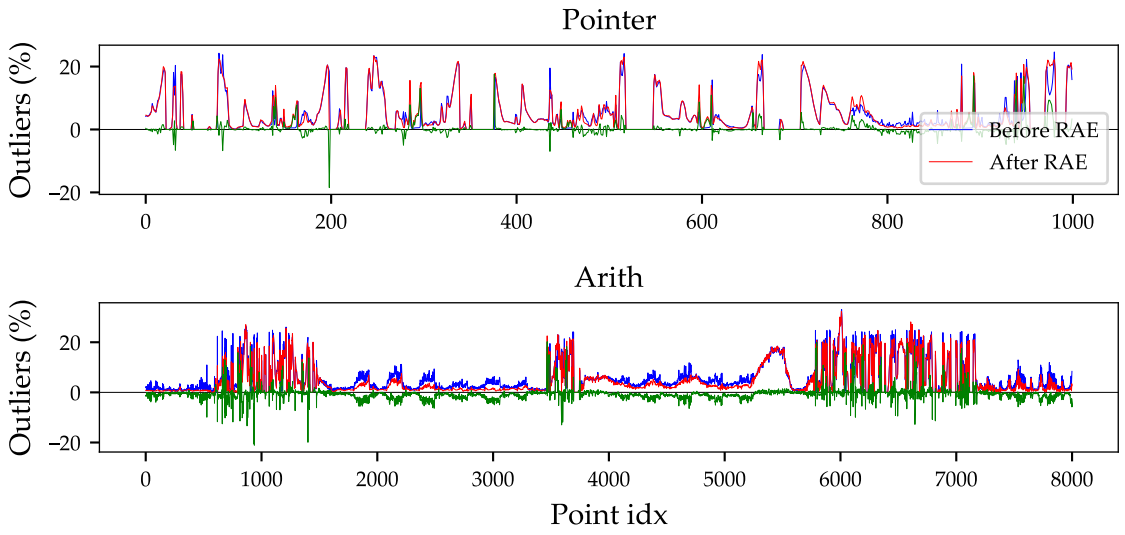

Figure 3.13: Percentage of outliers per time point (IQR) before (blue) and after (red) the application of the RAE. The green curve describes the improvement (< 0) or the deterioration (> 0).

nected autoencoder architecture was sufficient to get consistent reconstructions of the data. As noted above, no additional benefits have been observed in using CNN layers or more complex architectures compared to MLP. It is also noteworthy to say that further analysis of outliers and saturation mitigation success on resulting cleaned patterns can also be carried out in a complete unsupervised manner.

Still, the proposed optimization procedure emphasizes some limitations. First, the main drawback from this approach lies in the fact that no insight is given to the RAE about the nature of outliers to be mitigated. The identification and correction of said outliers has to be done undirected. While it has been shown to be successful in identifying outliers and saturated points in this study, it may fail to in others contexts, depending on the data with different distributions or properties. Thus, being able to integrate to knowledge of the outliers nature and location in the optimization procedure (e.g, using a modified loss function), could have some advantages to ensure more robustness in the outliers and saturated values mitigation process.

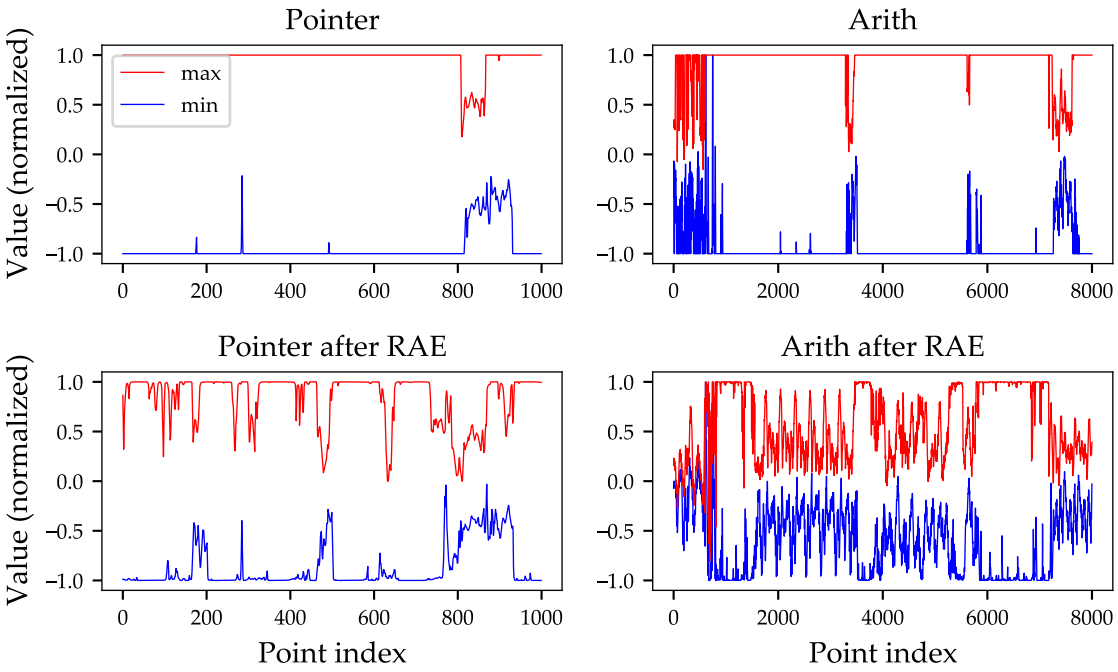

Figure 3.14: Minimum and maximum values for each time point before and after the RAE cleaning.

3.4 Generative Adversarial Networks

This section considers the use of Generative Adversarial Network (GAN) as an alternative to the robust autoencoder for outliers mitigation. As noted above, the previously considered method does not rely on any kind of supervision or prior knowledge in the mitigation process. As a result, self-supervised learning and driven architectures based on GAN are studied as an alternative in order to provide a more focused outliers correction procedure.

3.4.1 GAN architecture

The GAN architecture, proposed by Goodfellow *et al.* in 2014 [GPAM+14], is a class of generative models that are trained through an adversarial process. In its simplest form, it consists in two neural networks that train together on a zero-sum game. The discriminator network D learns to classify as authentic or fake the samples it is provided with. The generator network G learns to generate convincing samples from a noise latent space, which are then fed to the discriminator. By observing the discriminator D loss, the generator G aims at improving its sampling until D is not able to distinguish real from generated sample. The GAN objective is given by:

$$\begin{aligned}\min_G \max_D \mathcal{L}_{\text{GAN}}(G, D, X, Z) = \mathbb{E}_{x\sim X} \log D(x) \\ + \mathbb{E}_{z\sim Z} \log[1 - D(G(z))]\end{aligned} \tag{3.13}$$

where the discriminator D tries to maximize its probability of correctly predicting labels coming either from the training set or generated by G, and on the other hand, the generator G tries to minimize the right term of Equation 3.13, that is, the discriminator prediction error. This learning procedure can be seen as two-players minimax game, where each player tries

to minimize the adversary's maximum payoff.

Convergence is reached when an equilibrium is attained, where G is able to generate convincing samples, and further training would degrade D feedback. However, in practice on complex tasks, GAN training can sometimes lead to unstable convergences. In the case of a weak discriminator, where the generator is able to quickly learns the inner representation and successfully fool the discriminator, we speak of mode collapse. A simple illustration of the GAN architecture is given on Figure 3.15.

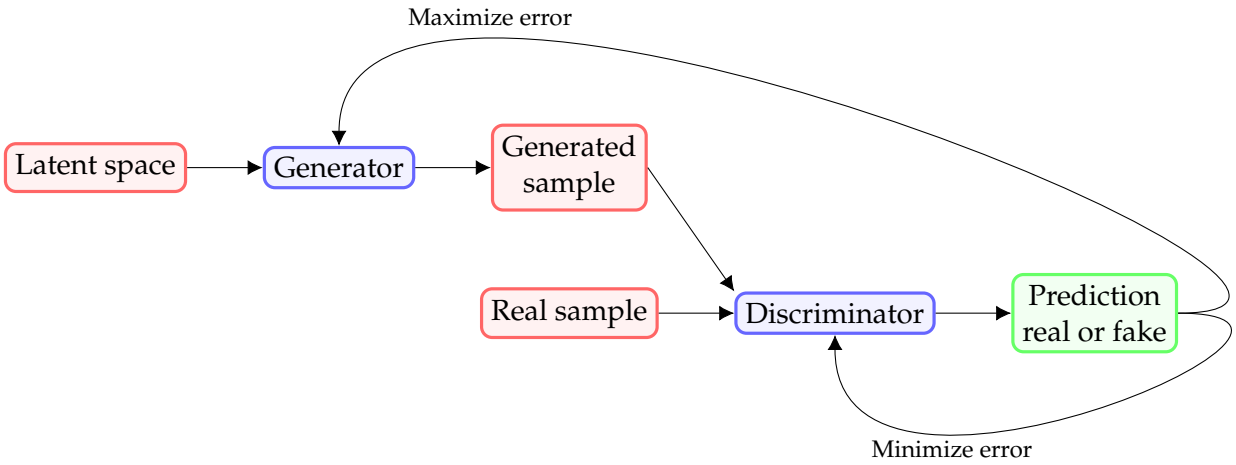

Figure 3.15: Illustration of the GAN architecture

3.4.2 CycleGAN

The CycleGAN is an extension of the GAN architecture. It was first proposed by Zhu *et al.* [ZPIE20] in 2020 for image domain translation, as an unpaired improvement over the Pix2Pix conditional GAN [IZZE18]. Given two sets of images from different domains $\mathcal{A}$ and $\mathcal{B}$, it aims at finding the mapping $F_{\mathcal{A}} : \mathcal{A} \to \mathcal{B}$ and inverse mapping $F_{\mathcal{B}} : \mathcal{B} \to \mathcal{A}$. The CycleGAN architecture features two GAN, that is two pairs of generator and discriminator networks, namely D_A, G_A, D_B, G_B. The complete loss is given by:

$$\begin{aligned}\mathcal{L}(G_A, G_B, D_A, D_B) = \mathcal{L}_{\text{GAN}}(G_A, D_B, A, B) \\ + \mathcal{L}_{\text{GAN}}(G_B, D_A, B, A) \\ + \lambda \mathcal{L}_{\text{cyc}}(G_A, G_B)\end{aligned} \tag{3.14}$$

where a cycle consistency loss is added when switching back to original domain:

$$\begin{aligned}\mathcal{L}_{\text{cyc}}(G_A, G_B) = \mathbb{E}_{a\sim A}||G_B(G_A(a)) - a||_1 \\ + \mathbb{E}_{b\sim B}||G_A(G_B(b)) - b||_1\end{aligned} \tag{3.15}$$

This loss encourage that $G_B(G_A(a)) \approx a$ and $G_A(G_B(b)) \approx b$ to guarantee mappings bijections. As with the vanilla GAN architecture, the optimization aims at finding G_A^* and G_B^* such that:

$$G_A^*, G_B^* = \underset{G_A, G_B}{\operatorname{argmin}} \underset{D_A, D_B}{\operatorname{argmax}} \mathcal{L}(G_A, G_B, D_A, D_B) \tag{3.16}$$

Figure 3.16 shows a simplified illustration of the CycleGAN architecture and the associated data flows.

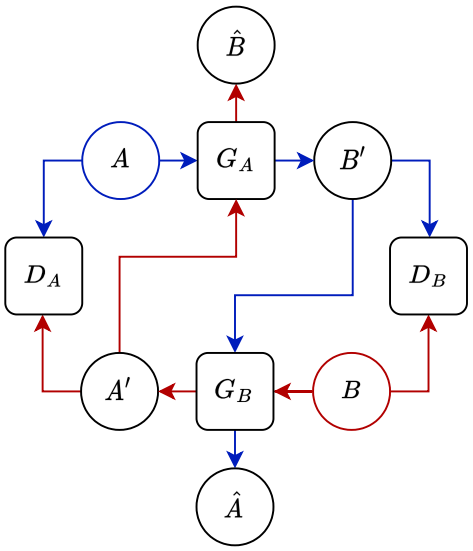

Figure 3.16: CycleGAN Architecture. Information flows from input data A and B are represented by blue and red colors.

3.4.3 Driven outliers mitigation

To formulate the outliers context, consider a matrix $\mathbf{X} \in \mathcal{M}_{n,m}(\mathbb{R})$ consisting in n measures of m points, and associated classes vector $\mathbf{Y} \in \mathbb{N}^n = (y_1, \ldots, y_n)$. Outliers can be defined as value outside the extended interquartile range of observed distributions:

$$\begin{aligned} f(\mathbf{X}) = \forall (i,j) \in (m,n) : \\ (x_{i,j} < Q_{1[x_j]} - 1.5 \times \text{IQR}_{[x_j]}) \vee \\ (x_{i,j} > Q_{3[x_j]} + 1.5 \times \text{IQR}_{[x_j]}) \end{aligned} \tag{3.17}$$

This allows to build the matrix $\mathbf{M} \in \mathcal{M}_{n,m}(\{0,1\})$, where $\mathbf{M} = f(\mathbf{X})$ the binary mask for each value of $\mathbf{X}$.

Through the optimization process, we aim to find alternative matrix $\mathbf{X}'$ such that the amount of outliers is lower than the base input matrix $\mathbf{X}$:

$$||f(\mathbf{X}')||_0 \leq ||f(\mathbf{X})||_0 \tag{3.18}$$

where $||\cdot||_0$ stands for the count of nonzero elements in the matrix. In addition, considering ℓ as an arbitrary leakage evaluation function, we would like to ensure the conservation or improvement of leakages levels:

$$\ell(\mathbf{X}', \mathbf{Y}) \geq \ell(\mathbf{X}, \mathbf{Y}) \tag{3.19}$$

assuming that higher leakage translate to better attack performances (that is not always the case as seen above).

In order to achieve this in an unsupervised manner, we first want to split the input matrix $\mathbf{X}$ into two sets, $\mathbf{A}$ and $\mathbf{B}$. To do so, we build the matrix $\mathbf{H}$ of the Hamming distance of between each possible $(m_{i,:}, m_{j,:})$ pairs of rows of $\mathbf{M}$:

$$h_{i,j} = \mathrm{HW}(m_i \oplus m_j) \tag{3.20}$$

where HW denotes the Hamming weight, so that each entry $h_{i,j}$ is the number of unshared outliers (the divergence) between pattern i and j in terms of point $\mathbf{X}$. From this, is possible to build the two matrices $\mathbf{A} = (x_{i,:}, \ldots)$ and $\mathbf{B} = (x_{j,:}, \ldots)$ of patterns by successively selecting the remaining pattern pairs with the greatest number of unshared outliers in descending order $\mathrm{argmax}_{i,j}\ \mathbf{H}$. [1] The goal is to cancel out these pairs by applying the CycleGAN framework. The pair $(x_{i,:}; x_{j,:})$ being selected, $x_{i,:}$ is pushed into $\mathbf{A}$ while $x_{j,:}$ is pushed into $\mathbf{B}$. These two matrices constitute the datasets provided to the two generators and discriminators of the CycleGAN shown in Figure 3.16.

As it stands, the CycleGAN will regenerate all points of all patterns during its training. This is not the sought behavior. In fact, most of the values are not outliers and should not be corrected. We therefore added two multiplexers to the CycleGAN to replace only points identified as outliers in the original patterns. These multiplexers apply the following masks:

$$\begin{aligned} m(\mathbf{A}, G_B(\mathbf{B}), \mathbf{M_A}) &= (\mathbf{M_A} \circ G_B(\mathbf{B})) + (\neg \mathbf{M_A} \circ \mathbf{A}) \\ m(\mathbf{B}, G_A(\mathbf{A}), \mathbf{M_B}) &= (\mathbf{M_B} \circ G_A(\mathbf{A})) + (\neg \mathbf{M_B} \circ \mathbf{B}) \end{aligned} \tag{3.21}$$

where $\mathbf{M_A}$ and $\mathbf{M_B}$ are the associated outliers mask from matrix $\mathbf{M}$. The Figure 3.17 illustrates the updated CycleGAN architecture with the addition of the two multiplexers. As a

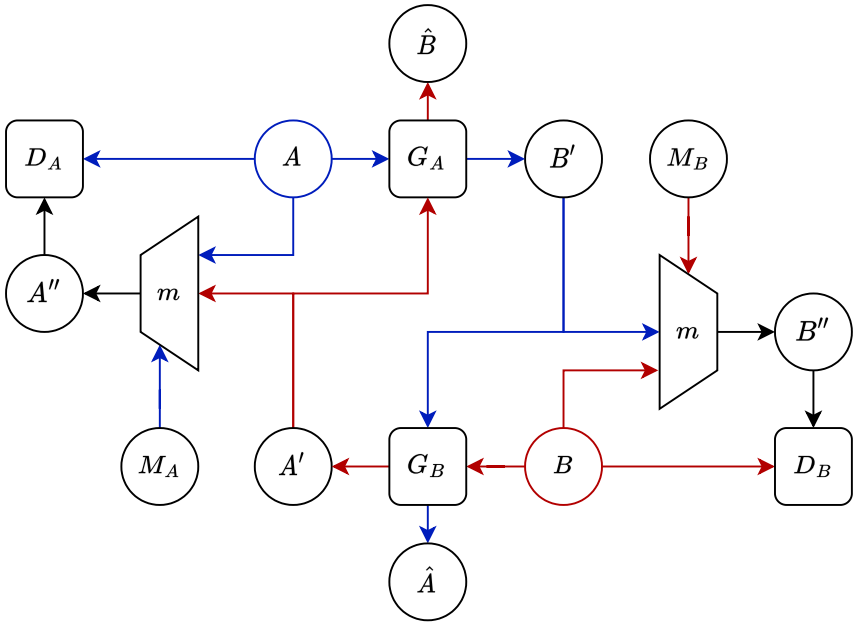

Figure 3.17: Modified CycleGAN architecture with the add of a multiplexer m

result, instead of feeding directly the domain translated $\mathbf{A}'$ and $\mathbf{B}'$ to discriminators, the outputs of multiplexers $\mathbf{A}''$ and $\mathbf{B}''$ are used.

[1] This is repeated $\frac{n}{2}$ times, while discarding $h_{i,:}$ and $h_{:,j}$ when selected.

This has several benefits. First, multiplexer results are only slight modifications of inputs $\mathbf{A}$ and $\mathbf{B}$ where outliers are replaced with generated values. This has the effect of reducing the training complexity of discriminators as focus is only made on some points, while the others are identical to the base patterns. Consequently, this action results in increased attentions on these outliers points, thus generators are trained more efficiently providing convincing outliers replacements values. Moreover, Equation 3.21 is fully, differentiable, so it does not affect the gradient descent back-propagation. Eventually, the global optimization procedure is improved as only some subparts of patterns differences are compared for assessing samples generation quality. The optimization procedure for outliers mitigation is given as follows:

1. Calculate the outlier matrix $\mathbf{M} = f(\mathbf{X})$, according to Equation 3.17.
2. Split the original input data into two subsets $\mathbf{A}$ and $\mathbf{B}$ with highest Hamming distance between pairs of rows of $\mathbf{M}$.
3. Train the CycleGAN until convergence (Loss equilibrium).
4. Replace outliers in initial traces $\mathbf{X}$ with generated points $\mathbf{X}'$ using $\mathbf{M}$ through the multiplexer m (Equation 3.21).
5. **Repeat** until $||\mathbf{M}||_0$ converges or no changes are observed.

At the end of the process, cleaned matrix $\mathbf{X}''$ should display a reduced amount of outliers without altering the legitimate points that are not targeted by the optimization.

3.4.3.1 Network Parametrization

For the application of the proposed architecture on SCA datasets, we considered the following network structure and parametrization. For the generators G_A and G_B, we use the LSGAN to prevent vanishing gradient problems [MLX+17]. Their architectures are defined as convolutional autoencoders with three downsampling layers and their upsampling symmetry. Each layer was followed by a ReLU activation, except the output layer that uses a Tanh activation for scaling the output to the desired $(-1, 1)$ range.

The discriminators D_A and D_B were defined as pixel discriminators, where each point (instead of the whole sample) is classified as real or fake [IZZE18]. ReLU activations were also used in each hidden layer except for the output layer.

The optimization procedure used a learning rate $\alpha = 10^{-5}$, a batch size of 16 samples and 10 epochs for the network training.

No extensive architecture exploration and hyperparameters tuning was done, as the aforementioned configuration allowed to provide satisfactory results. Still, different structure and parameters may allow some improvements.

3.4.3.2 Outliers mitigation performance

At the end, the above procedure yields the clean matrix $\mathbf{X}''$. Its analysis led to the results presented on Table 3.4. It compares the percentage of outliers and saturated values on the input data $\mathbf{X}$, the results obtained after applying the CycleGAN $\mathbf{X}''$, and we also report the result using the RAE $\mathbf{L}$. The CycleGAN architecture provides better performance than the RAE architecture to mitigate outliers identified using the extended interquantile range criterion. On $\mathcal{D}_{pointer}$, the GAN allows to divide the percentage of identified outliers by two,

	$\mathcal{D}_{pointer}$			$\mathcal{D}_{arith}$		
	X	L (RAE)	X'' (GAN)	X	L (RAE)	X'' (GAN)
Overall number of outliers (IQR)	4.32%	4.63%	**2.06%**	5.15%	4.67%	**1.26%**
% of points with >5% of outliers	27.5%	29.00%	**12.90%**	34.45%	27.35%	**6.88%**
Overall number of saturated values (SV)	30.27%	**1.19%**	29.10%	12.88%	**0.01%**	11.41%
% of points with >5% of saturated values	53.80%	**3.40%**	50.60%	29.15%	**0.07%**	21.36%
Overall number of outliers and SV	33.39%	**5.82%**	31.10%	16.54%	**4.68%**	12.61%
% of points with >5% of outliers and SV	65.20%	**34.50%**	52.60%	46.74%	27.98%	**22.61%**

Table 3.4: Percentage of outliers and saturated values obtained on original patterns, after applying the RAE and the CycleGAN. The best results are highlighted in bold.

while the use of the RAE causes an increase in the number of outliers. For $\mathcal{D}_{arith}$, the usage of the CycleGAN demonstrates an improvement of factor more than 3 for outliers mitigation, compared to the RAE. The fact that the GAN training benefits from the supervision of outliers masks and the multiplexer can explain this behavior, since the mitigation is completely unsupervised for the RAE, which has to identify and correct abnormal values.

When comparing the changes of saturated values, only minor changes are observed after application of the CycleGAN, but no degradation. This is caused by the outliers mitigation optimization where some outliers are also identified as saturated values. For information, the global percentage of points identified both as outliers and saturated values is 1.21% for $\mathcal{D}_{pointer}$ and 1.52% for $\mathcal{D}_{arith}$. As only outliers identified values are replaced generated ones, it is normal that only few saturated values are affected.

Regarding the effect of the CycleGAN on the saturated values, it seems to be limited to a few percent. Only minor changes are observed, but no degradation. This result was expected because the CycleGAN was designed to correct only outliers identified by the extended IQR criterion, and because most saturated values are not identified as such by this criterion. We recall here that some time points of both datasets consist mainly of saturated values and are not flagged as outliers: only 1.21% for $\mathcal{D}_{pointer}$ and 1.52% for $\mathcal{D}_{arith}$ of values are simultaneously identified as being saturated values and outliers. As only outliers identified values are replaced generated ones, it is normal that only few saturated values are affected.

Figure 3.18 shows the percentage of outliers per time point for the two datasets after applying the GAN framework.

For both datasets, there is a significant reduction in the percentage of outliers. Only a few time points of $\mathcal{D}_{pointer}$ (around index 250) show an increase in the percentage of outliers. The difference (shown in green) indicates that at some time points, the identified outliers are completely mitigated. If this figure 3.18 is compared with the results obtained using the RAE and reported in figure 3.13, it is straightforward to see that the CycleGAN outperforms the RAE for outliers mitigations.

3.4.3.3 Impact on leakage

The effect of the GAN on the $\mathcal{T}$-values computed with the knowledge of the labels is shown in Figure 3.19. The application of the framework does not induce degradation in the observed leakage, on the contrary, many increases are observed, as sought (Equation 3.19). To ensure that this outlier mitigation process does not impact the leakage and its clustering capacity, we plotted the BRR obtained before and after applying the GAN in Figure 3.20. For $\mathcal{D}_{pointer}$, significant increases (more than 25% for some of them) in BRR are observed, but no new leakage times are revealed. For $\mathcal{D}_{arith}$, the behavior is about the same, except

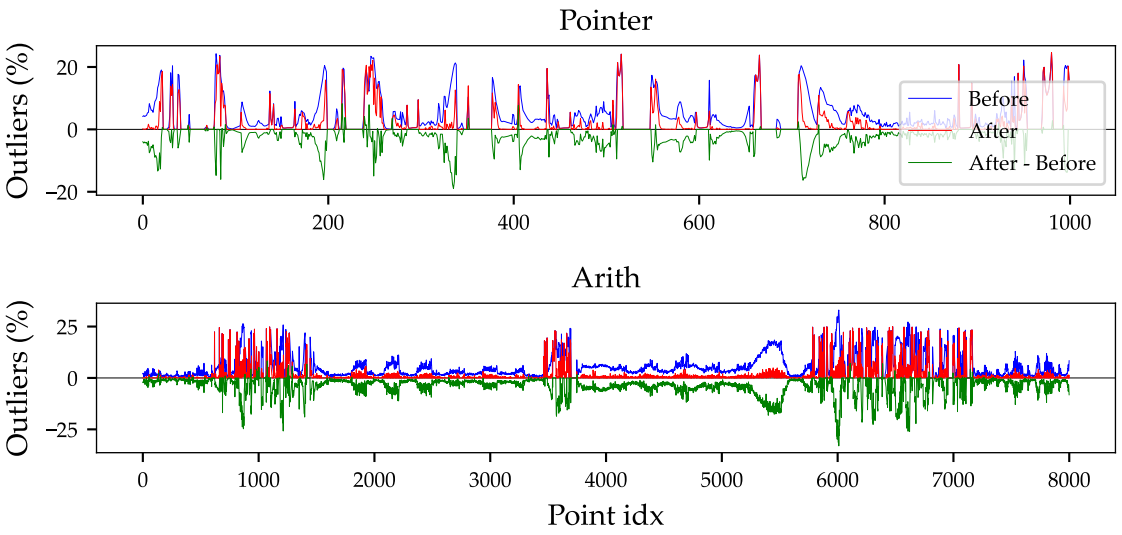

Figure 3.18: Percentage of outliers per time point (IQR) before (blue) and after (red) the application of the GAN. The green curve describes the improvement (< 0) or the deterioration (> 0)..

that some points (for instance between index 3800 and 5500) with initial BRR values close to 50% now have BRR values close to 80% and could potentially be considered as relevant PoI. Since these points had significant initial $\mathcal{T}$ values, one might think that the distribution associated with these points has changed. This point will be discussed later. Figure 3.21 allows to compare the BRR obtained on the raw dataset and the datasets provided by the RAE and GAN. Both methods exhibit similar performances in terms of leakage conservation and improvements. It is clear that the reduction of outliers and saturated values has a positive effect. However, for a given point, depending on whether the dominant anomalous values are outliers or saturated values, the RAE provides better performance or not compared to the GAN.

This shows that both methods are suitable for correcting parasite points from the input patterns without inducing degradation to the exploitable leakage. Based on the optimization provided, the CycleGAN is more suitable for outliers mitigation, while the RAE can achieve better results if patterns are highly saturated. Thus the choice of the method to consider will depend on the initial data distribution.

3.4.3.4 Impact on distributions

In the following, we quantify the global distribution divergences between raw patterns and cleaned patterns provided by either the RAE or the CycleGAN. The idea behind this is to assess whether the RAE or the CycleGAN causes major changes in the shape of the time point distributions. This task is done using the Jensen-Shannon divergence [FT04], which is based on the Kullback-Leibler divergence, but whose value is bounded between 0 and 1, where a 1 translates to an absolute divergence between the two studied distributions, while 0 means that they are identical. The Jensen-Shannon divergence is given by:

$$\mathrm{JSD}(P||Q) = \frac{1}{2}D_{KL}(P||M) + \frac{1}{2}D_{KL}(Q||M) \tag{3.22}$$

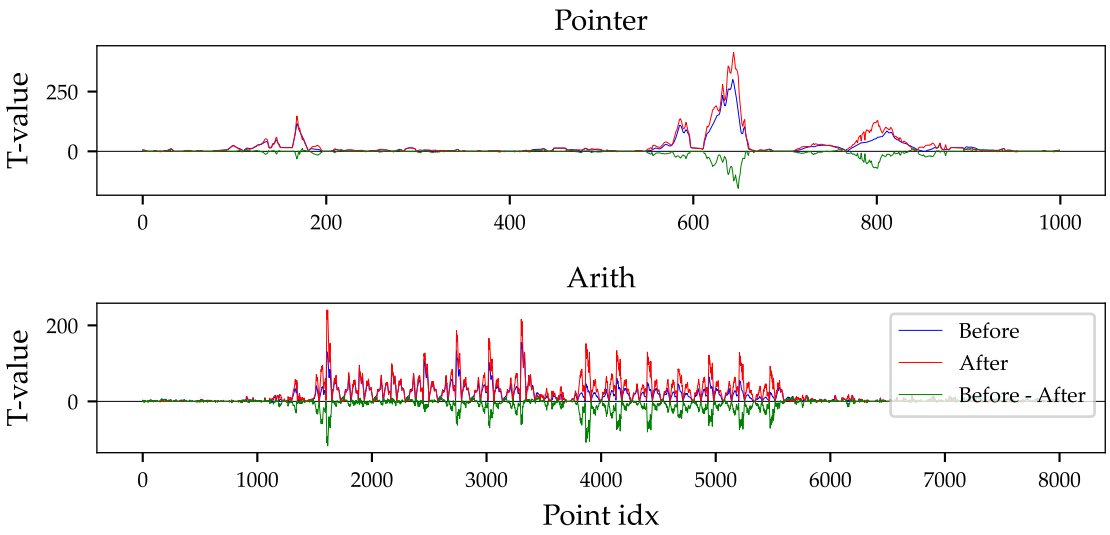

Figure 3.19: $\mathcal{T}$-values before (blue) and after (red) applying the GAN, and the difference in green.

with $M = \frac{1}{2}(P + Q)$ is the mixture distribution of P and Q. Figure 3.22 depicts the computed Jensen-Shannon divergence between corresponding time points before and after the application of either the RAE or the CycleGAN. Both methods have a similar impact on the distributions change. While the RAE shows slightly higher values on both sets, it does not rely on guided optimization for mitigation unlike the CycleGAN. This can however produce some unwanted side effects. While high measured values on $\mathcal{D}_{arith}$ are for most synonyms of highly saturated values correctly identified and mitigated, hence translating by high observed divergence values.

Both methods seem to have a similar impact on the distributions change. The Jensen-Shannon divergence (JSD) values obtained between the raw datasets and the cleaned datasets generated by RAE or GAN are all between 0 and 0.3 for both data sets. Thus, there are moderate changes in the shape of the distributions. Note, however, that the JSD between the raw data and the data generated by RAE are often higher than between the raw data and the data generated by GAN. This is not surprising, since only the outlier values (about 5% of all values) are modified by the GAN architecture we propose.

However, on some time points (for example index 838 on $\mathcal{D}_{pointer}$ set, for which the obtained JSD value is close to 0.3), the RAE produces a totally different distribution, even the input distribution is free of outliers or saturated values. The change in the shape of the distribution can be seen in Figure 3.23. As shown, both mean and variance are changed. This emphasizes that completely unsupervised mitigation process can sometimes result in unwanted behaviors. To possibly reduce this effect, a solution in the case of the RAE would be to only replace abnormal identified values with generated ones instead of considering the new cleaned patterns as a whole.

3.4.3.5 Updated univariate PoI identification

After setting up the above mitigation solutions, we wanted to assess whether they had a positive effect on the common methods for unsupervised PoI selection, as one would expect. To this aim, we applied the method of [PC16] (as described in the previous Chapter) to the cleaned datasets provided by the RAE and CycleGAN. We did not observe any

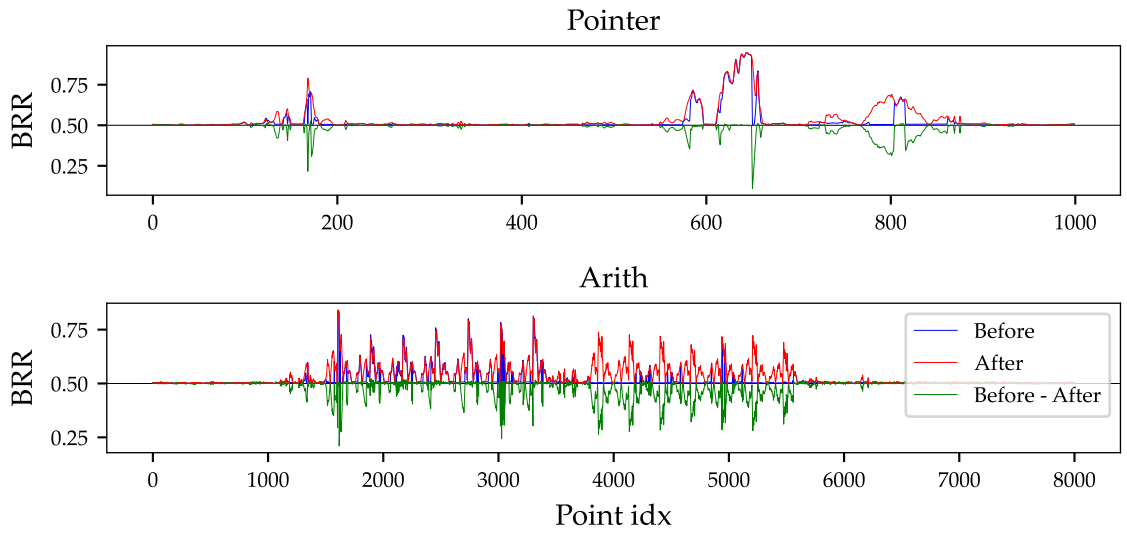

Figure 3.20: BRR before (blue) and after (red) applying the GAN, and the difference in green.

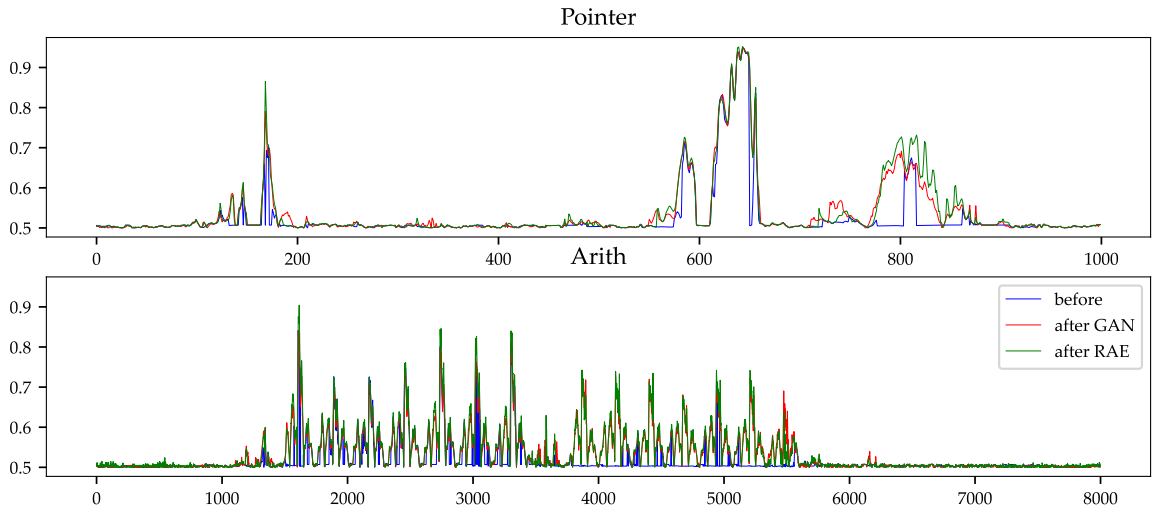

Figure 3.21: BRR obtained on the raw and cleaned datasets provided by the RAE and GAN

significant changes in the selected PoI list when applying the method described in [PC16] to the raw or cleaned datasets. Figure 3.24 supports this observation. The curves obtained considering the raw and cleaned datasets are very similar, although we observed that the unsupervised $\mathcal{T}$-values are lower at the time points where the RAE has corrected the saturation values. Using the RAE or CycleGAN, mitigation process do not produce significant changes in the proposed PoI selection from Perin *et al.* work [PC16]. With the RAE, the desaturation of points has the effect of lowering many previously identified high $\mathcal{T}$-values. Furthermore, this does not provide improvements on expected leakage areas (see Figure 2.2 for references). Using the CycleGAN, this has almost no effect on the PoI selection analysis, meaning that it is mostly caused by the presence of saturated values.

Overall, these results show that even on cleaned datasets with methods such as the RAE or CycleGAN, the PoI selection method proposed by [PC16] fails to extract relevant PoI on these datasets. This emphasizes that while leakage levels are higher and discrimination between classes easier, proposed state of the art PoI selection method is unable to provide

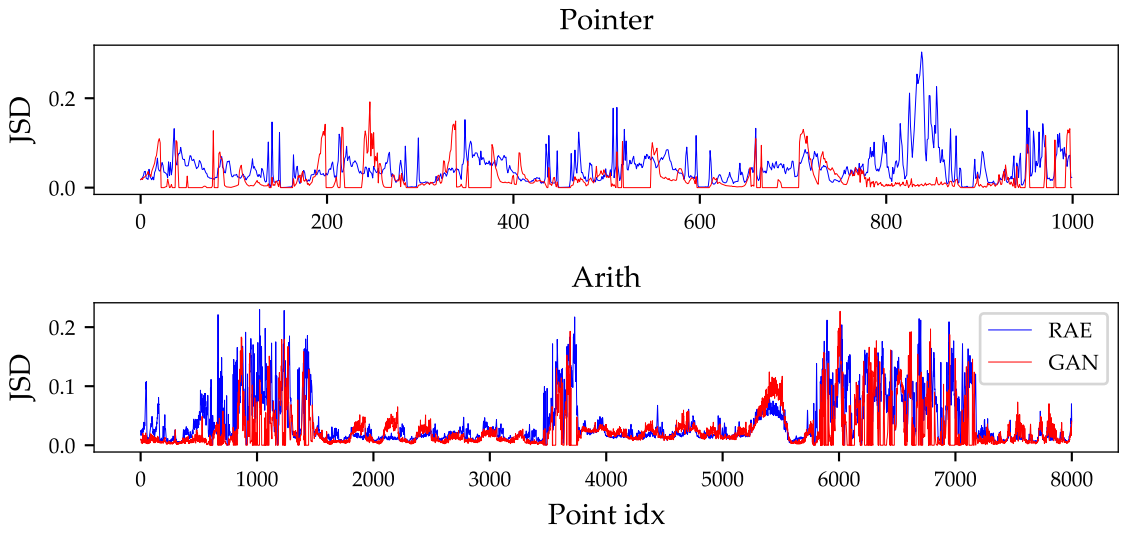

Figure 3.22: Jensen-Shannon divergence values between corresponding time points before and after the application of either the RAE or the CycleGAN.

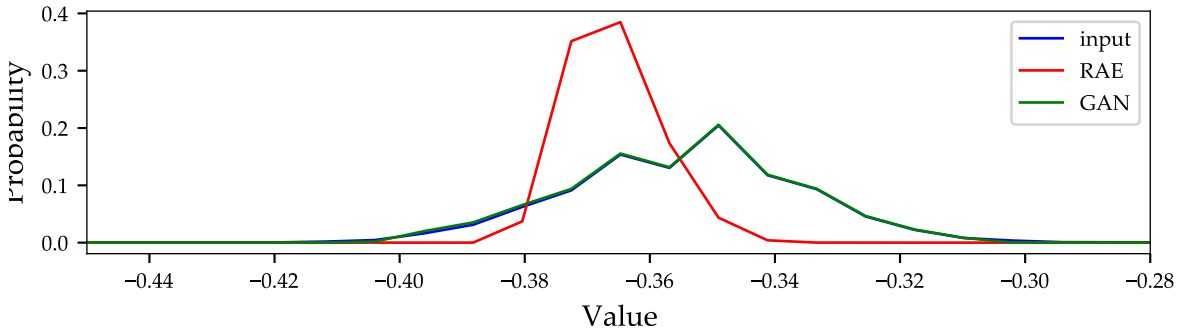

Figure 3.23: Distribution of time point 838 of $\mathcal{D}_{pointer}$ before and after applying the RAE or the GAN.

pertinent results. In the next chapter, an alternative method for selecting relevant PoI is presented.

3.4.4 Conclusion and Limits

We have investigated the interest of using autoencoder based architecture and GAN to mitigate the effect of outliers and saturated values on the detection and exploitation of the leakage. We found that the RAE and outlier-driven CycleGAN are efficient in correcting patterns without degrading the leakage. In addition, the outlier-driven CycleGAN framework we proposed offers more flexibility and control on the result than the RAE, since the values to be corrected are controlled while the others are secure and left unchanged. This prevents observing unexpected behaviors as with the RAE, where correctly distributed time points can be distorted through the process. Overall, we showed that problems of outliers and saturated values can be identified and treated with appropriate methods to enhance the exploitability of input patterns.

Still, the CycleGAN shows some limitations in its applicability. First, training GAN-based architectures can be difficult in practice, as GAN are known to exhibit problems such

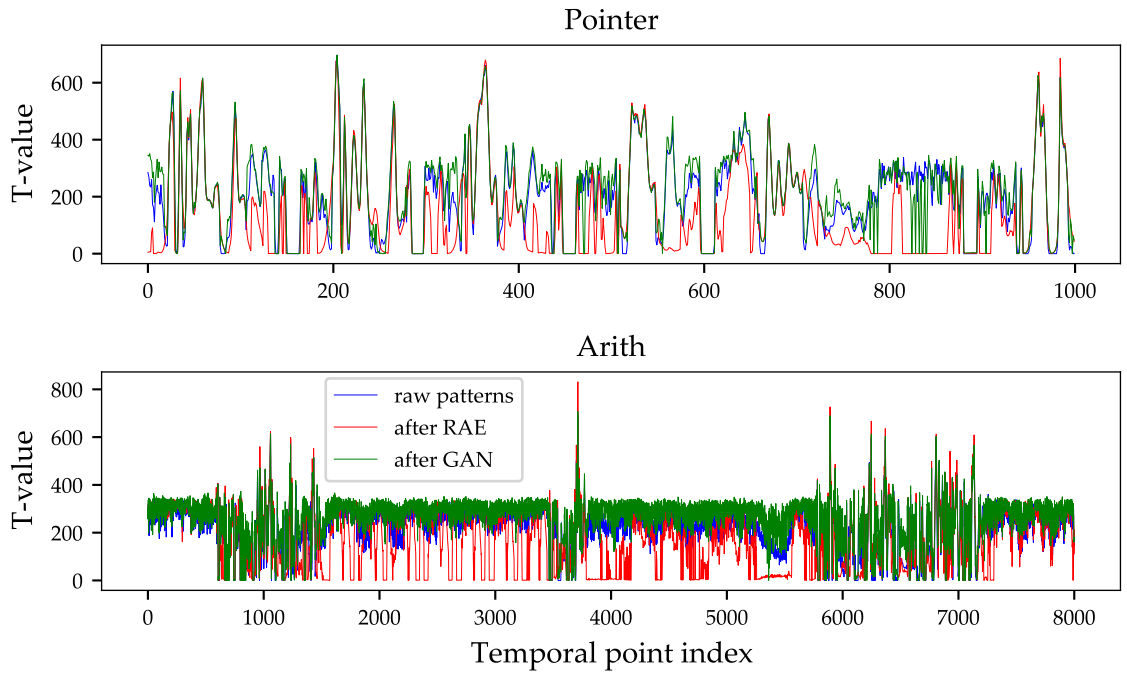

Figure 3.24: $\mathcal{T}$-values obtained after the initial clustering step of the attack proposed in [PC16] applied to the raw and cleaned datasets.

as hard convergence, mode collapse, or vanishing gradients. In addition, the architecture considered in this work could be improved for the outlier mitigation task by using a less complex one.

Finally, even if it seems possible to mitigate the effects of outliers and saturated values, we have shown, at least on two datasets, that it is not sufficient to make common PoI selection methods efficient when the leakage contained in traces is small. Indeed, the proposed state of the art method was not able to produce relevant PoI even after the cleaning process.

3.5 Self Organizing Maps

In this section, we consider the use of an alternative neural architecture for pattern extraction and visualization in the aim of partitioning patterns into clusters.

The following method was proposed by Kohonen in 1990 [Koh90] and is based on competitive learning instead of error learning. In this alternative training model, called Self Organizing Map (SOM), each node competes to represent a sample from the input data. This ensures preservation of distance between points through the mapping process. Furthermore, the SOM exhibits some properties such as the data topology conservation between the input high dimensional space and the lower subspace. They are used for various tasks in different fields such as banking [Kiv98], medical [CT07] or climate study [SL11].

3.5.1 Architecture definition

To describe the SOM architecture, let us consider that the set of input data is $X = \{\mathbf{x}_i\}_{i=1}^{n}$ with $\mathbf{x}_i \in \mathbb{R}^m$, and that the output of the SOM (or Kohonen layer) is a two-dimensional lattice V composed of a total of p nodes. Each output node $v_i \in V$ has an associated weight vector (or hidden layer) $\mathbf{w}_i \in \mathbb{R}^m$. This defines a weight matrix $W = \{\mathbf{w}_i\}_{i=1}^{p}$. Two topologies

for the projected space are usually considered, either a square grid or a hexagonal one. The latter one is often chosen because each node is surrounded by six immediate neighbors. The choice of the Konohen layer size defines the network capability. The Figure 3.25 gives a simplified view of the SOM architecture.

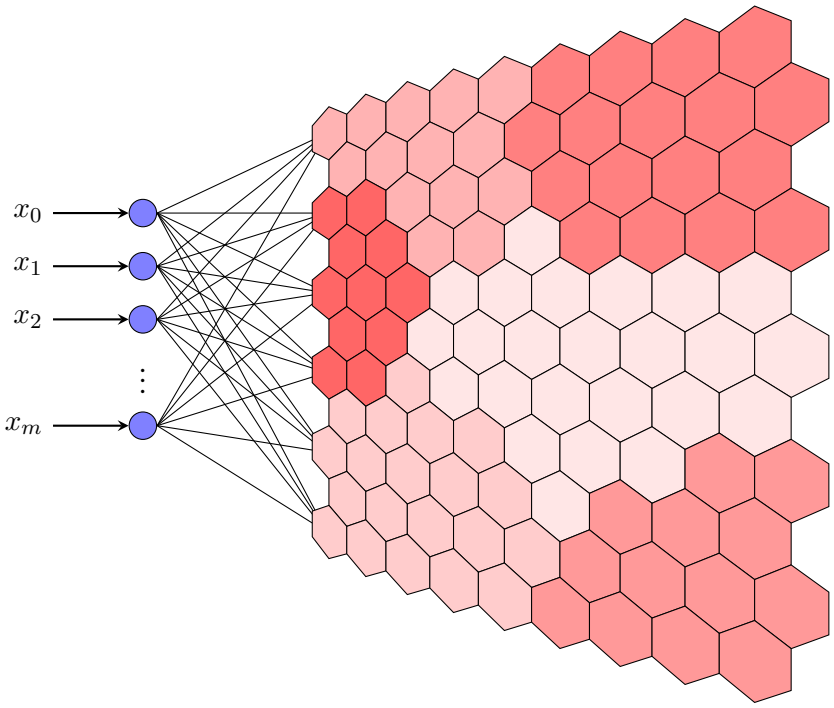

Figure 3.25: Simplified view of the SOM architecture

3.5.2 Optimization procedure

As mentioned above, the SOM training relies on competitive learning, so no loss function based on the prediction error is defined. Instead, a distance metric d is computed between a given input vector $\mathbf{x}_i$ and each weight vector $\mathbf{w}_j$, where $j \in \{1, \ldots, p\}$. The node with the smallest distance is called the Best Matching Unit (BMU) which index is given by:

$$\mathcal{B}(\mathbf{x}_i) = \underset{j \in \{1,\ldots,p\}}{\operatorname{argmin}} \; d(\mathbf{x}_i, \mathbf{w}_j). \tag{3.23}$$

This is then used for updating all weights at each step:

$$\mathbf{w}_j^{(s+1)} = \mathbf{w}_j^{(s)} + \alpha^{(s)} \cdot \theta(\mathcal{B}(\mathbf{x}_i), j)^{(s)} \cdot d(\mathbf{x}_i, \mathbf{w}_j^{(s)}) \tag{3.24}$$

where s is the current step, α the learning rate and θ a neighborhood function that defines how spatial neighbors of $\mathbf{w}_j$ are updated.

Different implementations can be used for θ, depending on the Kohonen layer topology and chosen update rules.

In its simplest form, one can set $\theta = 1$ for direct neighbors of the BMU while $\theta = 0$ for the rest of the nodes. Other functions such as the Mexican hat or the Gaussian method are also often considered [PK19]. The latter one (used in this work), while accurate, can cause longer training times due to the use of the exponential function. It is defined by:

$$\theta(\mathcal{B}(\mathbf{x}_i), j) = \exp\left(-\frac{||j - \mathcal{B}(\mathbf{x}_i)||}{2\sigma(s)^2}\right) \tag{3.25}$$

where the numerator is the distance on the map between the two nodes (depending on the topology considered), s the current step in the optimization and σ the radius parameter that controls the effect of the neighborhood function.

In the original work, the learning rate α decays over time as the number of steps s increases. This has the consequence to affect the range of θ function, and the convergence of weights from global to local estimates over time. However, this is only used in online learning, where at each step a training input $\mathbf{x}_i$ is randomly chosen from X. Instead, batch learning has been proposed as a more efficient alternative to online learning, where no learning rate is needed [KTKI02]. We used this approach in this work for faster training (see details in [PK19]). The network is trained for a defined number of epochs. Additionally, we consider an early stopping criterion, defined by the average global change of weights between steps. If the change is lower than a threshold ϵ for a given number of steps, this indicates network convergence and the training thus comes to an end.

3.5.3 Alternative for horizontal attacks

The use of the SOM can potentially provide some benefits over traditionally used attack schemes. In fact, it can be used as a reduction dimensionality technique.

Unlike the PCA which projects the input data into a p dimensional space, the SOM always maps the input to a two dimensional space (a plane), allowing for better visualization. In addition, the visualization of each weight across all nodes at a particular time point allows to observe the repartition of samples for this given point, allowing for better interpretation.

For these reasons, this architecture can be seen as a relevant candidate in the context of horizontal SCA. It is also important to note that the SOM is related to both the PCA and the k-means clustering algorithm.

If we define the SOM output layer with $p = 2$ nodes, it means that each input $\mathbf{x}$ can only be represented by one of the two available nodes, i.e. forcing the input to match one of the two nodes. In this case, if the Euclidean distance is chosen as distance metric d, the SOM behaves similarly to the k-means. Additionally, it has been showed that SOM is a non linear generalization of the PCA [Yin08, DCG16] and provides some advantages over PCA [LW05, LWM06]. As seen in previous sections, non-linear projections allow extracting more information and can be relevant in the context of SCA. For these reasons, we wanted to assess the applicability of the SOM architecture to high dimensional data, as well as its visualization and interpretation capacity after feature selection.

3.5.3.1 Application on high dimensional data

Let us compare the projection capability of a SOM network to assess if its structure can provide better low space representations compared to previously studied methods such as the PCA or autoencoder.

The SOM process was thus applied to the three datasets to obtain a two-dimensional space. For this experiment, we defined the number of output nodes to $p = 200$. While in the original SOM paper [Koh90], Kohonen advices to use $p = 5\sqrt{m}$ where m is the number of time points per pattern, we saw that in most experiments $p = 200$ was sufficient to avoid high nodes sparsity and allowed keeping reasonable training times, besides avoiding possible overfitting [PK19].

Figure 3.26 shows the projections we obtained into the two-dimensional lattice of all patterns for the three datasets considered. The points coloring represents the known class

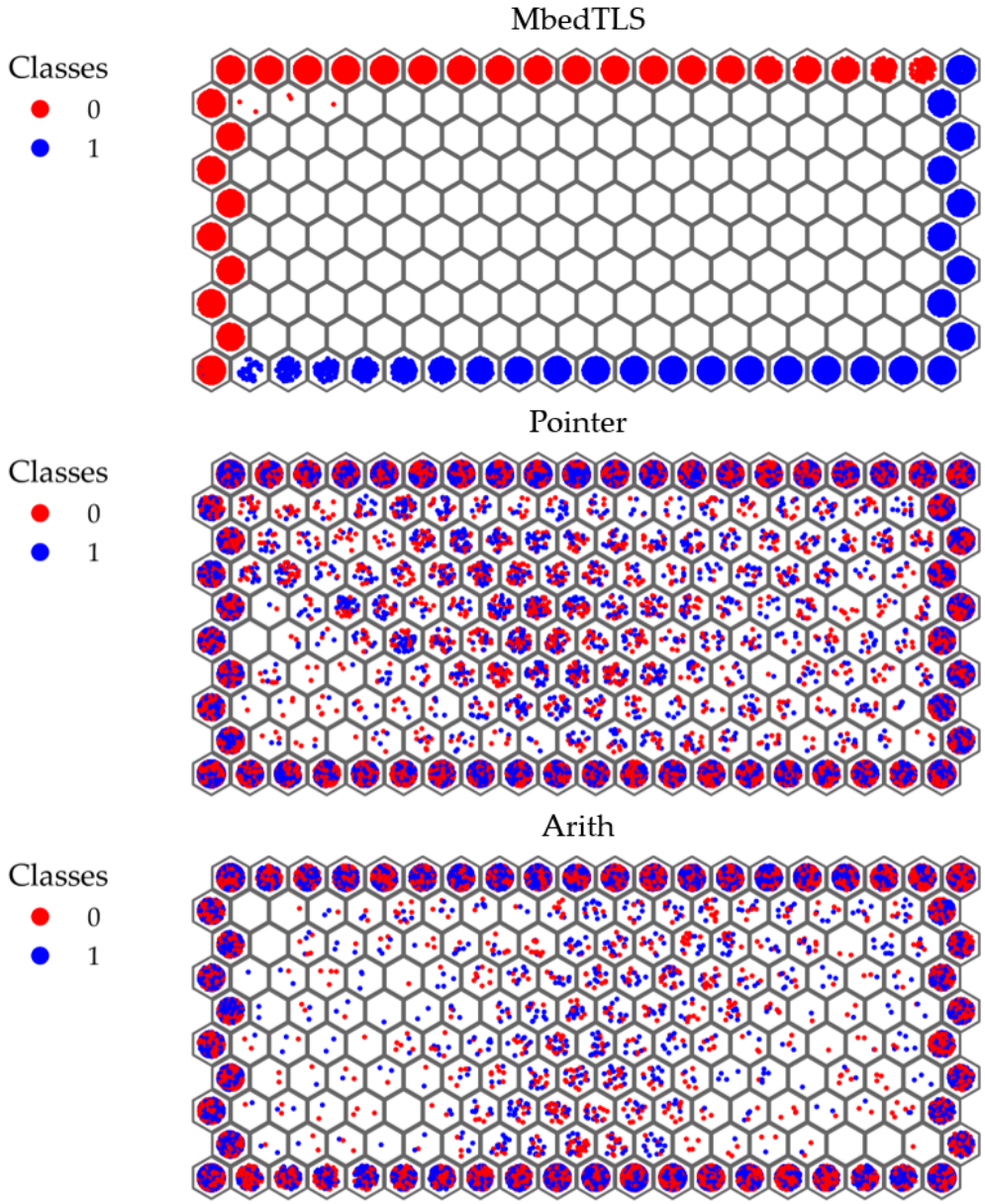

Figure 3.26: Projection obtained for the three datasets considered.

ownership which is used to observe the separability of patterns. Their position on the lattice map is determined by the BMU at the end of the network training (Equation 3.23). It can be seen that the $\mathcal{D}_{mbedtls}$ set is successfully projected onto the two-dimensional plane, as the classes are almost perfectly arranged and would be easily partitioned using a clustering algorithm. Only a reduced portion of the map is used to represent the input data in the considered subspace, meaning that the lattice size could be reduced for this dataset.

On the other hand, the projections from $\mathcal{D}_{pointer}$ and $\mathcal{D}_{arith}$ are not relevant in the context of classes discriminations. More output nodes are used, but the repartition of classes does not exhibit any exploitable behavior. As expected, the SOM is also sensitive to noisy data and cannot provide relevant projections if the amount of noise is too high. We thus applied the SOM procedure on the cleaned patterns provided by RAE and CycleGAN, but we did not get better results.

3.5.3.2 Application after features extraction

In this section, we study the potential added value of applying the SOM procedure before applying the k-means clustering algorithm during a clustering based horizontal attack. If we assume a correct feature extraction method, we want to know if the application of the SOM network has some benefits over clustering directly on extracted features.

As described above, the SOM allows projection to a two dimensional space. This can effect to enhance the visualization of the data. Furthermore, as the SOM preserves the topology from the input space to the projected space, this two dimensional space can be used for partitioning purposes afterwards. This is all the more interesting since the SOM projections have been shown quite resistant to noise because the SOM look for structure in the data, which is what the noise lacks [PK19].

For these reason, we compare the clustering accuracy of the space using the k-means algorithm. The k-means is applied either on the high dimensional data directly or on the projected two dimensional space from the SOM. To depict a comparable and relevant trends on the results, we consider the time points with the most leakage in terms of supervised $\mathcal{T}$-values. The results are presented on Figure 3.27 for different number of considered time points. For $\mathcal{D}_{pointer}$, the trends of the BRR vs the number of selected points obtained with

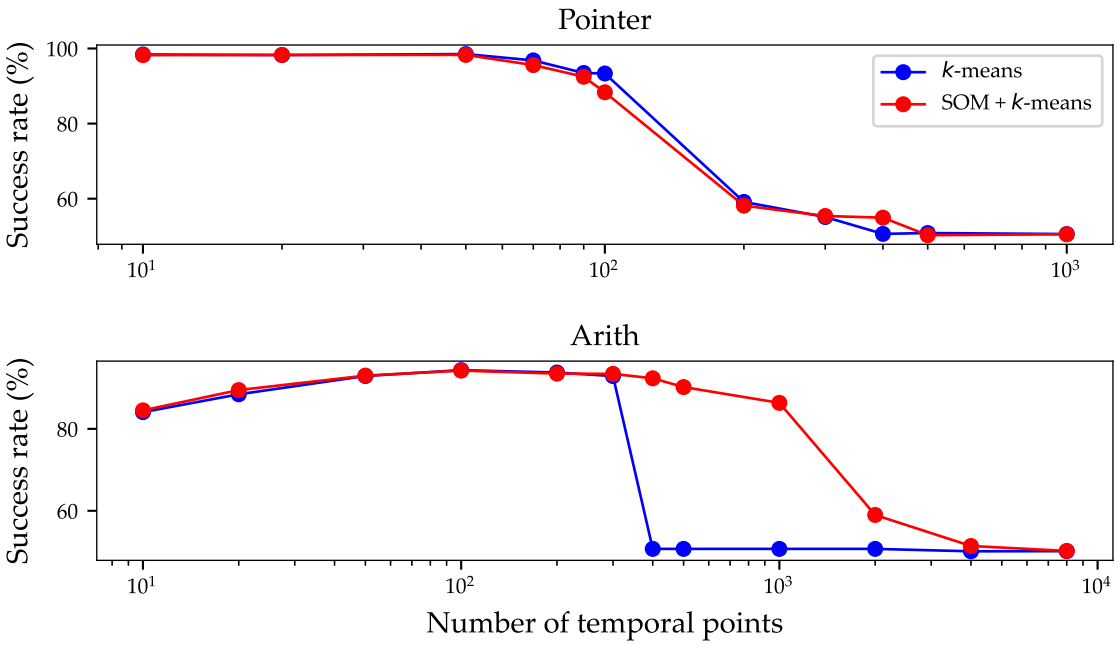

Figure 3.27: BRR vs. number of selected points with and without SOM dimensionality reduction.

and without applying the SOM procedure are nearly identical. There is no advantage in applying the SOM procedure. It should be noticed that both trends start decreasing as soon as the number of selected PoI rises above 100, which is the roughly the number of very large $\mathcal{T}$-values obtained when assessing the presence of leakages in a supervised manner (see Figure 2.2).

For $\mathcal{D}_{arith}$, the trends of the BRR vs the number of selected points obtained with and without applying the SOM procedure differs. Indeed, the BRR values obtained by using only the k-mean suddenly drop from about 90% to about 50% as soon as the number of selected points becomes higher than 300, although this number is well below the number of

very large $\mathcal{T}$-values obtained when assessing the presence of leakages (see Figure 2.2). This is not the case when the k-mean is applied to the projections obtained with the SOM. Indeed, the BRR is still close to 90% when 1000 points are selected. It even remains higher than 60% (value that is still better than that reported in [PC16]) when 2000 points are selected. Furthermore, we can see that the BRR only gets close to 50% when 4000 points are processed, i.e. when the number of points processed is about half the length of the pattern. This results support the claim of [PK19] according to which the projections provided by SOM have a certain resistance to noise. This suggests the interest of using SOM in the presence of some improperly chosen PoI during an attack.

To get more insight, we can observe the U-matrix of the SOM where the k-means fails to provide relevant partitioning, that is using the 300 best PoI on $\mathcal{D}_{arith}$. The U-matrix corresponds to the difference of each node weights with its neighbors [US90]. This can be used to visualize clusters on the two dimensional space. High values translates to large distance between nodes and thus a high distance between weight vectors and input space. Low values are synonyms of clusters formation where weights vectors are close in the input space. This visualization is particularly useful to estimate the distribution of clusters from the projection. This is not possible to do in the high dimensional space used when applying clustering directly. The U-matrix of the SOM projection on $\mathcal{D}_{arith}$ is depicted on Figure 3.28. One can

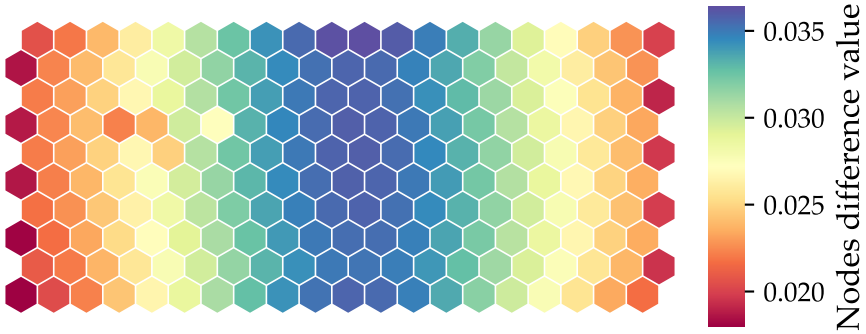

Figure 3.28: SOM U-matrix on $\mathcal{D}_{arith}$ using the 300 best PoI.

observe two clearly separated clusters illustrated by orange and red nodes colors with low nodes difference values. In this scenario, it is straightforward to partition into two classes the observed values, based on their associated BMU.

In practice, the SOM displays a better resilience to added noise that the k-means clustering applied directly on the 300 dimensional data. It can be seen as a relevant method way to assess and visualize the projection resulting from a PoI selection procedure. This allows to give insights of the correctness in the chosen PoI for the clustering phase. If the observed U-matrix does not depict several clusters, it can provide clues that clustering on this high dimensional data will not provide consistent partitioning.

3.5.4 Conclusion and limits

This section considered the SOM as an alternative method for dimensionality reduction and visualization of projections, in a way to ease the classes partitioning process. This showed promising results compared to clustering application on high dimensional data and provide an additional tool for assessing the relevance of the chosen time points in the context of a PoI selection strategy. On a clean dataset, such as $\mathcal{D}_{mbedtls}$, the SOM seem to work perfectly. On noisy dataset, such $\mathcal{D}_{pointer}$ and $\mathcal{D}_{arith}$, the SOM showed more robust results compared

to the direct application of clustering technique on high-dimensional data.

The main drawback of this approach is the need for parameterization of the SOM. This comprises the Kohonen map size and the choice of neighborhood function, although in the cases studied the Gaussian neighborhood function always showed better performance, assuming a trade-off in computation complexity.

Overall, while the SOM cannot be seen as a standalone alternative for patterns discrimination in a SCA context, it can be used as an extension tool in the analysis to validate the PoI selection results and provide useful insights about the partitioning capability of the considered space. We also saw that the usage of the SOM give some robustness to noise compared to applying clustering directly on the high dimensional data.

3.6 Global Conclusion

In this Chapter, the usage of unsupervised neural based architectures has been studied to provide improvements over state-of-the art method in the context of horizontal SCA.

Firstly, we considered compression alternative to traditionally used PCA method. This showed that autoencoders, while powerful to extract relevant features and detect non-linear relationships between data time points, were also sensitives to noise in high dimensional spaces. The same apply to variant architectures of the classical autoencoders.

Then, we studied two neural based candidates that can be used for abnormal values cleaning, that is outliers and saturated values. Using the RAE and CycleGAN, we showed that these architectures were able, in a completely unsupervised manner or using directional learning to successfully mitigate such values. This translated to improved leakage levels and better exploitability in a clustering context. This however did not provide improvements in the current state-of-the-art PoI selection method, emphasizing that the way the PoI are assessed and selected could be improved by considering a different leakage model.

Finally, the SOM architecture has been studied as a tool for improving visualization of high dimensional spaces and resilience to noise. This has showed that its application is relevant for analyzing the projected space partitioning capacity and provide an explainable tool instead of clustering directly on high dimensional spaces.

Overall, using neural networks in an unsupervised context can sometimes be tricky since no direct supervision of the training is possible, assuming the associated labels are not known. However, as showed these methods can provide relevant and powerful tools even they are optimized using alternative objectives that do not necessitate supervision of unknown labels.

In the end, it is still important to note that feature selection can be performed using compression methods in high dimensional space, but if the perceived relevant information is distilled in high amount of non-informative noise and additional behavior, this will not produce desirable effects. Instead, in these cases, univariate PoI selection is more prevalent, even it should be adapted based on the considered data.

Chapter 4

Statistical Machine Learning Approach

Abstract

This chapter provides a new points of interest selection method for univariate analysis. Additionally, a novel metric for the quantification of exploitable leakage is introduced and is showed to provide better estimation that traditional metrics. Parts of this chapter have led to the publication of [COM24] at CARDIS 2023 conference.

4.1 Introduction

Following results from previous chapter, reader can see that, while the application of neural network in a completely unsupervised attack context for horizontal attacks is relevant and showed interesting results for tasks such as traces cleaning and outliers mitigations, their impact for leakage selection is limited by the presence of high noise levels in the input patterns. As seen in practice, only few temporal points are usually conveying leakage information and thus applying such compression methods showed to be inefficient to extract relevant leakage.

Furthermore, the need of hyperparameters tuning can make the usage neural networks for unsupervised horizontal attacks inapplicable.

To that aim, univariate PoI selection is more appropriate in high noise environments. We saw in previous chapters limits of state-of-the-art methods for PoI selection. While neural networks are powerful tools in contexts of multivariate data analysis, they may not be the best fit for this particular problem. We would thus like to provide improvements over currently used univariate PoI selection methods without the use of neural networks models. Instead, classical machine learning approaches as well as statistical analysis is applied to improve previously proposed methods for horizontal attacks.

4.2 Univariate PoI Selection

In this part, we introduce a univariate PoI selection based on the Gaussian mixture model. Following previous Chapters, we propose an adapted leakage model and PoI selection framework more resilient to noise. Furthermore, we provide a metric to quantify and assess the quality of mixtures to be exploitable in a clustering process.

4.2.1 The ideal leakage model

Following previous works on the selection of PoI in the context of unsupervised horizontal attacks, namely Perin *et al.* [PITM14, PC16] and Nascimento *et al.* [NC17], we wish to provide an improved method for PoI selection. As seen in Chapter 2, the actual observed distributions of time points may often be different from the ones considered in current state-of-the-art contributions. In addition, some characteristics on the distributions of leaking time points (such as known prior) are not exploited to provide a better selection of relevant PoI. For example, it is necessary to redefine a leakage model that takes into account possible additional behaviors of other external or internal processes to the target Integrated Circuit (IC). To do this, we first formalize the initial leakage model and derive it so that it can be exploited under real-world conditions.

To fully understand the problematic of appropriate PoI selection, we define hereafter the expected leakage manifestation and the assumed leakage model for the context of horizontal attacks.

4.2.1.1 Gaussian mixture distribution

In a k class problem, where the patterns are uniformly sized and perfectly aligned, a leaking time can be modeled as a mixture of Gaussian components. Thus, the observed distribution $p(x)$ of a leaking time point is:

$$p(x) = \sum_{i=1}^{k} \pi_i \mathcal{N}(x|\mu_i, \sigma_i^2) \tag{4.1}$$

with π_i the mixture coefficient of the ith component s.t. $\pi_i \geq 0 \quad \forall i$ and $\sum_{i=1}^{k} = 1$, and where the normal distribution is defined by:

$$\mathcal{N}(x|\mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2} \tag{4.2}$$

Thus, a set of patterns can be viewed as a collection $X = (X_1, \ldots, X_m)$ of random variables $X_i \sim p(x)$, each of which time point is a Gaussian mixture with its own set of parameters $\theta = \{\pi_i, \mu_i, \sigma_i^2\}_{i=1}^{k}$.

4.2.1.2 Parameters estimation

In the context of unsupervised horizontal analysis, there is no knowledge about the underlying mixture components associated with each time point. However, it is possible to approximate them through estimation of the maximum likelihood of the mixture distribution. Given a set of observations $x = (x_1, \ldots, x_n)$, the likelihood is given by:

$$p(x|\theta) = \prod_{i=1}^{n} p(x_i|\theta) \tag{4.3}$$

while in practice the log-likelihood is preferred instead for numerical stability:

$$\begin{aligned} L(\theta) &= \log p(x|\theta) \\ &= \sum_{i=1}^{n} \log p(x_i|\theta) \end{aligned} \tag{4.4}$$

In terms of Gaussian mixture model, this gives:

$$L(\theta) = \sum_{i=1}^{n} \log \left[\sum_{j=1}^{k} \pi_j \mathcal{N}(x_i | \mu_j, \sigma_j^2) \right] \tag{4.5}$$

To provide the best estimate of the mixture model against the observed data, we want to find parameters θ that maximize this likelihood:

$$\theta^\star = \underset{\theta}{\operatorname{argmax}}\, L(\theta) \tag{4.6}$$

This is however intractable as the mapping from θ to $L(\theta)$ is not convex. Instead, we can use the EM to find a local maximum of the likelihood. Assuming that latent variables $z = (z_1, \ldots, z_n)$ are used to generate x, it becomes easy to solve Equation 4.6. The log-likelihood thus becomes:

$$L(\theta) = \log p(x, z|\theta) \tag{4.7}$$

Further details can be found in [Bon19]. The optimization procedure can then be described by the iterative application of two distinct steps. First, the expectation of the log-likelihood is computed given the current estimates of the parameters $\theta^{(t)}$:

$$Q(\theta|\theta^{(t)}) = \mathbb{E}_{z|x,\theta^{(t)}} \left[\log p(x, z|\theta) \right] \tag{4.8}$$

Then, update parameters such that it maximizes the log-likelihood.

$$\theta^{(t+1)} = \underset{\theta}{\operatorname{argmax}}\, Q(\theta|\theta^{(t)}) \tag{4.9}$$

The EM algorithm convergence is reached when no changes in the estimation of the log-likelihood are observed. From this procedure, the estimated parameters set θ can be used to assess the quality of the underlying mixture and components and whether or not a time point is a potential leakage point. In the next sections, we describe several criteria that should be considered to ensure relevant selection of PoI in the context of horizontal SCA.

4.2.1.3 Limits of the T-test

In Chapter 2, we showed that the T-test is not the most appropriate metric for quantifying leakage in a two-class problem. In fact, the use of first and second order moments (mean and variance) does not accurately reflect the exploitability of time points using a clustering approach. As said, it is due to several problems we highlighted.

On one hand, its value is not bounded and therefore no possible bijection can be made between the obtained $\mathcal{T}$-values and the correctness of the classifications of the considered time points using a clustering algorithm. This leads to the impossibility of quantifying the maximum value that can be considered satisfactory. For example, if the classes on the mixture are well separated and fully distinguishable, i.e., linearly separable, increasing the difference between the means or decreasing the class variances will result in higher $\mathcal{T}$-values, but will not necessarily translate into higher time point exploitability.

In the context of a SCA leakage analysis, whether it is carried out in a supervised way, knowing the class labels, or in an unsupervised way (following Perin's *et al.* work, for example [PC16]), the quality of a time point is often defined as follows. If the $\mathcal{T}$-value is greater than the critical value for the test, the time point contains information about the class leak-

age. However, further analysis to select the most relevant PoI among the non-rejected ones is not as straightforward as it should be. Often, the PoI with the highest $\mathcal{T}$-values (peaks) compared to the average $\mathcal{T}$-values obtained at each time point are chosen as the best candidates. This does not take into account the values, but only the amplitude compared to the average $\mathcal{T}$-value. This does not provide sufficient information about the actual leakage and exploitability of a given time point.

In addition, comparing $\mathcal{T}$-values from different time points can be confusing, since different scaling of the points would result in different $\mathcal{T}$-values even if the same amount of underlying leakage is expressed. For more insights about the limits of the T-test , see Section 2.3.2.3.

Instead of this, we want to be able to quantify the uncertainty in the observed mixture in a more general way. Assuming we know the underlying distribution of the classes, instead of considering moments such as the means or variances, we can assess the differences in the distributions themselves. More precisely, we would like to be able to characterize the leakage in a straightforward manner by computing a bounded score indicating how easily the components of the mixture can be separated.

4.2.1.4 The probability of confusion

To do this, we define the probability of confusion between mixture components. When observing a given value, it indicates the amount of uncertainty that exists in membership to identified classes of the value. Generalized to k classes, the posterior probability $p(c_i|x)$ of class c_i given value x can be computed by applying the Bayes's rule:

$$p(c_i|x) = \frac{p(c_i)p(x|c_i)}{p(x)} \tag{4.10}$$

where $p(c_i)$ is the prior probability of class c_i, that is the mixing coefficient π_i and $p(x|c_i)$ the likelihood of observing value x given class c_i. By applying the law of total probability, $p(x)$ can be expressed as the sum of weighted likelihoods over all classes:

$$p(c_i|x) = \frac{p(c_i)p(x|c_i)}{\sum_{j=1}^{k} p(c_j)p(x|c_j)} \tag{4.11}$$

Finally, previous equation can be reformulated using the Gaussian mixture model:

$$p(c_i|x) = \frac{\pi_i \mathcal{N}(x|\mu_i, \sigma_i^2)}{\sum_{j=1}^{k} \pi_j \mathcal{N}(x|\mu_j, \sigma_j^2)} \tag{4.12}$$

The probability of confusion p_{conf} is then given by:

$$p_{conf} = 1 - \int_{\mathbb{R}} p(x) \left[\max_{i=1}^{k} p(c_i|x) \right] dx \tag{4.13}$$

In cases where $k = 2$ classes are considered, this can be simplified to:

$$p_{conf} = \int_{\mathbb{R}} p(x) \left[\min_{i=1}^{k} p(c_i|x) \right] dx \tag{4.14}$$

The p_{conf} metric describes the overlap of components in a given mixture. Its bounds correspond to the separability of the mixture components. The lower bound $p_{conf} = 0$ means that components can be fully discriminated without errors. Conversely, the upper bound

$p_{conf} = 1 - \sum_{i=1}^{k} \frac{1}{k}\pi_i$ means that components are completely overlapped. If discrete random variables are considered instead of continous ones, the integrals of Equation 4.13 and 4.14 can be replaced with sums over the discrete values. Figure 4.1 shows illustrations of the mixture for $k = 2$ and $k = 3$ as well as their posterior probabilities and computed p_{conf} area. We

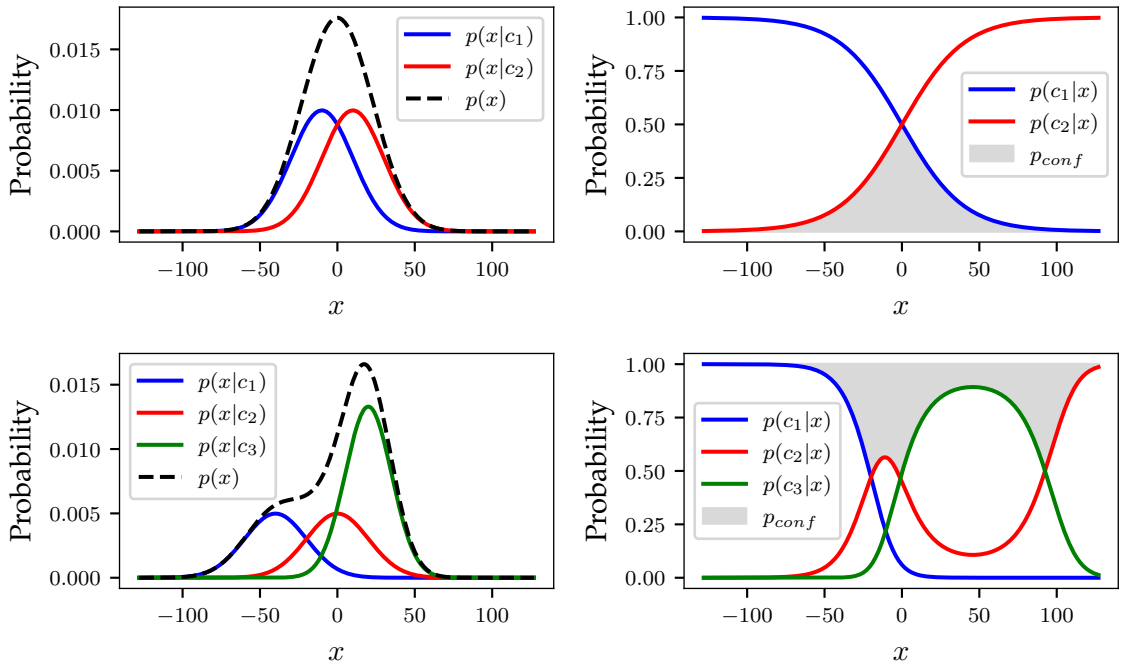

Figure 4.1: Illustration of the probability of confusion p_{conf}, for $k = 2$ (top row) and $k = 3$ (bottom row). The left plots show the mixture components, while the right plots show the posterior probabilities and the area materializing the p_{conf}.

can see that on complex mixtures, the p_{conf} provides a precise estimate of the components overlapping thus the membership uncertainty of values to the various components. To get more insight into the relevance of the proposed metric, we compared the p_{conf} to the T-test in the context of a supervised leakage assessment. For this purpose, the TVLA was performed, where the considered metric is applied at each time point. The results obtained on the three datasets chosen for this work are depicted on Figure 4.2 At first sight, the results obtained,

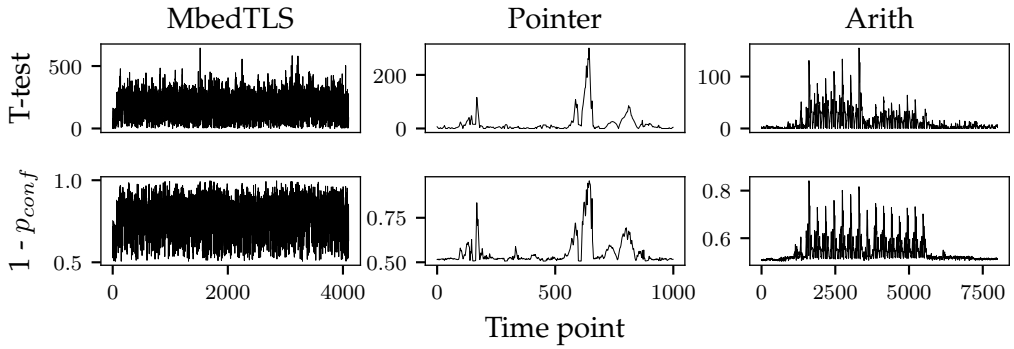

Figure 4.2: T-test (top row) and $1 - p_{conf}$ (bottom row, inverted for readability) applied on each time point, for the three datasets considered.

for the three datasets, using the T-Test and p_{conf} are similar. However, there are differences

especially for $\mathcal{D}_{pointer}$ and $\mathcal{D}_{arith}$. On $\mathcal{D}_{pointer}$, the p_{conf} metric gives (relatively) wider estimates of leakage peaks than the T-test (around time points 640 and 840), and higher values around time point 175. For $\mathcal{D}_{arith}$, more weight is given to the second part of the arithmetic swap, between points 3500 and 5500.

To further demonstrate of the relevance of the p_{conf} as an adapted PoI selection metric, we performed a clustering attack using up to the 100 most relevant PoI ranked according to either the T-test or the p_{conf}. The objective was to compare the robustness and exploitability by a clustering algorithm of these selected time points using both methods. Figure 4.3 shows the BRR obtained by applying the Fuzzy k-means on the most leaking time points on the three datasets considered. As expected, the p_{conf} ranks the times points more correctly

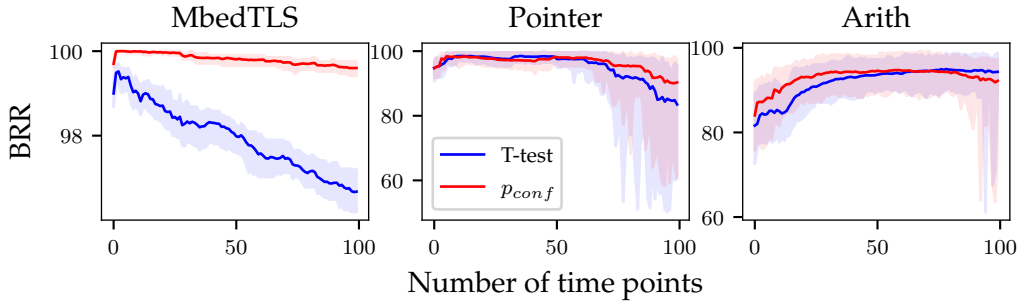

Figure 4.3: BRR scores obtained, on the three datasets, after applying the Fuzzy k-means clustering on the most leaking PoI, identified using the T-test or the p_{conf} metric with the knowledge of the labels.

according to their exploitability with clustering algorithm than the T-test. Indeed, on the three datasets considered, the use of the p_{conf} provides higher BRR with fewer time points. This confirms that the p_{conf} provides a more precise estimate of the underlying leakage exploitation.

4.2.2 Application for horizontal attacks

We have shown in previous sections that the proposed p_{conf} metric is relevant for assessing the leakage level of time points in a supervised attack context. While it is equally applicable in an unsupervised context, some prerequisites must be met in order for the values obtained to be meaningful. As mentioned in Chapter 2, applying the T-test statistic to a clustering result does not provide meaningful leakage measurements, even when adding a ratio parameter [PC16]. Therefore, we describe below the different criteria that should be checked in order to obtain acceptable PoI candidates.

4.2.2.1 Classes repartition

In the analysis of mixtures for the identification of relevant leakage candidate points, the repartition of classes is of crucial importance. We assume for the rest of this work that the expected classes repartition is known by the attacker beforehand. This requires the knowledge of the attacked algorithm and implementation details. We define the expected classes repartition when $k = 2$ as π_0. Each possible class of the secret d_i can be formulated as a Bernoulli trial, which Probability Mass Function (p.m.f) is given by:

$$P_{bern}(x) = \pi_0^x (1 - \pi_0)^{1-x} \tag{4.15}$$

where $x \in \{0, 1\}$ is the value of the outcome. Assuming the independence of trials for each value, we can thus formulate the process of generating classes values as a Bernoulli process. It is then possible, given these independent trials, to count the overall number of success as a Binomial distribution $B(n, \pi_0)$, with p.m.f:

$$P_{binom}(x, n) = \frac{n!}{x!(n-x)!}\pi_0^x(1-\pi_0)^{n-x} \tag{4.16}$$

where $x \in \{0, \ldots, n\}$ is the number of successful outcomes and n the total number of trials. Considering that x is observed and formulated as Y, we can define the following statistical test T_{repart}:

$$\begin{aligned} H_0 &: Y \sim B(n, \pi_0) \\ H_1 &: Y \nsim B(n, \pi_0) \end{aligned} \tag{4.17}$$

If Y does not follow a Binomial distribution with the given parameters, it does not result from the expected Bernoulli generation process. This test allows to ensure that the observed class repartition is relevant according to the prior π_0. Indeed, in a horizontal SCA context, if the selected PoI do not match the prior repartition expected from the targeted algorithm implementation, it is unlikely that it would provide relevant information about classes discrimination. For this reason, we would thus reject all time points if their repartition does not follow a Binomial distribution with parameter π_0.

4.2.2.2 Means difference

As seen in Chapter 2, the separability of components is related to their means. For this purpose, we look for a large difference between the component means $|\mu_1 - \mu_2|$. However, we will see in the following sections that a more relevant estimator of the separability of classes can be defined. Therefore, we do not directly quantify the component distances from the estimated mixture parameters.

4.2.2.3 Variance disparity

In addition, if we assume that the all classes are affected by the same measurement noise, the identified mixtures components should display a variance in a similar order of magnitude. We propose to assess the variance disparity between the two studied components with the F-test of variance equality T_{variance}.

$$F = \frac{\sigma_1^2}{\sigma_2^2} \tag{4.18}$$

so that null and alternative hypotheses are formulated as:

$$\begin{aligned} H_0 &: \sigma_1^2 = \sigma_2^2 \\ H_1 &: \sigma_1^2 \neq \sigma_2^2 \end{aligned} \tag{4.19}$$

If H_0 is rejected in favor of H_1, we conclude that the components do not belong the same leakage process. As a result, in the univariate analysis of all time points, a relevant point is expected to pass both the class repartition test and the variance test.

4.2.2.4 Additional noise components

The assumption that the points in exponentiation or scalar multiplication patterns follow a Gaussian mixture distribution is not always true. To be convinced, one can think about points associated only with internal multiplication operations or points associated with no operation. Such points are expected to be normally distributed due to measurement noise (at least to follow a distribution displaying a single mode) and do not contain any information about the secret bit of the exponent or scalar.

Similarly, if the leakage hidden in a point is too small with respect to the measurement noise, the distribution of the latter, while being a mixture, remains unimodal and is extremely difficult to exploit with clustering techniques.

On the other hand, the observation of the distribution can depict more than k components. As seen in Chapter 2, some time points on $\mathcal{D}_{arith}$ show more than two components. This is caused by the presence of outliers or saturated values and should be taken into consideration. Trying to model the mixture as the sum of $k = 2$ components while it is in reality built from $k > 2$ underlying components would lead to incorrect estimates of the densities and thus of the distribution parameters.

To avoid these eventualities, it may be necessary to redefine the actual observed mixture model by taking into account v additional noisy components, where each noise component is assumed to also follow a Gaussian distribution. This thus gives:

$$p(x) = \sum_{i=1}^{k} \pi_i \mathcal{N}(x|\mu_i, \sigma_i^2) + \sum_{i=1}^{v} \pi_i \mathcal{N}(x|\mu_i, \sigma_i^2) \tag{4.20}$$

with $\sum_{i=1}^{k+v} \pi_i = 1$. However, this redefinition of the expected leakage model affects the way the mixture is evaluated. Consequently, modified and additional conditions should be considered to classify an observed mixture as relevant.

For clarity, in the rest of this section, mixture components are sorted in decreasing order of mixing coefficient values s.t. $\pi_1 \geq \pi_2 \geq \ldots \geq \pi_{k+v}$.

First, we will consider the k informative components as the ones with the highest mixing coefficients. In case of $k = 2$, we want that $\pi_1 + \pi_2 \geq \beta$, where β is a threshold value. This ensures that information components are dominant over all observed components. If this is not the case, more noise than leakage will be observed and thus will not be exploitable in further analysis.

Second, in the presence of noise components, the classes repartition test should also be adapted. For this, we use the Multinomial distribution $M(x, n)$ instead of the previously considered Binomial distribution. In the case of $k = 2$ with equal classes repartition, the null and alternatives hypotheses are now given by:

$$\begin{aligned} H_0 &: \pi_1 = \pi_2 = \frac{1 - \sum_{i=k}^{v} \pi_i}{2} \\ H_1 &: \pi_1 \neq \pi_2 \end{aligned} \tag{4.21}$$

The null hypothesis is accepted if $p > \alpha$ where p is computed against the F-distribution. The variances disparity test of information components stays unchanged.

Finally, in the same way as before, if these tests are passed, we compute the p_{conf} of the mixture but only on the information components. In the presence of v supplementary noise

components, the weights of the noise components prior are included as follows:

$$p_{conf} = \sum_{i=k}^{v} \pi_i + \sum_{x \in X} P(x) \left[\min_{i=1}^{k} P(c_i|x) \right] \tag{4.22}$$

From this formulation, it means that the lower bound is defined by the sum of the weights of noise components in the mixture.

It is also very important to note that, in order to obtain a relevant estimation of the p_{conf} using the estimated mixture components through EM estimation, the discrete value bounds defined by the resolution and the settings of the oscilloscope should be taken in account. Indeed, as the produced variances can step out the bounds, estimating the p_{conf} outside the measured values range does not correctly reflect the actual overlapping. To prevent this, we calculate the p_{conf} only on the initial values $x \in X$. Figure 4.4 shows an example of p_{conf} computed on the discrete values, after fitting the Gaussian mixture model and estimating parameters using the EM algorithm.

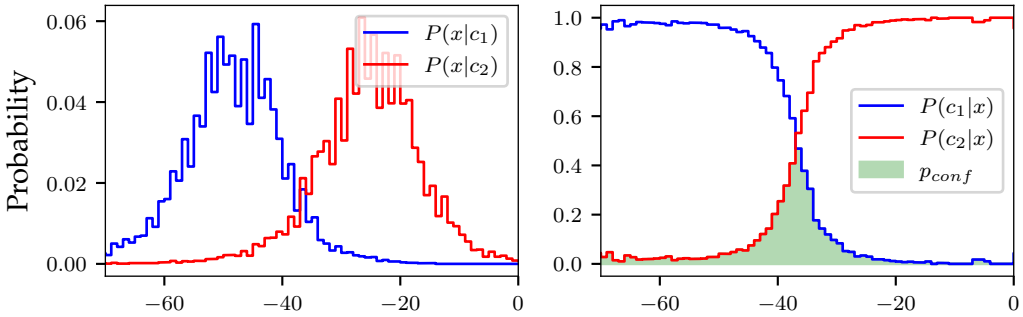

Figure 4.4: Computed p_{conf} on discrete data after the EM process (time point 641 on $\mathcal{D}_{pointer}$).

4.2.2.5 Proposed PoI selection

To perform an efficient PoI selection, based on the aforementioned criteria, we suggest and describe hereafter the following procedure. From the observed set matrix $\mathbf{X} \in \mathbb{R}^{n \times m}$. We consider each time point independently $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_m)$ where $\mathbf{x}_i$ is vector and the ith column of $\mathbf{X}$.

In order to provide consistent statistical analysis on the studied patterns, we consider a subset $\mathbf{r} \subseteq \mathbf{x}_i$ of q elements such that $\mathbf{r}$ elements are uniformly chosen.

Then, we estimate the parameters of the observed mixture of $\mathbf{r}$ using the EM algorithm. After convergence, we can apply the previously described statistical tests in order to assess the quality of the mixture. If the number of noise components $v = 0$, the Binomial test is applied, otherwise if $v > 0$, we use the Multinomial test instead. The choice of v is done by visual inspection and is fixed for all time points of a dataset.

Finally, if all tests are passed, we compute the probability of confusion p_{conf}, following Equation 4.14 if $v = 0$, or Equation 4.22 if $v > 0$. Otherwise, the obtained p_{conf} for that subset $\mathbf{r}$ is simply equal to 1 to reflect the non exploitability.

This analysis process is repeated h times over different subsets $\mathbf{r}$ of $\mathbf{x}_i$ and produced p_{conf} averaged. This approach is necessary to allow better generalization of the mixture model on the observed data and to prevent biases from random initialization of the EM algorithm. Note that the value of q has to be chosen sufficiently large to allows accurate density estimation.

This whole process is applied on each time point from $\mathbf{X}$. At the end, this yields a vector $\mathbf{b}$ of m elements such that b_i correspond to the p_{conf} from the analysis of the ith time point. We also define the PoI set $\mathcal{S}_{poi}$ with the indexes of all time points where $b_i < 1$, meaning that all tests passed. The complete analysis procedure is summarized on Algorithm 6.

Algorithm 6 PoI selection procedure with mixture statistical analysis

Require: $\mathbf{X}^{n\times m}, q, v, h$
Ensure: $\mathbf{b}^m, \mathcal{S}_{poi}$
  for $i \in \{1, \ldots, m\}$ **do**
    repeat h **times**
      $\mathbf{r} \subseteq \mathbf{x}_i : \mathbf{r} = (\{\mathbf{x}_{i_j}\})_{j=1}^{q}$ s.t. $j \sim \mathcal{U}(\{1, \ldots, n\})$.
      $\theta \leftarrow \mathrm{EM}(\mathbf{r}, v)$ ▷ Find mixture parameters by applying EM (Section 4.2.1.2)
      $T_1 \leftarrow T_{\text{repart}}(\pi)$ ▷ From section 4.2.2.1 if $v = 0$, Section 4.2.2.4 otherwise
      $T_2 \leftarrow T_{\text{variance}}(\sigma^2)$ ▷ From Section 4.2.2.3
      if T_1 and T_2 **then**
        $b_i \leftarrow b_i + p_{conf}$ ▷ Compute p_{conf} from Section 4.2.1.4
      else
        $b_i \leftarrow b_i + 1$ ▷ Maximum uncertainty otherwise
      end if
    end
    $b_i \leftarrow \frac{1}{h} b_i$ ▷ Average on h experiments
  end for
  $\mathcal{S}_{poi} = \{i \in \{1, \ldots, m\} | b_i < 1\}$
  return $\mathbf{b}, \mathcal{S}_{poi}$

From the application of the aforementioned procedure, we can decide to either exploit the set of identified PoI as it is for the attack phase, or we can expand and improve this set to strengthen the exploitability of PoI, as we describe in the next section.

4.2.2.6 Neighbors correlation analysis

Following the previously described PoI selection procedure, this process yields a set of PoI $\mathcal{S}_{poi}$ that describes the indexes of time points that are relevant candidates based on the carried statistical testing.

Still, in the context of acquired signals from a physical process, points at a time t and point $t + 1$ are more likely to share the same information than points far apart in time. Indeed, if we observe on Figure 4.5 the pairwise Spearman correlation matrix on $\mathcal{D}_{pointer}$ and $\mathcal{D}_{arith}$, we can see that points along the diagonal are highly correlated. We elaborate on the specific choice of the Spearman's correlation hereafter. Even some points that are far apart sometimes display high Spearman correlations. However, during the mixture analysis process, many time points which are close (and highly correlated) display different p_{conf} values. For example if we compare point indexes 643 and 640 on $\mathcal{D}_{pointer}$, displayed on Figure 4.6, their global distribution is almost the same, they share a high Spearman correlation of about 0.8, but point 643 gives a low p_{conf} value while point 640 fails to meet defined selection statistical criteria. This behavior can be due to miss estimation of the mixture components from two time points, or the presence of some outliers or saturated values.

To overcome this issue, we propose to extend the set of PoI $\mathcal{S}_{poi}$ by considering excluded time points if they share a high enough correlation with previously selected PoI. For this, we choose the Spearman correlation coefficient. While several candidates can be used to capture the correlation between two variables, we say that the Spearman correlation is one of

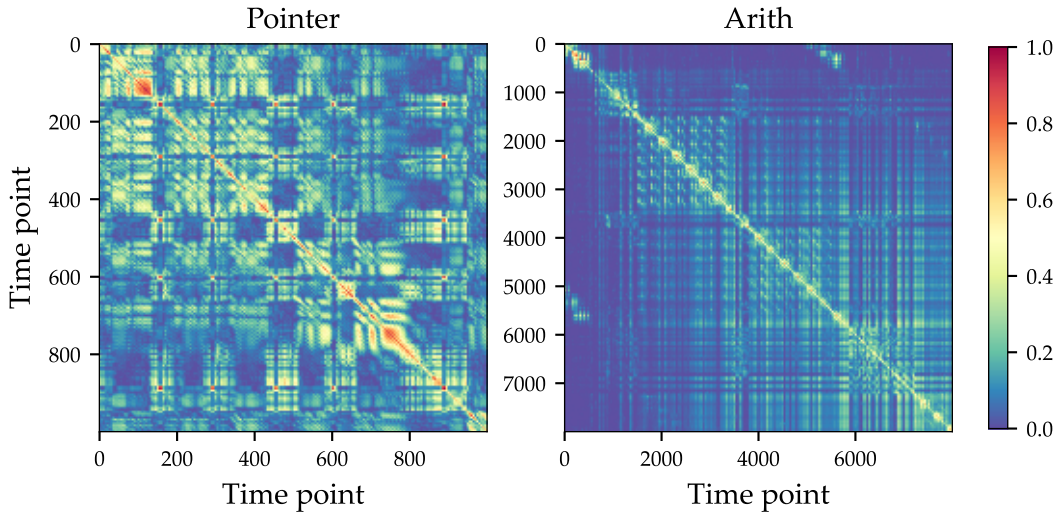

Figure 4.5: Spearman correlation (absolute values) matrices for $\mathcal{D}_{pointer}$ and $\mathcal{D}_{arith}$.

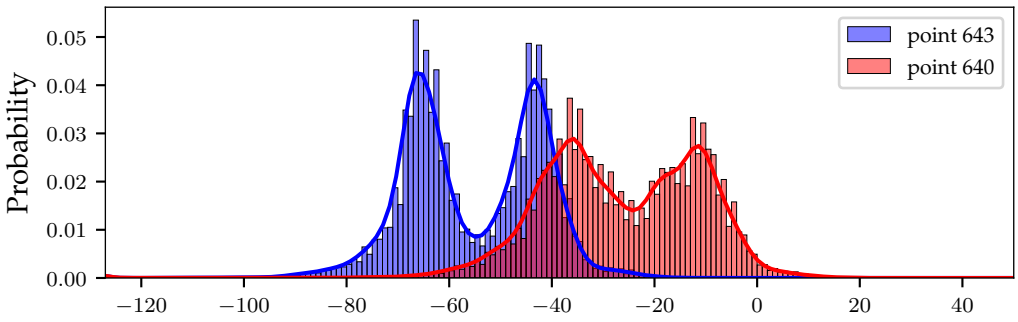

Figure 4.6: Comparison of time points 640 and 643 on $\mathcal{D}_{pointer}$.

the most appropriate one in this context. For instance, the Kullback-Leibler divergence (or symmetric Jensen-Shannon divergence) captures the global divergence on empirical distributions but can be costly to compute and is not bounded. The same is true for the Mutual Information. An alternative to information based measurements can be the Pearson's correlation coefficient, but it captures only linear relationships between the studied variables.

For these reasons, the Spearman's correlation [Spe04] is a relevant candidate for this task. It is very similar to the Pearson's correlation, but ranks are used instead of values. It is defined as:

$$\text{Spearman}(\mathbf{x}, \mathbf{y}) = \frac{\text{cov}(R(\mathbf{x}), R(\mathbf{y}))}{\sigma_{R(\mathbf{x})}, \sigma_{R(\mathbf{y})}} \tag{4.23}$$

where cov is the covariance, σ the standard deviation and $R(\cdot)$ the ordered rank indexes of variables elements. Please note that Equation 4.23 is equivalent to computing $\rho(R(X), R(Y))$, where ρ describes the Pearson's correlation coefficient.

The Spearman correlation can assess monotonic relationships between variables, even they are not linearly correlated. Furthermore, The Spearman's correlation coefficient is more

resilient to outliers as ranks are used, compared to the Pearson's correlation on values. Spearman values equals to -1 or 1 translates to a perfect correlation, that is when variables are perfectly monotonic functions of each others. In addition, mean shifts or different variances of the compared distributions do not affect the results of the Spearman correlation, since ranks are used instead of values.

We then define the following procedure. For each identified PoI during the previously described procedure $i \in \mathcal{S}_{poi}$, we compute the Spearman's correlation coefficient (and associated statistical test) between this point $\mathbf{x}_i$ and all other time points $\mathbf{x}_j : \forall j \in (1, \ldots, m)$, even excluded ones. If Spearman$(\mathbf{x}_i, \mathbf{x}_j) \geq \eta$, where η is a defined threshold (deliberately high) beyond which the correlation is considered sufficient, we say that time points $\mathbf{x}_i$ and $\mathbf{x}_j$ convey the same information. From this, we assign the same p_{conf} value to the time point at position j and add it to the PoI set if necessary.

This procedure has several advantages. First, it allows to recover some points that could have been miss-estimated using the EM algorithm or that might not have passed the statistical criteria due to too strict thresholds. Second, it also expands the region of interest that could have been detected by a single peak on a time point during the analysis. This has the effect to strengthen the set of PoI for the attack, as more PoI would now describe the same physical process (assuming that initially identified PoI carry leakage). The complete procedure is described on Algorithm 7.

Algorithm 7 PoI set improvement using Spearman's correlation

Require: $\mathbf{X}^{n \times m}, \mathbf{b}^m, \mathcal{S}_{poi}, \eta$
Ensure: $\mathbf{b}^m, \mathcal{S}_{poi}$
  for $i \in \mathcal{S}_{poi}$ **do**
    for $j \in \{1, \ldots, m\}$ **do**
      if Spearman$(\mathbf{x}_i, \mathbf{x}_j) \geq \eta$ and $b_j > b_i$ **then**
        $b_j \leftarrow b_i$
        if $j \notin \mathcal{S}_{poi}$ **then**
          $\mathcal{S}_{poi} \leftarrow \mathcal{S}_{poi} \cup \{j\}$
        end if
      end if
    end for
  end for
  return $\mathbf{b}, \mathcal{S}_{poi}$

Note that a variant of the algorithm is also possible, where instead of considering all points $j \in (1, \ldots, m)$, we can instead define a window of fixed size around each point i. This can be beneficial to reduce computational complexity or in cases where correlations are very localized on the pattern.

4.2.2.7 Choosing the number of PoI for the attack

From the PoI selection phase and additional enrichment of the PoI set using the neighbors correlation analysis (see previous Section), the attacker has now to choose an appropriate number of PoI to carry the attack on a single trace. If the amount of identified PoI in $\mathcal{S}_{poi}$ is fairly low, it is possible to use the whole set for the clustering process. However, if more than few dozen of points are considered, this can have a negative impact on the clustering (Curse of the dimensionality, as seen in Chapter 2 and 3). To prevent these effects, the right choice of the amount and order of PoI has to be done. We can make use of the estimated

p_{conf} to provide a relevant ranking of the exploitability of selected PoI. If we would like to only choose for example three PoI for the attack phase (to obtain visualizable clustering in three dimensions), we can simply select the time points with the lowest p_{conf} values.

However, it is also possible to characterize the set of PoI globally in order to try to achieve a more optimal clustering result by selecting a relevant subset of PoI. Several approaches can be considered for this task.

The first way can be to define a threshold on the maximum p_{conf} to consider for the attack. Indeed, the p_{conf} translates the uncertainty on a particular time point, so it makes sense to set a limit so that only points with the lowest p_{conf} are used. However, choosing the threshold value for this approach can be tricky because the range of p_{conf} values can vary depending on the target being considered. If the dataset is highly noisy, more points might be necessary.

Another approach for selecting the right number of PoI can be to use the elbow method. To illustrate this proposition, observe the values of the ranked p_{conf} values on the three considered datasets on Figure 4.7, after application of the proposed framework and the Spearman optimization procedure. It can be observed that at a certain point, the p_{conf} of

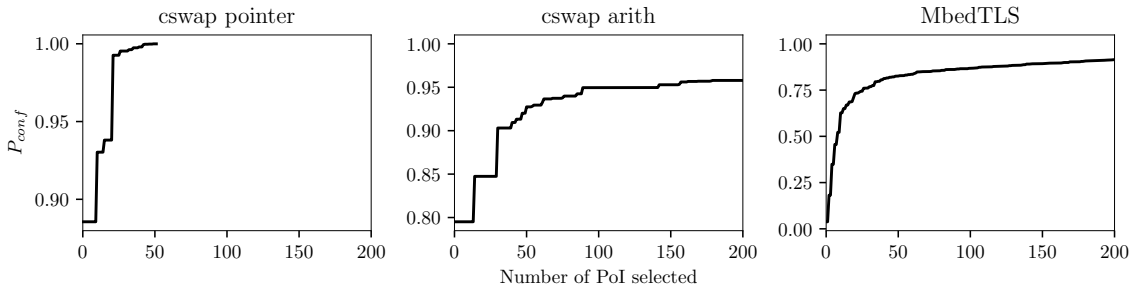

Figure 4.7: 200 first ranked p_{conf} values after the applying Spearman's augmentation on the three considered datasets.

the added points stabilize at a certain value. This can be used by considering the elbow of the p_{conf} curve (the knee in this case) to stop adding PoI that could have a negative impact on the clustering process. We show in the next section that the use of the p_{conf} elbow selector is relevant for the attack clustering phase.

4.2.2.8 Results

Following the PoI selection framework presented in the previous paragraphs, in this section we present the results obtained on the three target datasets considered for this work (see section 2.2.2 for details).

First, we define the following parameters for the proposed framework. For $\mathcal{D}_{pointer}$ and $\mathcal{D}_{mbedtls}$, we set the number of noise components $v = 0$, such that only $k = 2$ information components are sought during the EM fitting of the Gaussian mixtures. Indeed, while $\mathcal{D}_{pointer}$ contains many saturated values, the observed distributions almost always display two components. For $\mathcal{D}_{arith}$, we set the number of noise components $v = 2$, since many time points show additional behaviors, depicting more complex mixtures that would be miss estimated with incorrect choice of the number of underlying components. For all datasets, we define the Spearman threshold $|\eta| = 0.8$, such that only highly correlated points can be considered as describing the same behavior of identified PoI. We set the number of cycles for the analysis of each time point to $h = 100$, while considering at each step $q = 2048$ samples.

Finally, we set $\alpha = 10^{-3}$ as the significance level for all statistical tests.

One can observe on the Figure 4.8 the obtained p_{conf} values after the application of the proposed framework. Highest peaks of p_{conf} obtained from the unsupervised analysis co-

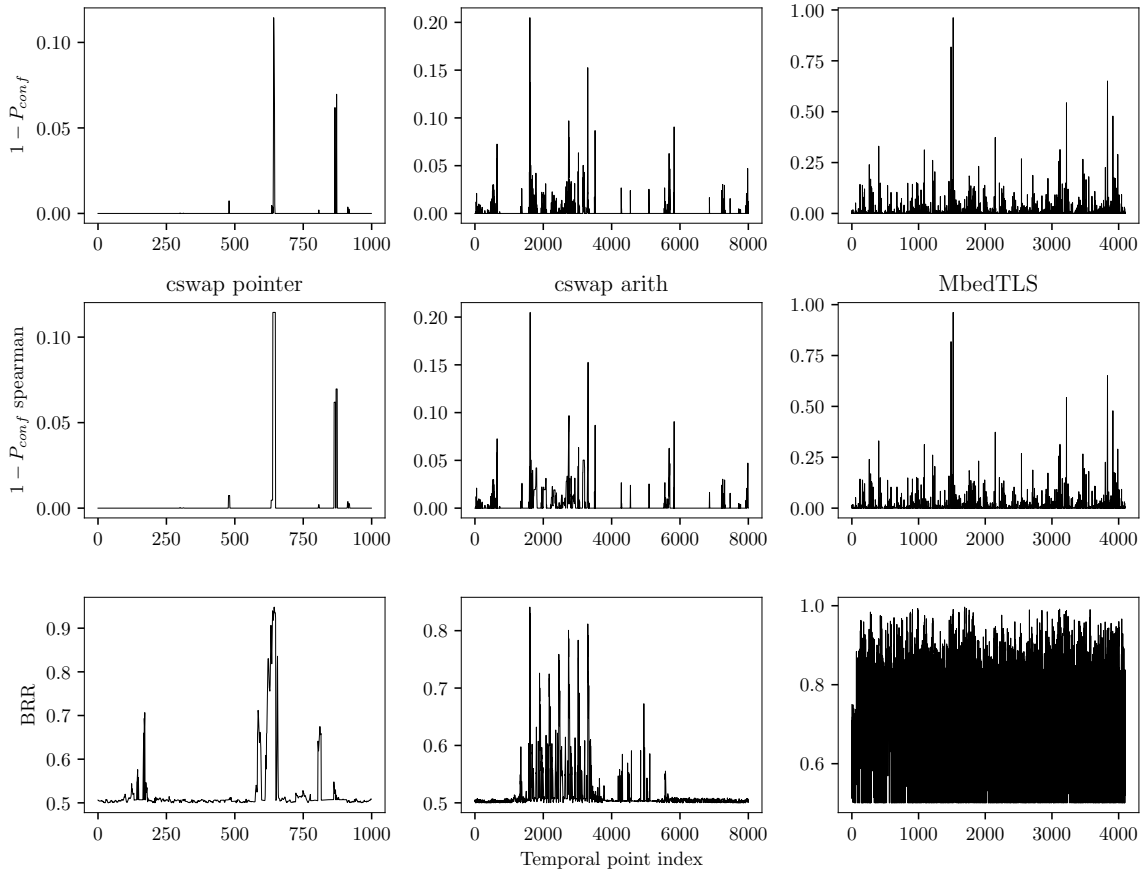

Figure 4.8: Obtained p_{conf} values (top row) and p_{conf} with added neighbors points using the Spearman correlation (middle row), as well as the supervised BRR values for reference (bottom row), on the three considered datasets.

incide with most prominent BRR values. This is clearly visible for points around 640 on $\mathcal{D}_{pointer}$ as well on points around 1500 for $\mathcal{D}_{arith}$. This provides evidence that the proposed method is able to identify relevant PoI according to the proposed leakage model. Moreover, the proposed p_{conf} metric is able to correctly translate the measured uncertainty in the observed mixtures.

It can also be seen that the Spearman analysis (middle row) allows to widen the area of interest on all datasets. This has the effect of adding more robustness to the attack process using the clustering algorithm.

Nevertheless, some PoI identified by their p_{conf} values can be seen as false positives when compared to the ground truth BRR values. This can be explained by the fact that some time points show the expected leakage distribution, but describe a different behavior. Since the proposed process is completely unsupervised, it is not possible to distinguish possible false positives considering an underlying leakage model if the ground truth is not known. Nevertheless, we say that it is preferable to choose a PoI area where more than one time point is identified with relevant p_{conf} values to avoid such false positives.

We show on Figure 4.9 the attack results using the Fuzzy k-means algorithm on the first PoI with the lowest p_{conf} values. This is repeated until the number of considered PoI reaches

200, or the number of elements in $\mathcal{S}_{poi}$ is exhausted. The results show that, on average over

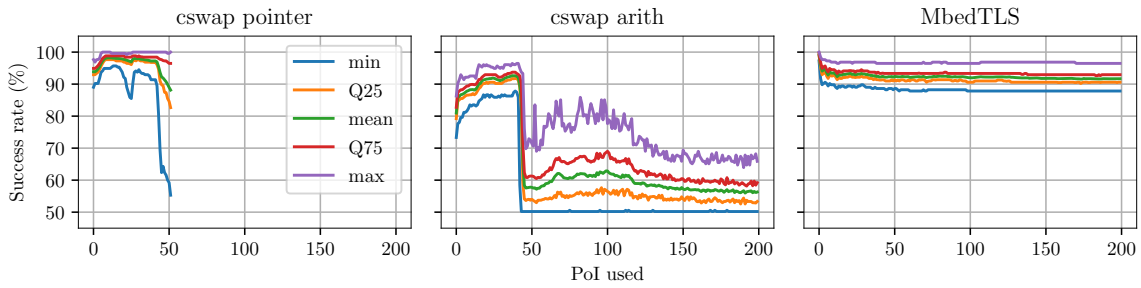

Figure 4.9: Minimum, maximum, mean, and quartile values of the success rates (BRR) obtained for the clustering attack using a single trace (255 patterns for $\mathcal{D}_{pointer}$ and $\mathcal{D}_{arith}$, 2048 for $\mathcal{D}_{mbedtls}$). The attack was performed 100 times on different traces.

100 attacks, based on the proposed PoI selection method, we are able to recover on average more than 90% of correct exponent/scalar bits on the three considered datasets.

For $\mathcal{D}_{pointer}$, the maximum BRR values obtained with only a few PoI are close to 100%. We can observe that the accumulation of more PoI has the effect of degrading the clustering quality and leads to a slow decrease of the BRR.

On $\mathcal{D}_{arith}$, the maximum BRR of about 96% is obtained by accumulating almost 50 PoI. However, if more PoI are added, the fuzzy k-means fails to provide a relevant partition from the given time points. This means that at some point the added points become uninformative or too noisy or carry some other information to allow for a relevant clustering.

For these two sets, the decrease in the success rate starts when the number of PoI is close to the elbow of the curve formed by the p_{conf} values sorted in ascending order, as previously shown in Figure 4.7. This shows that if the elbow method was considered for these two sets, it would provide relevant BRR for the number of selected time points.

For $\mathcal{D}_{mbedtls}$, high leakage levels allows to successfully recover on average 95% of bits of the exponent. Moreover, as this dataset is almost free of noise, adding more PoI only produce a small degradation of the obtained BRR.

When compared to the supervised clustering results on Figure 4.10, we can see that the unsupervised methodology provides competitive results, considering that there is no insight about the location of leaking points in the unsupervised analysis context. This sustains the

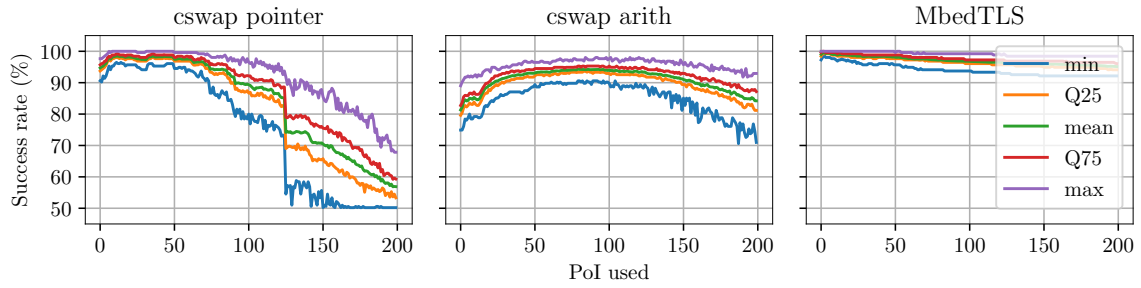

Figure 4.10: Minimum, maximum, mean, and quartile values of the success rates (BRR) obtained with the supervised clustering attack where PoI are chosen according to the highest $\mathcal{T}$-values, using a single trace (255 patterns for $\mathcal{D}_{pointer}$ and $\mathcal{D}_{arith}$, 2048 for $\mathcal{D}_{mbedtls}$). The attack was performed 100 times on different traces.

soundness of the proposed method for PoI selection. For comparison, Perin *et al.* reported an

average BRR of 52% on $\mathcal{D}_{pointer}$ and $\mathcal{D}_{arith}$ sets before applying the iterative bit correction framework from [PCBP20]. We also show on Figure 4.11 its application using the assignments provided by the clustering with the PoI from our analysis. This is repeated several time with different number of PoI considered. As it can be observed, on both sets $\mathcal{D}_{pointer}$

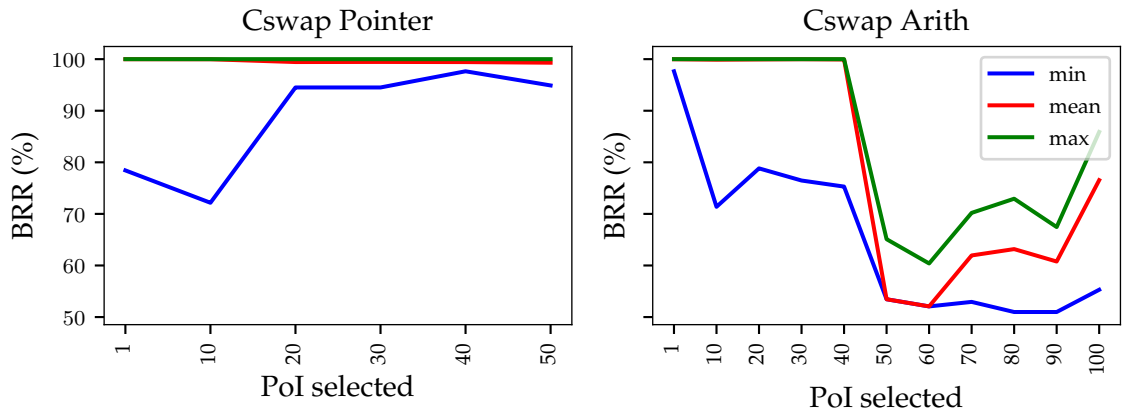

Figure 4.11: Minimum, maximum, mean values of the success rates (BRR) obtained after applying the bits correction framework from Perin *et al.* [PCBP20] on the assigned classes from the clustering using the PoI selected with the proposed mixture analysis.

and $\mathcal{D}_{arith}$, the average obtained BRR when considering the first 50 PoI is very close to 100%. From this we can see that the proposed mixture analysis for PoI selection is able to provide satisfactory enough initial assessment in order for the Perin *et al.* framework to correctly generalize and provide enhanced BRR.

Still, it is important to note that whatever the method considered (unsupervised or supervised), the accumulation of more time points, even if they carry relevant information about the leakage, will inevitably decrease the BRR obtained from the clustering process. In fact, as more points are added, they also add more noise, and the increasing dimensionality space becomes more difficult for the clustering algorithm to partition. This means that even in the context of a low leakage and high noise dataset, the number of PoI to consider for the attack should be reasonable (about 100) in order to provide a relevant representation for the clustering algorithm.

4.2.3 Application on cleaned patterns

Following the results presented so far, we would like to know if the application of the proposed framework to correct some outliers in patterns, namely from the application of the RAE or CycleGAN cleaning procedure (see Chapter 3), can have some advantages over the actual application on raw patterns.

As seen in the previous chapter, RAE and CycleGAN cleaning procedures both provided improved BRR values on the PoI conveying leakage. In addition, it allowed the identification of new leaking points that would not have been considered before the cleaning phase. Based on these results, the aim is therefore to assess whether this allows to identify new PoI when applying the framework to these cleaned traces.

4.2.3.1 Mixture analysis

For this purpose, the same parameters were considered as before for RAE and CycleGAN cleaning procedures. Then the PoI identification method was applied.

Figure 4.12 shows the p_{conf} values obtained after applying the mixture analysis to the resulting cleaned traces from the RAE and CycleGAN processes on $\mathcal{D}_{pointer}$. As before, we

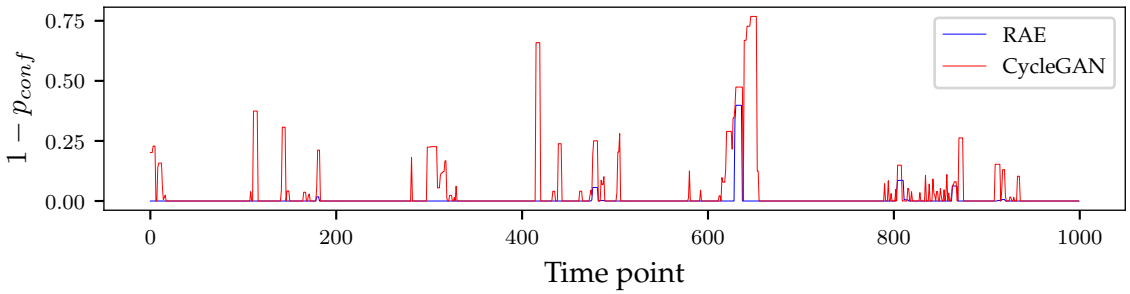

Figure 4.12: p_{conf} values obtained after applying the mixture analysis to patterns of $\mathcal{D}_{pointer}$ cleaned with the RAE and CycleGAN procedures.

considered $v = 0$ for this dataset, thus mixture of two components in the analysis.

For $\mathcal{D}_{arith}$, we experimented with different values of $v \in \{0, 1, 2\}$ to determine if the application of RAE and CycleGAN procedures can reduce the number of underlying components in the observed mixtures by mitigating noisy behavior. We show the results obtained in Figure 4.13.

In addition, in Figure 4.14 we show the results of the attack using the PoI provided by the proposed PoI identification method when applied to the cleaned traces obtained using the RAE and CycleGAN, with $v = 0$ for $\mathcal{D}_{pointer}$ and $v \in \{0, 1, 2\}$ for $\mathcal{D}_{arith}$. For $\mathcal{D}_{pointer}$, without the application of any cleaning procedure, the BRR starts to decrease as soon as the number of PoI kept raises above 40. The application of the RAE cause a degradation of the obtained results. Indeed, as soon as more than 30 PoI are kept the BRR starts to fall. On the contrary, the use of the cycleGAN has a positive effect by producing more robust BRR values. In fact, the decrease of the BRR occurs when more than 80 PoI are considered. This is direct evidence that the outlier of about 40 time points has been corrected to highlight the leakage.

On $\mathcal{D}_{arith}$, the CycleGAN cleaning process did not produce relevant results regardless the number of noise components considered. However, using the RAE, the choice of the number of noise components v had a huge impact on the quality of the selected PoI and the success of the attack. For $v = 1$ (thus three components), it yielded the best overall BRR, slightly below the one obtained using raw traces, but showing more robustness as the number of PoI increases. This suggests that while the quality of the selected PoI is a bit lower, the set of informative PoI is larger.

Overall, the results are mixed. While in some cases the application of the PoI selection procedure led comparable BRR values and robustness in some cases, in other cases it led to comparable degradation of the BRR values. This seems to be mostly due to the fact that some uninformative time points are now distributed in a way that matches the distribution sought by the PoI selection procedure. Indeed, most prevalent selected time points are correctly distributed according to the expected leakage model, but do not carry any exploitable leakage. Since, the whole process is applied in an unsupervised manner, the application of the proposed outliers and saturated values cleaning methods can thus have a negative

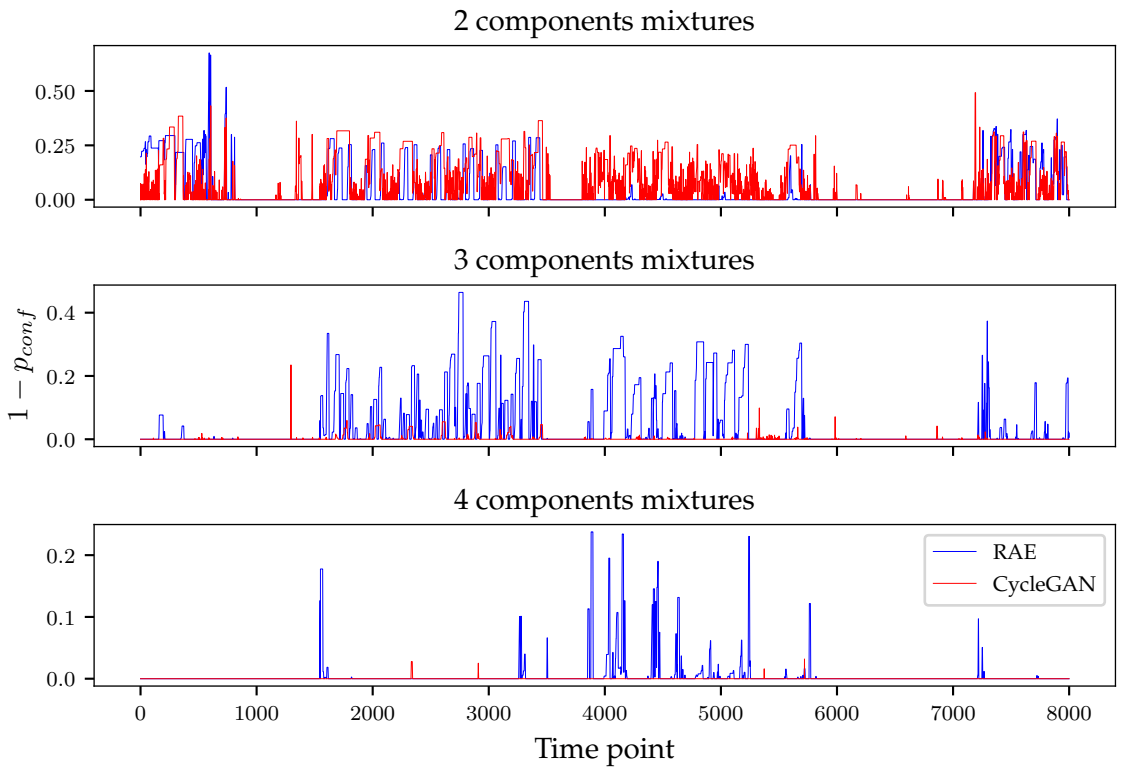

Figure 4.13: p_{conf} values obtained after applying the mixture analysis to patterns of $\mathcal{D}_{arith}$ cleaned with the RAE and CycleGAN procedures with different number of sought components in the mixtures.

impact on the PoI selection framework. Therefore, the application of the proposed PoI selection procedure on cleaned patterns should be done with this in mind. One solution to mitigate the result disparity can be, for example, to compare the obtained PoI between the raw patterns and the cleaned patterns from the RAE or CycleGAN. Another solution, which we describe below, can be to consider the PoI set obtained from the analysis on the original patterns, but perform the attack on the cleaned patterns.

4.2.3.2 Using the same PoI

As seen, in some cases, the action of cleaning the traces with the RAE or CycleGAN can have the effect of deteriorate the quality of patterns and thus of the PoI selection method. This can be caused by the fact that some time points after the cleaning phase are now described by distributions that match the awaited one from the statistical analysis framework. Still, we saw in previous Chapter that almost all time points which contain initial leakage are enhanced through the application of the mitigation process.

Thus, we consider the initial PoI selection on the original patterns to be optimal, since it cannot suffer from side effects caused by the cleaning process. However, it is still possible to use the results of RAE or CycleGAN. If we assume that the application of the cleaning process does not induce degradation on the original leakage time points (as seen in the previous chapter), we can use the selected PoI from the analysis on the original patterns to be applied on the cleaned patterns. Thus, if the leakage difference is significant, it would benefit from being applied to cleaned patterns. For this purpose, we perform the PoI selection and the neighbors analysis (as described in the previous sections) on the original patterns, and

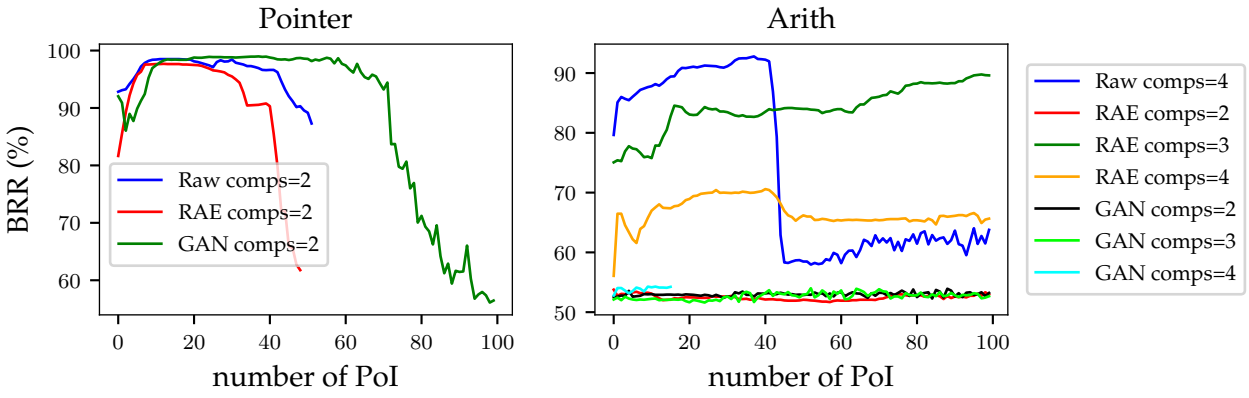

Figure 4.14: Obtained BRR values from the clustering attack using the raw patterns and cleaned patterns from the RAE and using CycleGAN, on $\mathcal{D}_{pointer}$ and $\mathcal{D}_{arith}$. Each time, a single trace is used (255 patterns), and the process is repeated 100 times.

carry out the clustering phase on the cleaned patterns from the RAE and the CycleGAN. The obtained results are shown on Figure 4.15.

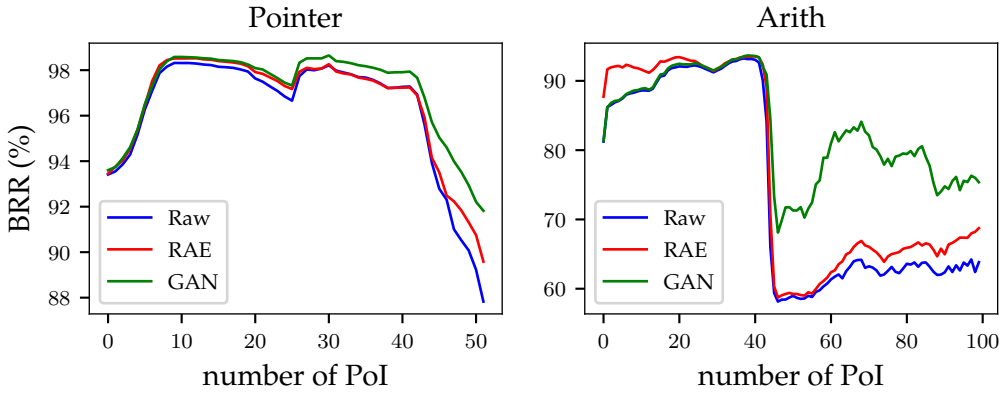

Figure 4.15: Obtained BRR values from the clustering attack using the cleaned patterns from the RAE and using CycleGAN, on $\mathcal{D}_{pointer}$ and $\mathcal{D}_{arith}$. Each time, a single trace is used (255 patterns), and the process is repeated 100 times.

As expected, using the cleaned patterns instead of the original ones does not degrade partitioning quality. Instead, it allows producing slightly higher BRR values. This is negligible in the case of $\mathcal{D}_{pointer}$, but for $\mathcal{D}_{arith}$ it allows to gain about 5% of accuracy using the cycleGAN, and add some robustness using the RAE. While this can be seen as a non-substantial change in the obtained results, it is still not negligible if the obtained BRR is insufficient to allow exponent or scalar recovery after partitioning.

In addition, we can apply Spearman neighbors analysis to the cleaned patterns instead, since the cleaning process may produce more highly correlated neighbors. For this experiment, we still consider the initially identified PoI from the analysis on the original patterns. The results are shown in the figure 4.16. On $\mathcal{D}_{pointer}$ this does not lead to slightly different results than the previously presented figure 4.15. On $\mathcal{D}_{arith}$, however, the action of applying the neighborhood analysis procedure to the cleaned patterns, using the PoI selected from the original patterns, leads to noticeable changes. With the RAE, this adds robustness to the partitioning success, since the BRR is still higher than 90% using 90 PoI, while it was highly degraded after 35 PoI using the original patterns. This means that the effect of the RAE al-

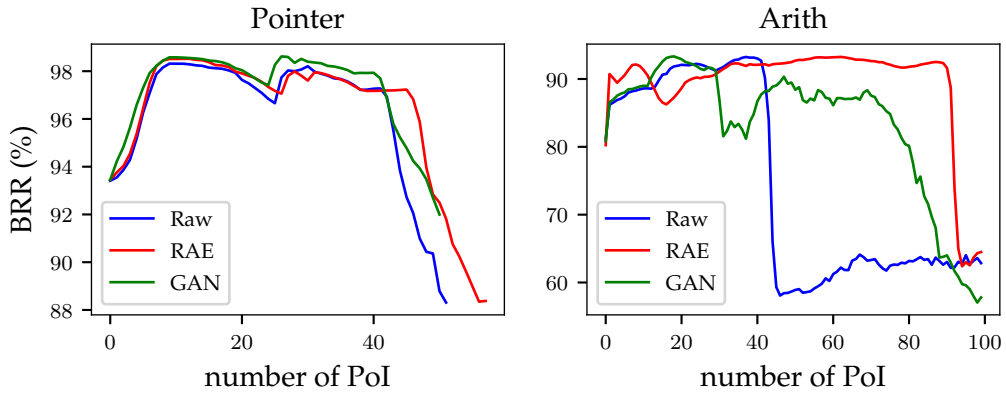

Figure 4.16: BRR values obtained after the clustering attack using the cleaned patterns from the RAE and using CycleGAN and applying the Spearman neighbors analysis on cleaned patterns, on $\mathcal{D}_{pointer}$ and $\mathcal{D}_{arith}$. Each time, a single trace is used (255 patterns), and the process is repeated 100 times.

lows to increase the rank correlation between neighboring time points and thus to produce more robust PoI regions. However, the opposite is observed when using the cycleGAN. In fact, its use results in degraded neighbor correlations, since fewer PoI are selected that transcribe the same behavior (thus increasing the leakage).

4.2.4 Adaptive mixture analysis

In the previous sections, the number of components searched and estimated in a mixture was chosen by visual inspection. While this showed relevant results in terms of identified PoI and success rates, we would like to know if removing this constraint by fixing the number of components to look for in each mixture can give better results. Indeed, as seen earlier, the number of underlying mixture components can vary at each time point, and thus restricting the analysis to a certain number of observed noise components v may result in some time point distributions not being correctly estimated.

However, a major problem with this approach is that the number of components must be specified a priori to the EM algorithm. Attempting to fit a mixture with a high number of components will often result in a partitioning of the main components if the underlying mixture component number is lower than expected. An alternative method would be to consider a Bayesian Gaussian mixture model instead of a non-Bayesian one, but this would introduce additional parameters and add complexity to the unsupervised analysis.

Instead, we propose to build a set of Gaussian mixture models $G = \{g_k, g_{k+1}, \ldots, g_{k+v}\}$ with different numbers of components in order to find the model that fits the best the observed measurements. While this method results in higher computational complexity, it is necessary to be able to compare models quality against the observed data. After building this mixture model set for each time point, two approaches can be considered to find the most appropriate number of underlying components in the mixture.

4.2.4.1 Model goodness of fit

On one hand, we can evaluate the goodness of fit of the model to the provided data based on a chosen metric M. Two typical metrics can be considered for this approach: the Akaike Information Criterion and the Jensen-Shannon divergence.

Akaike Information Criterion The Akaike Information Criterion (AIC) provides a criterion to know how much information is lost between models $g_i \in G$. It is used in many fields for Machine Learning model selection. The AIC for each model is computed as:

$$\mathrm{AIC}(\ell) = 2k - 2\ln(\ell) \tag{4.24}$$

where ℓ is the log likelihood and k the number of components in the mixture fitted. The AIC rewards the goodness of fit of the model while penalizing the increasing number of parameters in order to avoid overfitting. Lower AIC values translate better model fitting with the data.

Jensen-Shannon divergence From a fitted model on a given number of components k, this yields the set of parameters θ. The idea of this approach is to generate new samples q following the estimated mixture distribution Q. This samples can then be used to calculate the divergence between the actual point distribution and the sampled distribution from the fitted model with a particular provided number of components. The divergence can be computed using the Jensen-Shannon Divergence (JSD).

$$JSD(P,Q) = \frac{1}{2}\Big[D_{KL}(P||M) + D_{KL}(Q||M)\Big] \tag{4.25}$$

$$M = \frac{1}{2}(P+Q) \tag{4.26}$$

$$D_{KL}(P||Q) = \sum_{x \in X} P(x)\log\frac{P(x)}{Q(x)} \tag{4.27}$$

where P is the actual mixture point distribution and Q is the estimated distribution sampled from the model. The JSD is symmetric and normalized $0 \leq JSD(P,Q) \leq 1$ unlike the Kullback-Leibler divergence and thus can be used as a distance metric by taking the squared root of the divergence $\sqrt{JSD}$.

It is then possible to find the model b that minimizes the metrics in order to know which number of parameters fits the best the actual data.

$$b = \underset{G}{\operatorname{argmin}}\, M(g_i) \tag{4.28}$$

Another method is to apply the elbow method, that is, the point where the score decrease becomes significantly lower, below a defined threshold β.

$$b = \arg\Delta\{M(g_i), M(g_{i+1})\} < \beta \tag{4.29}$$

While this approach seems relevant in theory, it was not applicable in practice because the obtained values of AIC and JSD for different numbers of components to fit were not different enough to provide a strict consensus on the underlying number of components on the observed mixtures. Furthermore, the application of statistical tests often failed on the selected mixture at all-time points.

4.2.4.2 Lowest number of components

Instead, we consider an alternative approach to select the best number of components to represent the mixture of each time point. More precisely, we looked for the lowest number

of components allowing passing all the statistical tests involved in the PoI selection method. Indeed, the lower the complexity of the mixture, the higher its exploitability should be. To achieve this, we use the procedure described earlier in the Algorithm 6, but by increasing the value of $v \in \{0, \ldots, 4\}$ until the statistical tests of weight partitioning and variance disparity are successful. This indicates that the mixture is relevant according to the defined criteria. In the following, we describe the results obtained with this approach. First, we show in Figure 4.17 the number of components selected for each time point. Values equal to zero translates to rejected points unable to meet the selection criteria for any number of components. Unsurprisingly, several time points show a high number of components. We have

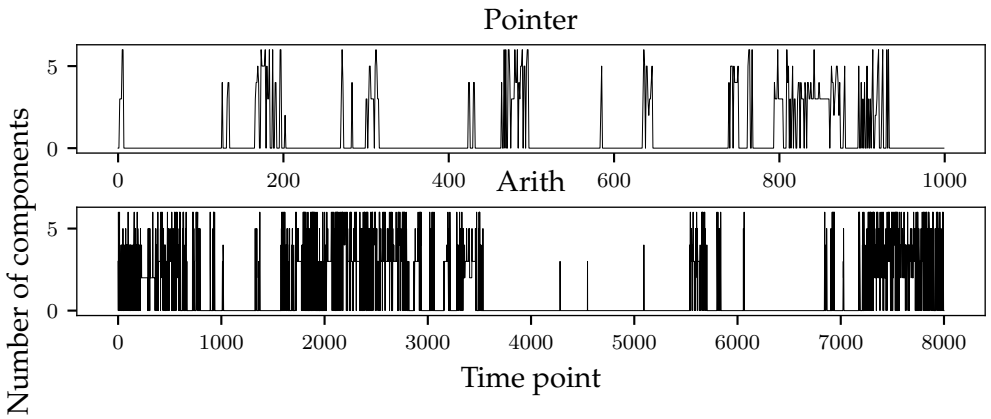

Figure 4.17: Minimum number of components allowing to meet the selection criteria for each time point of $\mathcal{D}_{pointer}$ and $\mathcal{D}_{arith}$.

seen through experimentation that as the number of components considered in the mixture increases, it almost always matches the defined selection criteria at some point. The downside is that above a certain threshold, the estimated mixtures are not meaningful compared to the original data. The selection criteria bounds are strict, using statistical tests with small significance levels to prevent false positives. This has the effect of rejecting mixtures with a low number of components and instead considering more complex mixtures. For example, in Figure 4.18 we show the six mixture components estimated using the EM of a time point that only passes all tests when at least 6 components are considered. While this time point

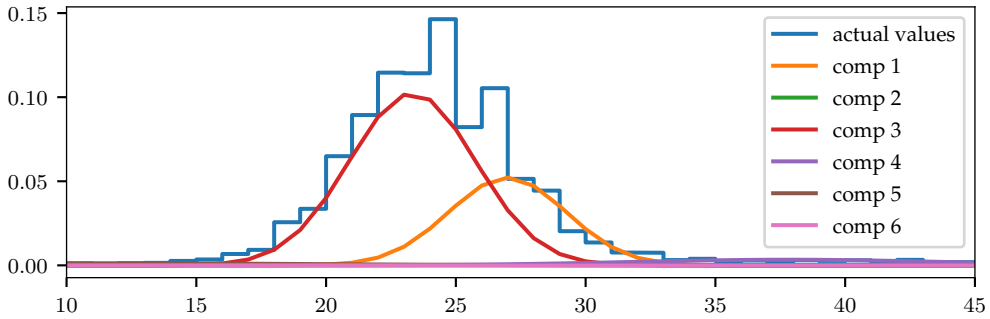

Figure 4.18: Mixture estimation with 6 components (time point 466 on $\mathcal{D}_{pointer}$).

could be modeled as a mixture of two or three components, the resulting six-component mixture shows four of the six components with very low mixing coefficients. In fact, these additional components do not carry any relevant information.

To get more insight, we plotted in Figure 4.19 the percentage of time points versus the

minimum number of components required in the modeling to pass all tests. We can see that

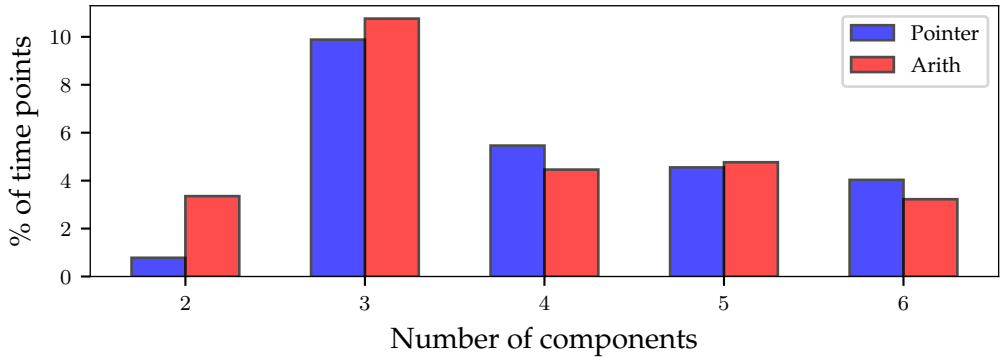

Figure 4.19: Percentage of time points per number of components.

most of the time points are described by a mixture of three components, i.e. with $v = 1$ additional noise component. In addition, the percentage of time points with more than two components is higher than those with only two components, so $v = 0$. Finally, the figure 4.20 shows the p_{conf} obtained for $\mathcal{D}_{pointer}$ and $\mathcal{D}_{arith}$ by applying the adaptive framework. As more choice for mixture estimation with more than a fixed number of components, we get

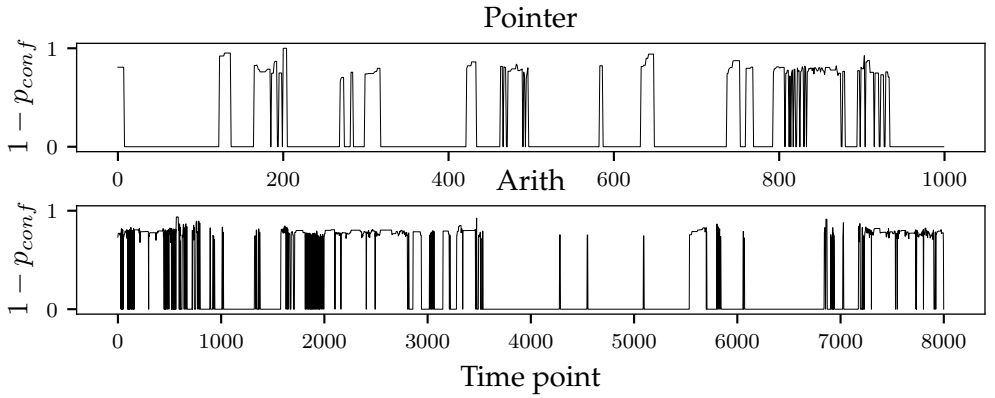

Figure 4.20: p_{conf} values obtained by applying the adaptive PoI selection procedure on $\mathcal{D}_{pointer}$ and $\mathcal{D}_{arith}$.

a larger number of PoI p_{conf} values less than 1. This suggests that more leakage information is captured by using more mixture components.

However, the application of the clustering attack process using the fuzzy k-means algorithm on the resulting sets of PoI obtained with the adaptive selection approach did not give better results than the previously considered method using a fixed number of components for a given data set. On $\mathcal{D}_{pointer}$, we obtained similar results as before. However, on $\mathcal{D}_{arith}$, this approach led to a degradation of the obtained BRR during the attack, regardless of the number of PoI chosen.

4.2.4.3 Application on cleaned patterns

We also applied this alternative adaptive method to cleaned patterns produced by the RAE and CycleGAN (see details in Chapter 3). The idea was to assess whether the outlier reduction process has an effect on the number of components identified in the mixture formed by each time point and its associated p_{conf}. In Figure 4.21, we show the percentage of time

points versus the minimum number of components required to pass all statistical tests. It

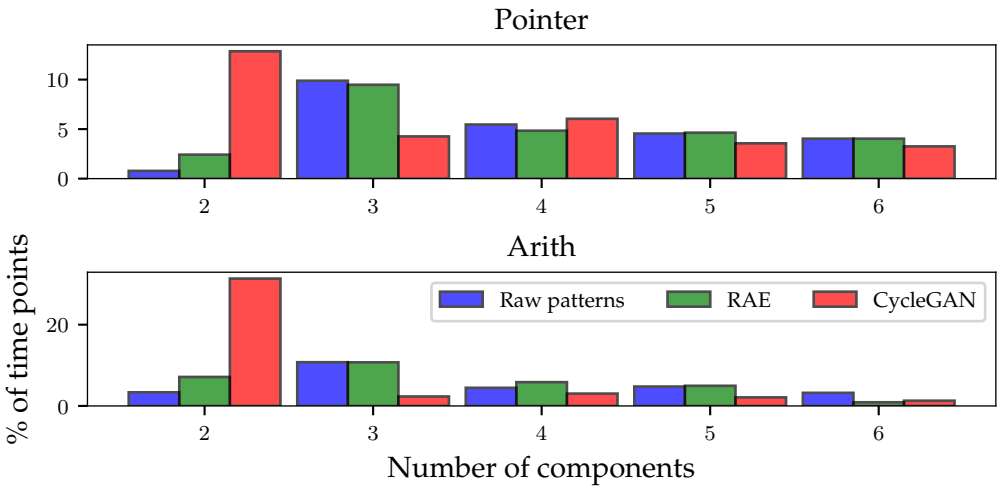

Figure 4.21: Percentage of time points versus the minimum number of components required to pass all statistical tests.

can be observed that the adaptive mixture analysis applied to the original patterns or to the patterns cleaned with the RAE produced similar distributions of time points against the minimum number of components required to pass all statistical tests. Otherwise, the saturated value reduction procedure based on a cycleGAN seems to reduce the number of components required to model the time points so that to pass all statistical tests. This means that outlier reduction has a more important effect on the number of underlying components in the mixture than the saturation cleaning provided by the RAE.

Figure 4.22 shows the p_{conf} values obtained for each time point of $\mathcal{D}_{pointer}$ and $\mathcal{D}_{arith}$ using the cleaned traces from the RAE and CycleGAN. The reported results also include the PoI set improvement using the Spearman correlation (see Algorithm 7). The results show

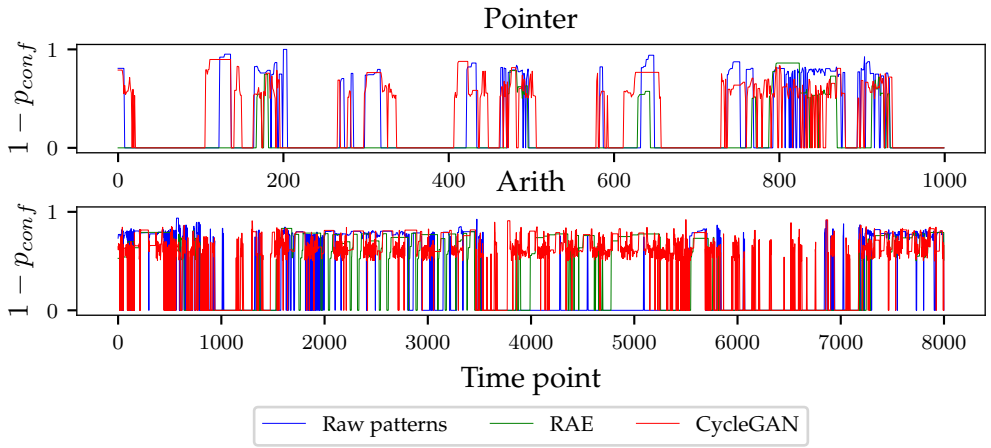

Figure 4.22: p_{conf} values obtained from the adaptive PoI selection procedure on $\mathcal{D}_{pointer}$ and $\mathcal{D}_{arith}$.

that while the cycleGAN identifies more PoI, most of which can be modeled with two components, many non-informative points are also included in the PoI set for $\mathcal{D}_{arith}$. On $\mathcal{D}_{pointer}$, the cycleGAN widens the identified leakage area about time point 600 compared to the results on raw patterns and cleaned ones using the RAE, but it also produces many false

positive time PoI.

4.2.4.4 Limits

We have seen in this section that the proposed methods for adaptive mixture analysis are theoretically relevant to provide a finer analysis of each underlying mixture model associated with each time point.

In practice however, they did not provide significant improvements over the previously proposed selection method with a fixed number of components to identify using the EM algorithm. While we obtained similar results on $\mathcal{D}_{pointer}$, the adaptive approach had the effect of degrading the results obtained on $\mathcal{D}_{arith}$. It thus seems that allowing a higher number of components leads to more complex mixture distributions which are less meaningful and less usable or at least harder to exploit.

Moreover, fitting multiple mixture models with different numbers of components adds a non-negligible computational complexity. The resulting trade-off between the quality of the results obtained and the CPU time cost of the procedure is not favorable for this adaptive approach, which is thus not really relevant in a real attack context.

4.3 Conclusion

In this chapter, we studied univariate PoI selection based on Gaussian mixture models. While this has been considered in previous work, we derived a leakage model to account for additional behavior in the analysis. Previous approaches always searched the observed data for mixtures with a number of components equal to the secret manipulated by the cryptographic algorithms. We have seen that this is not always the case due to outliers or measurement problems, and that the observed data is actually a more complex Gaussian mixture. Thus, the number of searched components in the observed data must be set accordingly after a careful inspection of the statistical distribution shapes.

Following this initial result, we provided an efficient framework for analyzing and selecting PoI candidates based on statistical tests. In addition, we defined a bounded and comparable metric across time points, p_{conf}, to quantify the exploitability of identified PoI and the ability to rank them accordingly. This allowed the selection of the most relevant PoI for the attack.

This metric can also be used to select the right number of PoI to consider for the attack phase. When compared to the actual ground truth leakage levels, it was shown that this metric is able to successfully reflect the leakage manifestation on the observed mixture components, while being applicable in a completely unsupervised manner. The application of the proposed framework allowed providing improved attack results on the considered targets, compared to the current state-of-the-art obtained results.

We then considered an alternative to this framework using an adaptive mixture model with an unbounded number of components to identify. While this approach does not limit the mixture to only feature a predefined and fixed number of noise components, it did not provide much improvement over the previously proposed method. Indeed, this resulted in more complex and less meaningful mixtures as the number of additional noise components increased.

Finally, we also studied the application of the framework after cleaned input traces using previously proposed neural network based traces correction methods using the RAE and CycleGAN architectures. They seem to offer some advantages on the obtained results when

applied after the selection of PoI on raw traces. However, systematically applying them before the analysis may be counterproductive.

Chapter 5

Conclusion & Perspectives

To conclude the work presented, in this final chapter we provide a summary of the contributions made in this work, as well as the perspectives and axes of improvement for future work.

5.1 Contributions

Through this work, we studied several axes for the improvement of horizontal side channel attacks.

In Chapter 2, we pointed out the limits of using metrics such as the Welch's T-test statistic for the selection of points of interests. We also showed that the applicability of dimensionality reduction methods such as the Principal Component Analysis are limited in noisy environments. Finally, we showed that outliers and saturated values can have a high negative impact on the selection of relevant PoI.

In Chapter 3, we proposed several methods for improving horizontal attacks using the power of neural networks architectures. We showed that unsupervised compression networks such as autoencoder based architectures, while having higher capacity than the PCA, did not perform well in the presence of high levels of noise but can be a relevant alternative otherwise. We used robust autoencoder based architecture and generative adversarial networks as tools for outliers and saturated values mitigation in the signal. Through this, we showed that this had a positive impact in the observed classes leakage.

Finally, Chapter 4 proposed an alternative method for univariate points of interest selection based on Gaussian mixture models and statistical analysis. It resulted in increased performance when compared to state of the art and competitive results against supervised PoI selection. We provided a sound approach based on weights repartition and variance criteria for identifying the best candidates time points. In addition, we introduced an alternative metric for assessing the exploitability of a given time point, by quantifying the overlap between the mixture components. This metric is applicable in supervised context where labels are known, or in an unsupervised context where class estimation is performed using mixture estimation methods. Overall, the proposed alternative PoI framework was shown to provide improved results compared to the state-of-the art, especially on noisy datasets where exploitability was not possible before.

5.2 Perspectives

From this thesis, several perspectives emerge for future works.

5.2.1 Horizontal attacks

Through this work, we presented methodologies to provide improved PoI selection to the state-of-the-art. However, the problem of horizontal side-channel attacks is still prone to improvements. We believe that while univariate leakage assessment is well studied, multivariate leakage exploitability still has room for improvement. Indeed, leakage exploitation in a multivariate manner could offer potential advantages over considering independent univariate time points. While we have attempted to approximate this using neighborhood analysis through rank correlation, we believe that it could be improved. Multivariate analysis can provide many advantages over univariate analysis, but describes new challenges and problems in univariate side-channel analysis that could be covered in a whole dissertation.

5.2.2 Deep Learning

As seen in this work, deep learning using neural networks is a powerful tool for tackling various problems in side channel analysis. In both supervised and unsupervised modes, neural networks have been shown to be a relevant candidate for leakage exploitation or trace quality improvement.

However, their applicability in an unsupervised context remains limited. This is especially true for complex architectures with high hyperparameter tuning needed, which is not possible in horizontal situations. Thus, the application of neural-based solutions to complex side-channel problems should be reserved for tasks where traditional machine learning is not possible. Indeed, the uncertainty of failing to recover a secret can be reduced in cases where neither tuning nor architecture search is required. More generally, it is common to see cases where traditional machine learning can easily outperform neural network based architectures in the case of non-optimal parameters search. This highlights the trade off of applying heavy optimization techniques where more classical methods can easily solve these problems. The fact that neural networks are universal function approximators does not mean that they should be blindly applied to any problem. Furthermore, the applicability of a specific defined architecture and parameters is often tied to the data it is trained on, and does not necessary transpose to different data.

Nevertheless, we believe that the use of neural networks for unsupervised tasks in the context of side channel analysis is relevant, but must be measured based on the problem being addressed. For instance, self-supervised learning is a relevant area of learning that can be applied to horizontal attacks, and we think that further exploration of such techniques could benefit to the improvement of unsupervised side channel attacks. Indeed, neural networks have been shown to be applicable to many domains and to provide improved performance over traditionally used methods. We believe that many methods from other domains (especially signal processing) could be applicable to the side channel domain, as they reflect the same problematics.

5.3 Final Words

To conclude this thesis, we would like to offer some thoughts as a general takeaway for the reader.

It is important to note that unsupervised horizontal side-channel attacks are difficult to perform in practice. In fact, this is the worst case attack scenario for a given attacker. In this work, we have assumed knowledge of the attacked cryptographic algorithm, but this

is not always the case. Moreover, the success of the attack will not primarily rely on the exploitation method, quite the opposite. In fact, preprocessing phases are crucial to enable the exploitation of the acquired patterns. Without them, even the most powerful method would not provide satisfactory results. Again, in this work we considered perfectly aligned patterns with no or minimal jitter. Thus, we believe that most of the effort should be put into these phases to provide the best quality patterns possible, since it is impossible to characterize an underlying leakage in an unsupervised context. Thus, it is necessary to exclude that unsatisfactory results could be due to misalignment or poor processing phases, starting with a correct configuration of the acquisition material.

Overall, we hope that the presented work in this thesis manuscript can benefit to readers wishing to get an overview of the current state-of-the-art of clustering based horizontal side channel attacks, as well as to support potential future studies.

Bibliography

[APSQ06] C. Archambeau, E. Peeters, F. X. Standaert, and J. J. Quisquater. *Template Attacks in Principal Subspaces*, volume 4249 of *Lecture Notes in Computer Science*, page 1–14. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006. 15

[BA24] Sana Boussam and Ninon Calleja Albillos. Keep it unsupervised: Horizontal attacks meet simple classifiers. In Shivam Bhasin and Thomas Roche, editors, *Smart Card Research and Advanced Applications*, Lecture Notes in Computer Science, page 213–234, Cham, 2024. Springer Nature Switzerland. 17, 41, 44

[BBR+18] Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeswar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, and R. Devon Hjelm. Mine: Mutual information neural estimation. *arXiv:1801.04062 [cs, stat]*, June 2018. arXiv: 1801.04062. 55

[BCC57] R. Bellman, Rand Corporation, and Karreman Mathematics Research Collection. *Dynamic Programming*. Rand Corporation research study. Princeton University Press, 1957. 37

[BCD+] G. Becker, J. Cooper, E. DeMulder, G. Goodwill, J. Jaffe, G. Kenworthy, T. Kouzminov, A. Leiserson, M. Marson, P.Rohatgi, and S. Saab. Test vector leakage assessment (tvla) methodologyin practice. 12

[BCO04] Eric Brier, Christophe Clavier, and Francis Olivier. Correlation power analysis with a leakage model. In Marc Joye and Jean-Jacques Quisquater, editors, *Cryptographic Hardware and Embedded Systems - CHES 2004*, Lecture Notes in Computer Science, page 16–29, Berlin, Heidelberg, 2004. Springer. 12

[BCV14] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. (arXiv:1206.5538), April 2014. arXiv:1206.5538 [cs]. 51

[BDGN14] Shivam Bhasin, Jean-Luc Danger, Sylvain Guilley, and Zakaria Najm. Nicv: Normalized inter-class variance for detection of side-channel leakage. In *2014 International Symposium on Electromagnetic Compatibility, Tokyo*, page 310–313, May 2014. 12, 22

[Beh70] Javad Behboodian. On the modes of a mixture of two normal distributions, 1970. 26, VII

[Bel61] Richard E. Bellman. *Adaptive Control Processes*. Princeton University Press, Princeton, 1961. 37

[BHvW12] Lejla Batina, Jip Hogenboom, and Jasper G. J. van Woudenberg. Getting more from pca: First results of using principal component analysis for extensive power analysis. In Orr Dunkelman, editor, *Topics in Cryptology – CT-RSA 2012*, Lecture Notes in Computer Science, page 383–397, Berlin, Heidelberg, 2012. Springer. 15

[BJPW14] Aurélie Bauer, Eliane Jaulmes, Emmanuel Prouff, and Justine Wild. Horizontal collision correlation attack on elliptic curves. In Tanja Lange, Kristin Lauter, and Petr Lisoněk, editors, *Selected Areas in Cryptography – SAC 2013*, Lecture Notes in Computer Science, page 553–570, Berlin, Heidelberg, 2014. Springer. 14

[BJZP20] Sunitha Basodi, Chunyan Ji, Haiping Zhang, and Yi Pan. Gradient amplification: An efficient way to train deep neural networks. (arXiv:2006.10560), June 2020. arXiv:2006.10560 [cs, stat]. 49

[BNSQ03] Lilian Bohy, Michael Neve, David Samyde, and Jean-Jacques Quisquater. Principal and independent component analysis for crypto-systems with hardware unmasked units. January 2003. 15

[Bon19] Thomas Bonald. Expectation-maximization for the gaussian mixture model. 2019. 87

[BPS+20] Ryad Benadjila, Emmanuel Prouff, Rémi Strullu, Eleonora Cagli, and Cécile Dumas. Deep learning for side-channel analysis and introduction to ascad database. *Journal of Cryptographic Engineering*, 10(2):163–188, June 2020. 48

[CCC+19] Mathieu Carbone, Vincent Conin, Marie-Angela Cornélie, François Dassance, Guillaume Dufresne, Cécile Dumas, Emmanuel Prouff, and Alexandre Venelli. Deep learning to evaluate secure rsa implementations. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, page 132–161, February 2019. 48

[CDP16] Eleonora Cagli, Cécile Dumas, and Emmanuel Prouff. *Enhancing Dimensionality Reduction Methods for Side-Channel Attacks*, volume 9514 of *Lecture Notes in Computer Science*, page 15–33. Springer International Publishing, Cham, 2016. 17, 41

[CFG+10] Christophe Clavier, Benoit Feix, Georges Gagnerot, Mylène Roussellet, and Vincent Verneuil. Horizontal correlation analysis on exponentiation. In Miguel Soriano, Sihan Qing, and Javier López, editors, *Information and Communications Security*, Lecture Notes in Computer Science, page 46–61, Berlin, Heidelberg, 2010. Springer. 13, 14

[CFG+12] Christophe Clavier, Benoit Feix, Georges Gagnerot, Christophe Giraud, Mylène Roussellet, and Vincent Verneuil. Rosetta for single trace analysis. In Steven Galbraith and Mridul Nandi, editors, *Progress in Cryptology - INDOCRYPT 2012*, Lecture Notes in Computer Science, page 140–155, Berlin, Heidelberg, 2012. Springer. 14

[CLM20] Valence Cristiani, Maxime Lecomte, and Philippe Maurine. Leakage assessment through neural estimation of the mutual information. In *International Conference on Applied Cryptography and Network Security (ACNS)*, volume 12418

of *Lecture Notes in Computer Science*, page 144–162, Rome, Italy, October 2020. 12, 55

[CLMW09] Emmanuel J. Candes, Xiaodong Li, Yi Ma, and John Wright. Robust principal component analysis? (arXiv:0912.3599), December 2009. arXiv:0912.3599 [cs, math]. 61, 62, XIV

[COM24] Gauthier Cler, Sebastien Ordas, and Philippe Maurine. Bernoulli at the root of horizontal side channel attacks. In Shivam Bhasin and Thomas Roche, editors, *Smart Card Research and Advanced Applications*, page 107–126, Cham, 2024. Springer Nature Switzerland. 85

[Cor99] Jean-Sébastien Coron. Resistance against differential power analysis for elliptic curve cryptosystems, 1999. 11, 13, XIII

[CRR03] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. Template attacks. In Burton S. Kaliski, çetin K. Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002*, Lecture Notes in Computer Science, page 13–28, Berlin, Heidelberg, 2003. Springer. 12

[CT07] Ping-Lin Chang and Wei-Guang Teng. Exploiting the self-organizing map for medical image segmentation. In *Twentieth IEEE International Symposium on Computer-Based Medical Systems (CBMS'07)*, page 281–288, June 2007. 78

[CUH16] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). (arXiv:1511.07289), February 2016. arXiv:1511.07289 [cs]. 49

[Cyb89] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, December 1989. 49

[DB79] David L. Davies and Donald W. Bouldin. A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1(2):224–227, April 1979. 16

[DCG16] Gopal Das, Manojit Chattopadhyay, and Sumeet Gupta. A comparison of self-organising maps and principal components analysis. *International Journal of Market Research*, 58, July 2016. 80

[DH76] W. Diffie and M. Hellman. New directions in cryptography, November 1976. 4

[DLO+19] Ibrahima Diop, Yanis Linge, Thomas Ordas, Pierre-Yvan Liardet, and Philippe Maurine. From theory to practice: horizontal attacks on protected implementations of modular exponentiations, April 2019. 20

[DLR77] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm, 1977. 14, 23, XVI

[DMG+17] Nat Dilokthanakul, Pedro A. M. Mediano, Marta Garnelo, Matthew C. H. Lee, Hugh Salimbeni, Kai Arulkumaran, and Murray Shanahan. Deep unsupervised clustering with gaussian mixture variational autoencoders. (arXiv:1611.02648), January 2017. arXiv:1611.02648 [cs, stat]. 58

[DV99] G.A. Darbellay and I. Vajda. Estimation of the information by an adaptive partitioning of the observation space. *IEEE Transactions on Information Theory*, 45(4):1315–1321, May 1999. 55

[Fis19] R. A. Fisher. Xv.—the correlation between relatives on the supposition of mendelian inheritance. *Transactions of the Royal Society of Edinburgh*, 52(2):399–433, 1919. 22

[FT04] B. Fuglede and F. Topsoe. Jensen-shannon divergence and hilbert space embedding. In *International Symposium onInformation Theory, 2004. ISIT 2004. Proceedings.*, pages 31–, June 2004. 74

[GB10] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, page 249–256. JMLR Workshop and Conference Proceedings, March 2010. 52

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org. 51

[GBTP08] Benedikt Gierlichs, Lejla Batina, Pim Tuyls, and Bart Preneel. Mutual information analysis. In Elisabeth Oswald and Pankaj Rohatgi, editors, *Cryptographic Hardware and Embedded Systems – CHES 2008*, Lecture Notes in Computer Science, page 426–442, Berlin, Heidelberg, 2008. Springer. 12

[GJJR] Gilbert Goodwill, Benjamin Jun, Josh Jaffe, and Pankaj Rohatgi. A testing methodology for side channel resistance validation. In *NIST non-invasive attack testing workshop, 2011*. 12, 22

[GPAM+14] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. (arXiv:1406.2661), June 2014. arXiv:1406.2661 [cs, stat]. 68, XV

[HCL06] R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, page 1735–1742, June 2006. 50

[HIM+13] Johann Heyszl, Andreas Ibing, Stefan Mangard, Fabrizio De Santis, and Georg Sigl. Clustering algorithms for non-profiled single-execution attacks on exponentiations. (2013/438), 2013. Publication info: Published elsewhere. CARDIS 2013, Springer. The final publication is available at link.springer.com. 14

[HKT15] Neil Hanley, HeeSeok Kim, and Michael Tunstall. Exploiting collisions in addition chain-based exponentiation algorithms using a single trace. In Kaisa Nyberg, editor, *Topics in Cryptology — CT-RSA 2015*, Lecture Notes in Computer Science, page 431–448, Cham, 2015. Springer International Publishing. 14

[Hor91] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257, January 1991. 49

[Hot33] H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24(6):417–441, 1933. 37

[HS] Michael Hutter and Peter Schwabe. Cryto jedi μnacl library for curve25519. https://munacl.cryptojedi.org/curve25519-msp430.shtml. 20

[HS06] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, July 2006. 51

[IZZE18] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. (arXiv:1611.07004), November 2018. arXiv:1611.07004 [cs]. 69, 72

[JCB84] William Full James C. Bezdek, Robert Ehrlich. Fcm: The fuzzy c-means clustering algorithm, 1984. 23

[JJHG00] Nathalie Japkowicz, Stephen Jose Hanson, and Mark A. Gluck. Nonlinear autoassociation is not equivalent to pca. *Neural Comput.*, 12(3):531–545, mar 2000. 52

[Jol02] I. T. Jolliffe. *Principal Component Analysis*. Springer Series in Statistics. Springer-Verlag, New York, 2002. 14, XIV

[JT19] Longlong Jing and Yingli Tian. Self-supervised visual feature learning with deep neural networks: A survey. (arXiv:1902.06162), February 2019. arXiv:1902.06162 [cs]. 50

[JY03] Marc Joye and Sung-Ming Yen. The montgomery powering ladder, 2003. 11

[KB17] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. (arXiv:1412.6980), January 2017. arXiv:1412.6980 [cs]. 52

[Kiv98] Kimmo Kiviluoto. Predicting bankruptcies with the self-organizing map. *Neurocomputing*, 21(1):191–201, November 1998. 78

[KJJ99] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis, 1999. 10

[KLM96] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. (arXiv:cs/9605103), April 1996. arXiv:cs/9605103. 51

[Koc95] C. K. Koc. Analysis of sliding window techniques for exponentiation. *Computers & Mathematics with Applications*, 30(10):17–24, 1995. 13

[Koc96] Paul C. Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems, 1996. 9, 10, 11, XIII

[Koh90] T. Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, September 1990. 78, 80, XV

[KPH+19] Jaehun Kim, Stjepan Picek, Annelie Heuser, Shivam Bhasin, and Alan Hanjalic. Make some noise. unleashing the power of convolutional neural networks for profiled side-channel analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, page 148–179, May 2019. 48

[Kra91] Mark A. Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal*, 37(2):233–243, 1991. 51, XIV

[KTKI02] Makoto Kinouchi, Naoshi Takada, Yoshihiro Kudo, and Toshimichi Ikemura. Quick learning for batch-learning self-organizing map. *Genome Informatics*, 13:266–267, 2002. 80

[Kul97] Solomon Kullback. *Information Theory and Statistics*. Courier Corporation, July 1997. Google-Books-ID: 05LwShwkhFYC. 55

[KW22] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. (arXiv:1312.6114), December 2022. arXiv:1312.6114 [cs, stat]. 58

[KYG+22] Tung Kieu, Bin Yang, Chenjuan Guo, Christian S. Jensen, Yan Zhao, Feiteng Huang, and Kai Zheng. Robust and explainable autoencoders for unsupervised time series outlier detection—extended version. (arXiv:2204.03341), April 2022. arXiv:2204.03341 [cs]. 61, XIV

[Llo82] S. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982. 14

[LW05] Yonggang Liu and Robert H. Weisberg. Patterns of ocean current variability on the west florida shelf using the self-organizing map. *Journal of Geophysical Research: Oceans*, 110(C6), 2005. 80

[LWM06] Yonggang Liu, Robert H. Weisberg, and Christopher N. K. Mooers. Performance evaluation of the self-organizing map for feature extraction. *Journal of Geophysical Research: Oceans*, 111(C5), 2006. 80

[Mac67] J. B. MacQueen. Some methods for classification and analysis of multivariate observations, 1967. 23

[MBvLM12] Dimitrios Mavroeidis, Lejla Batina, Twan van Laarhoven, and Elena Marchiori. *PCA, Eigenvector Localization and Clustering for Side-Channel Attacks on Cryptographic Hardware Devices*, volume 7523 of *Lecture Notes in Computer Science*, page 253–268. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. 15

[MDP19] Loïc Masure, Cécile Dumas, and Emmanuel Prouff. Gradient visualization for general characterization in profiling attacks. In Ilia Polian and Marc Stöttinger, editors, *Constructive Side-Channel Analysis and Secure Design*, page 145–167, Cham, 2019. Springer International Publishing. 48

[MLX+17] Xudong Mao, Qing Li, Haoran Xie, Raymond Y. K. Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. (arXiv:1611.04076), April 2017. arXiv:1611.04076 [cs]. 72

[Mon87] Peter L Montgomery. Speeding the pollard and elliptic curve methods of factorization, 1987. 11

[MRL95] Young-Il Moon, Balaji Rajagopalan, and Upmanu Lall. Estimation of mutual information using kernel density estimators. *Physical Review E*, 52(3):2318–2321, September 1995. 55

[NC17] Erick Nascimento and Lukasz Chmielewski. Horizontal clustering side-channel attacks on embedded ECC implementations (extended version). Cryptology ePrint Archive, Report 2017/1204, 2017. 16, 17, 19, 20, 32, 43, 44, 86

[NH10] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, page 807–814, Madison, WI, USA, June 2010. Omnipress. 49

[PC16] Guilherme Perin and Łukasz Chmielewski. A semi-parametric approach for side-channel attacks on protected rsa implementations, 2016. 16, 17, 19, 24, 25, 36, 43, 44, 64, 75, 76, 78, 83, 86, 87, 90, IX, XIV

[PCBP20] Guilherme Perin, Łukasz Chmielewski, Lejla Batina, and Stjepan Picek. Keep it unsupervised: Horizontal attacks meet deep learning, Dec. 2020. 16, 17, 19, 20, 29, 43, 44, 100, X

[Pea01] Karl Pearson. Liii. on lines and planes of closest fit to systems of points in space, 1901. 37

[PITM14] Guilherme Perin, Laurent Imbert, Lionel Torres, and Philippe Maurine. Attacking randomized exponentiations using unsupervised learning, April 2014. 15, 16, 17, 19, 23, 25, 43, 44, 86, XI, XIV

[PK19] Ravi Ponmalai and Chandrika Kamath. *Self-Organizing Maps and Their Applications to Data Analysis*. Number LLNL-TR-791165. September 2019. 79, 80, 82, 83

[Pla18] Elad Plaut. From principal subspaces to principal components with linear autoencoders. (arXiv:1804.10253), December 2018. arXiv:1804.10253 [cs, stat]. 52

[PSK+18] Stjepan Picek, Ioannis Petros Samiotis, Jaehun Kim, Annelie Heuser, Shivam Bhasin, and Axel Legay. On the performance of convolutional neural networks for side-channel analysis. In Anupam Chattopadhyay, Chester Rebeiro, and Yuval Yarom, editors, *Security, Privacy, and Applied Cryptography Engineering*, page 157–176, Cham, 2018. Springer International Publishing. 48

[RSA78] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, feb 1978. 5

[SA08] François-Xavier Standaert and Cedric Archambeau. *Using Subspace-Based Template Attacks to Compare and Combine Power and Electromagnetic Information Leakages*, volume 5154 of *Lecture Notes in Computer Science*, page 411–425. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. 15

[SHKS15] Robert Specht, Johann Heyszl, Martin Kleinsteuber, and Georg Sigl. Improving non-profiled attacks on exponentiations based on clustering and extracting leakage from multi-channel high-resolution em measurements. In Stefan Mangard and Axel Y. Poschmann, editors, *Constructive Side-Channel Analysis and Secure Design*, Lecture Notes in Computer Science, page 3–19, Cham, 2015. Springer International Publishing. 14, 15, 19, XIV

[Sho97] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, October 1997. arXiv:quant-ph/9508027. 5

[SL11] Scott C. Sheridan and Cameron C. Lee. The self-organizing map in synoptic climatological research. *Progress in Physical Geography: Earth and Environment*, 35(1):109–119, February 2011. 78

[Spe04] C. Spearman. The proof and measurement of association between two things. *The American Journal of Psychology*, 15(1):72–101, 1904. 95

[SSSK08] Taiji Suzuki, Masashi Sugiyama, Jun Sese, and Takafumi Kanamori. Approximating mutual information by maximum likelihood density ratio estimation. In *Proceedings of the Workshop on New Challenges for Feature Selection in Data Mining and Knowledge Discovery at ECML/PKDD 2008*, page 5–20. PMLR, September 2008. 55

[SZ15] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv:1409.1556 [cs]*, April 2015. arXiv: 1409.1556. 49

[US90] Alfred Ultsch and H. Peter Siemon. Kohonen's self organizing feature maps for exploratory data analysis. In Bernard Widrow and Bernard Angeniol, editors, *Proceedings of the International Neural Network Conference (INNC-90), Paris, France, July 9–13, 1990 1. Dordrecht, Netherlands*, volume 1, pages 305–308, Dordrecht, Netherlands, 1990. Kluwer Academic Press. 83

[vdMH08] L.J.P. van der Maaten and G.E. Hinton. Visualizing high-dimensional data using t-sne. *Journal of Machine Learning Research*, 9(nov):2579–2605, 2008. 57

[Wal01] C. D. Walter. Sliding windows succumbs to big mac attack. In Çetin K. Koç, David Naccache, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems — CHES 2001*, Lecture Notes in Computer Science, page 286–299, Berlin, Heidelberg, 2001. Springer. 11, 13, 14

[WEL47] B. L. WELCH. The generalization of student's problem when several different population variances are involved, January 1947. 22

[WvWM11] Marc F. Witteman, Jasper G. J. van Woudenberg, and Federico Menarini. Defeating rsa multiply-always and message blinding countermeasures. In Aggelos Kiayias, editor, *Topics in Cryptology – CT-RSA 2011*, Lecture Notes in Computer Science, page 77–88, Berlin, Heidelberg, 2011. Springer. 14

[Yin08] Hujun Yin. Learning nonlinear principal manifolds by self-organising maps. In Alexander N. Gorban, Balázs Kégl, Donald C. Wunsch, and Andrei Y. Zinovyev, editors, *Principal Manifolds for Data Visualization and Dimension Reduction*, Lecture Notes in Computational Science and Enginee, page 68–95, Berlin, Heidelberg, 2008. Springer. 80

[ZBHV20] Gabriel Zaid, Lilian Bossuet, Amaury Habrard, and Alexandre Venelli. Methodology for efficient cnn architectures in profiling attacks. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, page 1–36, 2020. 48

[ZP17] Chong Zhou and Randy C. Paffenroth. Anomaly detection with robust deep autoencoders. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, page 665–674, New York, NY, USA, 2017. Association for Computing Machinery. 61

[ZPIE20] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. (arXiv:1703.10593), August 2020. arXiv:1703.10593 [cs]. 69

Appendix A

Notes

Global means and variance can be computed from mixture components parameters by applying the laws of total expectations and variance:

$$\begin{aligned} \overline{\mu} &= \mathbb{E}\left[\mathbb{E}(X|Z)\right] \\ &= \sum_{i=1}^{k} \pi_i \mu_i \end{aligned} \tag{A.1}$$

$$\begin{aligned} \text{Var}(X) &= \mathbb{E}\left[\text{Var}(X|Z)\right] + \text{Var}\left(\mathbb{E}[X|Z]\right) \\ &= \mathbb{E}\left[\sigma^2(Z)\right] + \text{Var}\left(\mu(Z)\right) \\ &= \sum_{i=1}^{k} \pi_i \sigma_i^2 + \sum_{i=1}^{k} \pi_i (\mu_i - \overline{\mu})^2 \end{aligned} \tag{A.2}$$

Appendix B

List of acronyms

ADMM Alternating Direction Method of Multipliers. 59

AES Advanced Encryption Standard. 5

AIC Akaike Information Criterion. 103

BMU Best Matching Unit. 77, 79, 81

BRR Bit Recovery Rate. 29, 35, 41–43, 55, 56, 61–64, 71, 72, 74, 80, 81, 88, 96–99, 101, 102, 105, IV–VII

CNN Convolutional Neural Network. 46, 60, 65

CPA Correlation Power Analysis. 12

CRT Chinese Remainder Theorem. 6, 7, 9

DA Double and Add. 10

DLSCA Deep Learning Side Channel Attacks. 13

DoM Difference of Means. 15, 35

DPA Differential Power Analysis. 10, 11, III

ECC Elliptic Curve Cryptography. 7, 8, 10, 14

ECDLP Elliptic Curve Discrete Logarithm Problem. 7

ECSM Elliptic Curve Scalar Multiplication. 20

EM Expectation Maximization. 15, 85, 91, 92, 94, 95, 102, 104, 107, V

EV Explained Variance. 37

GAN Generative Adversarial Network. 66–75, 79, 82, 98–102, 105–107, V–VII, XIII

GMVAE Gaussian Mixture Variational Autoencoder. 56, 57

IC Integrated Circuit. 84

IQR Inter-Quantile Range. 32, 33, 63, IV

JSD Jensen-Shannon Divergence. 103

LIM Long-Integer Multiplication. 13, 14

LSB Least Significant Bit. 6, 20

MI Mutual Information. 53, 54, 60

MINE Mutual Information Neural Estimator. 53, 60, 61, VII

MLP Multilayer Perceptrons. 13, 46, 65

MSB Most Significant Bit. 6, 20

MSE Mean Squared Error. 50

p.d.f Probability Density Function. 16, 25

p.m.f Probability Mass Function. 88, 89

PC Principal Component. 39, 41, 42, IV, VII

PCA Principal Component Analysis. 1, 15, 36–42, 44, 49–51, 53–57, 59, 78, 82, 109, IV

PoI Points of Interest. 1, 2, 13, 15–17, 19, 21–25, 27–29, 31, 32, 34–37, 39, 41–44, 54, 56, 57, 62, 72–76, 81–86, 88, 89, 91, 92, 94–102, 104–107, 109, 110, V, VI

RAE Robust Autoencoder. 61–66, 70–75, 79, 82, 98–102, 105–107, IV–VII, XIII

RSA Rivest-Shamir-Adleman. 5, 7, 8, 20

SCA Side Channel Attacks. 4, 8, 9, 12, 13, 15, 20–22, 32, 37, 38, 45, 46, 48, 53, 56, 57, 70, 78, 82, 85, 89

SM Square and Multiply. 10, 20

SOM Self Organizing Map. 76–82, V

SOST Sum Of Squared $\mathcal{T}$-values. 16

SPA Simple Power Analysis. 9, 10, 13, III

SVD Singular Value Decomposition. 37

TA Template Attacks. 12

TVLA Test Vector Leakage Assessment. 22, 52, 87

Appendix C

Glossary

Curse of Dimensionality As dimensionality of data increases, space volume growth exponentially, resulting in fewer observations per region. 37

Montgomery Ladder Constant time implementation for modular exponentiation or scalar multiplication. 20

Square and Multiply Binary exponentiation that perform a square operation if the assessed bit is equal to 0, or a square followed by a multiply operation otherwise.. 6

List of Figures

1.1 Illustration of key generation (left) encryption/decryption (middle) and signature process (right) in asymmetric cryptography. 5
1.2 Simplified schema of a side channel analysis. 9
1.3 SPA by visual inspection on Square and Multiply binary exponentiation (Algorithm 1). Recovered private exponent is $d = 101101$. 9
1.4 Illustration of the DPA. Upper and middle plots show averaged traces with $Y = 0$ and $Y = 1$ respectively. Lower plot shows the difference. 10

2.1 A randomly selected pattern and the average pattern for the three considered datasets 21
2.2 Welch's statistic values obtained for the three datasets considered 23
2.3 Comparison of supervised $\mathcal{T}$-values with the unsupervised $\mathcal{T}$-values obtained after clustering of each time point for $\mathcal{D}_{mbedtls}$ dataset. 25
2.4 Histograms of the four most leaking points (and their index) according to $\mathcal{T}$-values obtained after applying the unsupervised approach of Perin *et al.* to the $\mathcal{D}_{mbedtls}$ dataset. The results are the same with supervised approach. . . . 25
2.5 Illustration of simulated univariate leakages. Top figure displays a high leaking point, characterized by high difference of means. Bottom figure displays completely overlapping components, resulting in non-exploitable point. Middle figure shows two different behavior but not enough separated to results in a bimodal mixture. Note that bimodality arises only when $|\mu_1 - \mu_2| > 2 \times \min(\sigma_1, \sigma_2)$ [Beh70]. 26
2.6 Comparison of supervised vs unsupervised $\mathcal{T}$-values (after univariate clustering on each time point) for $\mathcal{D}_{pointer}$ dataset. 27
2.7 Histogram of the four most leaking points and their index according to $\mathcal{T}$-values on unsupervised (top row) and supervised (bottom row) approaches for $\mathcal{D}_{pointer}$. 28
2.8 Comparison of supervised vs unsupervised T-test (after clustering) T-test values for $\mathcal{D}_{arith}$. 28
2.9 Histogram of the four most leaking points of $\mathcal{D}_{arith}$ (and their index) according to $\mathcal{T}$-values for the unsupervised (top row) and supervised (bottom row) approaches. 29
2.10 BRR. 30
2.11 Comparison of $\mathcal{T}$-values obtained in supervised manner (left) and in unsupervised manner (right) on simulated mixtures. 31
2.12 Heatmap of estimated $\max(\sigma_1, \sigma_2)$. Left: supervised approach ($\max(\sigma_1, \sigma_2) = \sigma_1 = \sigma_2$). Right: unsupervised approach with σ values estimated from after classification of values by the k-means algorithm. 31

2.13 Evolution of the percentage of identified outliers using the IQR method when adding saturation on both sides of the distribution (top) or on a single side (bottom). . . . 33
2.14 Distribution of points. . . . 33
2.15 Percentage of outliers values per time points for $\mathcal{D}_{mbedtls}$ (top), $\mathcal{D}_{pointer}$ (middle) and $\mathcal{D}_{arith}$ (bottom) according to Equation 2.9. . . . 34
2.16 Percentage of saturation values per time points for $\mathcal{D}_{mbedtls}$ (top), $\mathcal{D}_{pointer}$ (middle) and $\mathcal{D}_{arith}$ (bottom). . . . 35
2.17 Visualization of clustering partitions obtained for different percentages of added outliers. . . . 36
2.18 Metrics evolution. . . . 36
2.19 Visualization of the first two components obtained by decomposition in principal components of the synthetic datasets with $p_{noise} = 1$ (left), $p_{noise} = 2$ (middle) and $p_{noise} = 3$ additional noise dimensions. . . . 39
2.20 Visualization of the first two components obtained by decomposition in principal components of $\mathcal{D}_{mbedtls}$ (left), $\mathcal{D}_{pointer}$(middle) and $\mathcal{D}_{arith}$ (right). For readability purpose, only 2000 points are displayed, sampled uniformly from all patterns. Nevertheless, the PCA is applied on the entire datasets. . . . 39
2.21 Explained Variance ratios of the first thirty components for the considered datasets. . . . 40
2.22 Heatmap of the contribution of each temporal point from the input to the first 100 principal components extracted from the PCA. . . . 40
2.23 Produced ELV for each time point on the first 10 principal components extracted from the PCA. . . . 41
2.24 Cumulative ELV for each time point on the first 20 principal components extracted from the PCA. . . . 41
2.25 T-test on each PC . . . 42

3.1 Example of a one dimensional convolution. . . . 48
3.2 Illustration of main activation functions . . . 50
3.3 Autoencoder architecture illustration using fully connected layers. . . . 52
3.4 Pairwise Pearson correlation . . . 53
3.5 $\mathcal{T}$-values obtained in a supervised manner for each dimension of the projected subspace with using either a linear autoencoder, a non-linear autoencoder or a PCA. From left to right the results for $\mathcal{D}_{mbedtls}$, $\mathcal{D}_{pointer}$ and $\mathcal{D}_{arith}$. 54
3.6 Mutual Information between the code and labels for different code sizes. . . . 56
3.7 Averaged BRR vs p obtained, for the three datasets considered, by applying the Fuzzy k-means to the projections /codes provided by the PCA, linear autoencoder and non-linear autoencoder. . . . 57
3.8 t-SNE projections into a two-dimensional space of 1000 patterns extracted from the code. Left column: $\mathcal{D}_{mbedtls}$. Middle column: $\mathcal{D}_{pointer}$. Right column: $\mathcal{D}_{arith}$. . . . 60
3.9 $\mathcal{T}$-test comparison before (blue) and after (red) application of the RAE, and the difference (green). . . . 63
3.10 BRR values before (blue) and after (red) application of the RAE, and the difference (green). . . . 64
3.11 Scatter plot of the BRR and $\mathcal{T}$-test values for each time point before and after application of the RAE. . . . 65

3.12 Percentage of time points with a BRR higher than a given threshold obtained after and before applying the RAE. 66
3.13 Percentage of outliers per time point (IQR) before (blue) and after (red) the application of the RAE. The green curve describes the improvement (< 0) or the deterioration (> 0). 67
3.14 Minimum and maximum values for each time point before and after the RAE cleaning. 68
3.15 Illustration of the GAN architecture 69
3.16 CycleGAN Architecture. Information flows from input data A and B are represented by blue and red colors. 70
3.17 Modified CycleGAN architecture with the add of a multiplexer m 71
3.18 Percentage of outliers per time point (IQR) before (blue) and after (red) the application of the GAN. The green curve describes the improvement (< 0) or the deterioration (> 0).. 74
3.19 $\mathcal{T}$-values before (blue) and after (red) applying the GAN, and the difference in green. 75
3.20 BRR before (blue) and after (red) applying the GAN, and the difference in green. 76
3.21 BRR obtained on the raw and cleaned datasets provided by the RAE and GAN 76
3.22 Jensen-Shannon divergence values between corresponding time points before and after the application of either the RAE or the CycleGAN. 77
3.23 Distribution of time point 838 of $\mathcal{D}_{pointer}$ before and after applying the RAE or the GAN. 77
3.24 $\mathcal{T}$-values obtained after the initial clustering step of the attack proposed in [PC16] applied to the raw and cleaned datasets. 78
3.25 Simplified view of the SOM architecture 79
3.26 Projection obtained for the three datasets considered. 81
3.27 BRR vs. number of selected points with and without SOM dimensionality reduction. 82
3.28 SOM U-matrix on $\mathcal{D}_{arith}$ using the 300 best PoI. 83

4.1 Illustration of the probability of confusion p_{conf}, for $k = 2$ (top row) and $k = 3$ (bottom row). The left plots show the mixture components, while the right plots show the posterior probabilities and the area materializing the p_{conf}. 89
4.2 T-test (top row) and $1 - p_{conf}$ (bottom row, inverted for readability) applied on each time point, for the three datasets considered. 89
4.3 BRR scores obtained, on the three datasets, after applying the Fuzzy k-means clustering on the most leaking PoI, identified using the T-test or the p_{conf} metric with the knowledge of the labels. 90
4.4 Computed p_{conf} on discrete data after the EM process (time point 641 on $\mathcal{D}_{pointer}$). 93
4.5 Spearman correlation (absolute values) matrices for $\mathcal{D}_{pointer}$ and $\mathcal{D}_{arith}$. 95
4.6 Comparison of time points 640 and 643 on $\mathcal{D}_{pointer}$. 95
4.7 200 first ranked p_{conf} values after the applying Spearman's augmentation on the three considered datasets. 97
4.8 Obtained p_{conf} values (top row) and p_{conf} with added neighbors points using the Spearman correlation (middle row), as well as the supervised BRR values for reference (bottom row), on the three considered datasets. 98

4.9 Minimum, maximum, mean, and quartile values of the success rates (BRR) obtained for the clustering attack using a single trace (255 patterns for $\mathcal{D}_{pointer}$ and $\mathcal{D}_{arith}$, 2048 for $\mathcal{D}_{mbedtls}$). The attack was performed 100 times on different traces. . . . 99
4.10 Minimum, maximum, mean, and quartile values of the success rates (BRR) obtained with the supervised clustering attack where PoI are chosen according to the highest $\mathcal{T}$-values, using a single trace (255 patterns for $\mathcal{D}_{pointer}$ and $\mathcal{D}_{arith}$, 2048 for $\mathcal{D}_{mbedtls}$). The attack was performed 100 times on different traces. . . . 99
4.11 Minimum, maximum, mean values of the success rates (BRR) obtained after applying the bits correction framework from Perin *et al.* [PCBP20] on the assigned classes from the clustering using the PoI selected with the proposed mixture analysis. . . . 100
4.12 p_{conf} values obtained after applying the mixture analysis to patterns of $\mathcal{D}_{pointer}$ cleaned with the RAE and CycleGAN procedures. . . . 101
4.13 p_{conf} values obtained after applying the mixture analysis to patterns of $\mathcal{D}_{arith}$ cleaned with the RAE and CycleGAN procedures with different number of sought components in the mixtures. . . . 102
4.14 Obtained BRR values from the clustering attack using the raw patterns and cleaned patterns from the RAE and using CycleGAN, on $\mathcal{D}_{pointer}$ and $\mathcal{D}_{arith}$. Each time, a single trace is used (255 patterns), and the process is repeated 100 times. . . . 103
4.15 Obtained BRR values from the clustering attack using the cleaned patterns from the RAE and using CycleGAN, on $\mathcal{D}_{pointer}$ and $\mathcal{D}_{arith}$. Each time, a single trace is used (255 patterns), and the process is repeated 100 times. . . . 103
4.16 BRR values obtained after the clustering attack using the cleaned patterns from the RAE and using CycleGAN and applying the Spearman neighbors analysis on cleaned patterns, on $\mathcal{D}_{pointer}$ and $\mathcal{D}_{arith}$. Each time, a single trace is used (255 patterns), and the process is repeated 100 times. . . . 104
4.17 Minimum number of components allowing to meet the selection criteria for each time point of $\mathcal{D}_{pointer}$ and $\mathcal{D}_{arith}$. . . . 106
4.18 Mixture estimation with 6 components (time point 466 on $\mathcal{D}_{pointer}$). . . . 106
4.19 Percentage of time points per number of components. . . . 107
4.20 p_{conf} values obtained by applying the adaptive PoI selection procedure on $\mathcal{D}_{pointer}$ and $\mathcal{D}_{arith}$. . . . 107
4.21 Percentage of time points versus the minimum number of components required to pass all statistical tests. . . . 108
4.22 p_{conf} values obtained from the adaptive PoI selection procedure on $\mathcal{D}_{pointer}$ and $\mathcal{D}_{arith}$. . . . 108

List of Tables

1.1 Summary of [PITM14] methodology. 16

2.1 Mixture parameters for T-test experiment 30
2.2 Percentage of outliers and saturated values on each considered dataset. 34
2.3 Mixture parameters for PCA experiment 38
2.4 BRR obtained after apply clustering on k first PC. 42

3.1 Activation functions 49
3.2 Mutual information estimates (using MINE) of each matrix and the ground truth labels vector $\mathbf{Y}$. 63
3.3 Comparison of the percentage of outliers and saturated values before and after the RAE optimization. Best obtained results are highlighted with bold font. 66
3.4 Percentage of outliers and saturated values obtained on original patterns, after applying the RAE and the CycleGAN. The best results are highlighted in bold. 73

Résumé

Dans un monde hyper-connecté ou de grandes quantités d'informations digitales circulent et sont échangées, le sécurité des ces données potentiellement sensibles est d'une importance cruciale. Afin de sécuriser ces communications, des algorithmes cryptographiques sont communément utiliser afin de garantir la non-divulgation de l'information. Ces algorithmes éprouvés, sont basés sur des problèmes dit "durs" qui signifient qu'il est calculatoirement impossible de déchiffrer les contenus des messages chiffrés. Dans les travaux présentés ici, nous considérons l'algorithme RSA, basé sur l'impossibilité de factoriser de grand nombres, et le chiffrement à base de courbes elliptiques qui appuie sa sécurité sur le problème du logarithme discret. Bien que comme indiqué, ces algorithmes se basent sur des problèmes durs, il est possible d'exploiter leur fonctionnement en utilisant des sources auxiliaires. Par exemple, si l'on considère un algorithme de chiffrement qui est utilisé sur un système embarqué, tel qu'une carte à puce, un microcontrôleur ou un objet connecté, il est possible d'observer le comportement indirect de celui-ci lors de l'exécution des routines cryptographiques. Différents canaux dit auxiliaires peuvent être considérés, comment notamment l'observation de la consommation de courant du système, ses émanations électromagnétiques ou acoustiques. En utilisant des méthodes adaptées, il est alors possible d'exploiter ces canaux afin de récupérer l'information secrète, sans perturber le comportement du système. Ce procédé d'exploitation est communément défini par le terme d'attaques par canaux auxiliaires.

Depuis l'introduction en 1996 par Kocher des attaques par simple inspection des mesures (Simple Power Analysis) [Koc96], les attaques par canaux auxiliaires ont constamment été améliorées afin de passer outre les contre-mesures mises en place par les concepteurs de systèmes embarqués pour prévenir ces dites attaques. Néanmoins, certaines contre-mesures comme l'application de masquage et l'utilisation d'exposants aléatoires à chaque chiffrement rendent l'exploitation multi-traces et l'application d'outils statistiques proposés impossibles [Koc96, Cor99]. Dans ces cas de figure, l'attaquant peut uniquement exploiter le signal résultant d'une seule trace d'exponentiation ou de multiplication scalaire. Cette catégorie d'attaques constitue le cas le plus défavorable pour l'attaquent et est connue sous le nom d'attaques horizontales.

Une approche couramment utilisée pour l'exploitation est l'attaque par partitionnement. Elle consiste à diviser en plusieurs paquets (ou clusters en anglais) les séquences observées afin de pouvoir identifier les différents comportements et ainsi reconstruire l'information secrète manipulée. Par exemple, l'implémentation de l'exponentiation modulaire du RSA, considère et manipule souvent chaque bit de l'exposant secret un à un et effectue une opération précise en fonction de sa valeur. Il est donc possible d'identifier et de séparer les comportements observés dans le signal en fonction de la valeur du bit manipulé. Différentes

approches ont été proposées dans la littérature scientifique afin de sélectionner de manière optimale des parties de trace pour exploitation dans l'algorithme de partitionnement (le plus souvent le k-moyennes). D'une part, certains travaux ont proposé l'application de méthodes de réduction de dimensions (notamment l'analyse en composantes principales [Jol02]) afin de compresser les motifs d'entrée [SHKS15]. D'autre part, plusieurs contributions proposent l'analyse univarié de chaque point temporel afin de sélectionner des points d'intérêts basé sur un modèle de fuite défini [PITM14, PC16]. Bien que ces méthode soient pertinentes dans le cadre de signaux de bonne qualité, elles sont peu ou pas applicables en présence de bruits de mesures ou de comportements parasites. En effet, les méthodes utilisées ne prennent pas en compte les possibles artefacts présents dans le signal observé ce qui résulte en des estimations biaisées dans des cas d'usage plus complexes.

Dans ces travaux, nous nous intéressons donc à l'étude de méthodes pouvant être applicable dans ces conditions.

Nous étudions d'abord l'utilisation de réseaux de neurones de types auto-encodeurs comme alternative aux méthodes traditionnelles de réduction de dimension [Kra91]. Ces architectures de réseaux, appliquées dans différents domaines tel que le traitement numérique des images ou le domaine de la santé sont capables de compresser des données de grandes dimensions vers un sous espace de dimension réduite. L'avantage de ce type d'architecture réside dans le fait que leur optimisation peut être effectuée de manière complètement non supervisée. De plus, les auto-encodeurs sont capables d'extraire des relations non linéaires entre les points temporels d'entrées, contrairement à l'analyse en composantes principales qui n'est capable que d'extraire uniquement des corrélations linéaires. Appliqués à différents cas d'étude, nous avons pu constater que les auto-encodeurs produisent des performances similaires aux transformations en composantes principales, mais sont toutefois capables de compresser plus de données dans un sous espace plus petit, en exploitant les corrélations non linéaires entre les points d'entrée. Néanmoins, nous avons également pu observer que les auto-encoders étaient plus sensibles aux bruits parasites que les méthodes de réductions de dimensions classiques. De ce fait, bien qu'ils constituent une alternative pertinente, ils ne permettent pas d'obtenir de meilleurs résultats. De plus, la nécessité de régler les paramètres du réseau rendent leur application plus compliquée, particulièrement dans un contexte non supervisé, contrairement à l'analyse en composantes principales dont le comportement est déterministe.

Nous nous penchons donc ensuite sur des méthodes intelligentes pouvant mitiger les comportements parasites identifiés dans les signaux. Ces comportements sont par exemple des valeurs extrêmes par rapport au signal moyen mesuré, des saturations sur la plage de mesures ou des bruits supplémentaires provenant de sources directes. Cela peut être causé par un mauvais réglage du matériel de mesure, notamment la dynamique de l'oscilloscope. Nous avons donc identifié deux candidats pouvant convenir à la résolution de ce problème.

Le premier candidat est l'auto-encodeur robuste [KYG+22]. C'est une variation des auto-encodeurs précédemment étudié, basée sur l'analyse robuste en composantes principales [CLMW09], mais pouvant exploiter des corrélations non linéaires. Cette approche consiste à factoriser les données d'entrée $\mathbf{X}$ en deux matrices $\mathbf{L}$ et $\mathbf{S}$ qui représentent respectivement les données propres et les valeurs parasite extraites. Cette factorisation est effectuée par apprentissage et descente de gradient ainsi que par optimisation proximale. Les résultats obtenus

ont montré la pertinence de cette méthode. En effet, l'application de l'auto-encodeur robuste a permis la mitigation d'un pourcentage non négligeable de valeurs saturées présentes dans le signal d'origine. De plus, cette correction a eu pour effet d'augmenter les niveaux de fuites observés et ainsi la capacité de séparation des différentes classes. Néanmoins, bien que les données parasites soient en partie mitigées par cette méthode, le processus est totalement autonome, même si celles-ci peuvent être identifiées au préalable. Cela peut potentiellement avoir pour conséquence des effets de bords car les données non perturbées peuvent être également altérées durant ce processus.

Nous avons donc considéré une autre méthode, basée sur les réseaux de neurones antagonistes génératifs [GPAM+14]. Ces réseaux sont construits à partir d'une paire de deux réseaux, un discriminateur et un générateur. Le premier suit une architecture classique, qui effectue une classification binaire sur les données d'entrée, c'est à dire prédire si l'entrée donnée est réelle ou synthétique. Le but du second est de générer de la donnée synthétique et de faire croire au discriminateur qu'elle est bien réelle. L'apprentissage de ces deux réseaux se fait donc en concurrence, et a pour objectif de générer des données synthétiques convaincantes. Ce type de réseaux est souvent utilisés pour générer des images synthétiques. Dans notre cas d'application, nous utilisons les réseaux antagonistes afin de corriger les valeurs parasites identifiés sur le signal récupéré. L'objectif est de générer des points temporels synthétiques à la place des valeurs parasites, de tel sorte que le signal final modifié soit toujours convainquant pour le discriminateur. Pour ce faire, les valeurs parasites sont identifiées comme étant en dehors de 1.5 fois l'écart interquartile. Les résultats obtenus à l'issue de l'application de cette méthode montrent la pertinence des réseaux antagonistes pour la mitigation des valeurs parasites. En effet, la modification et correction ciblée (contrairement à l'autoencoder robuste) permet d'augmenter la qualité des points temporels, et de ce fait les niveaux de fuites observés et leur exploitabilité. De plus, certains points temporels, initialement considérés comme non exploitable du fait de leur haut niveaux de saturation, sont désormais exploitables. Il est toutefois important de noter que l'application de cette méthode, bien que pertinente et performance, requiert la paramétrisation complexe du réseau et de ses paramètres. Bien que lors de nos expériences, une architecture simple a suffi à corriger les valeurs aberrantes du signal, ce réseau peut être difficile à appliquer en pratique dans un contexte d'attaques horizontales.

Enfin, nous avons étudié l'utilisation de réseaux de neurones comme méthode alternative pour la réduction de dimension et la visualisation des projections résultantes. Nous considérons pour cela les cartes auto adaptatives de Konohen [Koh90]. Contrairement aux architectures précédemment présentées et couramment utilisées qui optimisent les paramètres de leur réseau en générant une erreur puis en la minimisant, l'apprentissage est fait ici de manière compétitive. Lors du traitement de chaque élément en entrée de réseau, il est associé au neurone vainqueur, qui est le plus similaire (défini par une distance). Les cartes auto adaptatives permettent une visualisation de la projection résultante sur un espace en 2 dimensions. De plus, cette architecture préserve la topologie et donc les distances entre les points d'entrée et projetés. L'application de cette architecture sur les signaux acquis a montré une meilleure résistance au bruit durant la phase de partitionnement, comparé au partitionnement direct. Cela permet également une meilleure analyse de la projection en deux dimensions, comparé à partitionnement multidimensionnel qui n'est pas facilement

interprétable. Toutefois, il est toujours nécessaire de sélectionner de manière optimale les points d'intérêts pour obtenir un partitionnement pertinent.

Les deux méthodes précédemment considérés pour la mitigation de valeurs parasites sont donc pertinentes car elles permettent d'améliorer la qualité des signaux d'entrées. Néanmoins, appliquer les méthodes de l'état de l'art pour la sélection de points d'intérêts sur des signaux issus de l'application de l'autoencoder robuste ou des réseaux antagonistes ne permet pas de sélectionner des points d'intérêt plus pertinents. En effet, le modèle de fuite choisi pour les méthodes de l'état de l'art univarié considère qu'un point temporel est porteur d'information de fuite uniquement si le nombre de composantes observées sur ce point temporel est égale au nombre de classes à partitionner. Ce n'est pas toujours le cas, car les comportements additionnels, parasites ou non, peuvent augmenter le nombre de composantes observées sur un point temporel. Nous proposons donc une nouvelle approche pour sélectionner des points d'intérêts pertinents qui se veut résistante au bruit et aux comportements additionnels qui pourraient survenir dans l'acquisition du signal. Pour ce faire, nous définissons un nouveau modèle de fuite comme ceci. Si l'on considère que l'opération appliquée lors de l'exponentiation modulaire ou la multiplication scalaire dépend de la valeur du bit observé, on peut formuler la valeur du bit comme un tirage de Bernoulli avec comme paramètre $p = 0.5$. De manière globale, la séquence de bits correspondant à la valeur secrète manipulée peut être formulée comme un processus de Bernoulli, et donc comme une variable aléatoire suivant une loi Binomiale de paramètre p. Afin d'évaluer cette répartition de classes, il est d'abord nécessaire d'estimer le nombre et les quantités des composantes formant chaque point temporel. Nous formulons chaque point temporel comme une mixture de distributions gaussiennes, avec leurs propres paramètres. Cette estimation est effectuée en utilisant l'algorithme d'espérance-maximisation [DLR77]. Il est donc possible suite à cette estimation d'évaluer et de rejeter tous les points temporels dont la répartition des classes ne suit pas une loi Binomiale de paramètre $p = 0.5$.

De plus, il est également possible d'évaluer la différence des variances entres les composantes d'informations identifiées. En effet, si l'on considère que le même processus physique est utilisé peu importe la valeur manipulée, toutes les composantes d'informations devraient être affectée par le même bruit gaussien. De ce fait, nous appliquons également un test statistique d'égalité des variances observées.

Toutefois, il est important de noter que même dans un contexte où deux classes sont attendues, il est possible d'observer des mixtures composées de composantes de bruit additionnels. Nous proposons donc une modification de la méthodologie afin d'inclure ces composantes parasite dans l'analyse. Sans cela, cela causerai des mauvaise estimations des composantes de la mixture.

Enfin, suite à la sélection de points temporels pertinent par rapport au modèle de fuite considéré, nous proposons une nouvelle métrique pour quantifier la fuite et donc l'exploitabilité d'un point temporel. Nous montrons qu'une métrique comme le T-test de Welsch n'est pas toujours adaptée pour mesurer efficacement la fuite sous jacente sur un point temporel. Au lieu d'utiliser les moments d'ordre un et deux (moyenne et variance) calculés lors de l'estimation des densitées, nous proposons d'évaluer directement les différences entres les ditributions des composantes à travers une métrique que nous nommons la p_{conf}. Cela permet de fournir une estimation bornée et comparable entre les points temporels, même avec

une echelle différente. De plus, nous montrons que la p_{conf} fournit une estimation plus précise que le T-test. Cette métrique peut être appliqué aussi bien dans le contexte d'une analyse de fuite supervisée que dans le cadre des attaques horizontales. Grâce à cette approche, nous obtenons des résultats supérieurs à ceux présentés dans l'état de l'art. De plus, l'application de cette méthode sur des patterns corrigées grâce aux techniques de réseaux de neuronnes non supervisées permet d'obtenir une amélioration des résultats comparé à l'application sur les patterns bruts.

Pour conclure ces travaux, nous dressons ci-après les perspectives possibles pour des études ultérieures sur les thématiques considérées.

Tout d'abord, l'utilisation de réseaux de neuronnes à apprentissage non supervisée appliqué a l'amélioration de la qualité des traces et des niveaux de fuites dans le contexte des attaques horizontales par canaux cachés s'est montrée pertinente. En effet, nous avons montré que les autoencoders robustes (RAE) ainsi que le CycleGAN étaient capables de corriger les valeurs anormales et parasites observées dans le signal d'entrée. Cela a permis d'améliorer les résultats d'attaques par partitionnement. Toutefois, il est nécessaire de souligner que ces méthodes présentent tout de même certaines limites. En effet, leur performance nécessite souvent un choix optimal d'hyper paramètres, qui n'est pas possible dans un contexte non supervisé. Dans la cas du CycleGAN, la position des valeurs extrèmes à corriger est fournie, ce qui rend l'apprentissage et la correction plus supervisée et pertinente dans le contexte des attaques horizontales.

Ensuite, concernant l'identification de fuites, nous pensons que l'analyse multivarié pourrait potentiellement produire de meilleurs point d'intérêts par rapport à l'analyse univariée proposée dans ces travaux. En effet, nous avons pu voir que le renforcement par analyse des points voisins avec la corrélation de Spearman permettait d'obtenir des niveaux de fuites améliorés. Néanmoins, la complexité d'analyse et de visualization de fuites multivariées peuvent rendrent leur application difficile. Ce point pourrait donc être un axe pertinent pour de futurs travaux.

Abstract

Recently introduced to the field of side channel analysis, neural networks have showed to be a powerful and relevant alternative to template attacks. However, their applicability is limited to profiled attack context, as supervised training is needed in order to build a relevant generalized model. When profiling on an open device is not possible, and vertical attacks cannot be applied, the only left possible approach is horizontal attacks. While several contributions have been made for tackling horizontal attacks on asymmetric cryptography algorithms implementations such as RSA or elliptic curve cryptography, their performance remains low and their applicability hard in real life scenario with the presence of high noise. Still, another neural network family known as unsupervised learning neural networks exists, which would not require an open device access and. It must be known if these networks unsupervised learning paradigm and their associated topology can be applied to the context of side-channel attacks and if such is the case, whether or not they can provide better results than traditional methods. Thus, In this work, several approaches are considered to improve clustering based horizontal side channel attacks efficiency. A novel methodology based on statistical analysis is also introduced for univariate points of interest selection. Additionally, an alternative metric for quantifying points of interest exploitability in a clustering attack is proposed and compared to commonly used metrics. The proposed methods allow providing significant improvement over state of the art attacks performance and giving a better explainability of obtained results.

Résumé

Récemment, l'utilisation de réseaux de neurones dans le cadre des attaques par canaux auxiliaires de type profilées sur systèmes ouverts a démontré son efficacité jusqu'à presque remplacer les techniques précédemment utilisées. Néanmoins, lorsqu'il n'est pas possible d'avoir accès à un système ouvert et lorsque les attaques dites verticales ne sont pas envisageables, la principale option restante est de mettre en oeuvre des attaques horizontales. C'est généralement le cas pour des implémentations d'algorithmes de cryptographie asymétrique (RSA, ECC). Les attaques horizontales existantes s'appuient sur des techniques de classification ou clustering issues du monde des statistiques. Même s'il a été apporté l'efficacité de ces techniques dans certaines circonstances, en pratique le taux de succès d'attaques basés sur ces techniques reste tout de même relativement faible, en particulier dans le cas d'environements avec la présence de hauts niveaux de bruits. Il existe toutefois une autre catégorie de réseaux de neurones dit à apprentissage non-supervisé, pour lesquels l'accès à un système ouvert n'est pas nécessaire. Il s'agit alors de déterminer si ce type de réseau peut être utilisé dans le contexte des attaques par canaux auxiliaire et, le cas échéant, si le gain par rapport aux techniques d'attaques déjà utilisées est suffisant. L'objectif de ces travaux est donc de valider cette idée en identifiant et en proposant une ou plusieurs topologies de réseaux de neurones adaptées aux attaques horizontales par clustering et d'en vérifier l'efficacité sur l'implémentation d'algorithmes cryptographiques asymétrique. De plus, une méthodologie alternative pour la sélection univariée de points d'intérêts basée sur des méthodes statistique est proposée, ainsi qu'une nouvelle métrique pour la quantification de fuite et l'exploitabilité des points sélectionnés. Les résultats obtenus montrent une amélioration des taux de succès obtenus par rapport aux méthodes proposées dans l'état de l'art.