MINISTRY OF EDUCATION & TRAINING MINISTRY OF NATIONAL DEFENSE

MILITARY TECHNICAL ACADEMY

DO NGOC TUAN

SIDE CHANNEL ATTACK BASED ON DISTRIBUTION OF SAMPLING CORRELATION AND MULTI-OUTPUT NEURAL NETWORK FOR HARDWARE SECURITY

A Thesis for the Degree of Doctor of Philosophy

HA NOI - 2023

MINISTRY OF EDUCATION & TRAINING MINISTRY OF NATIONAL DEFENSE

MILITARY TECHNICAL ACADEMY

DO NGOC TUAN

SIDE CHANNEL ATTACK BASED ON DISTRIBUTION OF SAMPLING CORRELATION AND MULTI-OUTPUT NEURAL NETWORK FOR HARDWARE SECURITY

A Thesis for the Degree of Doctor of Philosophy

Specialization: Electronic Engineering
Specialization code: 9 52 02 03

SUPERVISORS
1. Assoc. Prof. HOANG VAN PHUC
2. Prof. PHAM CONG KHA

HA NOI - 2023

ASSURANCE

I hereby declare that this thesis was carried out by myself under the guidance of my supervisors. The presented results and data in the thesis are reliable and have not been published anywhere in the form of books, monographs or articles. The references in the thesis are cited in accordance with the university's regulations.

Ha Noi, June 08, 2023

Author

Do Ngoc Tuan

ACKNOWLEDGEMENTS

This work would not have been possible without the support of mentors, my colleagues and friends. Specifically, I would like to thank my advisors, Assoc. Prof. Hoang Van Phuc and Prof. Pham Cong Kha for their excellent guidance and generous support throughout my Ph.D. course. I am very grateful to have their trust in my ability, and I have often benefited from their insight and advice. I would like to express my sincere thanks to Dr. Doan Van Sang for his enthusiastic support in the deep learning domain.

Besides my advisors, my sincere thanks go to my lecturers in the Faculty of Radio Electronic Engineering and Institute of System Integration, especially teachers in the Department of Microprocessors, who shared a variety of experiments for me to complete my research.

Finally but not least, my gratitude is to my family members who support my studies with strong encouragement and sympathy. Especially, my deepest love is for my parents, my wife, and my little sons, who are always my endless inspiration and motivation for me to overcome all obstacles. Without their invaluable help, this work would have never been completed.

Author

Do Ngoc Tuan

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

Abbreviation	Definition
AES	Advanced Encryption Standard
CMOS	Complementary Metal Oxide Semiconductor
CNN	Convolutional Neural Network
CPA	Correlation Power Analysis
DDLA	Differential deep learning analysis
DL	Deep learning
DLSCA	Deep learning based side-channel attack
DPA	Differential power analysis
DUT	Device under test
ELU	Exponential Linear Unit
GPIO	General Purpose Input-Output
HD	Hamming Distance
HW	Hamming Weight
MESF	Maximizing Estimated SNR First
MLP	Multilayer Perceptron
PCA	Principle component analysis
POI	Points of interest
PPA	Partial power analysis

PUF	Physical unclonable function
PGE	Partial Guessing Entropy
RELU	Rectified linear unit
RISC-V	Reduced Instruction Set Computer Five
SCA	Side-channel attack
SNR	Signal and noise ratio
SoC	System on chip
SR	Success rate
TA	Template attack

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

Symbol	Meaning
$\mathbf{x}$	x vector
$\mathbf{X}$	X matrix
p_i	The plaintext (Byte) number i
k_j	The subkey corresponding byte j
$\boldsymbol{t}_i$	The power trace number i
$t(\tau)$	The power consumption of a single sample
t_d	The data-dependent component of a power trace
t_o	The operation-dependent component
$t_{sw.noise}$	The switching noise component
$t_{el.noise}$	The electronic noise component
t_{const}	The constant component
t_{exp}	The exploitable component
t_{noise}	The noise component
σ^2	The variance
μ	The mean
Δ_F	The differential trace using function F
ρ	The correlation
r	The estimate of correlation

η	Learning rate
$\mathbf{R}_\rho$	The matrix of correlation coefficient
$\mathbb{R}^D$	The real coordinate space of dimension D
θ	The trainable parameter
Φ	The cumulative distribution function
γ	The momentum value
$\mathcal{L}_X(\theta)$	The loss function
∇	The gradient

INTRODUCTION

Motivation

In recent years, along with the rapid development of industry revolution 4.0, electronic devices have been applied to many categories of consumer electronics, including home appliances, telecommunication, transport, and important domains such as military, healthcare, and banking. To ensure data confidentiality, integrity, and validity of such devices, cryptographic algorithms are increasingly being applied, which make use of a secret key to turn conventional information (plaintext) into an unintelligible form (ciphertext).

Cryptographic algorithms can be either software implementation in microprocessors, microcontrollers, smart cards, or hardware implementation in Field Programmable Gate Array (FPGA) and Application Specific Integrated Circuit (ASIC) platforms. For some specific application processors, cryptographic functions are often available as a built-in accelerator, accessible through an Application Programming Interface (API). However, any computation is eventually performed by a part of the hardware, which leaks symptoms of its operations like power consumption, electromagnetic radiation, or temperature variations. By utilizing such leakage information, adversaries can mount the so-called Side-Channel Attacks (SCA) to reveal the secret information of the system. Since the first work in the late 1990s [2], Kocher *et al.* have raised the awareness

of the fact that "*provably secure*" cryptography is potentially vulnerable to SCA.

In general, SCAs are categorized into two main groups: profiled and non-profiled attacks [3]. Regarding profiled attacks, Template attack (TA) [4] and Deep learning (DL) based attacks [5], are considered a worst-case security risk as an attacker has full access to a copy version of the target device. The attacker is able to characterize the side-channel leakage of the target device by exploiting side-channel information collected from the reference device prior to the attack. In contrast, Differential Power Analysis (DPA) [2], Correlation Power Analysis (CPA) [6], and Differential DL Analysis (DDLA) [7] are the non-profiled SCA techniques, which perform without any prior knowledge about the expected side-channel leakage that is directly used for key extraction.

From a technical point of view, SCA can be classified into statistic-based attacks (CPA, DPA, TA, etc.) and machine learning (ML) based attacks (DL-based attacks, DDLA, etc.). In the machine learning domain, DL-based attacks have received a special attention from the research community because of their superiority over statistic-based attacks. Concretely, statistic-based attacks require pre-processing techniques on side-channel data in the case of common countermeasures applied [8–11]. It leads to a high cost and a time-consuming evaluation process. In contrast, DL-based attacks could reveal the secret key without any pre-processing techniques [12]. From the above analysis, an efficient SCA can be achieved by improving the existing statistic techniques or applying appropriate DL architectures.

The past twenty years have seen various investigations on the threats from SCA. Different countermeasures have been proposed in order to

improve hardware security against SCA. However, regardless of the type and soundness of the deployed countermeasures, the real attacks must be repeatedly performed in order to clarify their effectiveness in the early stage of the product. In addition, it is nearly impossible for a designer to predict a priori the SCA resistance of their design before its detailed implementation and subsequent in-depth analysis. Consequently, countermeasures against SCA are often developed and implemented in an additional step at the end of the design process of a device. Therefore, research on efficient SCA is crucial for designing provably secure countermeasures, validating the effectiveness of protected schemes as well as detecting potential vulnerabilities.

Research Objectives

The thesis focuses on proposing efficient SCA techniques to speed up and enhance the success rate of attacks on cryptographic devices applying different SCA-protected schemes.

The specific objectives of this research can be summarized as follows:

- To propose low complexity SCA techniques based on CPA for both unprotected and protected cryptographic devices applying masking and noise injection countermeasures.
- To break different SCA countermeasures by advanced deep learning (DL) techniques without pre-processing methods and to improve the efficiency of DL-based attacks in terms of attack time and success rate.

Research areas

- Block cipher (focused on AES algorithm), cryptography for embedded systems, and Microprocessor architecture (focused on RISC-V).
- Hardware security, non-profiled SCA using power consumption data, SCA countermeasures.
- Statistical methods, single-output deep learning, and multiple-output deep learning.

Research method

In this thesis, both the theoretical analysis and hardware-based experiments are implemented to evaluate the performance of the proposed techniques.

- The analytical methods are used to process the side-channel data, the correlation function, and the parameters of the proposed methods.
- The hardware-based experiments are conducted to collect side-channel data on the device under test (DUT), perform DL training to reveal the secret key, and compare the performance of the proposed and other previous works.

Thesis contribution

In this thesis, new SCA techniques are proposed to reduce the computation time and enhance the success rate. Concretely, the proposed techniques focus on enhancing the efficiency of the most commonly used non-profiled SCA techniques like CPA and DDLA in different evaluation conditions, such as high dimensional data, imbalanced datasets, and the

presence of SCA countermeasures. The major contributions of the thesis can be summarized as follows.

- To propose low-complexity CPA techniques (P-CPA and BP-CPA) based on the distribution of sampling correlation and the power trace biasing technique. The proposed techniques have been used to speed up the SCA attacks and enhance the success rate against different SCA protected schemes, such as masking and noise injection. The thesis also suggests countermeasures based on the disadvantages of the proposed techniques. This contribution is presented in [C1], [J1].
- To propose a dimensional reduction technique for DLSCA in a non-profiled context using the P-CPA method. Furthermore, a novel labeling technique, namely Significant Hamming Weight (SHW), is proposed to solve imbalanced dataset problems and reduce SCA data size. The efficiency of proposed methods is investigated on different DL architectures, such as Multi-layer perceptron (MLP) and Convolutional neural network (CNN). This contribution is presented in [C2, C3], [J3, J5] and [P1]
- To propose new non-profiled SCA based on multi-output and multi-loss deep neural networks (MO-DLSCA). The proposed techniques mitigate the drawbacks of DDLA attacks and enhance the performance of non-profiled DLSCA regarding the attack time and the success rate. Significantly, the attack time is from several hours to less than half an hour in some cases. The security solutions against MO-DLSCA itself are also presented. This contribution is presented in [J2, J4], and [C4].

Thesis structure

The thesis is organized into four chapters as follows:

- Chapter 1: Side-channel attacks.

  This chapter briefly introduces SCA. Specifically, the background of SCA, side-channel data, measurement setups, and the data used in the thesis are shown in this chapter. Then, the general knowledge of SCA methods, SCA countermeasures, and applications of SCA for hardware security is provided. Finally, a comprehensive review of recent research on the non-profiled SCA is presented along with some research directions that promote the contribution of this work in the subsequent chapters.

- Chapter 2: Low complexity correlation power analysis attacks.

  This chapter presents a theoretical analysis of the sampling distribution of the correlation coefficient and proposes a new POI extractor technique named partial correlation power analysis (P-CPA). By combining P-CPA and power trace biasing technique, another POI extractor called power trace biasing based P-CPA (BP-CPA) is proposed in order to improve the probability of taking the correct samples and reduce the number of needed traces. Based on the proposed POI extractors, two auto-CPA algorithms that automatically perform the POI selection, together with key recovering, are introduced. Details analysis of experimental results is also provided to clarify the efficiency of the proposed techniques in SCA security testing regarding execution time and success rate.

- Chapter 3: Dimensionality reduction and labeling methods for effi-

cient deep learning based non-profiled SCA.

In this chapter, non-profiled DLSCA techniques are investigated in different security testing scenarios. Regarding the high dimensional data issue, the P-CPA technique is applied to select the most informative sample points and reduce the size of input data significantly. Related to the imbalanced dataset problem, a new labeling technique called SHW based on Hamming weight is proposed. In this chapter, various DLSCA attacks are performed, which clarify the efficiency of SCA evaluation using SHW in comparison with to 9-HW and LSB labeling techniques on unprotected and protected datasets. This chapter also provides the attack results on cryptographic devices that applied noise injection and de-synchronized countermeasures.

- Chapter 4: Multi-output deep learning based non-profiled SCA.

  This chapter proposes two novel deep learning architectures based on multi-output classification and multi-output regression. Simultaneously, the details of reconstructed dataset methods corresponding to the proposed models are also provided. The experimental results are presented and analyzed to demonstrate the efficiency of proposed models for SCA on secure platforms that apply different countermeasures. Finally, the main drawback of the proposed techniques as well as the resistance methods are discussed.

Chapter 1
SIDE-CHANNEL ATTACKS

1.1. Introduction

1.1.1. Attacks on cryptographic devices

In recent years, several kinds of attacks on cryptographic devices have been launched. The goal of all these attacks is to reveal the secret keys of cryptographic devices. However, the techniques that are used to achieve such goal are manifold. Attacks on cryptographic devices differ significantly in terms of cost, time, equipment, and expertise needed [1].

In general, the attacks fall into two large groups, as depicted in Fig. 1.1. This classification depends on whether an adversary observes some parameters of the implementation or influences its execution.

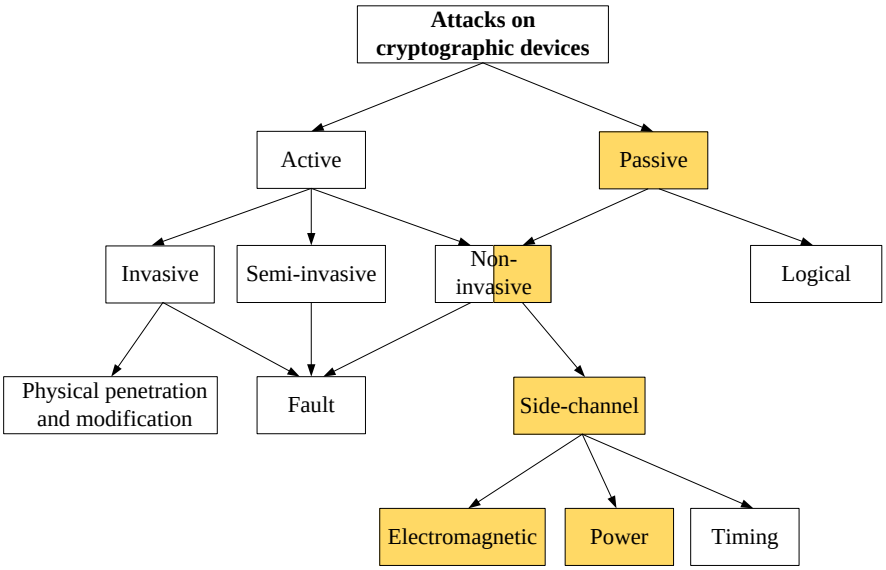

Figure 1.1: Classification of attacks on cryptographic devices [1]. The yellow parts indicate the attacks that are covered in this thesis.

The first group is *active attacks*, which exploit results of influencing an implementation, or manipulating it, such that it behaves abnormally. They comprise fault attacks where errors are introduced into the execution of a cryptographic algorithm or a protocol. Other active attacks do not directly exploit the properties of a cryptographic algorithm or a protocol, such as dumping the device's memory that contains the secret key.

The second group is *passive attacks*. This group exploits the results of observing an implementation while it works (largely) as intended. The adversary can feed inputs but does not interfere with the execution of an algorithm or a protocol. In particular, passive non-invasive attacks have received a lot of attention from the hardware research community. These attacks are usually referred to as *side-channel attacks* (SCA), where physical leakage of a device during an algorithm execution is observed. Other passive attacks are based on higher-level logic, where some parameters of a protocol execution, for instance, error messages, are observed.

1.1.2. Side-channel attack

The basic idea of SCA is to reveal the secret key of a cryptographic device by analyzing its side-channel data. As shown in Fig. 1.1, the leakage data that are usually exploited to attack are timing [13], power consumption [2], and electromagnetic (EM) radiation [14], because they are relatively easy to be acquired from various electronic devices. Other more specific side-channel data types were exploited in SCA, for instance, acoustic [15], optical [16], and thermal [17], but they are rarely used in practice for implementing SCA.

In practice, power consumption and electromagnetic data are linked since moving electric charges in the device is a source of electromagnetic radiation. Consequently, the power consumption of a cryptographic device can be measured directly by inserting a power measurement circuit or indirectly by an EM probe [18]. This thesis, therefore, focuses on SCA using power consumption data as indicated by the parts highlighted in yellow in Fig. 1.1. The concept "side-channel data" will refer to power consumption data for the rest of the thesis. It is worth noting that the attacks that perform successfully on power consumption are also applied successfully to EM data used in this thesis.

1.1.3. Classification of SCA

Basically, SCA attacks can be classified into two approaches: profiled and non-profiled attacks. *Profiled attacks* use a reference device, which is identical (or very close) to the target device, to build a database stocking power consumption information dedicated to a type of device [3], as depicted in Fig.1.2. This class of attack was initially proposed in [19] and then developed under the well-known "Template attack" [4]. In particular, profiled attacks take place in two stages: the profiling stage and the key extraction stage. In the *profiling stage*, a large number of power traces recorded from the reference device are used to build a template for each hypothesis key based on multivariate-Gaussian distribution. Then, the *key extraction stage* uses a small number of power traces (very small compared to the profiling stage) collected from the target device to find the correct key based on the template, which has been pre-built in the profiling stage. Such detection method is based on the maximum likelihood metric.

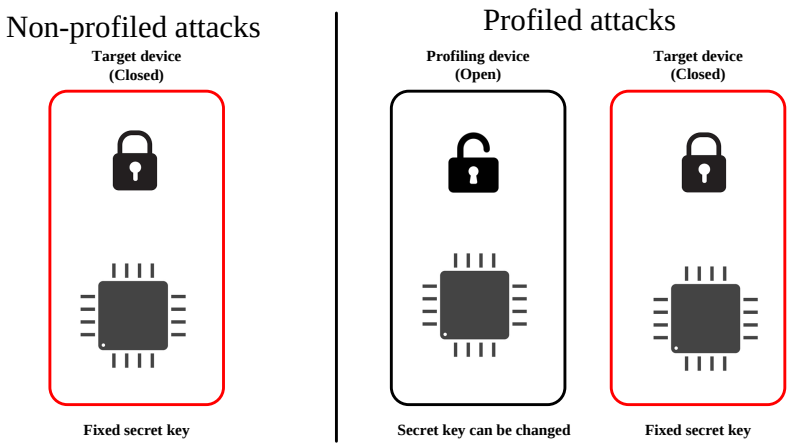

Figure 1.2: Non-profiled and profiled attack categories.

Even though profiled attacks are the most powerful of the SCAs, they have also been considered the worst-case security analysis, and the condition of this method is sometimes difficult to satisfy in practice, especially when the targets are flexible and highly customizable, such as FPGA-based designs or open-source architectures. Another case is closed products like smart cards running banking applications. The attacker does not have control of the keys and is usually limited by a transaction counter. In such cases, profiled attacks can not be performed. However, the secret device is still threatened by a method called a non-profiled SCA attack.

Non-profiled attacks are based on the relationship between the power consumption model and real measurements. In this case, the leakage information is exploited at the same point in time over different operations of devices. By computing the correlation of the ground-truth models and the measurements recorded from the target device, the non-profiled SCA can recover the secret key. Therefore, the non-profiled attacks do not require any reference device. On the one hand, it is suitable to ease the evaluation and low-cost testing process (no reference device needed).

On the other hand, non-profiled SCA can be considered a high-security scenario compared to profiled SCA since the attackers do not have any extra information from the target device except input/output data. For these reasons, this thesis place importance on investigating non-profiled SCA techniques with respect to both statistic-based and DL-based.

To mount an SCA attack, side-channel data is of prime importance. In the next section, a basic knowledge of side-channel data and measurement setup are briefly introduced.

1.2. Side-channel data and measurement setup

Digital circuits consume power whenever they perform computations. The current from a power supply is drawn and then dissipated as heat. The power consumption of digital circuits is a important topic. It determines whether a device needs to be cooled or not and which kind of supply is necessary. More importantly, in the case of cryptographic devices, it determines whether a device can be attacked or not. Therefore, power consumption is usually exploited for evaluating the safety of a designed circuit.

1.2.1. Power consumption of CMOS circuit

It is commonly known that digital circuits (ASIC or FPGA) are built based on logic cells. The most commonly used logic cell technology is Complementary meta-oxide semiconductor (CMOS), which is based on a complementary pull-up and pull-down network. Fig 1.3. illustrates a CMOS inverter cell, P1 is conducting, and N1 is insulating if the input a is set to GND. In contrast, P1 is insulating, and N1 is conducting if the input a is set to V_{DD}. The logic cells in the circuit process the input signal

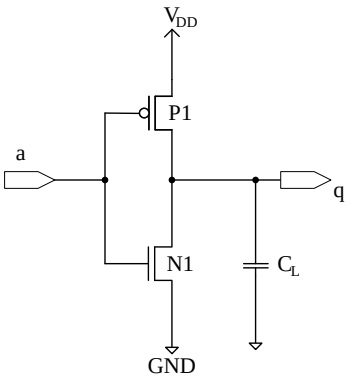

Figure 1.3: Lumped-C model of a CMOS inverter.

and draw the current from the power supply. Therefore, the total power consumption of a CMOS circuit is the sum of the power consumption of logic cells making up the circuit. Let $i_{DD}(t)$ and $p_{cir}(t)$ denote the instantaneous current and power consumption of the circuit, respectively. The average power consumption P_{cir} over time T is calculated as follows:

$$P_{\text{cir}} = \frac{1}{T}\int_0^T p_{\text{cir}}\,(t)\,dt = \frac{V_{\text{DD}}}{T}\int_0^T i_{\text{DD}}\,(t)\,dt \tag{1.1}$$

Digital systems usually draw both *dynamic* and *static* power [20]. Dynamic power is used for charging the capacitance as signals change between 0 and 1. On the other hand, static power is used even when signals do not change and the system is idle. In the SCA domain, dynamic power consumption is considered to exploit in most cases.

At a fixed moment of time, an output state of a logic cell normally falls into one of four transitions, such as $1 \rightarrow 1$, $0 \rightarrow 0$, $1 \rightarrow 0$, and $0 \rightarrow 1$. In the two cases $1 \rightarrow 1$, $0 \rightarrow 0$, there are no switching activities. Therefore, the power consumption contains only static power, whereas switching activities are detected in the case of $1 \rightarrow 0$ and $0 \rightarrow 1$. As a result, dynamic power consumption occurs. In other words, the dynamic

power consumption depends on the data processed by the CMOS circuit.

There are two reasons for a CMOS cell's dynamic power consumption. The first one is that the load capacitance of the cell needs to be charged. Let P_{chrg} be the power consumption caused by charging current. The average charging power P_{chrg} that is consumed by a cell during the time T can be calculated by the formula 1.2.

$$P_{chrg} = \frac{1}{T} \int_0^T p_{chrg}(t)\, dt = \lambda \times f \times C_L \times V_{\mathrm{DD}}^2 \tag{1.2}$$

where C_L and f denote the output capacitance and the clock frequency, respectively. λ is the so-called activity factor of the cell, which corresponds to the average number of $0 \rightarrow 1$ transitions occurred each clock cycle on the output of a logic cell.

The second component of dynamic power consumption is caused by the temporary short circuit that occurs in a logic cell during the switching of the output, denoted as P_{sc}. The average of P_{sc} during the time T can be calculated as follows:

$$P_{sc} = \frac{1}{T} \int_0^T p_{sc}(t)\, dt = \lambda \times f \times I_{peak} \times t_{sc} \tag{1.3}$$

where t_{sc} denotes the time for short circuit exits, I_{peak} denotes the value of the current peak that is caused by a short circuit during the switching.

Unlike the charging current, a short circuit happens in both cases of $1 \rightarrow 0$ and $0 \rightarrow 1$. Therefore, we can conclude that the power consumption of $0 \rightarrow 1$ transition is greater than $1 \rightarrow 0$ one.

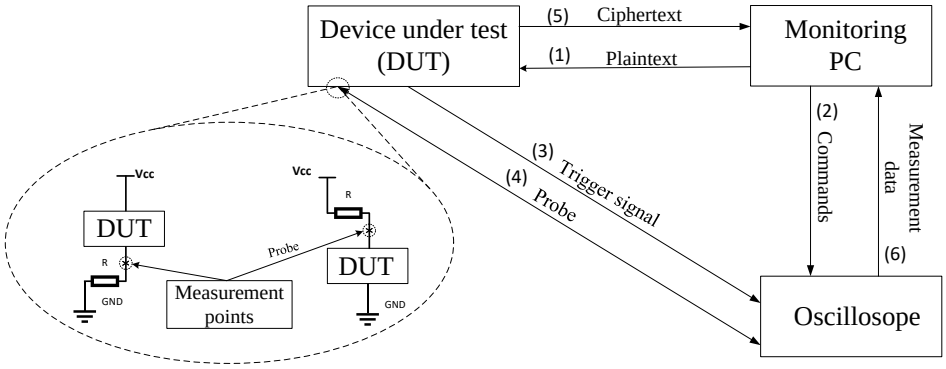

Figure 1.4: Power consumption measurement setups.

1.2.2. Measurement setups

For power analysis attacks, it is necessary to measure the power consumption of a cryptographic device while it executes cryptographic algorithms. This subsection briefly introduces the power measurement procedure, including setups and the needed components.

Fig 1.4 depicts a typical measurement setup that requires some components, such as a personal monitoring computer (PC), the device under test (DUT), and a digital oscilloscope.

Device under test: In practice, DUT is a cryptographic device that usually provides an interface to communicate with a monitoring PC and produces the trigger signal for a digital oscilloscope. The DUT can be an embedded device performing encrypt/decrypt data or hardware implementing the cryptographic algorithm on FPGA or ASIC. In order to measure the power consumption, a very small register (typical resistance values are 1 Ωto 50 Ω) is inserted into the GND or V_{DD} line of DUT. The measurement points are then determined as depicted in Fig.1.4.

Digital oscilloscope: A digital oscilloscope is employed to measure the power side-channel data when the target operates the cryptographic algorithm. The oscilloscope is configured with two analog channels. Two

passive probes are used to gather power traces from the target device in the experimental workplace. One probe is applied to acquire the analog signal from the core V_{DD} node of DUT. The second one detects the trigger signal provided by the target through a GPIO pin. The oscilloscope is remotely controlled by a monitoring PC.

Monitoring PC: The monitoring PC is utilized to operate the whole auto-measuring system. It communicates with the oscilloscope through the LAN port and with the target device through a USB port.

The numbers in Fig 1.4 illustrate in which sequence the component interact with each other when a power trace is recorded. Firstly, the PC sends a plaintext to the target device (1) and commands the oscilloscope to capture the power traces (2). In this setup, the digital oscilloscope is configured in TRIGGER MODE (measure data when trigger occurred). When the target SoC executes encryption/decryption, a trigger signal is emitted (3). Subsequently, the oscilloscope records the measurement data when the trigger is on (4). After completing encryption, the monitoring PC receives the ciphertext/plaintext corresponding to the input (5). Finally, the measured data from the oscilloscope are transferred to PC (6).

Power side-channel data can be seen as a vector of values proportioning to the device's drawn current. These data are recorded by a digital oscilloscope and so-called "power trace." For the rest of the thesis, the concept "power trace" will be used to describe the power side-channel data.

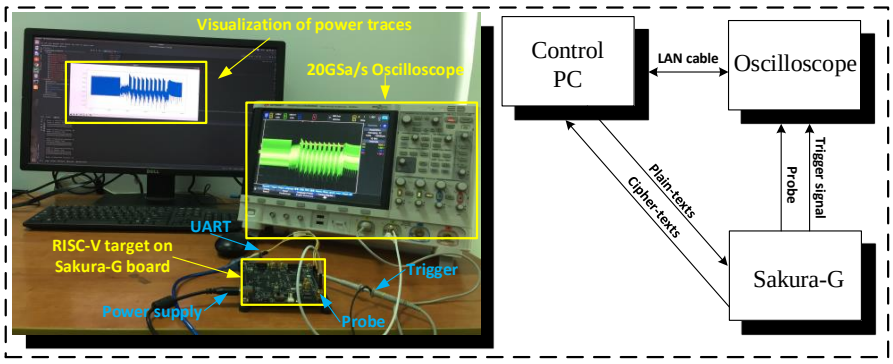

Figure 1.5: Test platform: RISC-V power traces acquisition on Sakura-G board.

1.2.3. The data used in thesis

The datasets used in this thesis are divided into unprotected and protected ones. Based on the wide range of applications and the number of related publications published in the SCA domain [7, 21–28], the AES-128 algorithm is selected to perform the power consumption data acquisition. Regarding unprotected AES-128, the side-channel data were recorded from ChipWhisperer (CW) [29] and Sakura-G board [30], which are popular platforms in the SCA research community. To investigate the SCA attacks on protected AES algorithm, the thesis uses the public ASCAD [12] and CHES2018-CTF [31] datasets. Simultaneously, two other simulated datasets are created to investigate different SCA countermeasures.

Unprotected dataset

-RISC-V MCU on Sakura-G: An experimental system is set up to collect power traces automatically as described in Section 1.2.2. As depicted in Fig.1.5, the experimental system consists of a Keysight DSOX6004A oscilloscope, a monitoring PC, and a Sakura-G FPGA board, in which Sakura-G FPGA board is the test platform. It is worth noting that the

target device is a 32-bit Murax RISC-V MCU operating at 48 MHz. This MCU is implemented on the Sakura-G board and then programmed an AES-128 software in C language. For all encryption processes, the secret key is fixed, and the plaintexts are chosen randomly. To record power consumption of the target, one probe of the DSOX6004A oscilloscope is applied to acquire the analog signal from the core V_{DD} node of the Spartan-6 chip at 125 MSa/s sample rate. The second one detects the trigger signal provided by the RISC-V target through pin GPIO2. A monitoring PC remotely controls the oscilloscope through Python software and a VISA COM library. The power traces and ciphertexts corresponding to the plaintexts are saved to NumPy files for later analysis. As a result, 10,000 power traces of the Sakura-G board are collected. Each power trace contains 9,919 samples.

- *ChipWhisperer board:* Similar to the RISC-V target, we set up an automatic system using CW to acquire power traces. CW is an all-in-one platform, including a 10-bit 105MSa/s ADC chip, a Spartan-6 FPGA for controlling, and an Atmel Xmega chip that serves as the target device. CW is controlled by a personal computer using Python software, which repeatedly sends the plaintexts to the CW board and receives the side-channel data along with corresponding ciphertexts via the USB port. All capturing processes are automatically done by Spartan-6 FPGA and an ADC chip. Accordingly, 10,000 power traces have been collected from the CW board. Each power trace contains 10,000 samples, which record the power consumption of approximately two first rounds of the AES-128 algorithm as depicted in Fig. 1.6.

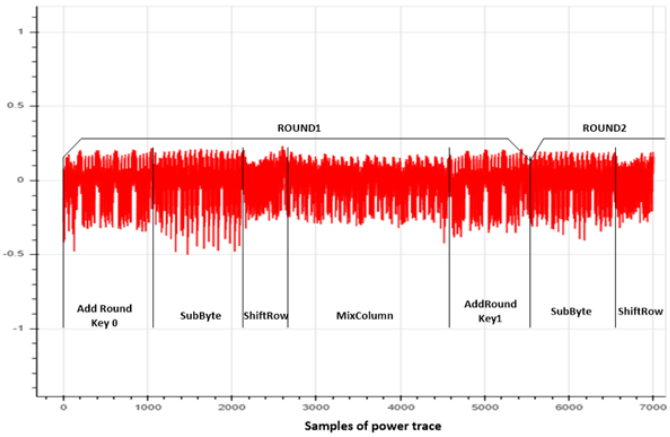

Figure 1.6: An example of CW power trace divided into different parts corresponding to different functions of AES-128 encryption.

Protected dataset

- *Noise generation countermeasure*: To simulate noise generation countermeasure in this case study, different levels of Gaussian noise were added to each sample of unprotected power traces (CW or Sakura-G) as follows:

$$t_{noise}\left(i,m\right) = t(i,m) + \sigma \times randn\left(1,m\right) + mean \tag{1.4}$$

where *randn* returns a vector of numbers drawn from the standard normal distribution, σ and *mean* are the standard deviation and mean value of Gaussian noise ($mean$= 0), respectively.

The values of σ depend on the standard deviation of all samples (σ_{pt}) in a set of power traces. Regarding RISC-V data, the σ_{pt} is in range [0.001;0.08]. Therefore, the range of σ [0.004;0.01] is chosen. Similarly, different ranges of σ are selected for other data as shown in Table 1.1.

It is noted that the selected values are also determined based on the results of the actual attacks. It means that the typical methods, such as CPA and DDLA, can not attack the noisy data successfully, or the performance of these attacks (e.g., success rate) decreases significantly.

Table 1.1: The values of standard deviation of Gaussian noise (σ) added on different data.

Data	σ_{pt} [min,max]	σ [min,max]	Step
RISC-V	[0.001,0.08]	[0.004,0.01]	0.002
ASCAD	[0.5,6.4]	[0.5,1.5]	0.5
CW	[0.003,0.06]	[0.025,0.075]	0.025

- *De-synchronized countermeasure*: Regarding the de-synchronized power traces countermeasure, these protected data can be achieved by using the random delay technique on software AES implementation, as illustrated [32]. To simulate this countermeasure, each power trace recorded from CW or Sakura-G is shifted randomly to a constant value as described in Algorithm 1.

Algorithm 1 Creating de-synchronized power traces

Input: N power traces T, shift_value = ψ, number of samples = β
Output: Shifted power traces: $\mathrm{T}_{\mathrm{shifted}}$

1: **for** $i \in 1 : N$ **do**
2: sh= randi([1:ψ])
3: $\mathrm{T}_{\mathrm{shift}} = \mathrm{T}(i, sh: sh{+}\beta)$;
4: $\mathrm{T}_{\mathrm{shifted}}(i,:) = \mathrm{T}_{\mathrm{shift}}$;
5: sh= 0;
6: **end for**
7:

- *Boolean masking countermeasure:*

ASCAD (ANSSI SCA Database): This is a set of databases that aims at providing a benchmarking reference for the SCA community: the purpose is to have something similar to the MNIST database that the Machine Learning community has been using for quite a while now to evaluate the classification algorithm's performance. This dataset provides electromagnetic radiation data of an 8-bit ATMega8515 board with the first-order protected software AES implementation. This thesis uses the first version of the ASCAD dataset, which is captured at a sampling rate

of 2GSa/s. The length of these power traces is 100,000 samples. This dataset consists of two sets of traces: a profiling set of 50,000 traces to train DL networks and an attack set of 10,000 traces to test the efficiency of the trained models in a profiled context. It is worth noting that 700 samples corresponding to the output of the 3^{rd} S-box processing during the first round are taken to construct the ASCAD.h5 file. Additional bytes can be generated automatically using the provided Python script. The data and scripts are available on the ASCAD GitHub repository[1].

CHES-CTF 2018: This dataset refers to the CHES Capture-the-flag (CTF) AES-128 trace set, released in 2018 for the conference on Cryptographic Hardware and Embedded Systems (CHES). This database contains 45,000 power traces that record the masked AES-128 encryption on a 32-bit STM microcontroller. This thesis considers a pre-processed version of the dataset, which includes a fixed key for all power traces, and each trace consists of 2200 samples. The pre-processed dataset is available at *http://aisylabdatasets.ewi.tudelft.nl/.*

1.3. Non-profiled SCA methods

1.3.1. Attack strategy

There exists a general attack strategy that is applied for all non-profiled attacks. Most of them follow the *"divide-and-conquer"* strategy. It independently recovers the individual chunks of the secret key. This is possible because small chunks of the secret key would be processed independently at some point in a cryptographic algorithm (AES, DES, etc.). The goal of non-profiled attacks is to reveal the secret key of the secure device based on a large number of power traces that have been

[1] https://github.com/ANSSI-FR/ASCAD/tree/master/ATMEGA_AES_v1

recorded while the devices encrypt or decrypt different data blocks. It can be described as following steps:

Step 1: Selecting an intermediate results of cryptographic algorithm.

Intermediate results of the cryptographic algorithm are the results of the function $f(d, k)$, where it depends on both known non-constant data d and a chunk (a byte) of the secret key k. In most attack scenarios, d is ether the plaintext of the ciphertext.

Step 2: Measuring power consumption.

Let us assumes that the target device processes N consecutive functions $f_n^{d_i}$ while simulated with random input data d_i. During each run, while the functions are processed, the power consumption $\boldsymbol{t}_i$ is recorded, and the corresponding output is gotten back. The processing in the device can be represented by the matrixes

$$\boldsymbol{F} = \begin{pmatrix} f_1^{d_1} & f_2^{d_1} & \dots & \boldsymbol{f}_n & \dots & f_N^{d_1} \\ f_1^{d_2} & f_2^{d_2} & \dots & & \dots & f_N^{d_2} \\ f_1^{d_3} & f_2^{d_3} & \dots & & \dots & f_N^{d_3} \\ f_1^{d_4} & f_2^{d_4} & \dots & & \dots & f_N^{d_4} \\ \vdots & \vdots & \dots & & \vdots & \vdots \\ f_1^{d_D} & f_2^{d_D} & \dots & & \dots & f_N^{d_D} \end{pmatrix}; \tag{1.5}$$

$$\boldsymbol{T} = \begin{pmatrix} t_{1,1} & t_{1,2} & \dots & \boldsymbol{t}_n & \dots & t_{1,S} \\ t_{2,1} & t_{2,2} & \dots & & \dots & t_{2,S} \\ \vdots & \vdots & \dots & & \dots & \vdots \\ \boldsymbol{t}_i & & & & & \\ \vdots & \vdots & \dots & & \dots & \vdots \\ t_{D,1} & t_{D,2} & \dots & & \dots & t_{D,S} \end{pmatrix}. \tag{1.6}$$

Each column vector $\boldsymbol{f}_n$ of $\boldsymbol{F}$ executes the same functions. The matrix

$\boldsymbol{T}$ contains the corresponding recorded trace $\boldsymbol{t}_i$ with respect to the D runs. Each trace consists of $S > N$ samples, such that each function f_n is described by at least one sample.

Step 3: Calculating hypothetical intermediate value.

Hypothetical intermediate values are the intermediate values that are calculated for every possible choice of k. These possible choices are denoted by vector $\boldsymbol{k} = (k_1, \ldots, k_K)$, where K denotes the total number of possible choices for k. Given the data vector $\boldsymbol{d}$ and the key hypotheses $\boldsymbol{k}$, an attacker can easily calculate hypothetical intermediate values $f\,(d, k)$ for all D encryption runs and for all K key hypotheses. This calculation (1.7) results in a matrix $\boldsymbol{V}$ of size $D \times K$.

$$\boldsymbol{V} = v_{i,j} = f\,(d_i, k_j) \quad i = 1, \ldots, D \quad j = 1, \ldots, K \tag{1.7}$$

Step 4: Calculating hypothetical power consumption value.

Hypothetical power consumption $h_{i,j}$ are the values, which are calculated by a predict function $f_{prd}(.)$ with respect to hypothetical intermediate value as follows:

$$h_{i,j} = f_{prd}\,(v_{i,j}) \tag{1.8}$$

The quality of predict function strongly depends on the attacker's knowledge of the analyzed device. The better the attacker's prediction function matches the device's actual power consumption characteristic, the more effective the attack becomes. The prediction function is also known as the power consumption model, which is discussed in the next part.

Step 5: Determining the correct key.

Regardless of whether the attacks are statistic or DL based, the goal of non-profiled attacks can be achieved by comparing the hypothetical power consumption values of each key hypothesis with the recorded

traces at every position.

In the case of CPA, for example, the values of correlation ρ_k will be used to determine the correct key k_{cr}. The attacks rest on the following fact: if the power traces and $f_{prd}(.)$ are well-chosen, then $\boldsymbol{V}$ is highly correlated to power traces, and thus the coefficient $\rho_{k_{cr}}$ corresponding to the correct key guess must be greater than every coefficient ρ_{k_j} where $k_j \neq k_{cr}$. Especially, the coefficient $\rho_{k_{cr}}$ achieves the highest value at the sample t_{ct}. This sample contains the power consumption values that depend on the intermediate values v_{ck}.

In the case of DLSCA, the model tries to approximate a function $Net(\boldsymbol{t}_i, h_{i,j})$, which is used to predict the value of $h_{i,j}$ in correspondence with key hypothesis k_j from unseen power traces $\boldsymbol{t}_i$. Then the training metrics such as accuracy or loss are used instead of ρ_k.

Attack point selection

To reveal the correct key from a set of key hypotheses, one usually assumes a very low correlation between correct and incorrect guesses. This assumption which depends on the structure of f is fairly realistic if f is highly non-linear [33]. The Sbox function of block cipher based on substitution–permutation network is an example. A one-bit difference at an Sbox input leads to a difference of several bits at the output. Consequently, even if a key hypothesis is only wrong in one bit, the output of the Sbox is different in several bits. When attacking the output of the Sbox, the correlation for all wrong key guesses is therefore significantly smaller then the correlation for the correct one [20]. In contrast, for the linear part correlated with the plaintext, the linear operations tend to give a poor separation between correct and incorrect keys. As a result,

it tends to be giving the "*ghost peaks*" [6]. Various works have demonstrated the efficiency of non-linear output (*e.g* Sbox output) on different algorithms DES [6, 34], AES [6, 7, 26, 34–36], GIFT64, PRESENT or PICCOLO algorithm [37].

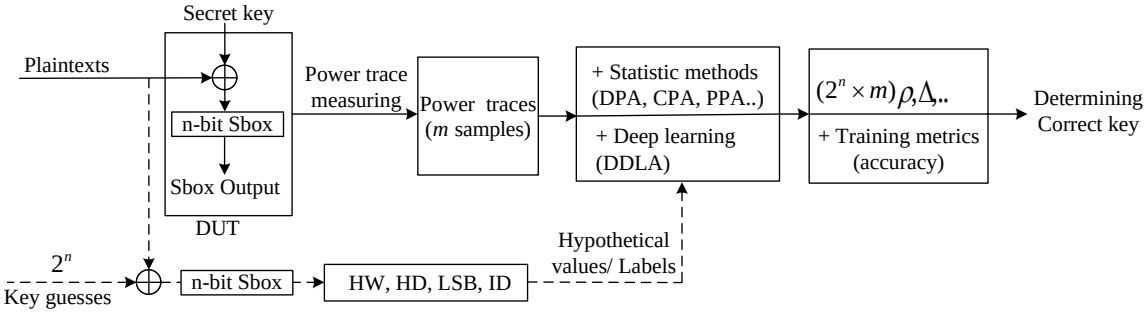

Figure 1.7: Non-profiled SCA procedure on Sbox output of block cipher

Fig 1.7 illustrates the SCA attacks on Sbox output. It is noted that the attackers usually chose the *first-round* Sbox, which is directly related to the original secret key. For example, the secret key of the *first round* of AES algorithm [7,12,26] can be taken by SCA attacks. It is also true for other block ciphers like SM4 or DES [38]. Based on the efficiency of Sbox output in SCA attacks, the thesis considers the first round Sbox output as the main attack position for all proposals. In addition, our approach is independent of the chosen block cipher algorithm.

1.3.2. Power consumption models

In reality, most embedded devices set the bus lines to a 'precharge' state which is halfway between a high and a low state before setting the bus lines to the final state. Thus, on every clock cycle, we can measure the current flow in the VCC line, and the value of this current would be expected to be linearly related to the number of lines going from the precharge state to the high state: the higher the current peak, the more lines switched high [39]. In other words, the instantaneous power usage

is highly dependent on the processed intermediate data.

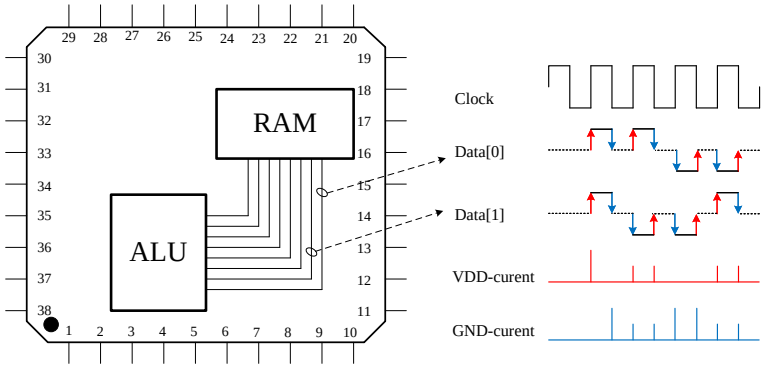

Figure 1.8: Example of power consumption trace of a pre-charged bus

In the non-profiled SCA context, the most important thing determining the success of an attack is power consumption models. This is due to the fact that attackers usually have only very limited knowledge about the implementation of the target device [18]. Therefore, it is often necessary to use a power model to simulate the power consumption related to processed data. This part briefly introduces some generic power consumption models, which are commonly used for power analysis attacks.

- Hamming distance: Hamming Distance (HD) model is usually used to map the transitions that occur at the output of cells of a netlist to power consumption values. Therefore, the HD model is appropriate to describe the power consumption of buses and registers on hardware implementation. However, the attacker must determine the consecutive data values v_0 and v_1 that are processed by these components of a circuit. The power measurement that is recorded from the process of changing the bus or register value is proportional to the number of bit transitions, which is calculated as follows:

$$HD\left(v_0, v_1\right) = HW\left(v_0 \oplus v_1\right) \tag{1.9}$$

where HW denotes Hamming Weight (HW) of a value, which corre-

sponds to the number of bits that are set to one. It is worth noting that if the attacker has no information about the netlist at all or the consecutive data values for some known part of the netlist, the HD model can not be applied [18]. In this case, another power model based on HW is selected.

- Hamming weight: The HW model is simpler than the HD model and is used in the case that only one data value transferred over a bus is known. The attacker assumes that power consumption is proportional to the number of bits that are set in the processed data value [18, 39]. As illustrated in Fig. 1.8, the HW model is useful if a pre-charged bus is used. It means that the HW model is particularly well applicable to software implementations and can be calculated as below:

$$h_{i,j} = HW\left(f\left(d_i, k_j\right)\right) \tag{1.10}$$

For example, the HW model of ASCAD data on the third Sbox can be expressed as follows:

$$h_{p_i^3,k} = HW\left(Sbox\left[p_i^3 \oplus k_3\right]\right) \tag{1.11}$$

where p_i^3 and k_3 are the third byte of plaintext number i and the key, respectively. Similarly, the HW model of RISC-V or CW data can be calculated straightforwardly.

- Least significant bit/Most significant bit: The power models presented above are commonly used in traditional statistical SCA techniques. In DLSCA attacks, power models are usually used to label the training data. The Least Significant Bit/Most Significant Bit (LSB/MSB) is a popular model used in the non-profiled DLSCA [7, 21, 40]. By taking LSB/MSB as a power model, we can calculate the label

$L_{j,i}$ of each power trace according to the following formula:

$$L_{i,j} = LSB/MSB(f\left(d_i, k_j\right)). \tag{1.12}$$

- Indentity: The last model is also the simplest model. Identity exploits directly the output values of intermediate calculus g for simulating the power consumption, as shown in Eqs. (1.13). This power model is commonly used in profiled attacks [12, 41, 42].

$$L_{i,j} = f\left(d_i, k_j\right) \tag{1.13}$$

1.3.3. Attack methods

This part describes the most common non-profiled SCA techniques, such as DPA, PPA, CPA, and DDLA.

Differential power analysis (DPA)

Differential power analysis (DPA) was introduced by Kocher et al. [2]. This analysis is based on the fact that the power consumption to manipulate one bit to 1 is different from the power consumption to manipulate it to 0. To test different key hypothesis k_j, DPA use D plaintexts (or cipher text) d_i and a boolean function $F(d_i, b, k_j)$. This boolean function computes the value of an examined bit b. For example, a bit of Sbox output $h_{i,j} = LSB(Sbox(d_i \oplus k_j))$. DPA computes a differential trace $\Delta_F\left(b\right)$ as the difference between the average of the traces for which $F(d_i, b, k_j)$ is 1 and the average of the traces for which $F(d_i, b, k_j)$ is 0. According to [2], $\Delta_F\left(b\right)$ is calculated as follows:

$$\Delta_F\left(b\right) = \frac{\sum_{i=1}^{D} F\left(d_i, b, k_j\right) \boldsymbol{t}_i}{\sum_{i=1}^{D} F\left(d_i, b, k_j\right)} - \frac{\sum_{i=1}^{D} \left(1 - F\left(d_i, b, k_j\right)\right) \boldsymbol{t}_i}{\sum_{i=1}^{D} \left(1 - F\left(d_i, b, k_j\right)\right)} \tag{1.14}$$

If the bits calculated during the cryptographic algorithm are statistically uniformly distributed and if the number of power traces is suffi-

cient, $\Delta_F(b)$ tends to become 0 for the wrong hypothesis key, whereas $\Delta_F(b) \neq 0$ for the correct key k_{cr}. Especially at the instant t_{ct} where the bit b is handled, this is the DPA peak.

Partitioning power analysis (PPA)

In the previous attack, only one bit was used to estimate the power consumption. To enhance the original DPA, some authors have introduced *m-bit* DPA attacks which mean that m bits are used instead of only one-bit [43–45]. In order to generalize the multi-bit DPA methods, Thanh-Ha Le et al. have proposed the Partitioning Power Analysis (PPA) method based on the Hamming distance [34].

They consider m-bit set $\beta = b_1 b_2 \dots b_m$ and divide D power traces $\boldsymbol{t}_i$ into $(m+1)$ partitions (classes) $G_0, G_1, \dots, G_m$.

$$G_c = \{\boldsymbol{t}_{i;i=1\dots D} | HD(d_i, \beta, k_j) = c\} \tag{1.15}$$

where $HD(d_i, \beta, k_j)$ denotes the Hamming distance between a previous state and the actual state of β, corresponding to the plaintext d_i and the hypothesis key k_j. The decision signal of PPA is given as follows:

$$\sum\nolimits_H(\beta) = \sum_{j=0}^{m} \gamma_c \frac{\sum_{G_c} \boldsymbol{t}_i}{N_c} \tag{1.16}$$

where γ_c denotes the chosen weights, N_c denotes the cardinality of each class G_c.

Correlation Power Analysis (CPA)

Correlation power analysis (CPA) was proposed by Brier *et al.* in [6], and can be considered a special form of the PPA attack [34]. CPA exploits the correlation between the real power consumption and the power consumption model Hamming weight [46,47] or Hamming distance [6] of manipulated data.

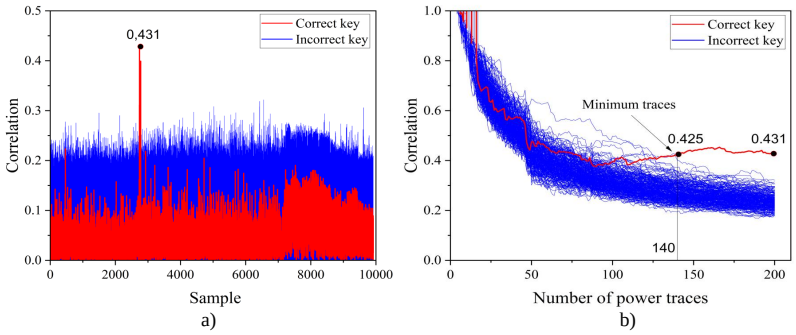

Figure 1.9: Attack results of CPA and online CPA.

In the CPA attack, the Pearson correlation coefficient is the common measure to determine the linear relationship between two variables. The definition of the Pearson correlation coefficient r is shown in equation (1.17). r estimates the correlation ρ between two variables based on D power traces. $\overline{h}_j$ and $\overline{t}_s$ are the average values of the power consumption model and real power consumption at the instant t_s $(1 \leq s \leq S)$, respectively.

$$r_{j,s} = \frac{\sum_{i=1}^{D}(h_{i,j} - \overline{h}_j)(t_{i,s} - \overline{t}_s)}{\sqrt{\sum_{j=1}^{D}(h_{i,j} - \overline{h}_j)^2 \sum_{j=1}^{D}(t_{i,s} - \overline{t}_s)^2}} \tag{1.17}$$

The Pearson correlation between the power consumption model and the real power consumption is calculated for every value of k and t_s. It results in the matrix $\boldsymbol{R} = r_{1...K,1...S}$ of correlation coefficients. There is an alternative form of the correlation equation for online calculations, and it allows us to add one trace per time without re-summing all of the past data. This form is presented in (1.18).

$$r_{j,s} = \frac{D\sum_{i=1}^{D} h_{i,j}t_{i,s} - \sum_{i=1}^{D} h_{i,j}\sum_{i=1}^{D} t_{i,s}}{\sqrt{\left(\left(\sum_{i=1}^{D} h_{i,j}\right)^2 - D\sum_{i=1}^{D} h_{i,j}^2\right)\left(\left(\sum_{i=1}^{D} t_{i,s}\right)^2 - D\sum_{i=1}^{D} t_{i,s}^2\right)}} \tag{1.18}$$

Fig. 1.9 illustrates the attack results using the conventional CPA and the online calculation CPA. In addition to performing attacks without re-summing all of the past data, online CPA allows the attacker determines the minimum power traces needed for a successful attack, as depicted in Fig. 1.9.b. In contrast, conventional CPA estimates the correlation of all samples over a given set of power traces. As illustrated in Fig. 1.9.a, the correlation values of approximately 10,000 samples were calculated over 200 power traces. As expected, a clear peak was detected between the correlation of correct and incorrect keys. It is noted that the peak value has exactly the same value as that of online CPA at 200th sample (peak value: 0.431).

Deep learning based non-profiled SCA

a) Deep learning

The main objective of deep learning is to classify some data $\boldsymbol{x} \in \mathbb{R}^S$ based on their labels $z(\boldsymbol{x}) \in \boldsymbol{Z}$, where S is the dimension of the data to classify, and $\boldsymbol{Z}$ is the set of classification labels. The goal of DL is to produce a function $Net : \mathbb{R}^S \rightarrow \mathbb{R}^{|\boldsymbol{Z}|}$ which takes as input data to classify $\boldsymbol{x} \in \mathbb{R}^S$, and outputs a score vector $\boldsymbol{y} = Net(\boldsymbol{x}) \in \mathbb{R}^{|Z|}$. In other words, the final results are the score vector $\boldsymbol{y}$ based on $Net(\boldsymbol{x})$ and updating the trainable parameters θ. Typically, prior to performing DL, the values of trainable parameters θ are randomly chosen from a normal distribution, like the Xavier scheme. Two main operations, forward and backward propagation, are performed to obtain the expected results.

Forward propagation: Each neuron in hidden and output layers

takes as input a vector $\boldsymbol{x}$ and outputs a weighted sum as formula (1.19).

$$W_{sum} = b + \sum_{i=1}^{S} w_i x_i \tag{1.19}$$

where w_i are called the weights and b the bias of a neuron, the calculated weighted sum W_{sum} is then passed to the activation function, which adds non-linearity to the network. There are some commonly used and popular activation functions such as Sigmoid, Hyperbolic tangent (Tanh), ReLU, ELU, and Softmax. To this end, the trainable parameters θ have not been updated yet. Therefore, the achieved results might not be optimum.

Backward propagation: This is the core of DL because each weight can be updated to obtain the expected results. First, an error function $E : \mathbb{R}^S \to \mathbb{R}$ such as the Euclidean distance [2] between the output of the network $DL(\boldsymbol{x})$ and the expected output Z:

$$E(\boldsymbol{x}) = \left(\sum_{i=1}^{|Z|} \left(Z(\boldsymbol{x})[i] - DL(\boldsymbol{x})[i] \right)^2 \right)^{\frac{1}{2}} \tag{1.20}$$

The error function computes the gap between the network output and the expected results. To quantify the error of the network over a whole set of training data $\boldsymbol{X} = (\boldsymbol{x}_i)_{1 \leqslant i \leqslant D}$, a loss function is defined as follows:

$$\mathcal{L}_X = \frac{1}{D} \sum_{i=1}^{D} E\left(\boldsymbol{x}_i\right) \tag{1.21}$$

This lost function can be seen as a function $\mathcal{L}_X(\theta)$, which depends on the trainable parameters θ. For successful training, deep learning needs to find the optimal minimizing of the loss function $\mathcal{L}_X$, and the value calculated by this function is referred to as simply "loss." In DL, the

[2] The error and loss functions presented here are given only as examples. There exist actually many different error/loss functions which can be used in Deep Learning.

preferred method is the Gradient Descent technique. DL model will do a series of iterations, and in each iteration, the gradient of loss function $\nabla \mathcal{L}_X(\theta)$ is computed. After that, θ is updated by using the following formula:

$$\theta^{(t+1)} = \theta^{(t)} - \eta \nabla \mathcal{L}_X\left(\theta^{(t)}\right) \tag{1.22}$$

where η is called the learning rate. This parameter control how quickly the model is adapted to the problem and has a small positive value, often in the range between 0 and 1.

When all the training samples have been used, they are shuffled, and the process, including forward propagation and backward propagation, is repeated. Each iteration over all sets of training data is called an epoch. Many epochs will be run until the loss converges and reaches its minimum. On each epoch, the number of correctly predicted will be used to calculate another important metric called "*accuracy*" as follows [48]:

$$accuracy = \frac{C}{A} \tag{1.23}$$

where C denotes the number of samples predicted correctly, and A is the number of total predicted samples.

b) Differential deep learning analysis (DDLA)

To apply DL in a non-profiled context, the main idea is similar to the PPA method. The authors in [7] introduced a new non-profiled SCA attack based on deep learning called differential deep learning analysis (DDLA). Accordingly, the attackers divide the given traces into different partitions corresponding to the value of partition function h. For example, h can be defined as the Hamming Weight (HW) of the target function $Sbox(d_i \oplus k_j)$. Then, for each guess key $k_j \in \boldsymbol{k}$, the attacker

computes a series of hypothesis intermediate values and groups the traces corresponding to the values $h_{i,j} = HW(Sbox(d_i \oplus k_j))$. This step is similar to labeling the data (power trace) in deep learning. Finally, attacker performs a DL training using the traces $(\boldsymbol{t}_i)_{1 \leqslant i \leqslant D}$ as training data, and the series $(h_{i,j})_{1 \leqslant i \leqslant D}$ as the corresponding classification labels.

Algorithm 2 Differential Deep Learning Analysis (DDLA) [7]

Input: D traces $(\boldsymbol{t}_i)_{1 \leq i \leq D}$, corresponding plaintexts $(d_i)_{1 \leq i \leq D}$, and K key hypotheses. A network Net and number of epochs n_e

Output: $k_{cr} \in \boldsymbol{k}$

1: Set training data as $X = (\boldsymbol{t}_i)_{1 \leq i \leq D}$.
2: **for** $k_j \in \boldsymbol{k}$ **do**
3: Re-initialize trainable parameters of Net
4: Compute the series of hypothetical values $(h_{i,j})_{1 \leq i \leq D}$
5: Set training labels as $Y_i = (h_{i,j})_{1 \leq i \leq D}$
6: Perform Deep Learning training: $DL\,(Net, X, Y_i, n_e)$
7: **end for**
8: **return** key k_{cr} which leads to the best DL training metrics

Unlike profiled DLSCA, DDLA detects the correct key by evaluating the consistency of the partitioned power traces using different hypothesis keys. Algorithm 2 summarizes the DDLA procedure to perform a non-profiled attack using deep learning. When the correct hypothesis key $k_{cr} \in \boldsymbol{k}$ is used, the series of intermediate results (labels) $Y_{cr} = (h_{i,cr})_{1 \leq i \leq D}$ will be correctly computed (assigned).

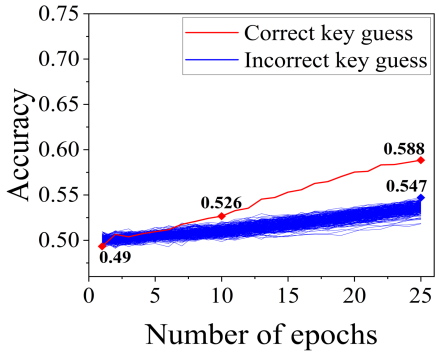

Figure 1.10: An example of attack results using DDLA algorithm.

Consequently, the training metrics (loss, accuracy) achieved by the

DL-based model trained on the dataset that is constructed from the correct key will be high because the input data are correlated to the labels. In the case of incorrect guess keys, the labels used for the training will be incompatible with the input data. Fig. 1.10 provides an intuitive example of attack results using DDLA. Accordingly, the training metrics of the model training on datasets that are created by the incorrect key will be low and chaotic (blue curve in the graph), whereas the accuracy of the model training on datasets reconstructed from the correct key (red curve) is high and stable.

1.3.4. Side-channel attack metrics

Number of traces

SCA starts from the physical observables of a device. Therefore the number of measurements is the most common metric to evaluate side-channel security. Several certification bodies and governmental agencies have integrated this metric into their guidelines, for instance, the current ISO/IEC 17825:2016. In addition, various proposals of SCA attack use this metric to quantify their efficiency [12, 49–52]. In this thesis, the number of traces is considered the main metrics to illustrate the data complexity of the proposals.

Number of epochs

In machine learning, an epoch means one complete pass of the entire training dataset through the algorithm. Therefore, with the same size as the training dataset, the number of epochs can be used to evaluate the efficiency of DL-based SCA in terms of computation time. However, it is only useful in the case of the early stopping technique being applied. This is a regularization technique for deep neural networks that stops

training when parameter updates no longer begin to yield improvements on a validation set [53,54]. In this case, translating the number of epochs into the side-channel field results in the metric of time complexity.

Success rate

The success rate metric requires the correct key to be known and is computed in the following way [55]. Let vector $[guess_1, guess_2, \ldots, guess_k]$ denote the output of an SCA attack, and let us assume that the correct key is equal to k_c. The success rate of the SCA experiment number i (i.e., SR^i) is equal to 1 if the best guess is equal to the correct key, that is if $guess_1 = k_c$. Otherwise, the success rate is equal to 0, as described in the following formula.

$$SR^i = \begin{cases} 1, if\ (k_{cr} = guess_1) \\ 0, otherwise \end{cases} \tag{1.24}$$

$$SR = \frac{1}{p} \sum_{i=1}^{p} SR^i \tag{1.25}$$

To ensure statistical stability, it is common practice to repeat the SR^i computation using multiple experiments and various keys. The final success rate metric (SR) is estimated with formula 1.25.

Partial Guessing Entropy

Similar to SR, a key guessing vector $[g_1, g_2, \ldots, g_k]$ in decreasing order of probability is the output of an attack, where g_1 denotes the most likely and g_k the least likely key candidate. Guessing Entropy (GE) is a popular SCA metrics in the SCA research community [56,57]. However, the partial guessing entropy (PGE) is prefer in some case. The "partial" refers to the fact that we are finding the guessing entropy on each subkey. This gives us a PGE for each of the 16 subkeys. A PGE of 0 indicates

the subkey is perfectly known, a PGE of 10 indicates that 10 guesses were "incorrectly" ranked higher than the correct guess. To improve consistency the PGE for each subkey is averaged over several attacks (trials) [58].

Attack time

Apart from the number of traces and success rate, the attack time is an important metric in SCA evaluation. This metric is widely used in many works to quantity the efficiency of SCA attacks [7, 40, 59]. The attack time is considered as the execution time from starting analysis/train (i.e., CPA/DDLA) of the data to the time when the secret key is taken successfully. In this thesis, the attack time is used as the primary metric to evaluate the efficiency of SCA evaluation.

1.4. SCA for hardware security

1.4.1. Developing appropriate countermeasures

To develop a defense technique, an attack method must be known in detail. Therefore, research on SCA methods is crucial for developing SCA countermeasures. Numerous countermeasures have been developed and applied to counteract the power analysis attacks. The goal of these countermeasures is to eliminate the dependency on the intermediate values and the power consumption of a cryptographic device. They are categorized into two main groups: masking countermeasures and hiding countermeasures. This current subsection presents general knowledge about these countermeasures and discusses how they work.

Masking

Masking countermeasures can defend power analysis by randomizing the intermediate values v that are processed by the cryptographic device. The randomizing of v can be achieved by concealing it with a random value m called a mask, which the attacker does not know. In practice, several masks can be used to conceal v. Therefore, masking can be called secret sharing with d shared and presented as follows:

$$v = m_1 * m_2 \ldots * m_d \tag{1.26}$$

the operation $*$ is typically defined according to the operations that are used in the cryptographic algorithm (*e.g.* the exclusive or for the most popular Boolean masking already proposed in [60]).

The soundness of the masking countermeasure is implied by the fact that the complexity of recovering a secret key by power analysis on v shared into several pieces grows exponentially with the number of d shares [61]. From the implementation point of view, the primary advantage of masking countermeasures is that the device's power consumption characteristics do not need to be altered. In addition, masking countermeasures can be deployed easily at the software level of cryptographic systems.

Despite having many advantages, the drawback of masking countermeasures is that they require modifications to cryptographic algorithms. This requirement causes an increment in the computational complexity of the cryptographic algorithm, decreases the performance, and enlarges the resource usage of the algorithm's implementation on a secret device.

Hiding

Apart from masking techniques, another well-known group of countermeasures is hiding countermeasures. The main purpose of the hiding method is to break or at least reduce the dependencies of processed intermediate values and the real power consumption. To achieve these requirements, two approaches are commonly used, as described below:

The first is to randomize the power consumption of the target device on each clock cycle by performing the operations of the executed cryptographic algorithm at different moments of time during each execution. These methods consequently only affect the time dimension of the power traces. This can be done by randomly inserting dummy operations [62, 63], clock randomization [64], shuffling [65] or other [66–69]

The second is to make the power consumption consumes an equal amount of power for all operations and all processed data values. Therefore, this approach affects the amplitude dimension of the power consumption [70–73].

Different from masking countermeasures, hiding techniques do not modify the cryptographic algorithm and are free of unexpected mathematical vulnerabilities. The drawback of hiding methods is that, in practice, the data dependency cannot be entirely eliminated. Therefore, applying a hiding countermeasure only increases the amount of power consumption traces that an attacker must analyze in order to effectively reveal the secret key. Hiding countermeasures cannot completely prevent power analysis attacks.

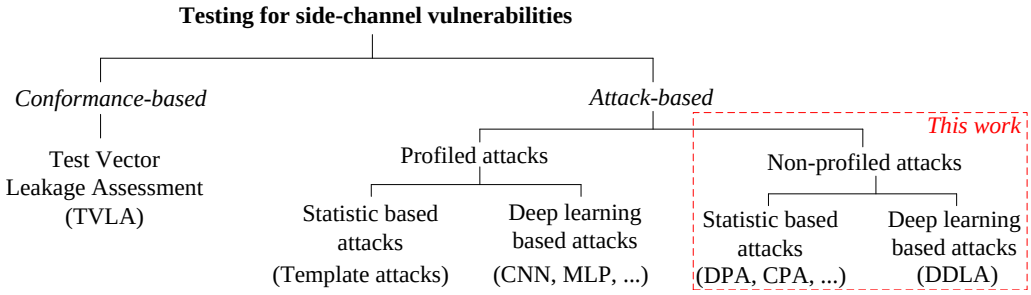

Figure 1.11: Classification of side-channel security evaluation

1.4.2. Hardware security evaluation

To go against the side-channel threats, the designers must provide guaranteed security on their designs. The goal of hardware security evaluation is to evaluate the effectiveness of countermeasures and detect the potential SCA threat. In this context, efficient validation and evaluation methodology for testing side-channel vulnerability has gathered significant interest in the research community. In particular, there exist today two popular security certification programs "*conformance-based*" and "*attack-based*" testing, as depicted in Fig. 1.11.

Conformance-based evaluation

Conformance-based testing aims to develop cost-effective procedures to verify minimum properties for secure side-channel implementations. Examples include the FIPS 140-3 standard [74], and the popular TVLA methodology [75]. This testing employs a simplified approach for merely detecting the presence of any leakage, independent of the attack method and leakage models. Therefore, the conformance-based testing mechanism can only detect the presence of side-channel vulnerability. As mentioned above, SCA countermeasures cannot completely prevent power analysis attacks. Therefore, it is important to know more than the answer "*yes or no*" of the presence of SCA leakage. In other words, the

side-channel vulnerabilities need to be quantified in more detail, such as the data complexity, the right leakage model, or the attack time.

Attack-based evaluation

Unlike conformance-based evaluation, attack-based testing rather aims at defining a common framework for evaluating implementations by analyzing different attack strategies. Obviously, the efficiency of the SCA evaluation process depends on the computational complexity of SCA attacks corresponding to applied SCA countermeasures. This testing needs a significant effort required to keep track of all existing SCA attacks. However, the advantage of this methodology is that it can quantify side-channel susceptibility while also finding applications in comparing the vulnerability of two designs.

1.4.3. The related works and research directions

From the analysis above, research on SCA techniques is crucial for hardware security. Especially in the case of side-channel security evaluation. Various works have been published in the SCA domain to introduce efficient SCA techniques, which can be applied for SCA evaluation as well as propose potential countermeasures. This part provides a review of the related works and indicates the potential research directions with respect to two main groups: statistic-based attacks and DL-based attacks.

Statistic-based attacks

In terms of the statistic-based non-profiled SCA techniques, DPA and CPA are the common attacks used in SCA evaluation. They have demonstrated the efficiency in attacking both unprotected and protected de-

vices. Especially, the CPA technique has been used to break different block ciphers such as DES, AES [34], PRESENT, PICCOLO, and GIFT [37]. In addition, it is also the most commonly used technique to clarify the efficiency of SCA countermeasures [52, 62, 76–78]. Therefore, improving the efficiency of CPA has received significant interest in recent years.

Kim *et al.* [35] proposed a method to extract a small set of traces with a high signal-to-noise ratio (SNR) distributed in both tails of the distribution range. This method aims to enlarge the variance of the exploitable consumption component in the power trace. Similarly, an empirical method uses the adaptive chosen-plaintext CPA attacks (ACP-CPA) [36]. The authors tried to resolve the drawback of discarding too many traces in the extractor proposed by Yongade *et al.* [35]. However, this technique requires many requests to choose adaptive plaintexts for all bytes. Hence, it makes SCA security testing a time-consuming process.

Recently, an improvement in the power traces extractor presented by Ou *et al.* [51] called Maximizing Estimated SNR First (MESF). Unlike the proposed in [35], the novelty of this work is that this technique extracts the subset of power traces with the smallest estimated noise and maximizes the variance of the data-dependent power consumption. Consequently, by using the high SNR samples, the computation complexity of the attacks can be reduced instead of increasing the success rate (SR). However, Ou's techniques are based on the mean power consumption of plaintext byte values. This constraint limits these techniques to scenarios where the attacker has a large number of traces to estimate the mean values. Hence, it leads to the high computational cost of performing an

attack with a large number of power traces.

In summary, the power trace extractor is an efficient pre-processing technique that improves correlation in power analysis attacks. However, in terms of computational complexity, it is not an optimal way since the correlation coefficient needs to be calculated on all samples of power trace. Based on this drawback, a new SCA technique was proposed in this thesis, which automatically selects a small subset of useful sample points (Point of Interest: POI) together with key recovery. This proposal will be described in Chapter 2.

DL-based attacks

As presented previously, statistic-based attacks have been performed successfully on both unprotected and protected devices. However, in the case of SCA countermeasures added, a dedicated pre-processing technique is required for each specific attack. It leads to an increment cost and time-consuming process of SCA evaluation. Fortunately, by applying DL techniques in the SCA domain, pre-processing techniques are no longer required. However, DL techniques are usually considered an alternative to profiling attacks. Research on DL-based non-profiled SCA is still quite a new field.

Indeed, Timon introduced the first DL-based attacks, namely Different Deep Learning Analysis (DDLA), in a non-profiled scenario in TCHES 2019 [7]. The main idea of DDLA is to observe the trend of training metrics, such as “loss”, and “accuracy,” to discriminate the correct key from a set of hypothesis keys. Accordingly, to reveal one byte key, the attacker/evaluator must repeatedly perform the training process for each hypothesis key. Following Timon’s approach, some other

works have been presented to investigate the performance or enhance the efficiency of DDLA-based attacks. In [21], the author investigated the performance of DDLA attacks on the datasets by applying different countermeasures such as masking and correlated noise generation. The experimental results of their work indicated that a hiding countermeasure might provide higher protection against non-profiled DLSCA. Similarly, the authors in [23] reported on the results of DDLA attacks against AES software implementation protected with two types of masking countermeasures. One is the table re-computation masking countermeasure, and the other is the Rotating Sboxes Masking (RSM) countermeasure. Additionally, the authors proposed using regularization in DDLA. They concluded that using L1/L2 regularization has a significant advantage on the performance of DDLA compared to results without it.

Overall, the mentioned non-profiled SCA techniques have demonstrated the efficiency of deep learning in SCA evaluation without reference devices. In addition, DDLA could break the different protected schemes without any pre-processing techniques. However, there is no report on some complex conditions, such as high-dimension data input or different labeling techniques. Especially in the case of HW labeling, the imbalanced dataset occurs. These issues will be investigated in chapter 3. A dimensionality reduction and new labeling techniques will be proposed to deal with these problems.

Most recently, Kwon et al. [40] have indicated a major issue of the original DDLA and introduced a new approach based on a multi-label neural network. The main drawback of DDLA is the requirement of a training process for each key hypothesis. It means that the original DDLA technique is not optimized in terms of execution time. The au-

thors have introduced a parallel architecture to mitigate the mentioned issue. Accordingly, their proposed models can simultaneously predict a total of 256 hypothesis keys. Kwon's work can be considered as a multi-label SCA approach as in [25]. Despite being a very fast attack technique, the parallel architecture requires high memory usage. They have also introduced a shared-layer-based model to mitigate the drawback of parallel architecture. However, the important SCA metric, such as the success rate of attacks, has not been investigated, especially in applying noise injection countermeasures.

An alternative and often more effective approach in the DL domain is to develop a single neural network model that can learn multiple related tasks (i.e., outputs) at the same time, called multiple-output learning (MOL) [79–81]. From the point of view of the SCA domain, MOL is a promising technique that could increase the performance of the SCA evaluation process. However, most of the proposed techniques only work in profiled contexts [25–27]. There is no report of non-profiled DLSCA using multi-output (multi-loss) architecture. Therefore, this approach will be investigated in Chapter 4.

In Viet Nam, there are only a few publications on the SCA domain, and most of them are published by the research teams from the Vietnam Academy of Cryptography Techniques, Vietnam National University, Hanoi or Le Quy Don Technical University. In addition, their recent works focus on profiled attacks.

In [82, 83], the authors proposed a preprocessing method for selecting POI based on the combination of variational mode decomposition (VMD) and Gram-Schmidt orthogonalization (GSO). VMD is used to decompose the power traces into sub-signals (modes), and POIs selec-

tion process based on GSO is conducted on these sub-signals. Selected POIs are then used for the Support Vector Machine (SVM) classifier to conduct profiled attacks.

In [41], the authors introduced an efficient CNN-based profiled attack. The novelty of their work is using the swarm-based method called Grey Wolf Optimizer (GWO) to automatically fine-tune the CNN hyperparameter.

Apart from proposing new SCA attacks, various works proposed secured designs and used attack-based testing methods to verify the security of their designs. The authors from Vietnam National University, Hanoi, performed CPA attacks to evaluate their proposed secure processor [77]. Most recently, K.-H Pham *et al.* have used first-order CPA attacks to demonstrate the efficiency of the new masking method for hardware-based AES implementation [78]. These works have shown that attack-based testing is the popular methodology for hardware security evaluation.

1.5. Summary

Based on the analysis of different SCA approaches, non-profiled SCA attacks are considered to be the most suitable and efficient techniques for evaluating the security level of modern electronic devices, which are flexible and highly customizable. This chapter presents the background knowledge of SCA, leakage data, and SCA for hardware security. In particular, it presents a comprehensive review of recent works on non-profiled SCA methods as well as outlines some research directions that promote the contributions of this work. Each issue will be dealt with in turn in the next chapters of the thesis.

Chapter 2
LOW COMPLEXITY CORRELATION POWER ANALYSIS ATTACKS

In scenarios (such as masked implementations [84, 85]) where sensitive data dependencies have been successfully eliminated from individual leakage points, the leakage information still persists in joint distributions of multiple points. However, the direct estimation of this information becomes exponentially hard as the number of shares in the masking scheme increases (just as the expected security level). Numerous previous works which aim to reduce the computation complexity of the CPA technique or enhance the results of attacks were discussed in the previous chapter. However, most of the previous works focused on improving first-order CPA [35, 36, 49], or only increasing the success rate of second-order attacks as in [51].

This chapter introduces improved CPA techniques, which can deal with either first-order or second-order leakages. The proposed techniques could be applied to any preprocessed second-order data. In addition, the proposals outperform non-profiled DLSCA in terms of data complexity and attack time, especially in the case of performing security testing on a noise-generation countermeasure. The results of this chapter were presented in the papers [C1], [J1].

2.1. The complexity of CPA attacks

In order to enhance the efficiency of CPA, this section presents the complexity of the CPA attacks on both unprotected and protected devices using masking countermeasure. Then, an important factor that impacts the complexity of CPA attacks will be indicated. This factor is the base for the proposals in this chapter.

As discussed in Section 1.3, the power traces correspond to the power consumption of the device while it executes a cryptographic algorithm using different data inputs and a fixed key k_{cr}. In addition, the intermediate value is a part of this algorithm. Therefore, the device needs to calculate the intermediate values $\boldsymbol{v}_{cr}$ using k_{cr} during the different executions of the algorithm. Consequently, the recorded traces depend on these intermediate values at same position. This position of the power traces, namely *correct sample*, is denoted as ct (*i.e.* the column $\boldsymbol{t}_{ct}$ of the matrix $\boldsymbol{T}$ contains the power consumption values that depend on the intermediate values $\boldsymbol{v}_{cr}$). By calculating hypothetical power consumption $\boldsymbol{h}_{cr}$ based on k_{cr}, the correlation between $\boldsymbol{h}_{cr}$ and $\boldsymbol{t}_{ct}$ is the strongest. In fact, they lead to the highest value in $\boldsymbol{R}$ (*i.e.* the highest value of the matrix $\boldsymbol{R}$ is $r_{cr,ct}$).

It is clear that the number of correlation coefficients of a CPA attack needs to calculate is the size of matrix $\boldsymbol{R}$. Therefore, the complexity of CPA is proportional to $K * S$. In other words, the complexity of CPA depends on the length S of the power traces. As depicted in Fig. 2.1, only one sample (in 10,000 samples) results in the highest correlation value ($r_{cr,ct}$) at the correct sample t_{ct}. It means that most correlation calculations on other samples are redundant. This problem becomes

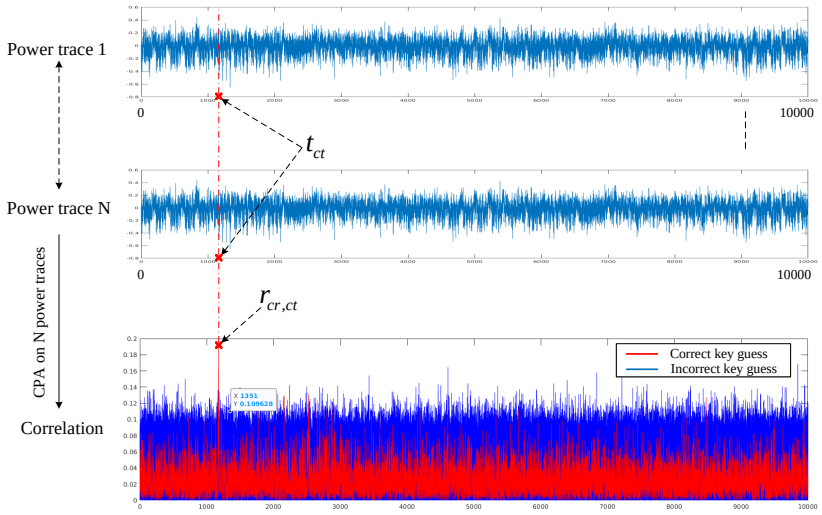

Figure 2.1: The complexity of first-order CPA attacks on high dimensional SCA data

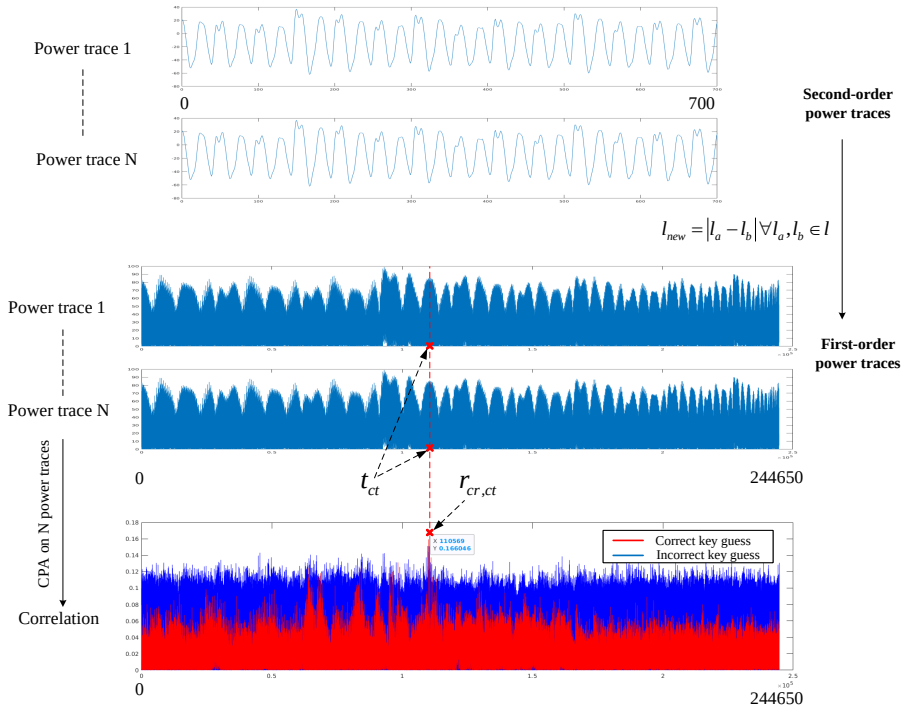

Figure 2.2: The complexity of second-order CPA attacks on high dimensional SCA data

more severe for second-order attacks. Indeed, the authors in [9] have shown that the complexity of second-order CPA is the square order in comparison to a standard CPA attack on an interval with the length of S.

Accordingly, the length of each power trace will increase from S to $\frac{S(S-1)}{2}$. Therefore, the computations of the second-order CPA is proportional to $\frac{S(S-1)}{2} * K$. Fig. 2.2 shows an intuitive example. Second-order attacks can reveal the secret key from only one correct sample (t_{ct}) from more than 244,000 samples of the processed power traces.

In practice, it is impossible to figure out the correct sample before mounting the attacks. However, the attacker can point out a group of samples, namely Points of Interest (POI) [86], which contains the correct sample t_{ct} for SCA attacks (the number of POI is very small compared to S). It is clear that performing CPA on POI will reduce significantly the number of computations. Therefore, extracting POI is crucial to speed up the CPA attack. In the next section, two efficient POI extractors based on the distribution of sample correlation are introduced.

2.2. Distribution of sampling correlation coefficients in CPA

2.2.1. Leakage characteristics of samples

According to [87], the power consumption of a single sample t_τ can be expressed as the sum of a data-dependent component $t_d(\tau)$, an operation-dependent component $t_o(\tau)$, switching noise $t_{sw.noise}(\tau)$, electronic noise $t_{el.noise}(\tau)$, and the constant component $t_{const}(\tau)$. All components are independent with each other as shown in (2.1).

$$t(\tau) = t_o(\tau) + t_d(\tau) + t_{sw.noise}(\tau) + t_{el.noise}(\tau) + t_{const}(\tau) \tag{2.1}$$

Let $t_{exp}(\tau)$ denote the exploitable component including the operation-dependent component $t_o(\tau)$ and data-dependent component $t_d(\tau)$, see 2.2. Let $t_{noise}(\tau)$ denote the noise component consist of the switching

noise $t_{sw.noise}(\tau)$ and the electronic noise $t_{el.noise}(\tau)$, as in (2.3).

$$t_{exp}(\tau) = t_d(\tau) + t_o(\tau) \tag{2.2}$$

$$t_{noise}(\tau) = t_{sw.noise}(\tau) + t_{el.noise}(\tau) \tag{2.3}$$

The SNR of the sample t_τ is the ratio of the variance of exploitable power consumption $t_{exp}(\tau)$ to the variance of noise component $t_{noise}(\tau)$. Therefore, the formula of SNR can be simplified as:

$$SNR = \frac{\sigma^2(t_{exp}(\tau))}{\sigma^2(t_{noise}(\tau))} \tag{2.4}$$

The correlation coefficient of a sample is estimated by formula (1.17), as indicated in Section 1.3.3. Based on SNR, the authors in [87] provided another correlation calculation method as follows:

$$\rho(h_k, t) = \frac{\rho(h_k, t_{exp})}{\sqrt{1 + \frac{1}{SNR}}} \tag{2.5}$$

For a conventional CPA, the correlation $\rho(h_k, t_{exp})$ between the power consumption model and the data-dependent component is a constant for a time sample. From equation (2.5), it is clear that the SNR determines the correlation $\rho(h_k, t)$, and $\rho(h_k, t)$ approaching a constant when the number of power traces used in attack is large enough.

2.2.2. Distribution of sampling correlation

According to Equation 1.17, the correlation coefficients are calculated at every point of power traces for each key hypothesis. In this case, S different values of r correspond to S points are calculated using n power traces. Therefore, the distribution of values of r on each point after repeated samples of n power traces is the sampling distribution [88].

By applying Fisher's transformation (if $n \geq 30$), the correlation coefficient ρ can be mapped to a random variable Z that has a normal

distribution, as in the equation (2.6). The mean of Z is then given by μ in (2.7) and the variance in (2.8).

$$R \mapsto Z = \frac{1}{2} \ln \frac{1+R}{1-R} \tag{2.6}$$

$$\mu = E(Z) = \frac{1}{2} \cdot \ln \frac{1+\rho}{1-\rho} \tag{2.7}$$

$$\sigma^2 = Var(Z) = \frac{1}{n-3} \tag{2.8}$$

In SCA security testing, the number of traces needed in a CPA attack is commonly used to measure the resistance of a design against these attacks. In order to reveal the correct subkey k_c, the number of power traces needs to be increased in an attack until a significant peak $\rho_{\max}$ is visible in the correlation matrix $\boldsymbol{R}$. Based on Fisher's transformation, the authors in [87] have proposed a method to calculate the lower bound of the number of power traces only based on the peak correlation. Concretely, they assume that the peak is determined by the distance between the sampling distribution with $\rho = 0$ and $\rho = \rho_{\max}$. Apply the formulas 2.6, 2.7 and 2.8, the distribution of two sampling distribution with $\rho = 0$ and $\rho = \rho_{\max}$ can be described as follows:

$$\begin{aligned}
R_0 \;\mapsto\; & Z_0 = \tfrac{1}{2} \ln \tfrac{1+R_0}{1-R_0} \\
& \mu_0 = E\left(Z_0\right) = \tfrac{1}{2} \ln \tfrac{1+0}{1-0} + \tfrac{0}{2 \cdot (n-1)} \\
& \sigma^2 = Var(Z_0) = \tfrac{1}{n-3} \\
R_1 \;\mapsto\; & Z_1 = \tfrac{1}{2} \ln \tfrac{1+R_1}{1-R_1} \\
& \mu_1 = E\left(Z_1\right) = \tfrac{1}{2} \ln \tfrac{1+\rho_{\max}}{1-\rho_{\max}} + \tfrac{\rho_{\max}}{2 \cdot (n-1)} \\
& \sigma^2 = Var(Z_1) = \tfrac{1}{n-3}
\end{aligned} \tag{2.9}$$

it is noted that the fraction $\rho_{\max}/(2 \cdot (n-1))$ approaches zero for large n and the $\rho_{\max}$ is small. In order to measure the distance between the distributions, the authors calculate the probability α that a value drawn from the distribution with $\rho = \rho_{\max}$ (Z_1) is bigger than the one that is drawn from the distribution with $\rho = 0$ (Z_0). According to [87], the number of traces n that is necessary to assert with a confidence of α that the two normal distributions Z_0 and Z_1 are different is given by:

$$n = 3 + 8 \cdot \frac{z_\alpha^2}{\left(\ln \frac{1+\rho_{max}}{1-\rho_{max}} - \ln \frac{1+\rho_0}{1-\rho_0}\right)^2} \tag{2.10}$$

where z_α is the quantile of the probability α. Since $\rho_0 = 0$, then the equation 2.10 becomes to:

$$n = 3 + 8 \left(\frac{z_\alpha}{\ln\left(\frac{1+\rho_{\max}}{1-\rho_{\max}}\right)} \right)^2 \tag{2.11}$$

In [20], the authors perform further analysis and provide several comprehensive examples of CPA attacks. Especially, the authors elaborate on issues like the simulation of CPA attacks and the calculation of the number of power traces that are needed to perform CPA attacks successfully. Table 6.1 in [20] provides the results to illustrate the relationship between $\rho_{\max}$ and the calculated number of traces according to (2.11). Especially, the authors indicate that since $\sigma = 1/\sqrt{n-3} \approx 1/\sqrt{n}$ with the large enough value of n, the estimators for all correlation coefficients before and after the attacked intermediate results is processed are essentially located in the interval $\pm 4\sigma = 4/\sqrt{n}$. Moreover, the authors also indicated the relationship between n and $\rho_{cr,ct}$ as follows:

$$n \approx \frac{28}{\rho_{cr,ct}^2}. \tag{2.12}$$

where $\rho_{cr,ct}$ denotes the correlation value that is calculated by formula 1.17 at position t_{ct} of power traces and the intermediate value $\boldsymbol{h}_{cr}$ using the correct key k_{cr}. In this case, $\rho_{cr,ct}$ equals to ρ_{max}. The purpose of the proposals in this chapter is not to find the lower bound of the number of power traces. Assuming the n is already sufficient and used in total to implement the CPA efficiently. The proposed techniques for reducing the complexity of CPA attacks on n power traces will be discussed in the next section.

2.3. Partial correlation power analysis (P-CPA)

As mentioned in the previous section, all values of the matrix $\boldsymbol{R}$ are drawn from one of two sampling distributions. By using Fisher's transformation, we assume that all values of the matrix $\boldsymbol{R} \rightarrow \boldsymbol{Z}$ will be drawn from two normal distributions $N_1(0, \sigma_n)$ and $N_2(\mu_{\rho_{\max}}, \sigma_n)$, where $\sigma_n = \frac{1}{\sqrt{n}}$ as illustrated in Fig. 2.3. From the equation (2.8), it is clear that an attacker can decrease the overlap between these two normal distributions by increasing the number of traces.

As explained in [20], the recorded traces are quite long compared to the interval during which the attacked intermediate results are processed. Usually, many operations are executed during the recording and are completely independent of the attacked intermediate values. Therefore, most correlation values $\rho_{k \neq cr, s \neq tc}$ are usually zero in practice. Indeed, it can assume that all values of ρ, except $\rho_{cr,tc}$, will be distributed in the form of $N_1(0, \sigma_n)$, and values of $\rho_{cr,tc}$ will be distributed in the form of $N_2(\mu_{\rho_{\max}}, \sigma_n)$. Hence, there are some observations as follows.

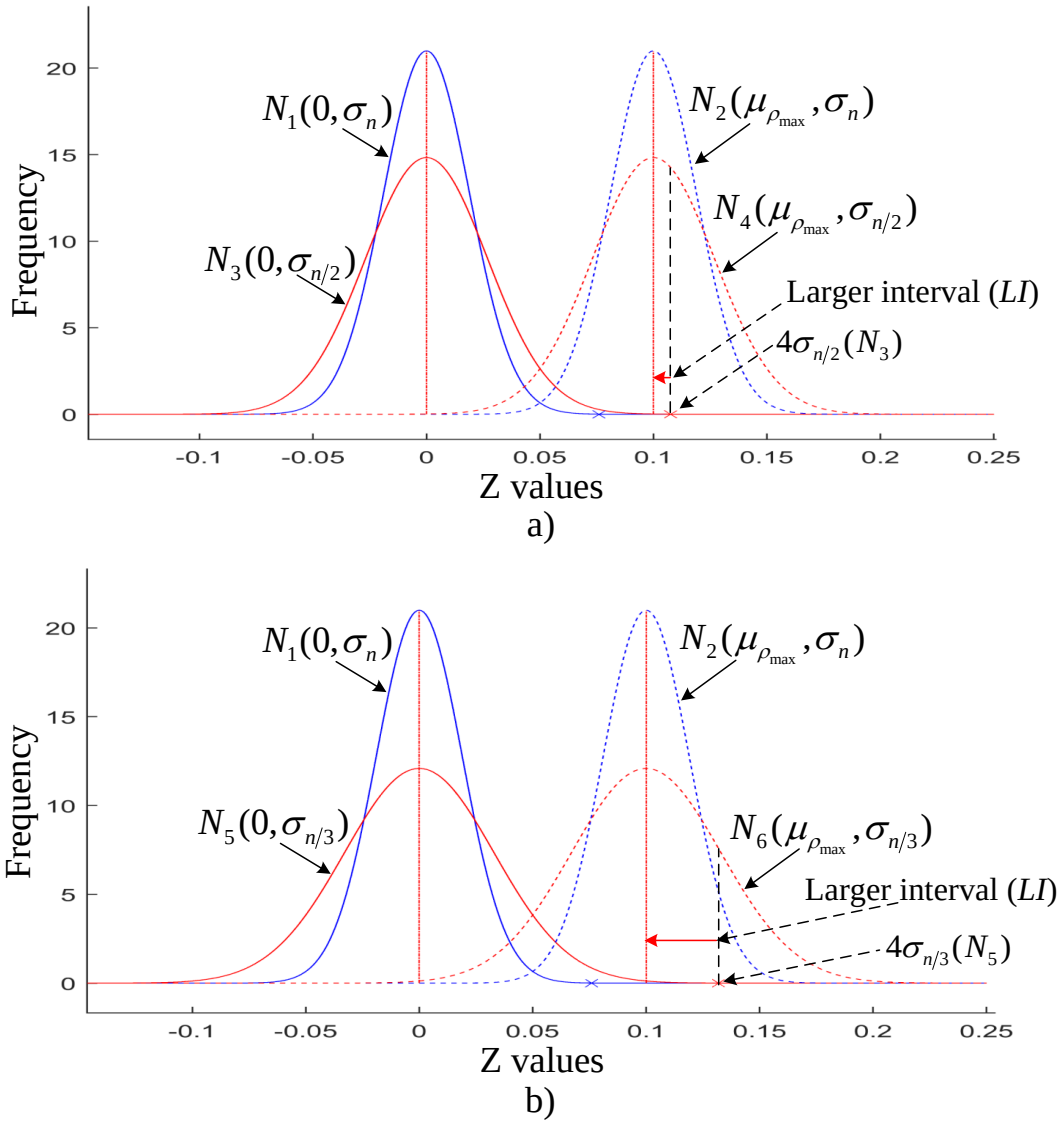

Figure 2.3: The probability density of Z values on different numbers of used power traces: a) n and $n/2$; b) n and $n/3$.

Firstly, if the number of traces reduces from n to $n/2$ and $n/3$, the standard deviation of the normal distribution $N_1(0, \sigma_n)$ will be changed from σ_n to $\sigma_{n/2}$, $\sigma_{n/3}$, respectively. The shape of distribution $N_1(0, \sigma_n)$ will be changed to $N_3(0, \sigma_{n/2})$, $N_5(0, \sigma_{n/3})$, respectively, as illustrated in Fig. 2.3.

Secondly, from the equation (2.5), the correlation $\rho_{cr,tc}$ will be a stable value when the number of traces is large enough. In this case, assuming

that the $\rho_{cr,tc}$ will distribute around and near the mean $\mu_{\rho_{\max}} = \frac{1}{2}\ln\frac{1+\rho_{\max}}{1-\rho_{\max}}$ and then keeps consistent. Therefore, the value of mean $\mu_{\rho_{\max}}$ will be used to determine the interval of POI.

Thirdly, if n is replaced by $n/2$, the correlation coefficients are located in new range $4\sigma_{n/2} = \frac{4\sqrt{2}}{\sqrt{n}}$. In addition, from Eqs. (2.12), we have $\rho_{\max} \approx \frac{\sqrt{28}}{\sqrt{n}}$. Therefore, it is easy to obtain:

$$4\sigma_{n/2} > \rho_{\max} \tag{2.13}$$

In this context, $\rho_{\max}$ is small ($\rho_{\max} < 0.2$), therefore $\mu_{\rho_{\max}} \approx \rho_{\max}$. It means that the right tail of distribution $N_3(0, \sigma_{n/2})$ is larger than $\mu_{\rho_{\max}}$. This larger interval (LI) can be calculated as $LI = 4\sigma_{\rho_0} - \mu_{\rho_{\max}}$, where σ_{ρ_0} denotes standard deviation of zero mean normal distribution in Fig. 2.3. The same calculations can be done for $n/3$. As a result, if LI containing the samples which have the highest correlation are taken, the correct samples t_{tc} can be detected. In other words, the POI can be taken based on the right tail of $N_3(0, \sigma_{n/2})$, $N_5(0, \sigma_{n/3})$.

Let's consider an intuitive example of an 8-bit Sbox output of a popular block cipher to illustrate the observations about the correlation. It is noted that with the non-linear property, Sbox function of AES-128 can be selected but not limited to other versions like AES-256 as well as other block ciphers such as PRESENT or GIFT. In this case, the ASCAD data containing the power consumption data of a boolean masking AES-128 implementation is selected. An online 2-order CPA is performed on the first 1200 power traces of ASCAD database. Then, the maximum correlation of each hypothesis key is taken. In this context, assume that the secret key is known and illustrated by a red line in Fig. 2.4. Two remarkable positions in which the numbers of traces are $\frac{n}{2}$ and $\frac{n}{3}$ are

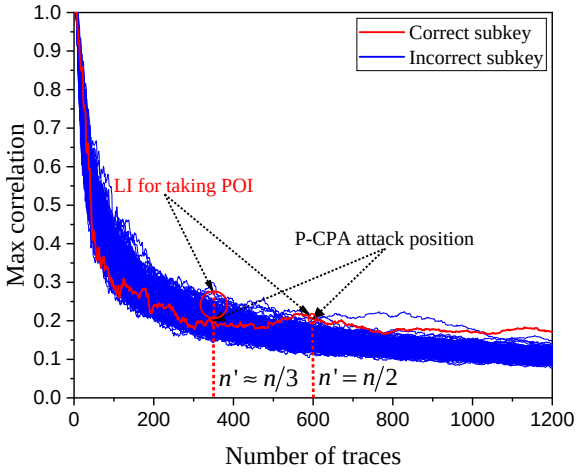

Figure 2.4: The remarkable positions to perform P-CPA.

determined, respectively.

In addition, the diameter of the red circle is used to illustrate the LI for taking POI. It is clear that at the position $n' = \frac{n}{2}$, the red circle is very small, and the maximum correlation of the correct key is higher than almost all other hypothesis keys. It means that if the top-down ε POIs, which have the highest correlation at $n' = \frac{n}{2}$ are taken, the correct samples t_{tc} could be detected. In contrast, at the position $n' = \frac{n}{3}$, the red circle is larger, and the correlation is very low compared with other hypothesis keys. Therefore, a larger number of POIs must be taken to reveal the correct samples t_{tc}. It is noted that the correlation matrix has the size of $K \times S$. Therefore, taking the sample t_{tc} when $n' = \frac{n}{3}$ is very challenging.

With ε selected POIs where ε is very small in comparison to S, the CPA computation complexity can be reduced proportionally with $\frac{S}{\varepsilon}$. The first proposal is completed by Algorithm 3. Suppose that Algorithm 3 is employed to perform an auto-CPA on matrix power traces $\boldsymbol{T}$ that has the size of $n \times S$, corresponding to a set of plaintexts which has a size of $n \times 16$, n' is set as $n' = \frac{n}{2}$ and $\varepsilon = 100$.

Algorithm 3 is divided into two phases. Firstly, a subset of power trace, which has a size of $n' \times S$ (Steps 1, 2) is taken. Then, the standard CPA is performed using the function $StandardCPA()$ on this subset of power traces. As a result, the matrix correlation $\boldsymbol{R}$ is achieved, and it is also the end of phase 1 (Step 4). Next, top-down 100 samples corresponding to the first 100 highest correlation values of the matrix $\boldsymbol{R}$ (Steps 7,8,9) are taken. After that, the second subset of power traces is selected with the size of $n \times 100$ and the corresponding subset of plaintexts (Steps 11, 12). Finally, phase 2 is performed (Step 14). The standard CPA is implemented again on the newest subset of power traces and plaintexts. The output of Algorithm 3 is the correct byte of the secret key.

Algorithm 3 Auto-CPA based on partial correlation power analysis: P-CPA

Input: $trace^{n \times S}$, plaintext$^{n \times 16}$, attack byte B, $n' = \frac{n}{2}$, $\varepsilon = 100$
Output: $\boldsymbol{k}[B]$

1: $Plt_0 =$ plaintext$^{n' \times 16}$
2: $Tr_0 = trace^{n' \times S}$
3: **for** k from 0 to K **do** ▷ Phase 1
4: $\quad \boldsymbol{R} \leftarrow StandardCPA(Plt_0, Tr_0)$
5: **end for**
6: **while** $ii \leq \varepsilon$ **do** ▷ Taking POIs
7: $\quad ii = ii + 1$
8: $\quad \bar{S}[ii] = s_{max} \leftarrow argmax(\boldsymbol{R})$ ▷ s_{max}: the index of the column containing max value
9: $\quad s_{max} = 0$
10: **end while**
11: $Plt_1 =$ plaintext$^{n \times 16}$
12: $Tr_1 = trace^{n \times \bar{S}}$ ▷ $\bar{S}$ has size of $(1 \times \varepsilon)$
13: **for** k from 0 to K **do** ▷ Phase 2
14: $\quad \boldsymbol{R} \leftarrow StandardCPA(Plt_1, Tr_1)$
15: **end for**
16: $\boldsymbol{k}[B] = line_{max} \leftarrow argmax(\boldsymbol{R})$ ▷ $line_{max}$: the index of the line containing max value

2.4. Partial correlation power analysis based on power trace biasing (BP-CPA)

As indicated previously, taking the sample t_{tc} when $n' = \frac{n}{3}$ is very challenging. Therefore, this section proposes another method based on

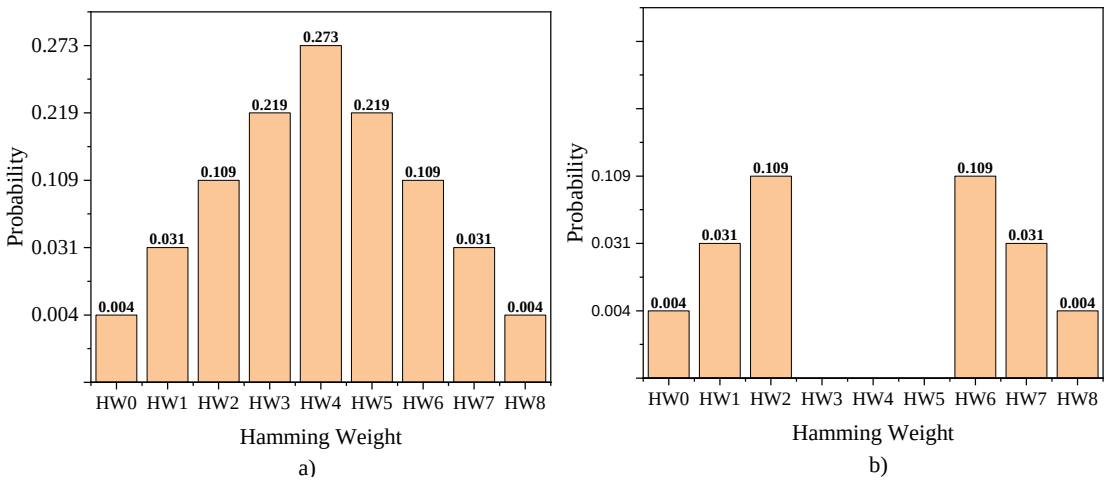

Figure 2.5: Probability distribution of the HW of a uniformly distributed 8-bit value. a) All HW; b) High variance HW.

the power trace biasing technique.

For each sample t_τ, the operation on all power traces is usually the same. Therefore, the variance of the operation-dependent power consumption $\sigma^2(t_o(\tau)) = 0$. Consequently, the SNR value in (2.4) can be further simplified as:

$$SNR = \frac{\sigma^2(t_d(\tau))}{\sigma^2(t_{noise}(\tau))} \tag{2.14}$$

In the conventional CPA, it is clear that if the plaintexts are chosen randomly, the intermediate values are uniformly distributed. In addition, each bit of the intermediate value is independent of the other bits, and the probability of each bit is 0.5. Therefore, HW follows a binomial distribution. Consequently, the HW with a big deviation appears with low probability. For example, the probability of each HW of 8-bit Sbox output is distributed as expressed in Fig 2.5.a. It is noted that HW of other bit-length Sbox output (such as 4-bit of DES or PRESENT algorithms) is also followed the binomial distribution. In this case, assume that $Var(t_d(\tau))$ proportions to $Var(HW)$. It means that the high values of the variance $\sigma^2(t_d(\tau))$ could be taken if the adaptive plaintexts

are chosen. Consequently, SNR increases, $\rho_{cr,tc}$ will increase according to formula 2.5, if the plaintexts corresponding to high $Var(HW)$ are selected as depicted in Fig 2.5.b. Based on this observation, a method for selecting the adaptive plaintexts which give the high SNR values is proposed. However, unlike the previous works [36], the proposal's novelty is that the plaintexts are selected from a given set.

Using the 8-bit Sbox as an example, if n' plain-texts which have the intermediate values corresponding to $HW = \{0, 1, 2, 6, 7, 8\}$ are taken, n' is approximately 28.9% of n. In other words, if the CPA is calculated on $n' = \frac{n}{3}$ for taking POI as the phase 1 of P-CPA, the probability of taking the correct sample t_{tc} is very high. The previous intuitive example is used again with further experiments. Fig. 2.6.(a) illustrates the correlation at the first 350 power traces of a given set of power traces (one-third of the previous example). Meanwhile, Fig. 2.6.b presents the correlation of 350 power traces selected by biasing technique. It clearly shows that the correlation of the correct key in Fig. 2.6.(b) is higher and more consistent than that in Fig. 2.6.(a). Therefore, it is easy to take the number of POIs that consist of the correct sample t_{tc}. The second proposed method of this chapter is presented by Algorithm 3.

Similar to Algorithm 3, Algorithm 4 is divided into two phases. However, Algorithm 4 is different from Algorithm 2 in the process of phase 1, and the value of $\varepsilon = 250$. The proposal does further steps to take out $n' \approx \frac{n}{3}$ power traces based on the biasing technique (Step 6,7). The rest of Algorithm 4 is processed the same as Algorithm 3.

An important practical aspect is the number of POIs that have to be taken in order to perform a successful attack. However, calculating the exact number of POIs that contain the correct sample is very chal-

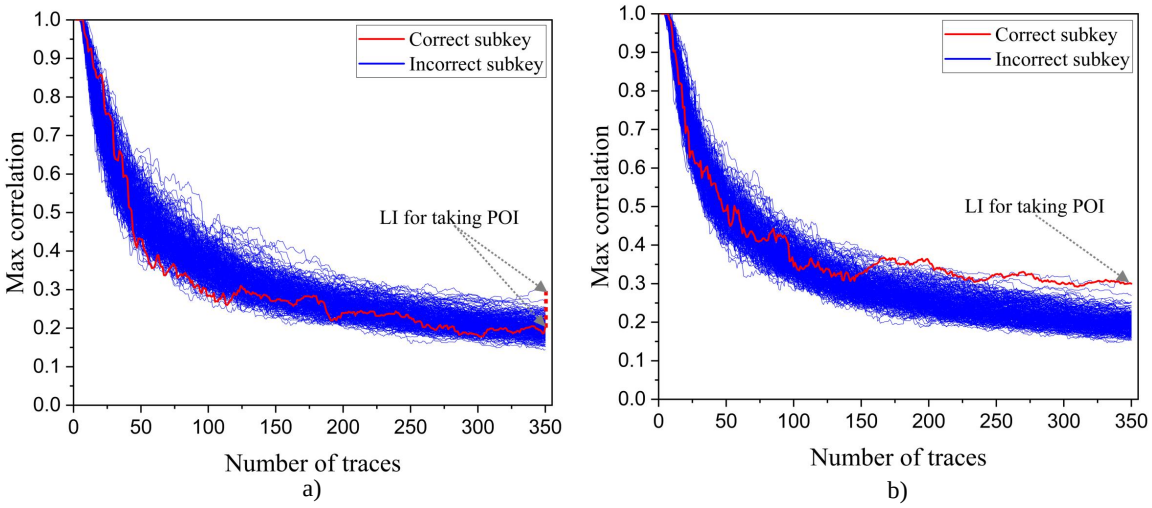

Figure 2.6: The correlation of the correct key for two CPA methods: a) Standard CPA; b) CPA with power trace biasing technique.

Algorithm 4 Auto-CPA based on power trace biasing based partial correlation power analysis: BP-CPA

Input: $trace^{n\times S}$, plaintext$^{n\times 16}$, attack byte B, $n'=0$, $\varepsilon=250$
Output: $\boldsymbol{k}[B]$

$Plt_0 =$ plaintext$^{n\times 16}$
for k from 0 to K **do** ▷ Phase 1
  for i from 1 to n **do**
    $h_{i,k} \leftarrow HW(SBOX((plaintext_{i,B}, k))$
    if $h_{i,k} = 0, 1, 2, 6, 7, 8$ **then**
      $n' = n' + 1$
      $trace_{n',S} = trace_{i,S}$
    end if
  end for
  $Tr_0 = trace^{n'\times S}$ ▷ $n' \approx \frac{n}{3}$
  $\boldsymbol{R} \leftarrow StandardCPA(Plt_0, Tr_0)$
end for
while $ii \leq \varepsilon$ **do** ▷ Taking POIs
  $ii = ii + 1$
  $\bar{S}[ii] = s_{max} \leftarrow argmax(\boldsymbol{R})$ ▷ s_{max}: the index of the column containing max value
  $s_{max} = 0$
end while
$Plt_1 =$ plaintext$^{n\times 16}$
$Tr_1 = trace^{n\times \bar{S}}$
for k from 0 to K **do** ▷ Phase 2
  $\boldsymbol{R} \leftarrow StandardCPA(Plt_1, Tr_1)$
end for
$\boldsymbol{k}[B] = line_{max} \leftarrow argmax(\boldsymbol{R})$ ▷ $line_{max}$: the index of the line containing max value

lenging in practice because of several reasons. Firstly, $\rho_{\max}$ is a random variable, and it is difficult to find the certain values of $\rho_{\max}$. Secondly, it is clear that LIs have different lengths corresponding to different values of n. Finally, there are so many correct samples that can be used to reveal the correct keys. Indeed, the authors in [20] have indicated that in their experiments, the intermediate result is used in several instructions. This is very typical for software implementations. Each time the micro-controller performs an operation that involves the attacked intermediate result, there is at least one peak in the matrix $\boldsymbol{R}$. Due to the reasons mentioned above, this proposal focuses only on finding the number of traces for the POI extractor. The theory of calculating exactly the number of POIs is out of the scope of this work. However, based on several practical attacks and simulations, the values of POI = 100 and POI = 250 are determined for $n/2$ and $n/3$, respectively. It can be enough to reduce n for POI extraction with a small number of POIs. This reasonable choice will be proven by the experiments in the next section.

2.5. Validation experiments

To prove the efficiency of proposals on different SCA contexts, this chapter investigates the performance of SCA attacks on the power consumption data of 8-bit Sbox outputs. The other bit-length Sbox could be performed similarly. Concretely, two power consumption datasets of the AES-128 implemented on two platforms, including the RISC-V processor and public ASCAD database, were used. All experiments were performed with MATLAB software on a personal computer with an Intel Core i5-9500 CPU and DDR4 24GB memory. In the experiments, the

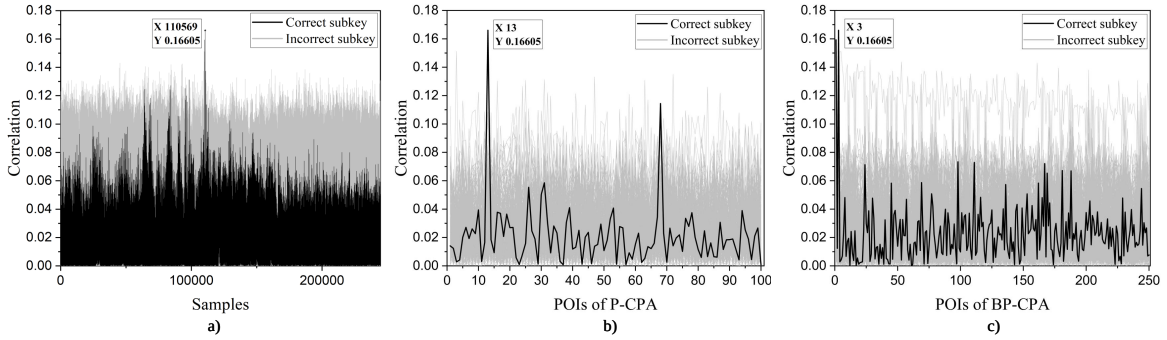

Figure 2.7: Attack results of standard CPA, P-CPA and BP-CPA methods on masking countermeasure: a) Standard CPA; b) P-CPA; c) BP-CPA.

average success rate and computation time are used as the metrics to evaluate the efficiency of the proposed methods.

2.5.1. Attack on masking data

As described in Section 1.2.3, the ASCAD data was collected from the software AES-128 implemented by the masking countermeasure with a different masking value for each bye of the secret key. This technique leads to increased computational complexity. In these experiments, the standard 2-order CPA (Std-CPA) is performed to determine the number of power traces needed for the attack. P-CPA and BP-CPA techniques are then performed to investigate the efficiency in terms of the computation time. In the pre-processing state of Std-CPA, a pre-processed trace which contains all values $|l_a - l_b| \, \forall l_a, l_b \in l$ is calculated. As a result, the new dataset with 244,650 samples on each power trace is formed. Next, the Std-CPA is implemented on the pre-processing traces. As depicted in Fig. 2.7(a), the Std-CPA was implemented successfully with 1200 ASCAD traces. The highest correlation is 0.16605, and the execution time of Std-CPA is 1188.4 seconds. Then, these traces are used to implement P-CPA and BP-CPA. Fig. 2.7.(b) shows that P-CPA can reveal the correct key on power traces that are used by Std-CPA. Especially, P-CPA

allows taking exactly the same sample, which has the highest correlation as Std-CPA ($\rho = 0.16605$). For BP-CPA, it is clear that this method works better than P-CPA and Std-CPA. BP-CPA only used 350 power traces for phase 1 (nearly 1/3 of total power traces needed in Std-CPA). In addition, the POIs of BP-CPA are taken accurately, and the position of the correct sample ($X = 3$) is lower than P-CPA ($X = 13$), as shown in Fig. 2.7. It means that the power trace biasing technique makes the values of the correct key higher and leads to a higher probability of a successful attack. Since the partial correlation is used, the computation time values of P-CPA and BP-CPA decline considerably from $1,188.4$ seconds to 583.54 seconds and 446.94 seconds, respectively. These results have clarified the efficiency of our proposed methods for masking protected devices.

2.5.2. Attack on noise injection data

This experiment aims to evaluate the proposed technique in different contexts of hiding countermeasures. The RISC-V power traces are selected and then added by the Gaussian noise centered in zero with several values of standard deviation to simulate different levels of hiding countermeasure, as described in formula 1.4. As same as the previous experiment, CPA attacks are performed by using three techniques of Std-CPA, P-CPA, and BP-CPA to compare the efficiency in the computation time. Table 2.1 presents the details of the parameters for each experiment in this work. The attacks are mounted and repeated in each case of noise 100 times, in which the power traces of each attack are randomly taken from 10,000. The results of the experiments are then averaged. As depicted in Fig. 2.8.(a), the computation times of three techniques on

Table 2.1: The parameter of traces for our experiments on noise added RISC-V power traces.

$\sigma(noise)$	0.004	0.006	0.008	0.01
n(Std-CPA)	400	800	1000	1600
n'(P-CPA)	200	400	500	800
n'(BP-CPA)	≈ 115.6	≈ 231.2	≈ 289	≈ 462.4

each level of noise are different. It is clear that the average computation time of Std-CPA is the highest in all experiments. The results indicate that the computation time values of P-CPA and BP-CPA are the 2^{nd} lowest and the lowest ones, respectively. These results demonstrate that BP-CPA is the most effective in terms of computation time, and it can reduce approximately two times compared with Std-CPA. However, it is worth noting that the time consumption values of P-CPA and BP-CPA methods when $\sigma = 0.004$ are nearly the same, and the reduction of the computation time is 1.5 times. This is an unexpected result because the number of traces for the attack is small, and phase 1 of BP-CPA needs some extra computation time to filter the high SNR power traces.

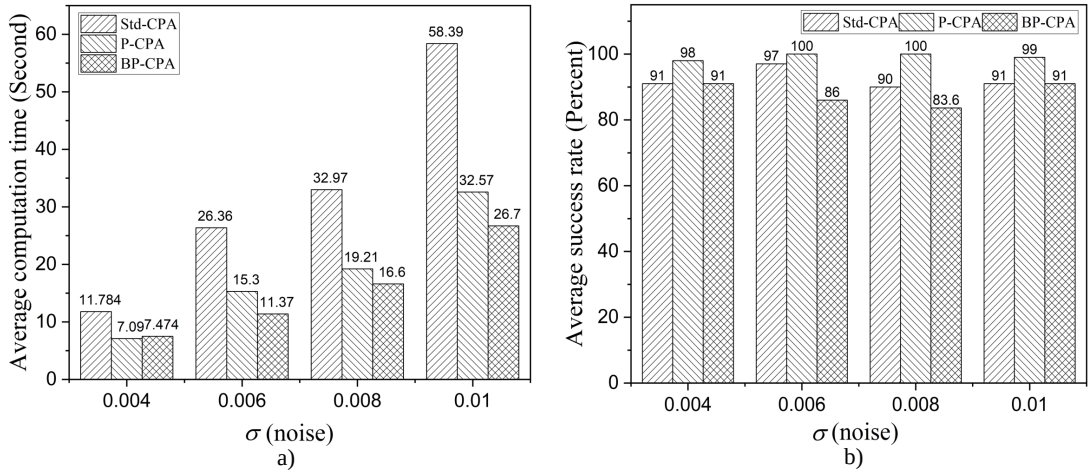

Figure 2.8: Average computation time and success rate of one subkey on noise added RISC-V power traces. a) Average computation time; b) Average success rate.

Comparing the reliability of proposed techniques, the averages of successful attacks is taken. As illustrated in Fig. 2.8.(b), P-CPA achieves the highest success rate, and the success rate of BP-CPA is lower than Std-CPA in cases of $\sigma = 0.006$ and $\sigma = 0.008$. However, in general, the results of the BP-CPA success rate are acceptable.

2.5.3. Combined hiding and masking

Next, the combination of hiding and masking techniques is considered. To simulate this scenario, the original ASCAD database is added with different levels of Gaussian noise as described in Section 1.2.3. In TCHES-2019, Timon demonstrated that DDLA could break masked cryptographic devices without any advance knowledge about the masking implementation [7]. In contrast, a second-order CPA attack needs to be pre-processed and performed in a squared computation complexity compared to the 1-order CPA. However, the standard second-order CPA can overcome the noise-generation-based hiding countermeasure. Therefore, in this experiment, the attacks are performed to compare the possibility of DDLA and CPA on a combined countermeasure. The same model of the MLP network on TCHES-2019 is reconstructed with 30,000 power traces as the original work [7]. In terms of CPA, we use a maximum 2,700 and minimum 1,200 power traces for the highest and the lowest levels of noise, respectively. It is noted that the original traces in each attack are the same, but the Gaussian noise is re-initialized and added on each implementation. Table 2.2 shows the parameter of our experiments in detail. For convenience, we choose the most effective proposed technique BP-CPA for this experiment. The attack results are shown in Fig. 2.9. As shown clearly in Fig. 2.9.(a, b, c), it is difficult to

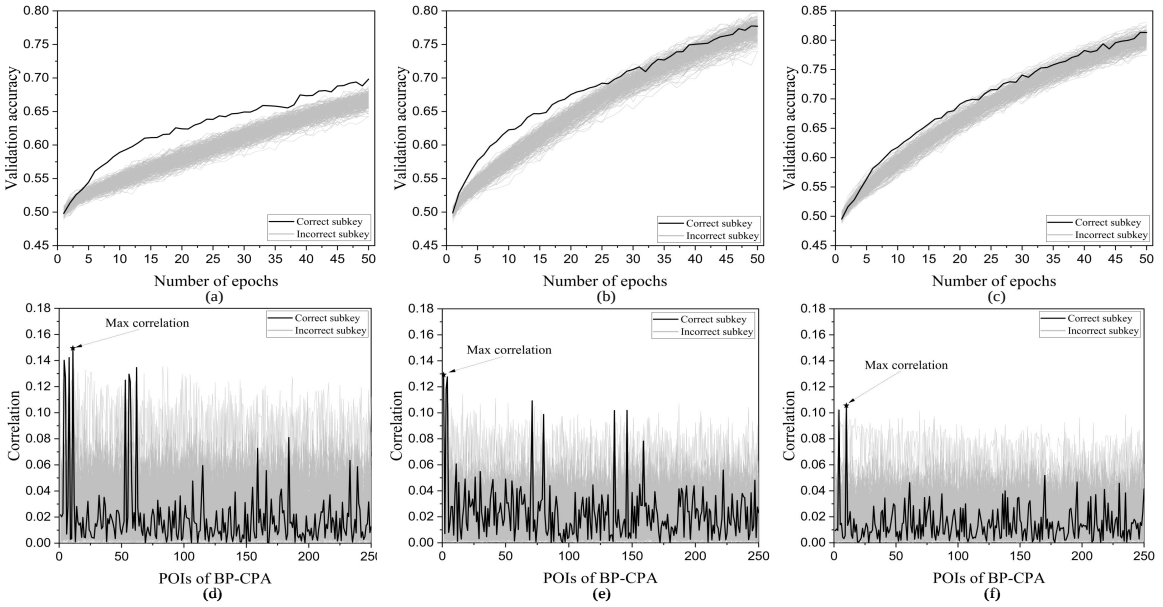

Figure 2.9: The experimental results on differential levels of noise added ASCAD database. Left column) $\sigma = 0.5$; Center column) $\sigma = 1.0$; Right column) $\sigma = 1.5$; a, b, c) DDLA attack; d, e, f) BP-CPA attack.

discriminate the correct subkey by DDLA when $\sigma = 1.0$ and it can not distinguish the correct subkey in the case of $\sigma = 1.5$.

In contrast, the proposed technique can reveal the correct subkey in all cases. The black lines in Fig. 2.9.(d, e, f) present the correlation of the correct subkey. It is worth noting that the correct samples which have the highest correlation are taken at the beginning of the POI axis. It means that the power trace biasing technique makes the correlation values of candidate samples higher than the rest. In addition, our algorithm takes the POI following the top-down strategy. Therefore, we can reduce the number of POIs to a value of less than 250. More interestingly, the maximum correlation value decreases when the number of traces increases, as explained in (2.11). Despite having the highest values, the black lines are not clearly distinguished from the rest (gray lines). This is because we take the minimum number of traces for successful attacks. These experimental results have indicated that the proposed

method could outperform the non-profiled DL-based attack (DDLA) on hiding-masking protected devices.

Table 2.2: The parameter of traces for the experiments on the noise added ASCAD database.

$\sigma(noise)$	0.5	1.0	1.5
n(Std-CPA)	1500	2200	2700
n'(BP-CPA)	≈ 433.5	≈ 635.8	≈ 780.3
TCHES-2019 [7]	30,000	30,000	30,000

After proving the efficiency of our proposed method, we decided to perform further experiments to evaluate the execution time of DDLA, Std-CPA, and our proposed techniques. To achieve reliable results, we repeat 50 times the experiment of CPA in Section IV.c. The computation time is then averaged and presented in Fig. 2.10.(a). The DDLA columns show the execution time of the DDLA technique. They are almost the same because we fixed the number of traces for all different levels of noise. The Std-CPA columns illustrate the computation time of Std-CPA. We increase the number of traces corresponding to the increase of the deviation of the noise. As a result, the execution time of Std-CPA increases significantly, and it reaches the highest value when $\sigma = 1.5$. In addition, the execution time of Std-CPA is higher than DDLA in all cases. These results indicate that the drawback of Std-CPA is time-consuming. Fortunately, the goal of this work is to resolve the limitation of Std-CPA. As expected, the BP-CPA columns present the time consumption of BP-CPA, which is lower than those of both DDLA and Std-CPA. Especially, the execution time of BP-CPA was reduced approximately by 2.6 times compared to Std-CPA in all cases. The results also indicate that our proposed technique outperforms DDLA

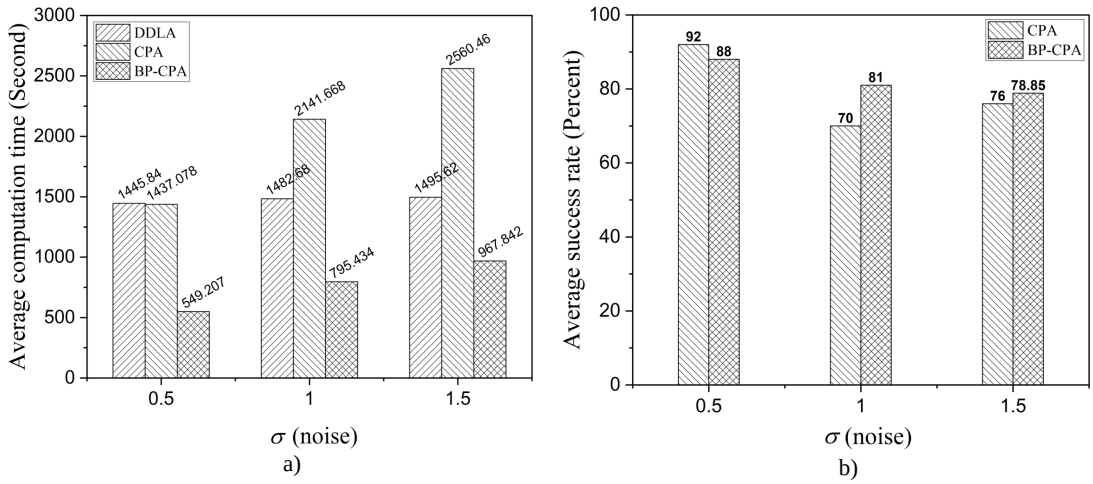

Figure 2.10: Average of computation time and success rate on different levels of Gaussian noise added ASCAD: a) Average of computation time; b) Average of success rate.

(967.842 seconds compared to 1495.62 seconds in the case of $\sigma = 1.5$) even though DDLA uses a huge number of power traces (2700 compared to 30000 power traces).

Considering the reliability, the comparison of the success rate between Std-CPA and BP-CPA is shown in Fig. 2.10.(b). In terms of the low noise level, the success rate of the proposal is slightly lower than Std-CPA. This is the limitation of the proposed technique. This chapter uses power trace biasing to increase the standard deviation of $t_d(\tau)$ on $n' = n/3$ power traces. However, the attack phase is still based on standard CPA. Therefore, the success rate of the proposed technique is less than or the same as Std-CPA in such cases. In contrast, the proposal has better results than Std-CPA in the remaining cases. From (2.14), it clearly shows that the higher the noise, the smaller the SNR value. Therefore, Std-CPA needs to use more power traces in order to discriminate the correct samples as obtained from (2.5). By using the POI extractor, the proposed technique has eliminated most of the noise-affected samples

that are not related to the correct key or the operation. Consequently, the proposed technique increases the probability of a successful attack in comparison to Std-CPA: 81% compared to 70% in the case of $\sigma = 1$ and 78.85% compared to 76% in the case of $\sigma = 1.5$.

2.6. More discussion

2.6.1. Use cases

Security testing on noise injection countermeasure

a) Scenario description

- Use case 1: To find out the level of noise injection against a given number of power traces. Considering level 3 in ISO/IEC 17825:2016, the DUT needs to pass the security checking with 10,000 power traces. In this case, the evaluator needs to try to modify the level of injected noise. The evaluator will perform the security testing using CPA and the proposed BP-CPA techniques.

- Use case 2: To find out the number of power traces that break the DUT with a given level of injection noise (σ).
  In this case, the level of additive noise is fixed. The evaluator needs to determine the number of required power traces that can break the countermeasure.

To simulate noise injection countermeasure in this case study, different levels of Gaussian noise were added to each sample of power traces as described in Section 1.2.3.

b) Results

- Use case 1: In the first scenario, the evaluator is required to perform many CPA attacks with N= 10000 power traces corresponding to

Table 2.3: The execution time of evaluation process

Level of noise (σ)	0.03	0.032	0.034	0.036	0.038	0.04	**Total time**
Attack time CPA (Second)	276.495	275.503	273.260	273.655	267.801	277.294	**1644.008**
Attack time BP-CPA (Second)	127.080	128.419	125.714	126.524	127.2731	170.7751	**805.785**
Successful attack	Yes	Yes	Yes	Yes	Yes	No	

Table 2.4: The execution time of evaluation process

Number of power traces	6000	7000	8000	9000	10000	**Total time**
Attack time CPA (Second)	149.439	177.544	219.383	236.648	275.431	**1058.445**
Attack time BP-CPA (Second)	74.065	82.837	103.067	110.233	126.912	**497.114**
Successful attack	No	No	No	No	Yes	

different levels of additive noise. The time process is recorded and shown in Table 2.3. It can be seen that both CPA and BP-CPA attacks provide the same result of a successful attack. However, in terms of execution time, the evaluator spends 1644.008 seconds (approximately 27 minutes) to find out the final results. Interestingly, by using BP-CPA, the execution time of the evaluation process reduces significantly to 805.785 seconds (approximately 13 minutes). This result has demonstrated that the proposed technique works better in the SCA evaluation process.

- Use case 2: Based on the procedure of attack-based testing described in Section 1.4.2. The evaluator must try a different number of power traces to determine the vulnerability of DUT. Unlike the validation experiments with N already known, he starts with N= 6000, and each step increase to 1000 power trace until he achieves a successful attack. The results of security testing with $\sigma = 0.036$ are presented in Table 2.4.

Table 2.5: The execution time of evaluation process

Number of power traces	2000	2500	3000	3500	**Total time**
2-order CPA (second)	3270.3	2722.9	2224.5	1701.6	**9919.406**
2-order BP-CPA (second)	1226.771	1013.073	831.0303	658.6923	**3729.566**
Successful attack	No	No	No	Yes	

Security testing on combined noise generation and boolean masking countermeasure

a) Scenario description

In this part, security testing on a DUT-equipped boolean masking countermeasure is presented. For convenience, ASCAD data is selected to simulate this scenario. In addition, each sample of power trace is added Gaussian noise as described in formula (1.4). The evaluator is required to perform second-order CPA attacks to find out the number of power traces, which can break the DUT countermeasure with $\sigma = 1.8$.

b) Results

In this case, various second-order attacks are performed. Similar to the previous scenario, CPA and BP-CPA attacks are selected to implement security tests. To perform second-order attacks, the new traces were reconstructed from original ASCAD data (700 samples/trace). The number of samples of new traces is 244650. For convenience, the execution time of pre-processing phase of second-order attacks is discarded. The results of the processes are presented in Table 2.5. The total execution time of 2-order CPA is 9919.406 seconds (approximately 2 hours 45 minutes), whereas 2-order BP-CPA is 3729.566 seconds (approximately 1 hour).

These results have demonstrated the efficiency of proposed techniques in SCA security testing.

2.6.2. The number of POI (ε)

As mentioned in the previous part, it is difficult to determine the right value of ε in P-CPA or BP-CPA. It is usually selected based on the results of real attacks. However, the value of ε should be initiated by the following criteria:

- ε is defined to ensure that it is a small number compared to the number of samples (l) on a power trace.
- As ε decreases, the probability of obtaining correct samples decreases, resulting in a lower success rate and faster attack time.
- As ε increases, the probability of obtaining correct samples increases, leading to a higher success rate and longer attack time.

2.6.3. More comparisons to DDLA

The results presented in the previous section indicate that the DDLA attacks face to "overfitting" problem. This issue leads to the phenomenon which allows the attacker can detect the correct key in the early epochs. In fact, it is impossible to know these early epochs prior to performing the actual attacks. However, to provide more comparisons between BP-CPA and DDLA, this part assumes that the early epochs are known.

Taking Fig. 2.9.c as an example, the correct key can be taken from epoch 10 to epoch 15. However, the accuracy of the correct key is quite close to the incorrect one. It means that the correct key may not be distinguished in some cases. To investigate this problem, additional experiments were performed to evaluate the success rate of attacks in early epochs. The results are presented in Table 2.6. By reducing the number

Table 2.6: The results of other attacks using DDLA with selected number of epochs

Attack method	No. of traces used	Noise (σ)	No. of epochs	Success rate (%)	Average time (second)	Repeated
DDLA	30,000	1.5	15	74	1158	50
DDLA	15,000	1.5	15	8%	561	50
BP-CPA	**2700**	**1.5**	**-**	**78.85**	**967.6**	**50**

of epochs, the performance of DDLA increases significantly. Especially, SR of DDLA attacks with 15 epochs is 74%. The attack time is reduced to 1158 seconds.

Compared to BP-CPA, the performance of DDLA (15 epochs) is nearly the same. However, it is noted that the number of power traces used in DDLA is many folds compared to BP-CPA (30,000 compared to 2700). To demonstrate this, the number of traces is reduced to 15,000, and the DDLA attacks are conducted again. The result was not unexpected. The SR of DDLA attacks reduces dramatically (from 74% to 8%) in this case. These results clarify that DDLA attacks require a huge number of traces compared to BP-CPA. In addition, different early epochs need to choose to correspond to different attacks.

In summary, the proposed technique outperforms DDLA regarding the attack time and the number of traces required. In the case of attacking early epochs, DDLA attacks provide performance nearly the same as BP-CPA.

2.6.4. Disadvantage and resistance against BP-CPA attacks

Despite the efficiency of the proposed technique, these proposals still contain several limitations as shown below:

Firstly, in the case of masking countermeasure, the number of traces for the attack is limited because of the complexity of second-order leakage data processing. In addition, this process requires high memory usage.

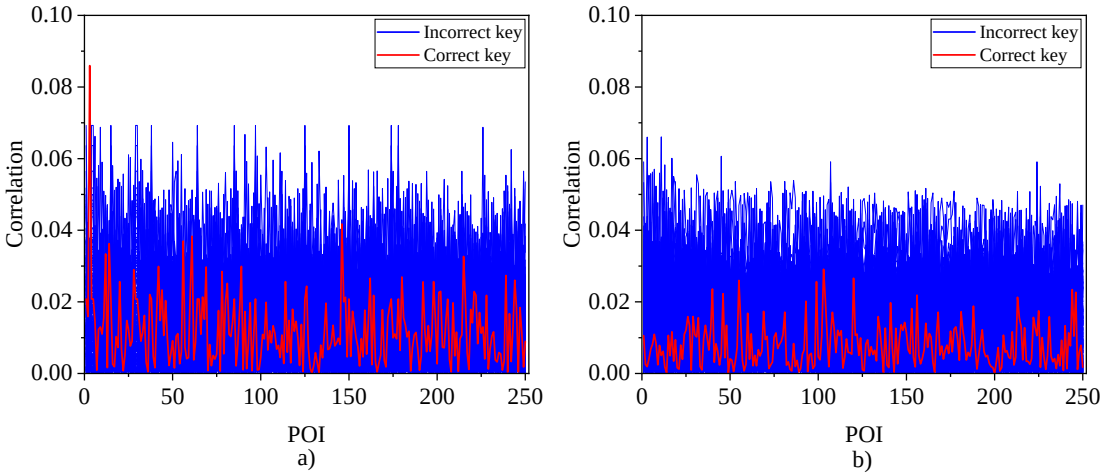

Figure 2.11: Experimental results of BP-CPA attack on de-synchronized countermeasure. a) Shifted value = 1; b) Shifted value = 5.

Therefore, in the case of a high number of second-order power traces, it is difficult to apply pre-processing techniques.

Secondly, other power consumption models have not been investigated. In this case, only HW model is investigated and applied to biasing power trace techniques. In means that the efficiency of proposed techniques has been clarified on software implementations of cryptographic algorithms. To attack hardware based designs of cryptographic algorithms, HD model needs to be considered. This is also the future work of this study.

Thirdly, the number of POI is determined manually. As stated previously, the number of POI is selected based on the practical attack results. In some cases, the number of POI is too small to cover the correct sample t_{ct}. Consequently, it leads to a failed attack. In other cases, the number of POI is too large. Hence, the attack time is not optimized. This issue is also investigated in the future work of this study.

Finally, in the case of hiding countermeasures, only the noise injection technique is investigated. Furthermore, the random delay will cause a

misaligned problem. Therefore, the POI extractor will not work correctly because it requires each operation of the cryptographic algorithm should be located at the same position in each power trace to find out the correlation. This observation is the motivation for the designer of cryptographic devices to randomize the execution of the cryptographic algorithm, i.e., the devices perform the operations of the algorithms at different moments of time during each execution.

To clarify this assumption, an additional experiment is performed on simulated de-synchronized power traces. These power traces are reconstructed by randomly shifting each power trace of RISC-V data in a maximum of 1 and 5 samples (denoted as data-sh1 and data-sh5, respectively). The attack results are depicted in Fig. 2.11. Obviously, the attack results of BP-CPA on data-sh1 show that the correct key can be revealed easily, whereas the attack results on data-sh5 show that the red curve of the correct key is very low compared to incorrect keys. The secret key is not discriminate anymore in this case. These results have demonstrated that the BP-CPA can not break de-synchronized countermeasure directly. In addition, de-synchronized is not difficult to integrate into a hardware design as well as software design. Therefore, this is also a good solution to counteract BP-CPA.

2.7. Summary

In this chapter, the complexity of the CPA attack on the side-channel data using different SCA countermeasures has been presented. To increase the performance of SCA security testing, this chapter introduces two new techniques called P-CPA and BP-CPA based on the sampling distribution of correlation coefficient and the biasing technique. Perfor-

mance comparisons between the proposed techniques and the conventional CPA are provided using various experimental results, especially in the case of protected datasets. In particular, P-CPA and BP-CPA reduce the computation time by approximately 2 and 2,6 times compared to standard CPA, respectively. Additionally, in the case of combining masking and noise generation countermeasures, BP-CPA outperforms both standard CPA and DDLA in terms of execution time. The success rate of the proposed techniques is also increased compared to conventional CPA. However, BP-CPA can not break de-synchronized countermeasure. This is also the solution to counteract BP-CPA on both hardware and software implementations of cryptographic devices. The results of this chapter are published in [C1] and [J1].

Chapter 3
DIMENSIONALITY REDUCTION AND LABELING METHODS FOR EFFICIENT DEEP LEARNING BASED NON-PROFILED SCA

In the non-profiled context, statistic-based attack such as CPA has been demonstrated to work well on breaking different SCA resistances. CPA has also been applied to various works to verify the efficiency of the deployed countermeasures. However, it requires a specific pre-processing for each type of countermeasure. It leads to a high-cost and time-consuming evaluation process. First introduced in 2018, DDLA can perform the attack successfully without any pre-processing techniques in the non-profiled context [7]. DDLA is the most used SCA method using deep learning in research papers.

Despite being effective on both unprotected and protected SCA data, DDLA still faces many issues, such as high-dimension data input, labeling techniques, or other common countermeasure like noise generators. This chapter introduces the techniques to improve the DDLA attack and mitigate the mentioned issues. Concretely, a dimensionality reduction method and a labeling technique are introduced to deal with high-dimensional data input, imbalance dataset, and to reduce the impact of additive noise. Simultaneously, different models based on popular architectures, such as MLP and CNN, are developed to apply the proposed techniques. The chapter's results are published in the papers [C2, C3], [J3, J5], and P[1]

3.1. Reducing data dimension using P-CPA

As indicated by Timon [7], the weights of the leakage sample directly impact the loss, as it is the sample that carries the useful information for the classification. In addition, there are a few leakage samples in a power trace related to the processed data and the secret key in the executions of the cryptographic algorithm. Therefore, high-dimensional data input will make the DL model spend more time optimizing the weights corresponding to the leakage features. In addition, one drawback of DDLA is that it is necessary to perform a DL training for each key guess (i.e., 256 training times in the case of AES-128) [7], the higher the data dimension, the more complex the network architecture.

Several techniques were proposed for improving the performance of non-profiled attacks by extracting the POI, as discussed in chapter 1. However, most of the previous works focused on profiled attacks. In terms of non-profiled scenarios, the authors in [89] exploited pattern recognition methods to filter interesting points of power trace for obtaining a successful attack. Alipour *et al.* [21] used a simple method in order to reduce the sampling points of each power trace by 50%. Accordingly, from each two neighboring sample points, the first one is kept, while the second is omitted.

In the previous chapter, the P-CPA technique is introduced and used for taking the most relevant samples in the power trace by computing the correlation between real traces with their model. Additionally, P-CPA requires only 50% of the given power traces for detecting the POI. Therefore, this method is suitable for power traces containing a large number of samples. Based on the advantages of P-CPA, this method

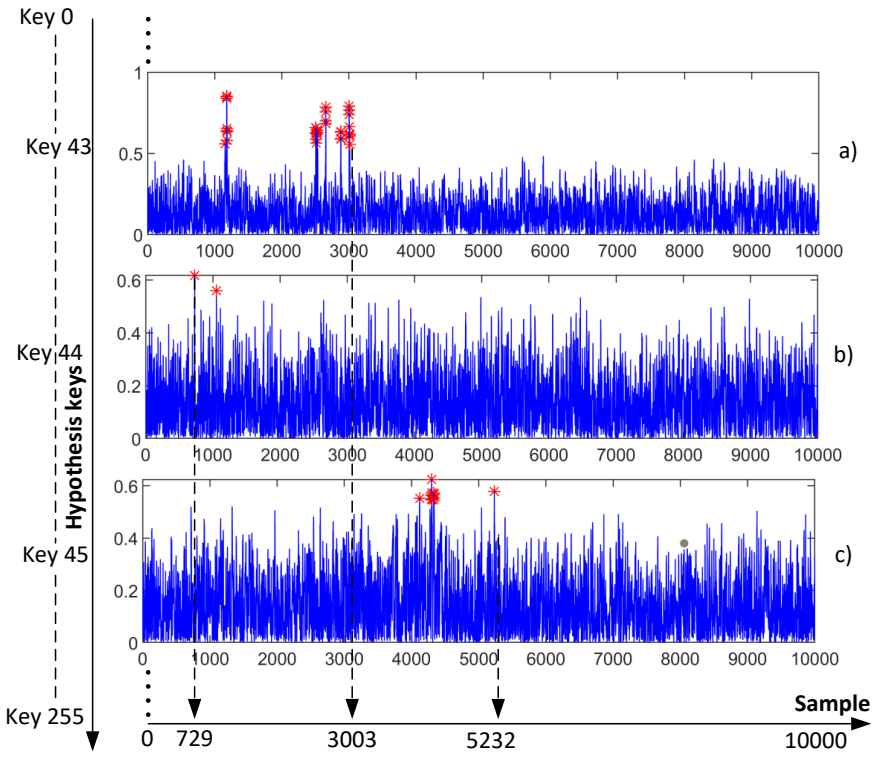

Figure 3.1: The positions on sample axis corresponding highest correlation values on all hypothesis keys will be taken (The red markers).

is used to reduce the number of features of data input in DDLA. To illustrate the dimensionality reduction method based on P-CPA, an intuitive example on the CW platform is shown in Fig. 3.1. Accordingly, for determining the positions of high correlation on a dataset containing 5,000 power traces of 10,000 samples each. Firstly, the correlation value of the real power trace with the HW model is calculated on 2,500 power traces. As a result, a matrix of correlation coefficients with a size of 256 $\times$ 10,000 is produced, in which each row corresponds to a hypothesis key. As depicted in Fig. 3.1, the correlation values of three rows (k= 43, 44, 45) in the correlation matrix are plotted. Next, from which 50 positions with the 50-top highest correlation values are located (red star points). Then, 50 relevant sample points of all power traces are extracted. Con-

sequently, the size of the new dataset can be reduced 200-fold compared to that of the original one. It can be explained as the correlation-based extraction can reduce the size of each power trace from 10,000 to 50 samples.

By determining 50 useful sample points based on the 50-top highest correlation values, a smaller dataset of power traces is generated and reconstructed following hypothesis keys and HW values, in which HWs play the role of labels. However, this method is only used in the case of first-order leakage or the pre-processed data of second-order power traces.

3.2. Significant HW Labeling

In terms of labeling techniques, two typical labeling techniques, including Hamming weight and Binary, have been applied in the DL-based non-profiled SCA [7]. The efficiency of DDLA using the Binary labeling method is proven in many works [7, 21–23]. Especially by using Binary Neural Network (BNN) with LSB/MSB labeling techniques on picture-formatted data, the authors in [22] show that the validation accuracy is significantly higher, and the number of learning epochs required to obtain the secret key can be reduced.

In contrast, no report on using the HW labeling method is published in the non-profiled deep learning context. Additionally, the authors in [22] have also indicated that the HW model causes the imbalance dataset problem in the non-profiled SCA scenario. Indeed, by observing Fig. 2.5, it is obvious that the distribution of intermediate values on each HW is imbalanced and symmetric about $HW4$ in the case of AES-128. To address this problem, the authors in [90] proposed a method based on

the data re-sampling technique. Accordingly, they used a random over-sampling method, the so-called SMOTE, to oversample for each class. In practice, SMOTE can be considered as a general case of the data augmentation (AU) technique, which is proposed in [91].

To mitigate the imbalanced dataset issue in non-profiled context, this section introduces a simple labeling technique based on Hamming weight. Unlike profiled DL-based SCA, DDLA uses the training metrics instead of the model's output for discriminating the correct key. Accordingly, training with the correct key always has better learning ability than incorrect ones. It means that if we use only three classes for training instead of nine classes, the DL model using the correct key still has better training metrics than the wrong key. As shown in Fig. 2.5, there are three significant HWs (HW = 3, 4, 5) that contain the most distribution of intermediate values. Moreover, the distribution of three significant HWs (SHW) is nearly balanced. Therefore, SHW is considered to use for classification in the non-profiled context.

Apart from balancing data, SHW labeling reduces significant measurements needed for the training process. Indeed, as depicted in Fig 2.5.a, it is clearly shown that SHW discards the intermediate values corresponding to $HW = 0, 1, 2, 6, 7, 8$ (about 30% of power traces). It is meaningful in the case of reducing the execution time in the non-profiled context. This property will be illustrated in the experimental results section.

3.3. Dataset reconstruction

For investigating the efficiency of the proposed techniques in this chapter, different datasets are reconstructed from the original ASCAD, RISC-V, and CW data as presented in Table 3.1. The character "x" in the

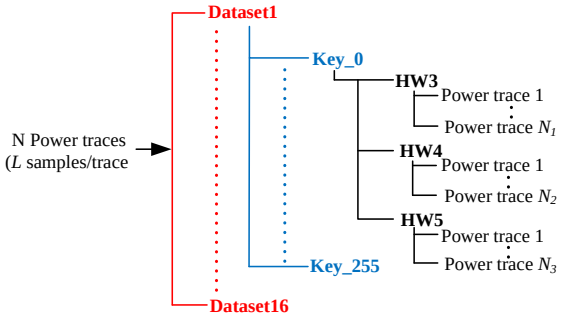

Figure 3.2: Structure of the new datasets: There are 16 folders (**Dataset1 to Dataset16**) corresponding to 16 bytes of secret key, each folder contains 256 files in **.csv** format which correspond to 256 hypothesis keys. N original power traces (L samples/trace) are calculated to form N_1 new traces and labeled ($HW = \{3, 4, 5\}$). Each new trace contains 50 samples which are highest correlation values.

name of each dataset indicates that it is an original or reconstructed one. Then, each dataset is described in a row for its properties. Dataset2x, for example, contains Dataset2O and Dataset2R. Dataset2O consists of 20,000 unmasked ASCAD power traces; each power trace has 700 samples. Dataset2R is reconstructed from Dataset2O, using P-CPA with $\varepsilon = 50$ to reduce the data dimension from 700 to 50. The dataset is labeled by the SHW labeling technique. For other cells using the symbol "-", the corresponding part is not applied. It is noted that the number of traces needed for creating Dataset2R is the same in Dataset2O. However, the number of traces of Dataset2R for the training process is reduced by approximately 30% in the case of SHW applied. It is true for all other datasets in this chapter.

To simulate de-synchronized countermeasure, $\text{Dataset3}_{\text{sh1}}$ and $\text{Dataset3}_{\text{sh5}}$ are generated from the original RISC-V data. Each power trace in the datasets is randomly shifted in a maximum of 1 and 5 samples, respectively. Regarding noise generation countermeasure, three different datasets corresponding to three different levels of Gaussian noise were

Table 3.1: The details of reconstructed datasets using different data.

Dataset	Original (O)			Reconstructed (R)				
	Data	No. of traces used	Dim	No. of traces for training	Dim	Labeling technique	Shifted values	RD method (ε)
Dataset1x	$\text{ASCAD}_{\text{unmask}}$	20,000	700	20,000	700	LSB	-	-
Dataset2x	$\text{ASCAD}_{\text{unmask}}$	20,000	700	≈7000	50	SHW	-	P-CPA (50)
Dataset3x	RISC-V	10,000	480	≈7000	480	SHW	-	-
$\text{Dataset3}_{\text{sh1x}}$	RISC-V	10,000	480	≈7000	480	SHW	1	-
$\text{Dataset3}_{\text{sh5x}}$	RISC-V	10,000	480	≈7000	480	SHW	5	-
Dataset4x	CW	5,000	10,000	≈3000	50	SHW	-	P-CPA (50)
$\text{Dataset4}_{\text{9-HWx}}$	CW	5,000	10,000	5000	50	9-HW	-	P-CPA (50)
$\text{ASCAD}_{\text{LSBx}}$	ASCAD	20,000	700	20,000	700	LSB	-	-
$\text{ASCAD}_{\text{SHWx}}$	ASCAD	20,000	700	≈14,000	700	SHW	-	-

x: O/R; RD: Reducing data dimension; -: Not used

employed on CW data as described in Table 3.2. Finally, by increasing the size of the attack dataset, we investigate the ability of DLSCA to break the noise generator if power traces are collected enough. To simulate this scenario, we chose $\sigma = 0.025$ and $\sigma = 0.055$ to form three new datasets from CW data as presented in Table 3.3.

For all reconstructed datasets, 16 folders corresponding to 16 sub-bytes of a secret key are obtained. Each folder contains 256 sub-folders corresponding to 256 values of the potential guessed key. In such a sub-folder, three folders named HW3, HW4, and HW5 are used for three labels of the CNN. Finally, for each label folder corresponding to the intermediate value ($HW = 3, 4, 5$), the power traces were partitioned as illustrated in Fig. 3.2. Each dataset is divided into two parts of training and validating data corresponding to 80% and 20% of the created dataset, respectively.

Table 3.2: The details of reconstructed datasets from noise-added CW power traces.

Dataset	Original (O)		Reconstructed (R)				
	No. of traces used	Dim	No. of traces for training	Dim	Labeling technique	Std of noise (σ)	RD method (ε)
$Dataset4_{1x}$	5000	10,000	≈3000	50	SHW	0.025	P-CPA (50)
$Dataset4_{2x}$	5000	10,000	≈3000	50	SHW	0.05	P-CPA (50)
$Dataset4_{3x}$	5000	10,000	≈3000	50	SHW	0.075	P-CPA (50)

x: O/R; RD: Reducing data dimension; -: Not used

Table 3.3: The details of reconstructed CW datasets for evaluating the noise generation based hiding countermeasure.

Dataset	Original (O)		Reconstructed (R)				
	No. of traces used	Dim	No. of traces for training	Dim	Labeling technique	Std of noise (σ)	RD method (ε)
$Dataset5_{1x}$	3000	10,000	≈2100	50	SHW	0.025	P-CPA (50)
$Dataset5_{2x}$	3000	10,000	≈2100	50	SHW	0.055	P-CPA (50)
$Dataset5_{3x}$	4000	10,000	≈2800	50	SHW	0.055	P-CPA (50)
$Dataset5_{4x}$	5000	10,000	≈3500	50	SHW	0.055	P-CPA (50)

x: O/R; RD: Reducing data dimension; -: Not used

3.4. Non-profiled DLSCA using significant HW labeling

While the original DDLA proposal [7] uses CNN and MLP as a building block, one can use advanced DL techniques like recurrent neural networks (RNN) or long short-term memory (LSTM), especially in the case of sequence-based data like SCA data. This chapter, however, continues to use MLP and CNN architecture for two distinct reasons. Firstly, RNN and LSTM are not currently well-studied for SCA use cases. Secondly, the wide variety of results available for the use of CNN or MLP with SCA datasets helps us to benchmark our results. Therefore, two new instances of MLP and CNN architectures are introduced in this section.

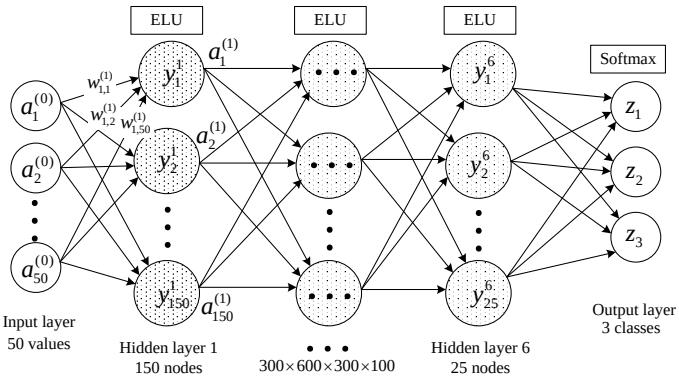

Figure 3.3: The proposed Multi-layer perceptron architecture.

3.4.1. MLP_{SHW}

The proposed MLP network comprises an input layer, output layers, and six hidden layers. The number of nodes in the input layer is assigned according to the number of samples in a power trace. As depicted in Figure 3.3, all arrows represent the weights. Prior to the implementation training phase, the values of weights and bias are randomly chosen from a normal distribution using the Xavier scheme.

As explained in Section1.3.3, a procedure called *forward propagation* is performed. In DL-based SCA, the popular activation functions used in hidden layers are ELU and RELU, which are computed as formula (3.1) and (3.2), respectively. Our proposed model used ELU instead of ReLU to avoid the vanishing problem and produce negative outputs for each node in the hidden layer.

$$\text{ReLU} : F_{(y)} = \left\{ \begin{array}{ll} y: & y > 0 \\ 0: & y \leq 0 \end{array} \right\} \tag{3.1}$$

$$\text{ELU} : F_{(y)} = \left\{ \begin{array}{ll} y: & y > 0 \\ \phi \cdot (e^y - 1): & y \leq 0 \end{array} \right\} \tag{3.2}$$

For classification, the Softmax function is used in the output layer for

calculating the probability of each HW label. This function is calculated as

$$\text{SoftMax}: z\left(y\right)\left[i\right] = \frac{e^{y[i]}}{\sum_{j=1}^{C} e^{y[j]}} \tag{3.3}$$

where C is number of classes, in our case, $C = 3$ since our proposed model uses HW label.

Finally, *backward propagation* is implemented in order to update the weights to obtain the expected results. Since we have three labels, the categorical cross-entropy loss between the ground-truth and prediction labels are computed as follows:

$$\mathcal{L}_X(\boldsymbol{w}) = -\sum_{j=1}^{3} y_{true} \ln\left(z\right) \tag{3.4}$$

where y_{true} is the grouth-true values of HW classes.

Then, we use stochastic gradient descent with momentum (SGDM) optimizer to find the optimal minimizing of the loss function. Deep learning will do a series of iterations t, and in each iteration, the gradient of loss function $\nabla\mathcal{L}_X\left(\boldsymbol{w}\right)$ is computed. After that, $\boldsymbol{w}$ is updated by using the formula as follows:

$$\begin{aligned} \boldsymbol{v_t} &= \gamma \boldsymbol{v_{t-1}} + \eta \nabla_w L_X\left(\boldsymbol{w}\right) \\ \boldsymbol{w^{t+1}} &= \boldsymbol{w^t} - \boldsymbol{v_t} \end{aligned} \tag{3.5}$$

where γ is the momentum value. In our case, γ is chosen equal 0.9, and the learning rate η is chosen equal 0.01.

When the correct hypothesis key k_{cr} is used, the series of intermediate results will be correctly computed. Consequently, the partition and the labels used for our model will be consistent with the corresponding

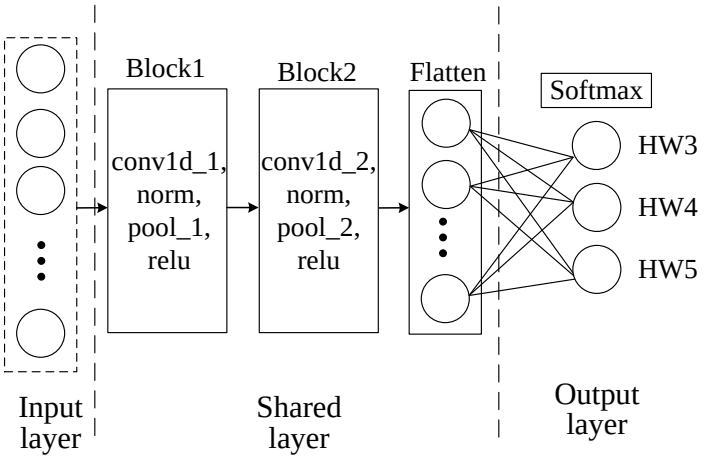

Figure 3.4: The proposed $\mathrm{CNN_{SHW}}$ architecture.

traces. In contrast, for all the incorrect guess keys, the labels used for the training will be incompatible with the traces. As a result, the training model with the dataset generated from k_{cr} provides better results with lower loss or higher accuracy than the other candidates. Therefore, the correct key can be obtained.

3.4.2. CNN_{SHW}

As demonstrated in [7], CNN is an efficient architecture to perform DDLA on de-synchronized power traces. This part introduces a new CNN model using the SHW label for the non-profiled SCA attack. The details of the proposed architecture are described as follows.

The proposed architecture is composed of an input layer and two 1-D convolutional (Conv1d) blocks in the middle, followed by the flattened layer and classification (output) layer. Each Conv1D block is formed by a Conv layer directly followed by a batch normalize layer and a pooling layer for selecting the informative and downsample the feature maps. The last layer of the Conv1D block is the activation layer, as described in Fig. 3.4.

Table 3.4: Hyperparameter of proposed CNN_{SHW} architecture

Model		CNN_{SHW}
Input size	RISC-V shifted	480
Convolutional layers (Activation)		Conv1d_1(N* 32×1 filters), Pool_1(2×1), Norm, Relu Conv1d_2(N* 16×1 filters), Pool_2(4×1), Norm, Relu
Number of filters (N)	Conv1d_1	4/8/16
	Conv1d_2	4/8/16
Output layer		3-Softmax
Batch size		200/500/1000
Learning rate		0.001

Similar to MLP, CNN_{SHW} performs *forward propagation* to calculate the output (3 nodes) from the input layer and *backward propagation* to update the learning metrics. However, unlike in MLP models where each neuron has a separate weight vector, neurons in CNN share weights. Neurons perform convolutions on the data, with the convolution filter being formed by the weights. In our case, two Conv1d layers corresponding to two blocks are designed with the same number of filters (4,8, and 16).

According to [7], on the first-order SCA data, DDLA only uses one main leakage area to classify the data. It means that the CNN model tries to find the most informative feature in the training phase. Therefore, the filter of the proposed model has the size of [32 1] for the first Conv1d block and [16 1] for the second block. The stride of [1 1] is used to extract the strongest features. The weighted sum of each filter is calculated as follows:

$$a_{1,j} = \sum_{n=0}^{f} w_{1,n} I_{1,j+n} + b \tag{3.6}$$

where I is the input data (or the output of the previous layer), b de-

notes the bias, f is the size of the filters, and $w_{1,n}$ stands for the element of filter (or convolutional kernel weights). Before feeding up to a non-linear activation function, the output of (3.6) is normalized. Finally, an FC layer, along with the Softmax function, plays a role as a classifier, which has three output classes corresponding to three HWs labels. In *backward propagation* stage, an optimizer is used to find the optimal parameters minimizing the loss function. In this case, the popular optimization algorithm called Adaptive Moment Estimation (ADAM) with default setting is employed to train the proposed model as follows:

(1) $m_t \leftarrow \beta_1 m_{t-1} + (1-\beta_1)\nabla\mathcal{L}_{total}(\theta_{t-1})$ (Update biased first moment estimate);

(2) $v_t \leftarrow \beta_2 v_{t-1} + (1-\beta_2)\nabla\mathcal{L}^2{}_{total}(\theta_{t-1})$ (Update biased second raw moment estimate);

(3) $\hat{m}_t \leftarrow m_t/(1-\beta_1^t)$ (Compute bias-corrected first moment estimate);

(4) $\hat{v}_t \leftarrow v_t/(1-\beta_2^t)$ (Compute bias-corrected second raw moment estimate);

(5) $\theta_t \leftarrow \theta_{t-1} - \eta\hat{m}_t/(\sqrt{\hat{v}_t}+\kappa)$ (Update the parameters)

where θ_t represents the set of parameters of the model at timestep t; m_t and v_t are the first and second moment vectors ($m_0 \leftarrow 0$ and $m_0 \leftarrow 0$) at timestep t, respectively; β_1 and β_2 are the first and second moment decay rates, respectively; η is the learning rate; and κ is a scalar value. In our case, η is chosen equal 0.001, other parameters are chosen as default setting [92] ($\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\kappa = 10^{-8}$). The details of the proposed CNN_{SHW} architecture are described in Table 3.4.

3.5. Validation experiments

In these experiments, the reconstructed datasets in Section 3.3 are used to implement training on the proposed models. Different models are performed to obtain the results. Firstly, with the unprotected dataset, only MLP architecture in TCHES2019 called MLP_{DDLA}, and the fine-tuned MLP_{SHW} are investigated. In the case of the protected dataset, the experiments aim to evaluate the efficiency of SHW compared to LSB. Therefore, the MLP_{DDLA} model is reused, except for the output layer and labeling technique, to train the masked data. Regarding hiding countermeasure, MLP_{SHW} and CNN_{SHW} is used to attack noise-generator and de-synchronized datasets, respectively. The experiments are performed on a personal computer with the configuration of Intel Core i5-9500 CPU and DDR4 24GB memory.

3.5.1. Taking the correct key and fine-tuning model

Unlike the profiled DLSCA, non-profiled DLSCA uses the trend of training metrics, such as loss and accuracy, to detect the correct key. This subsection indicates how to use accuracy for determining the correct key in a non-profiled context, and it can be applied to loss metrics as well. Apart from taking the correct key, this part aims to achieve better performance by choosing the right number of hidden layers and the size of each hidden layer. Specifically, four different models are proposed based on MLP architecture called MLP_{SHW1}, MLP_{SHW2}, MLP_{SHW3}, and MLP_{SHW4} corresponding to four different number of hidden layers. The details of the proposed models are presented in Table 3.5. In the case of CNN-based model, the proposed architecture starts with the original architecture described in [7], except for the output layer. The variant of

Table 3.5: Hyper-parameters of MLP models using in experiments.

Hyperparameters	MLP_{DDLA}	MLP_{SHW1}	MLP_{SHW2}	MLP_{SHW3}	MLP_{SHW4}
Input	700	50	50	50	50
Hidden layer	2	3	4	5	6
Neuron	20x10	150x100x25	150x300x100x25	150x300x300x100x25	150x300x600x300x100x25
Output	2	3	3	3	3, 9
Label	LSB	SHW	SHW	SHW	SHW, 9-HW
Optimizer	Adam	Sdgm	Sdgm	Sdgm	Sdgm
Activation	RELU	ELU	ELU	ELU	ELU, RELU
Learning rate	0.001	0.01	0.01	0.01	0.01
Batch size	1000	-	-	-	-
Initializing	Xaivier Initialization				

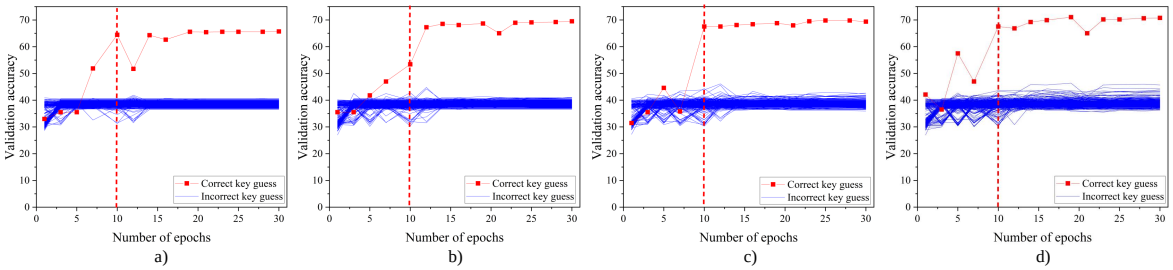

Figure 3.5: Results of validation accuracy using proposed MLP model with different number of layers. a) MLP_{SHW1}; b) MLP_{SHW2}; c) MLP_{SHW3}; d) MLP_{SHW4}

CNN_{SHW} will be investigated in the next part.

Firstly, the training processes on unprotected CW data (Dataset4R) are performed using proposed MLP models. As an illustration, Fig. 3.5a presents the validation accuracy obtained when performing MLP-based SCA attacks using CW dataset with $n_e = 30$ epochs per guess key. In the graph in Fig. 3.5a, the horizontal axis presents the number of training epochs, and the vertical axis shows the validation accuracy values. A total of 256 curves corresponding to 256 accuracy metrics are obtained. It means that these curves are the results of 256 pieces of training using 256 sub-dataset (blue) as shown in Fig.3.2. It can be seen that only one red curve has the highest values (around 65% from the 10^{th} epoch to the

last). In contrast, all other blue curves are lower and fluctuate around 40%. This graph demonstrates that the proposed model can learn the dataset formed by the correct key and can not learn the incorrect key datasets. Consequently, the correct key is found in our attack setup.

As mentioned above, besides finding the correct key, this experiment aims to find out the most effective model from different MLP architectures as presented in Table.3.5. Furthermore, these experimental results clarify the efficiency of SHW labeling compared to other techniques. In this case, the number of epochs and the value of accuracy are used as the metrics for determining the best model, which requires the smallest number of epochs and achieves the highest accuracy value to distinguish the correct key from the incorrect one. As depicted in Fig 3.5, all proposed MLP models provide good results in taking the correct key. There are no big differences in the results of different numbers of hidden layers. However, it can be seen that the most significant difference is in the first ten epochs. Regarding accuracy, MLP_{SHW1} and MLP_{SHW2} are lower than the rest. These results show that using more hidden layers (MLP_{SHW3}, MLP_{SHW4}) could achieve higher accuracy in the case of unprotected data like CW. Switching to number of epochs, it is clear to see that MLP_{SHW4} can distinguish the correct key with only five epochs. In addition, the accuracy at epoch 5^{th} of this model is the highest. Therefore, MLP_{SHW4} is selected for the next experiment. Next, by using MLP_{SHW4}, additional experiments are performed to investigate the efficiency of SHW labeling techniques compared to others. However, only HW labeling and LSB labeling techniques are investigated since identity labeling will naturally lead to similar DL metrics for all the key candidates as indicated in

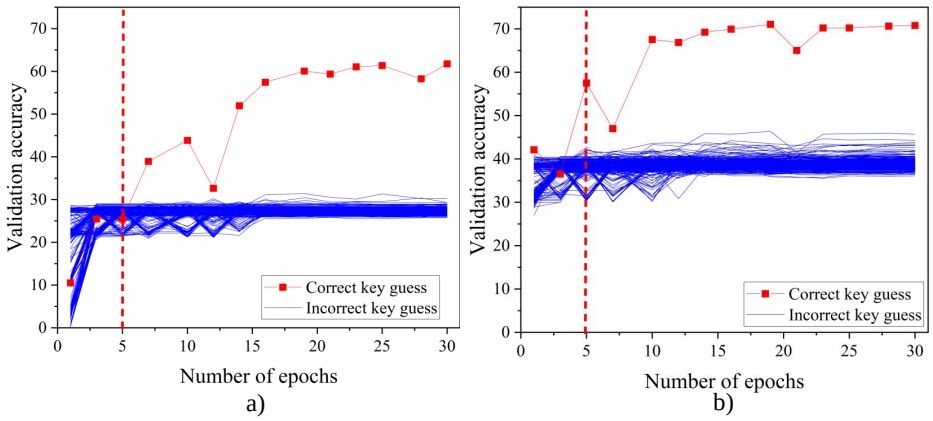

Figure 3.6: Experimental results of proposed model using a) 9-HW and b) SHW labeling technique.

[7]. HW labeling technique is commonly known as an effective power model, which contains nine classes (9-HW). In this experiment, we perform a DLSCA attack on $\text{Dataset4}_{\text{9-HWR}}$ using MLP_{SHW4} with 9-HW. The attack results are then compared to that of SHW, as shown in the previous experiment. As illustrated in Fig.3.6, both labeling techniques achieve good results in taking the correct key. However, it is clear that 9-HW technique has lower accuracy and requires at least seven epochs to discriminate the correct key. More importantly, the number of power traces needed for the SHW technique is approximately 3000 compared to 5000 power traces of 9-HW (less than approximately 30%).

In summary, in the case of using unprotected data like CW, MLP_{SHW4} provides stable results and requires fewer epochs than others. Furthermore, the model using SHW achieves better results and reduces the number of power traces needed for DLSCA. Therefore, for the rest of this chapter, we choose MLP_{SHW4} using the SHW label as the main MLP-based model for the experiments on the unprotected dataset.

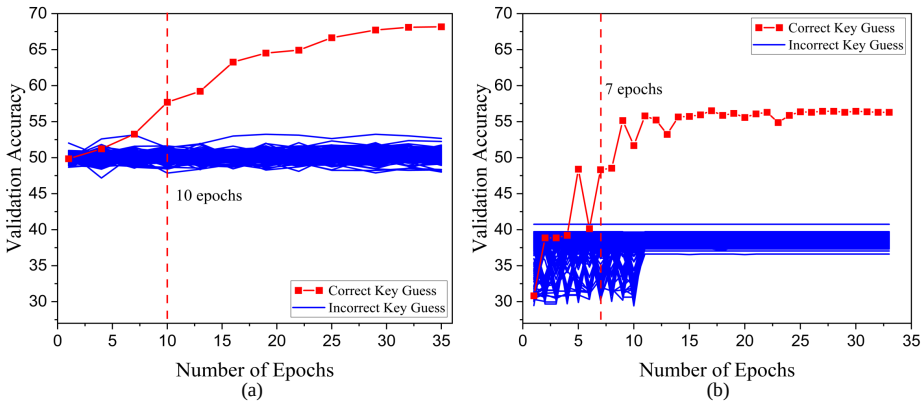

Figure 3.7: Results of ASCAD database for different models using SHW and LSB labeling technique: a) MLP_{DDLA} and LSB label; b) MLP_{SHW} and SHW label;

3.5.2. Unprotected data

To investigate the efficiency of the proposed model and labeling technique on the unprotected platform, we use the unmasked ASCAD dataset reconstructed in Section 3.3. Especially, the performance of DDLA using different labeling techniques, including SHW-based and LSB-based, is investigated. For simplicity, only MLP_{SHW4} model is used in this experiment. Firstly, the training processes using MLP_{DDLA} model on Dataset1R are performed with $n_e = 35$ epochs per guess key. It is important to note that the dimension of data input of MLP_{DDLA} is the same as the original one. However, MLP_{DDLA} are implemented on MATLAB framework instead of Pytorch as in the original work. Secondly, the MLP_{SHW4} model using SHW-based labeling technique is trained on Dataset2R with $n_e = 35$ epochs for each key guess. The attack results on the third byte are illustrated in Fig. 3.7, it can be seen that by using the SHW labels, the model has lower validation accuracy than TCHES 2019 because they use only two labels, which leads to the results for classification is at least 50%. Indeed, the result of MLP_{DDLA} on Fig. 3.7.a shows

that the validation accuracy of the correct key (red curve) increases gradually from 50% to approximately 68%, whereas the incorrect key ones (blue curves) fluctuate around 50%. Obviously, MLP_{DDLA} can discriminate the correct key after 10 epochs.

Switching to the proposed models, as it can be clearly seen in Fig. 3.7.b, the validation accuracy of correct key guess (red curve) increases in a zig-zag way from 30% to 55.6% and then keeps consistent. In contrast, the incorrect key guesses (blue curves) update the values only in the first ten epochs, then keep unchanged below 40%. These results demonstrate that the proposed model using SHW has the ability to reveal the correct key. Despite lower accuracy, the proposed technique gives a high probability of discriminating the correct key than the model in [7]. In addition, the number of epochs needed for determining the correct key is only seven compared to 10 of MLP_{DDLA} (red dash line).

3.5.3. Protected data

According to [7], the MLP-based model is suitable for masked data since it can combine second-order leakage samples automatically without any pre-processing technique. On the other hand, the CNN-based model is capable of fighting against hiding countermeasures. Therefore, in this experiment, an MLP-based model is used to investigate the efficiency of the proposed technique on a masked dataset. Regarding hiding countermeasures, both MLP-based and CNN-based models are selected to evaluate the performance.

Masking

a) Combining function in non-profiled DLSCA

As presented in Section 1.4.1, a very common countermeasure to

protect block ciphers implementation is to randomize their sensitive variables by masking techniques. Every sensitive variable v is randomly split into d shared $m_1, \ldots, m_d$ in such a way that the relation $m_1 * m_2 \ldots * m_d = v$ is satisfied for a group operation $\circ$.

In the case of first-order masking, let assume two leakages $L(t_1)$ and $L(t_2)$ at sample l_1 and l_2 such that:

$$\begin{aligned} L(t_1) &= \delta_1 + HW(v \oplus m) + B_1 \\ L(t_2) &= \delta_2 + HW(m) + B_2 \end{aligned} \tag{3.7}$$

where δ_1 and δ_2 denote the constant part of the leakage, $HW(.)$ is the Hamming Weight function, B_1 and B_2 are two Gaussian random variables (center zero and with standard deviation σ).

The input y_i^1 of node i in the first shared layer can be calculated from the S-dimension data input $a_j^{(0)}$ $(j < 0 \leq S)$ and the weight w_j as follows:

$$\begin{aligned} y_i^1 &= \sum_{j=0}^{m} \left(w_j \times a_j^{(0)} \right) \\ &= \sum_{j=0, j \neq l_1, l_2}^{m} \left(w_j \times a_j^{(0)} \right) + \left(w_{l_1} \times a_{l_1}^{(0)} \right) + \left(w_{l_2} \times a_{l_2}^{(0)} \right) \\ &= \sum_{j=0, j \neq l_1, l_2}^{m} \left(w_j \times a_j^{(0)} \right) + \left(w_{l_1} \times (\delta_1 + HW(Z \oplus M) + B_1) \right) \\ &+ \left(w_{l_2} \times (\delta_2 + HW(M) + B_2) \right) \\ &= \sum_{j=0, j \neq l_1, l_2}^{m} \left(w_j \times a_j^{(0)} \right) + \left(w_{l_1} \times L(t_1) \right) + \left(w_{l_2} \times L(t_2) \right) \end{aligned} \tag{3.8}$$

Because $(w_j)_{0 \leq j \leq S} \in R$, therefore $L(t_1)$ and $L(t_2)$ can be positive or negative. As the result, if the model is perfectly trained ($w_j \approx 0 \, (j \neq l_1, l_2)$), the function $F(.)$ can be satisfied:

$$F(L(t_1), L(t_2)) = |L(t_1) - L(t_2)| \tag{3.9}$$

in other words, the neural network can recombined the mask and the masked values following the *"absolute difference combining function"* to

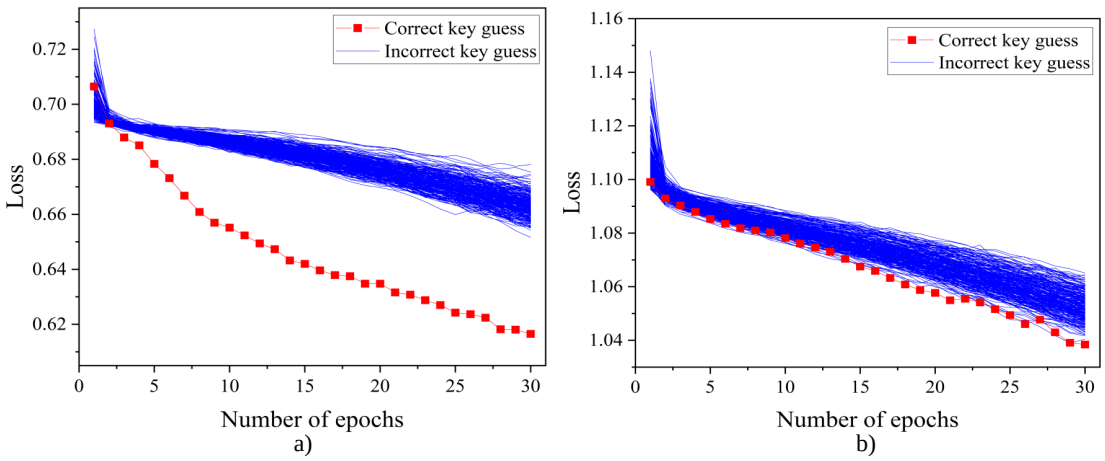

Figure 3.8: Experimental results on masked AES data using LSB and SHW labeling technique.

perform a high-order attack [33]. However, for this combining to work correctly, we assume that δ_1 be equal to δ_2 as indicated in [33].

b) Attack results

As presented in Section 3.1, POI selected dataset is only suitable in first-order leakage data. Therefore, in this experiment, only the efficiency of the proposed labeling technique is investigated. The reconstructed dataset method and proposed MLP architectures are out of scope of this experiment. A new dataset called $\text{ASCAD}_{\text{SHWR}}$ is created based on ASCAD data, which is labeled using SHW instead of LSB method. For convenience, MLP_{DDLA} model is selected to evaluate both LSB labeling and SHW labeling techniques.

Firstly, the attacks are launched by MLP_{DDLA} model, which is trained on the $\text{ASCAD}_{\text{LSBR}}$ dataset with two outputs (LSB label). Next, other attacks are implemented on MLP_{DDLA} model using the $\text{ASCAD}_{\text{SHWR}}$ dataset with three outputs (SHW label). The experimental results are shown in Fig. 3.8. It can be seen that LSB labeling provides a clear distinction between correct and incorrect keys, whereas it is only a small

gap between correct key and incorrect keys in the case of the SHW labeling method.

To investigate more about SHW on masked data, other attacks are performed on ASCAD. The known key-based analysis is then applied. The attack results are shown in Fig. 3.9. In these experiments, the PGE metric is selected to evaluate the efficiency of SHW. It is clear that the correct key usually has a lower loss (low PGE) value than the incorrect keys, as shown in Fig. 3.9.a,b,c. However, detecting the key through the normal method or even the "early stop" technique is not enough.

To reveal the secret key, a new distinguisher based on the inversion of PGE value is introduced. It contains two steps:

- The attacks are performed and repeated N times. PGE of all hypothesis keys is calculated on N attacks.
- Calculate the inversion of all PGE (IPGE) as follows:

$$IPGE = \frac{1}{PGE + \phi} \tag{3.10}$$

  where $\phi = 1$ is a constant to avoid division by zero in the case of $PGE = 0$. Therefore, IPGE reaches 1 when $PGE = 0$. The correct key is then determined based on the highest IPGE of the corresponding hypothesis key.

The attack results using IPGE are presented in Fig. 3.10. By using IPGE values, the correct key is clearly detected from other hypothesis keys. The results also indicate that the performance of the model depends on the hyperparameter of the model, such as batch size and number of epochs. As shown in Fig. 3.10.a, the attacks on 30 epochs training provide better results than that of using 20 epochs (another

peak occurs in Fig. 3.10.b). On the other hand, Fig. 3.10.c indicates that the batch size of 256 achieves better results compared to others (the correct key's IPGE value is greater).

The experiment results have clarified the correlation between the combined second-order leakage samples and the HW model. However, the poor results of SHW show that SHW is less efficient than the LSB label in this case. By applying the IPGE distinguisher, it is demonstrated that SHW labeling is able to reveal the subkey from an AES implementation-equipped masking. In addition, the SHW labeling technique helps reduce the number of measurements for each training process by approximately 30% compared to the LSB labeling technique.

Despite revealing the subkey successfully, the biggest disadvantage of this method is that it requires many iterations of attack to take the PGE values. However, this issue can be solved with other hyperparameters of the model.

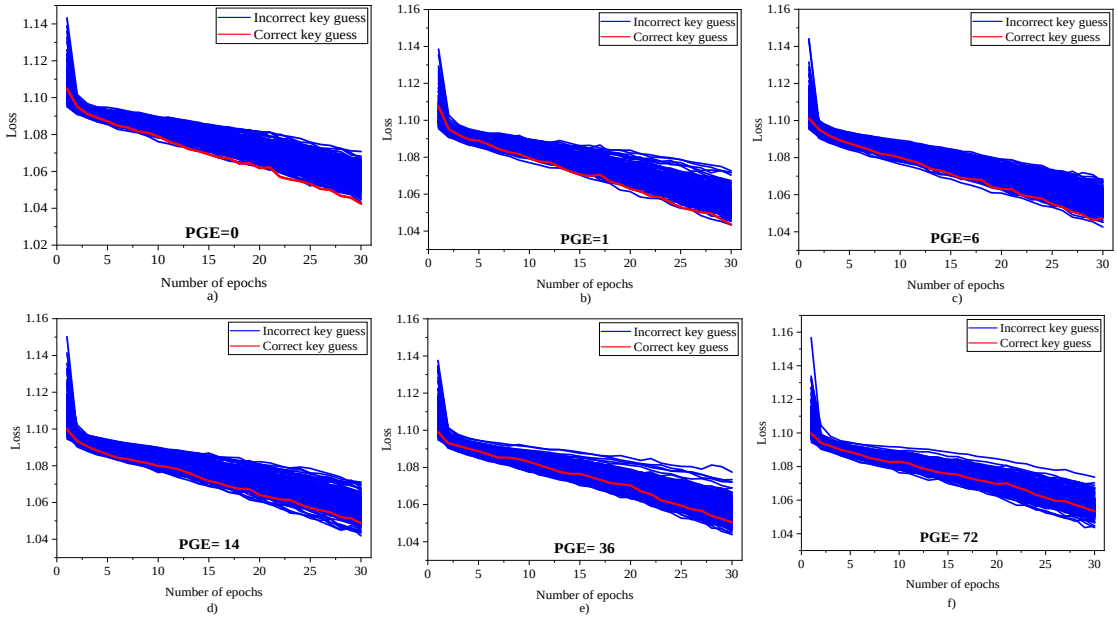

Figure 3.9: Partial Guessing Entropy of the correct key on different attacks using ASCAD data with SHW labeling technique.

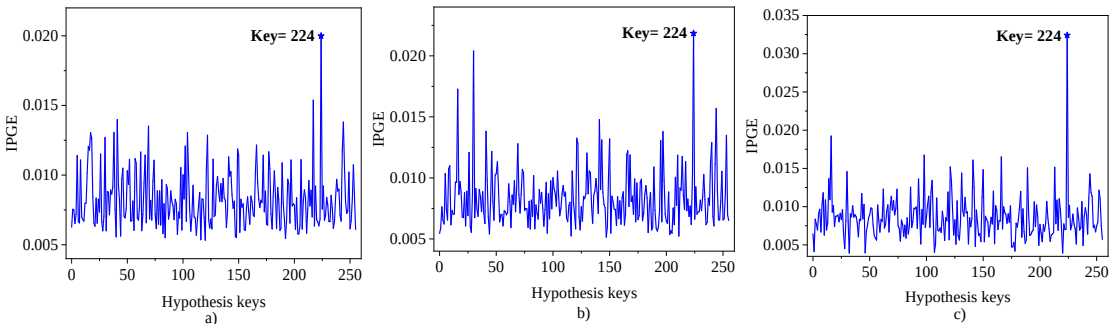

Figure 3.10: Attack results on ASCAD data using IPGE distinguisher and SHW label. a) 30 epochs, batchsize = 1000; b) 20 epochs, batchsize = 1000; c) 30 epoch, batchsize = 256

Noise generation based hiding countermeasure

This subsection evaluates the ability of non-profiled DLSCA against a noise generation countermeasure. In order to simulate the noise generator countermeasure, different levels of Gaussian noise are added to power traces as described in Section 1.2.3. As pointed out in the previous chapter, non-profiled SCA utilizes the relationship between real power consumption and the power consumption model. In addition, the authors in [20] showed that the additive noise contribute to values of the Signal-to-noise ratio as follows:

$$SNR = \frac{Var\left(P_{\text{exp}}\right)}{Var\left(P_{sw.noise} + P_{el.noise}\right)} \tag{3.11}$$

where $P_{e.noise}$ is electronic noise, $P_{sw.noise}$ is switching noise and P_{exp} denotes the exploitable power consumption.

In this case, additive noise plays a role as $P_{e.noise}$. Furthermore, in [1], the authors show that the correlation is proportional to $\sqrt{SNR}$. Therefore, when the Gaussian noise is introduced, the relationship between the data input and the output (label) of DDLA decreases. As a result, the DL model is more difficult to learn the data correctly.

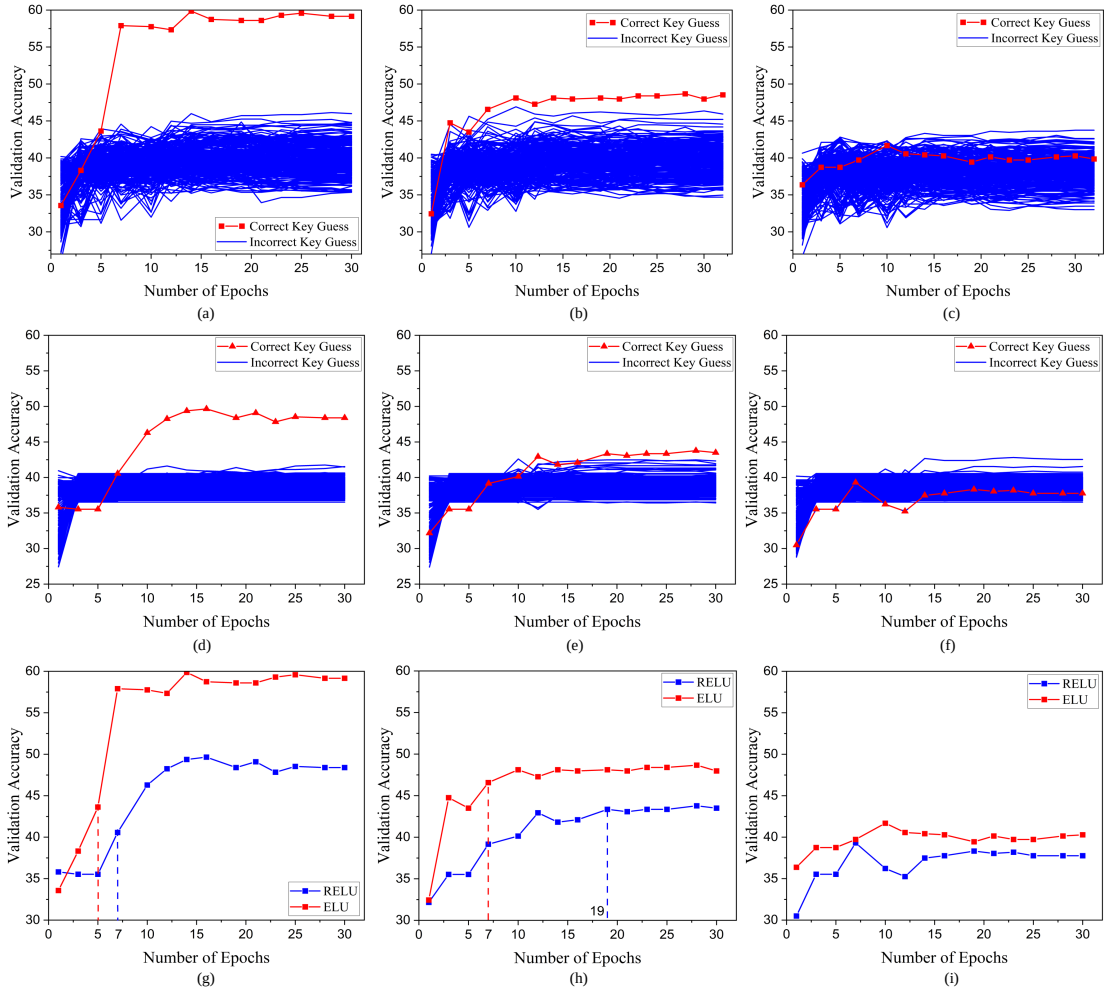

Figure 3.11: Attack results of the MLP_{SHW4} model fight against three levels of noise generation-based hiding countermeasure. Left column: 0.025, Center column: 0.05, Right column: 0.075; Fig (a),(b),(c) are results of MLP_{SHW4} using ELU activation function; Fig (d),(e),(f) are results of MLP_{SHW4} using ReLU activation function; Fig (g),(h),(i) compare the validation accuracy of MLP_{SHW4} using ELU and ReLU.

In this experiment, the datasets described in the Table 3.2 are chosen because the raw power traces from the CW platform are low in noise, and we assume that they have no impact on measurement equipment. The results are depicted in Fig. 3.11, where the top row and middle row illustrate the results of MLP_{SHW4} using ELU and RELU in different levels of noise. The bottom row shows the comparison of using ELU and RELU activation. Overall, MLP_{SHW4} provided good results in a

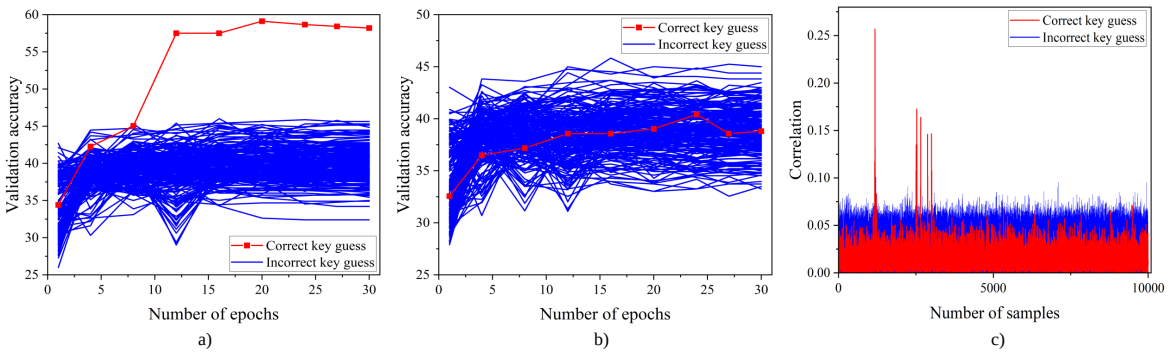

Figure 3.12: Non-profiled DLSCA and CPA attack results against hiding countermeasure. a) DLSCA, $\approx$ 2100 power traces, $\sigma = 0.025$; b) DLSCA, $\approx$ 2100 power traces, $\sigma = 0.055$; c) CPA, 3000 power traces, $\sigma = 0.055$

non-profiled context. However, MLP-based DDLA needs more epochs to discriminate the correct key than CNN-based DDLA.

In terms of the effect of noise, the curve accuracy of MLP_{SHW4} training on the dataset reconstructed from the correct key goes down when the standard deviation of Gaussian noise increases (from 0.025 to 0.075) as illustrated in Fig. 3.11.a,b,c. However, a significant difference in performance between ELU and RELU activation can be seen as depicted in Fig. 3.11.g,h,i. In the first case ($\sigma = 0.025$), it is clearly seen that the correct key accuracy of MLP_{SHW4} using ELU activation (red curve) increases drastically and reaches the peak of about 60%, whereas the RELU used model (blue curve) increases gradually from 36% to 49%. This means that ELU used model provides greater learning ability than RELU. In addition, the number of epochs needed for discriminating the correct key is only 5 (ELU- red dash line) compared to 7 (RELU- blue dash line). More interestingly, in the case of higher noise ($\sigma = 0.05$), ELU used MLP_{SHW4} needs only 7 epochs compared to 19 epochs of MLP_{SHW4} using RELU to determine the correct key. Especially, the results presented in Fig. 3.11i show that both models using ELU and RELU fail to recover

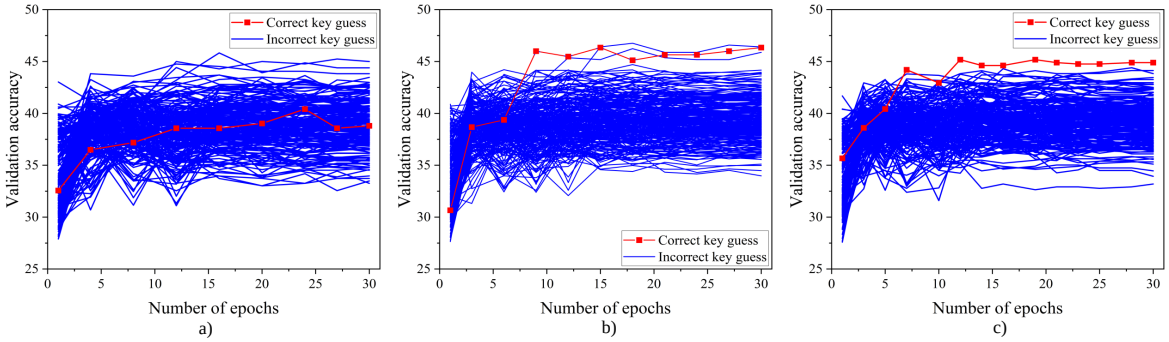

Figure 3.13: Experimental result of non-profiled DLSCA using different sizes of dataset . a) $\approx$ 2100 power traces, $\sigma = 0.055$; c) $\approx$ 2800 power traces, $\sigma = 0.055$; d) $\approx$ 3500 power traces, $\sigma = 0.055$.

the key due to the presence of a high level of noise ($\sigma = 0.075$). However, the red curve is still higher than the blue curve. These results clarify that MLP_{SHW4} using ELU outperforms the RELU in fighting against a noise generation-based hiding countermeasure.

In next experiment, all attacks are performed using MLP_{SHW4}. The results of non-profiled DLSCA on the low noise dataset ($\text{Dataset5}_{\text{1R}}$) are illustrated in Fig. 3.12a. Evidently, our model provides good performance with the presence of a small level of noise generator ($\sigma = 0.025$). However, by increasing the noise ($\sigma = 0.055$) and keeping the same size of the dataset ($\text{Dataset5}_{\text{2R}}$), the noise generator countermeasure fights against our MLP-based model successfully, as depicted in Fig. 3.12b. Compared to CPA attack, we perform a first-order CPA attack on the original dataset of $\text{Dataset5}_{\text{2R}}$ namely $\text{Dataset5}_{\text{2O}}$. The results shown in Fig. 3.12c indicate that the CPA attack outperforms non-profiled DLSCA in the case of applying noise-generator countermeasure. Next, we keep the level of noise ($\sigma = 0.055$) and increase the size of the dataset to 4000 and 5000 power traces, then the reconstructed datasets called $\text{Dataset5}_{\text{3R}}$ and $\text{Dataset5}_{\text{4R}}$ are used. Fig. 3.13 depicts the attack results.

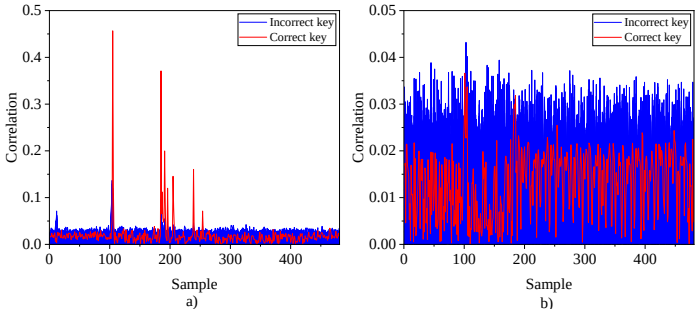

Figure 3.14: The CPA attack results on de-synchronized datasets. a) Small shifted value, success; b) High shifted value, failed

Interestingly, by increasing the size of the attack dataset, our model achieves better results and breaks the noise generator countermeasure successfully, as shown in Fig. 3.13c. These results clarify that increasing the attack set size can help mitigate the hiding countermeasure in the non-profiled scenario.

De-synchronized

In this part, we first clarify that the de-synchronized works well in defending the statistic-based attack such as CPA. Two CPA attacks are performed on $\text{Dataset3}_{\text{sh1R}}$ and $\text{Dataset3}_{\text{sh5R}}$. As depicted in Fig.3.14, it can be clearly seen that the CPA attack on $\text{Dataset3}_{\text{sh1R}}$ can reveal the secret key easily, whereas the attack on $\text{Dataset3}_{\text{sh5R}}$ failed. These results have clarified the efficiency of de-synchronized on defending CPA attacks.

To investigate the efficiency of the proposed CNN model using SHW label (CNN_{SHW}), we start with the base architecture as described in [7] (CNN_{LSB}), except the output layer. Then, we evaluate the impact of the different number of filters to find the most effective model. In addition, the comparisons are made for our proposed model and TCHES 2019. Firstly, the attack results of CNN_{LSB} and CNN_{SHW} using four fil-

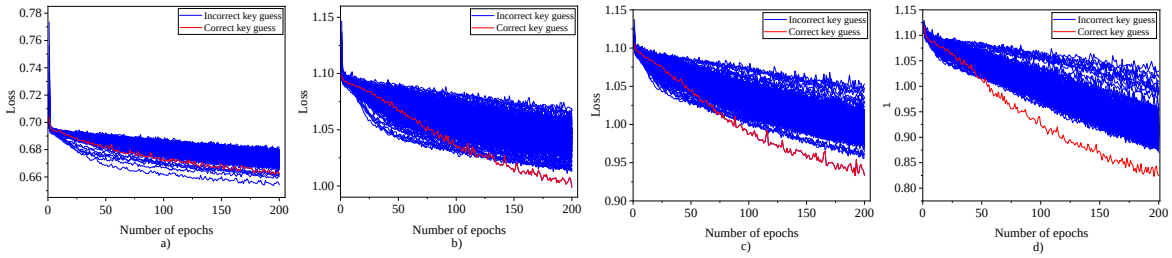

Figure 3.15: The attack results of CNN models using LSB and SHW labeling technique on de-synchronized datasets with different numbers of filters. a) LSB label, 4 filters; b) SHW label, 4 filters; c) SHW label, 8 filters; d) SHW label, 16 filters;

ters are shown in Fig. 3.15.a and Fig. 3.15.b, respectively. With the same condition of the training process (number of epochs, number of filters, activation, etc.), it can be seen that CNN_{LSB} can not reveal the secret key, whereas the secret key can be clearly discriminated at epoch 150 by CNN_{SHW}. More interestingly, CNN_{SHW} performs the attacks faster than CNN_{LSB} by about 20% (8.31 hours compared to 10.52 hours as shown in Table 3.8) since the model using SHW label reduces approximately 30% power traces needed for training. It is worth noting that by implementing various experiments, the results of CNN_{SHW} mentioned above are achieved with a batch size equal to 200; other values provide poor results.

In summary, these results have demonstrated that DL-based attack outperforms CPA in the same condition, such as the number of measurements for attack and de-synchronized countermeasure applied. Significantly, the CNN model using SHW achieves higher performance than that of using LSB labeling.

Table 3.6: The execution time of DDLA attacks on unprotected data using different labeling techniques.

Target	Model	Labeling technique	Number of traces		No. of samples	RD method (ε)	No. of epochs	Execution time
			Total	Used for training				
$CW_{\text{no mask}}$	MLP_{exp} [7]	LSB	3000	3000	10000	-	30	4 hrs 31 min 59 sec
	MLP_{SHW4}	SHW	3000	≈2100	50	P-CPA (50)	30	1 hr 39 min 30 sec
$ASCAD_{\text{unmasked}}$	MLP_{exp} [7]	LSB	10000	10000	700	-	30	6 hrs 53 min 44 sec
	MLP_{SHW4}	SHW	10000	≈7000	50	P-CPA (50)	30	6 hrs 0 min 41 sec

RD: Reducing data dimension; -: Not used

3.5.4. Complexity

This section analyzes the complexity of proposed methods compared to the previous work [7]. One drawback of DDLA is that it is necessary to perform DL training for each key guess. In the previous subsection, the number of epochs is used as the metric to quantify the efficiency of an attack. However, in practice, the time complexity could be reduced when the model uses the "early stopping" technique to finish the attacks. This technique is out of scope in this thesis and will be considered in our future work. To complete the performance analysis of DDLA attacks, we provide the execution time comparison for one key byte attack. Firstly, the execution times of the proposed model using the SHW labeling and LSB labeling technique were recorded. It is worth noting that all experiments are performed on a personal computer without a Graphic Processing Unit (GPU). It means that the execution time could be reduced dramatically when utilizing parallel computing (a multi-core CPU or GPU). The results are summarized in Table 3.6.

Firstly, the efficiency of the dimensionality reduction method and SHW is demonstrated on the CW dataset. The execution time of DDLA attacks using the SHW label on a reconstructed dataset is significantly faster compared to the original dataset using the LSB label (approxi-

Table 3.7: The execution time of DDLA model using LSB and SHW label.

Attack method	No. of traces training	No. of epochs	Batch size	Labeling technique	Attack time (hours)	Iteration required
MLP_{exp} [7]	20,000	30	1000	**LSB**	**0.24**	**1**
MLP_{exp} + IPGE	≈14,000	30	1000	SHW	3.3	20
		20	1000	**SHW**	**2.92**	**20**

Table 3.8: The execution time of CNN model using LSB and SHW label with different numbers of filters.

Model	No. of filters	Attack time (hours)
CNN_{LSB}	4	**10.52**
CNN_{SHW}	4	**8.31**
	8	10.26
	16	16.41

mately 1 hour 40 minutes compared to 4 hours 32 minutes) since the dimensionality of data input reduced from 10000 to 50. The same trend of attack time can be seen in the results of the ASCAD unmasked dataset. However, the attack time decreases slightly from 6 hours 53 minutes to 6 hours because the dimension input reduces from 700 to 50.

Next, we consider the case of the ASCAD masked dataset. In this context, the architecture, number of traces, and input dimensions are chosen the same as in Timon's work. It is noted that for utilizing the performance of multi-core, Keras framework is used instead of MATLAB for the rest of the experiments. As presented in Table 3.7, the execution time of MLP_{exp} [7] on 20,000 power traces using LSB is 0.24 hours to reveal the subkey. Regarding MLP_{exp} using SHW, it is not clear for taking the subkey after one training as illustrated in Fig. 3.8. Therefore, IPGE is applied to reveal the subkey. Consequently, only the execution time of MLP_{exp} combining IPGE is provided. Since IPGE requires many iterations, the execution time of MLP_{exp} using SHW is very long compared to LSB in this case (2.92 hours compared to 0.24 hours).

Finally, additional experiments are performed to investigate the complexity of CNN_{SHW} using different numbers of filters. The attack results are depicted in Fig. 3.15.b,c,d. With the same eight filters for both Conv1D blocks, CNN_{SHW} achieves better results than four filters when the correct key can be revealed earlier after 100 epochs. The same trend can be seen with CNN_{SHW} using 16 filters. The secret key can be distinguished after only 50 epochs. However, the negative impact of the increasing number of filters is that the attack time goes up significantly from 8.31 hours to 16.41 hours, corresponding to 4 filters and 16 filters, respectively.

In summary, the dimensionality of data input strongly impacts the performance of DDLA. The proposed dimensionality reduction method has demonstrated efficiency in the case of first-order leakage compared to Timon's work. In addition, the size of the training data can be determined by the label used. In our scenario, the models using SHW have achieved better results than that using the LSB labeling method in terms of execution time for non-protected or de-synchronized data. In the case of masking, the SR of attack is low due to the weak correlation between actual power consumption and the power model. In this case, the correct subkey is still reveal by repeating the training process and using IPGE as a distinguisher. It is noted that the neural networks used for the experiments in this chapter were purposely kept small to limit the complexity of the attacks. The architectures used are surely not optimal, and other hyper-parameters of the network might lead to better results.

3.6. Summary

In this chapter, non-profiled deep learning-based SCA techniques are presented and provide details investigations of different scenarios. Regarding the high dimensional data issue, the P-CPA technique is introduced to select the most informative sample points. Therefore, the size of the data input reduces significantly. Related to the imbalanced dataset problem, a new labeling technique called SHW based on Hamming weight is proposed. By using SHW for MLP and CNN, the models have been trained and revealed the correct key successfully. More interestingly, SHW helps the attacker reduce the measurements needed for training processes by approximately 30%. Various experimental results are reported in this chapter, which clarifies the efficiency of SHW compared to 9-HW and LSB labeling techniques on both unprotected and protected datasets. The results also indicate that it is difficult to distinguish the correct key based on SHW label in the case of masking countermeasure applied. This issue can be solved by utilizing IPGE distinguisher after repeating the attack many times. It therefore leads to a longer execution time compared to model using LSB label. Regarding noise generation countermeasure, the experimental results have demonstrated that increasing the attack dataset can account for the correlation noise-generator hiding technique.

Chapter 4
MO-DLSCA: MULTI-OUTPUT DEEP LEARNING BASED NON-PROFILED SCA

This chapter introduces new DL models based on multi-output classification and multi-output regression, which can predict all possible values of the key hypothesis in a single training without any reference device. As a result, the proposed methods can reduce the attack time from several hours to less than half an hour compared to previous work in a non-profiled context. In addition, this chapter first suggests using the identity labeling technique in the non-profiled SCA domain, which mitigates the imbalanced dataset issue and eases the SCA evaluation.

The validation results of the proposed techniques were presented in the papers [C4], [J2], and [J4].

4.1. Introduction

The efficiency of the DDLA technique has been demonstrated in different scenarios, especially in the case of masking and hiding countermeasures applied. However, DDLA spends several hours or even more to reveal only one byte of the secret key. The main reason for the aforementioned problem is that DDLA requires performing the training process many times to record the training metrics for all key hypotheses. It leads to a time-consuming and high-cost evaluation process. For example, with the AES-128 algorithm, the DDLA method (Algorithm 2) requires performing 256 training processes corresponding to 256 key guesses to

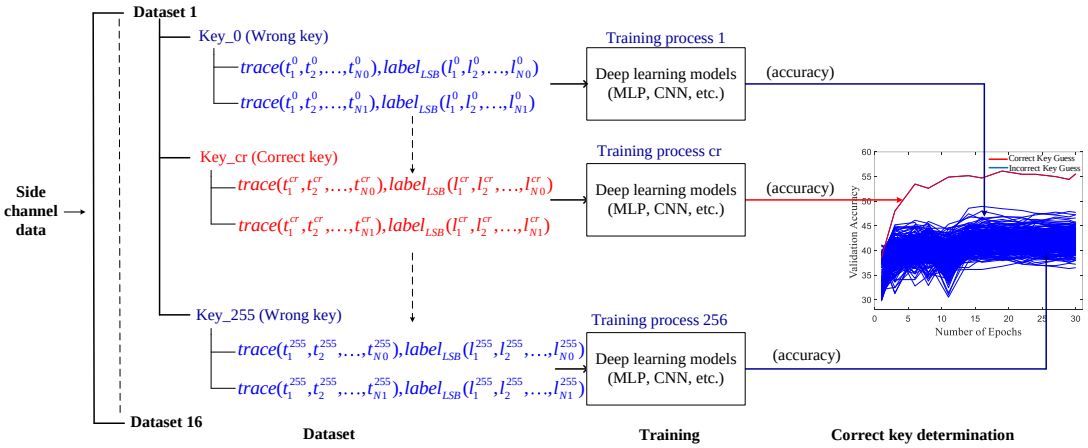

Figure 4.1: Differential deep learning analysis attacks perform the training process repeatedly to reveal the secret key.

determine the correct subkey byte, as illustrated in Fig. 4.1.

Most recently, the drawbacks of the original DDLA have been investigated and mitigated by Kwon et al. [40]. Their work is based on multi-label neural networks. Accordingly, they use a parallel architecture to predict a total of 256 hypothesis keys. The loss function used in the parallel network is the *binary cross-entropy*. The output layer consists of 256 nodes (corresponding to 256 key guesses).

To apply the multi-label in their model, the hypothesis intermediate value $h = LSB(Sbox(p \oplus k_i))$ is calculated for each key guess. Let us assume that the values for k_0 is **0**, for k_1 is **1**, for k_2 is **1**,..., and for k_{255} is **0**, then the label array "**011....0**" is achieved. One element of the array represents one key hypothesis. Therefore, it consists of the values for all key guesses. Since only a scalar loss value is achieved after a training process, the authors cannot use the *"loss metric"* for discriminating the correct key. Their solution is using a custom function to calculate the accuracy of each key guess by separating the output and then matching the hypothesis values to count the number of corrected predictions. This

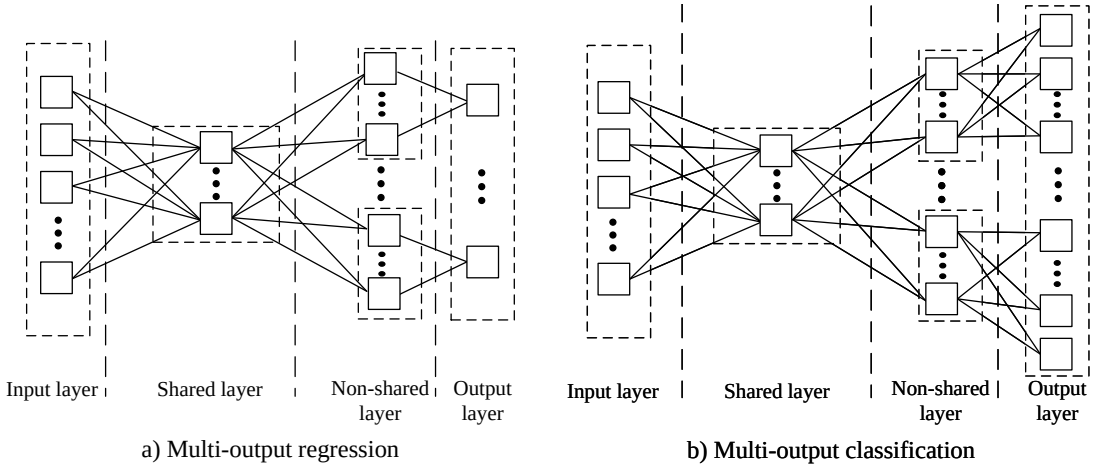

Figure 4.2: The structure of multi-output neural network

method requires extra calculations to achieve the final results. Therefore, the attack time is not optimized.

The authors in [40] have also introduced a shared-layer-based model to mitigate the drawback of parallel architecture. However, their proposed architectures are still based on the original DDLA. Therefore, only the attack time is investigated and enhanced. Another important metric, such as the success rate of attack, has not been considered.

Recently, an alternative and often more effective approach in the DL domain is to develop a single neural network model that can learn multiple related tasks (i.e., outputs) at the same time, called multiple-output learning (MOL) [79]. There are two mostly used models in MOL: multi-output regression (MOR) and multi-output classification (MOC), as illustrated in Fig. 4.2. Firstly, MOR is known in the literature as multi-target, multi-variate or multi-response regression, which aims to predict multiple real-valued output/target variables simultaneously. Next, MOC is usually used to perform different classification problems simultaneously. From the point of view of the SCA domain, MOR and MOC are promising techniques that could increase the performance of the SCA

evaluation process. Therefore, this chapter proposes and investigates the efficiency of MOL models based on two popular architectures, such as MLP and CNN. Concretely, MLP-based MOL models are selected to deal with the datasets that are equipped with masking or noise injection countermeasures. To break other protected scheme such as de-synchronized, the models based on CNN are applied.

4.2. Data preparation

To apply multi-output model in SCA domain, the data input (power traces) should be labeled by the values corresponding to the model outputs. The proposed models aim to predict all key hypotheses in one training process. Therefore, the number of network outputs is K, which corresponds to K key guesses (0 to K). However, it is different from Kwon's models; the outputs of the proposed model are divided into K branches. To benchmark the proposed models, this chapter uses the

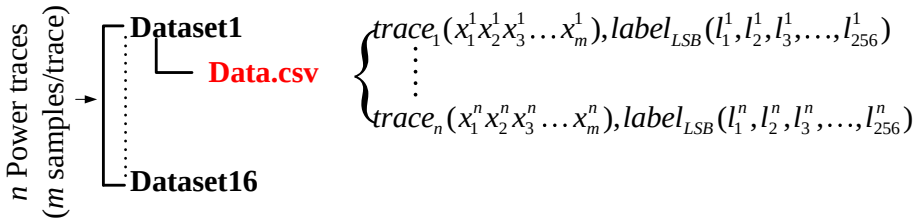

Figure 4.3: Structure of multi-output datasets used in this thesis.

same LSB labeling technique as in previous works [7, 40], which is calculated by the formula (1.12). Regarding the MOR model, the identity labeling technique is first selected to use in non-profiled DLSCA. In this case, the ID labels are calculated by the formula (1.13). Regarding the SHW label, it requires a different set of power traces corresponding to different key hypotheses. Therefore, SHW can not be applied in multi-output architecture. All multi-output datasets used in this chapter are

Table 4.1: The structure of reconstructed datasets.

Dataset	ASCAD		ChipWhisperer		CHES2018-CTF		Ground truth value
	No. of traces	Input	No. of traces	Input	No. of traces	Input	
Original	50000	700	10000	480	45000	2200	-
Dataset1	20000	700	-	-	-	-	LSB
Dataset2	20000	700	-	-	-	-	Identity
Dataset3	-	-	10000	480	-	-	LSB
Dataset4	-	-	10000	480	-	-	Identity
Dataset5	-	-	-	-	40000	2200	LSB
Dataset6	-	-	-	-	40000	2200	Identity

constructed as depicted in Fig. 4.3.

In order to evaluate the efficiency of the proposed models, the data recorded from different platforms are considered, including ASCAD data, CHES2018-CTF data, and the data captured from the CW board. Concretely, 20,000 power traces from the fixed key ASCAD dataset are used to evaluate the efficiency of the proposed model on the first-order masking countermeasure. To evaluate the efficiency of proposed models on different platforms, other masking datasets are reconstructed from 40,000 power traces CHES2018-CTF. In the case of CW data, 10,000 power traces with the size of 480 samples/trace are selected, which correspond to the power consumption of the first Sbox output process. In addition, this dataset simulates the de-synchronization countermeasure as described in Section 1.2.3. The structure of reconstructed datasets used in this chapter is shown in Table 4.1.

4.3. Proposed multi-output classification neural networks

This section introduces MOR and MOC models based on multi-layer perceptron architecture, which can predict 256 hypothesis keys simulta-

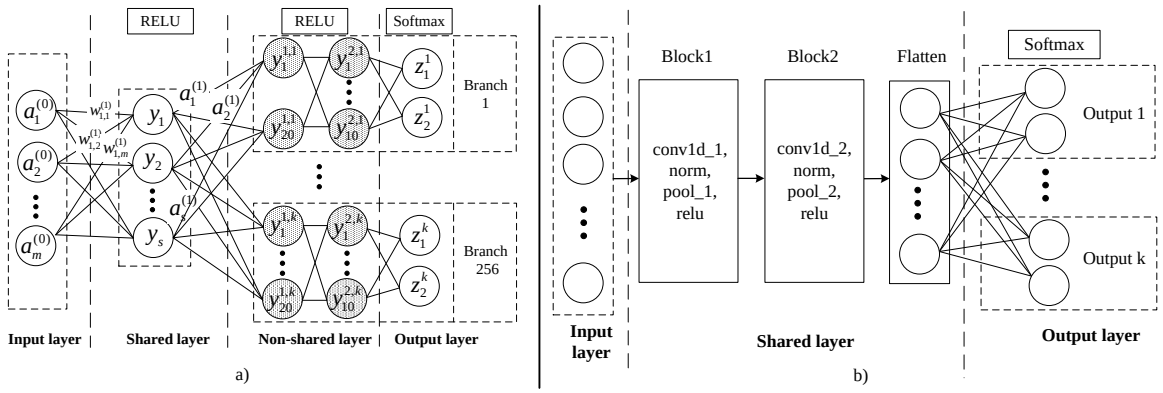

Figure 4.4: Structure of proposed multi-output classification neural network.

neously in a single training process.

4.3.1. MLP_{MOC}

This part first considers the non-profiled SCA as a classification problem. As depicted in Fig. 4.4.a, the overall architecture of the proposed network consists of an input layer, a shared layer followed by K branches corresponding to K hypothesis keys (for example, $K = 256$ in the case of 8-bit Sbox). Each branch contains an MLP architecture as same as MLP_{DDLA} (except the input layer) [7]. According to [23], the number of layers and the number of nodes in each layer are similar to the original MLP_{DDLA} model (hidden layer: 20x10-Relu, output layer: 2-Softmax). The input layer of the proposed model has the same size as the number of samples in the power trace.

The shared layer plays an important role in the proposed architecture. It can be utilized the max shared as in $\text{MLP}_{\text{max-shared}}$ [40]. However, using the same architecture of MLP_{DDLA} except the output layer, their architecture only decreases the execution time without enhancing the success rate, especially in the case of noisy data. In contrast, the proposal aims to decrease the computation time as well as enhance the success rate.

Therefore, we do not make the comparison to $\text{MLP}_{\text{max-shared}}$ in this work. In the proposed model, the shared layer is exploited to decrease the number of input features for each branch. In addition, the shared layer plays a role in reducing the impact of noise of data input. As a result, the proposed model can deal with noise generator hiding countermeasure better than MLP_{DDLA}. However, it may cause a negative impact on the convergence of the optimization algorithm compared to MLP_{DDLA}. Consequently, the number of epochs for obtaining good training metrics is higher than the model without using a shared layer. A formal description of our proposed model is presented below.

Let $\boldsymbol{W}^{[l]} = R^{u^{(l-1)} \times u^{(l)}}$ and $\boldsymbol{B}^{[l]} = R^{u^{(l)}}$ be the matrices of weight and bias parameters, respectively, of the shared layer number l^{th} containing $u^{(l)}$ units. Prior to the implementation training phase, the values of weights and bias are randomly chosen from a normal distribution using the Xavier scheme.

Firstly, the *"forward propagation"* procedure is performed. Given an example input $\boldsymbol{t}_i$, the activation value of the unit in the shared layer number l^{th} (presented as $\boldsymbol{A}$) is computed as follows:

$$\begin{aligned} \boldsymbol{Z}^{[l]} &= \boldsymbol{W}^{[l]} \boldsymbol{A}^{[l-1]} + \boldsymbol{B}^{[l]} \\ \boldsymbol{A}^{[l]} &= \sigma\left(\boldsymbol{Z}^{[l]}\right) \end{aligned} \tag{4.1}$$

where σ denotes the activation function and it is noted that $\boldsymbol{A}^{[0]} = \boldsymbol{t}_i$. There are some commonly used and popular activation functions such as Sigmoid, Hyperbolic tangent (Tanh), ReLU, ELU, and Softmax. In the proposed model, ReLU is chosen for the nodes in hidden layers since it is less computationally expensive than others.

Regarding the non-shared layers, the activation values of the first non-

shared layer of the branch k^{th} ($\boldsymbol{A}^{[1,k]} \in R^{u_{ns}}$) are computed as follows:

$$\begin{aligned} \boldsymbol{Z}^{[1,k]} &= \boldsymbol{W}^{[1,k]}\boldsymbol{A}^{[s]} + \boldsymbol{B}^{[1,k]} \\ \boldsymbol{A}^{[1,k]} &= \sigma\left(\boldsymbol{Z}^{[1,k]}\right) \end{aligned} \tag{4.2}$$

where $\boldsymbol{A}^{[s]}$ represents the activation values of the last shared layer. Consequently, the activation values of the last non-shared layer of the branch k^{th} is defined as:

$$\begin{aligned} \boldsymbol{Z}^{[ns,k]} &= \boldsymbol{W}^{[ns,k]}\boldsymbol{A}^{[ns-1,k]} + \boldsymbol{B}^{[ns,k]} \\ \boldsymbol{A}^{[ns,k]} &= \sigma\left(\boldsymbol{Z}^{[ns,k]}\right) \end{aligned} \tag{4.3}$$

In the proposed MOC model, four variants of the shared layer are investigated. They include a non-shared layer (0 node) and one shared layer of 50, 200, and 400 nodes, for comparison with MLP_{DDLA}. In the case of the model using the non-shared layer, each first hidden layer of a branch is fully connected to the input layer. In such a case, each branch of the proposed model can learn all input features independently, similar to MLP_{DDLA} architecture. However, the novelty of the proposed model is that the network parameters are updated for all hypothesis keys in each iteration instead of updating for only one guess key as MLP_{DDLA} architecture. Since the same structure is applied for all branches, the weights used for each branch are equivalent. Consequently, the loss function of the whole network is calculated as follows:

$$\mathcal{L}_{total} = \sum_{k=1}^{256} \gamma_k * \mathcal{L}^{[k]}(\theta) \tag{4.4}$$

where θ represents the set of all parameters of the model, γ_k is used as the weighted factor of branch number k^{th} and set as 1 for all branches (weights of each branch is equivalent), $\mathcal{L}^{[k]}$ denotes the loss results cal-

Table 4.2: Deep learning hyper-parameters of proposed models.

Model	MLP_{MOC}	CNN_{MOC}
Input size	700, 480	700, 480
Shared layer	0/50/ 200/400	conv1d_1 (4 32×1 filters), pool_1 (2×1), norm, relu conv1d_2 (4 16×1 filters), pool_2 (4×1), norm, relu
Branch	256	256
Hidden layer/branch	20×10	0
Output layer/branch	2	2
Activation	Relu, Softmax	Relu, Softmax
Optimizer	Adam	Adam
Learning rate	0.001	0.001
Batch size	100/500/1000	50/100/500/1000
Initializing	He_uniform	He_uniform

culated for the k^{th} branch, which can be generally defined as follows:

$$\mathcal{L}^{[k]}(\theta) = -\frac{1}{N_s}\sum_{j=1}^{2} y_{true} \ln(z) \tag{4.5}$$

where y_{true} and z are the ground-truth and the predicted values, respectively. N_s denotes the number of training samples.

Finally, the *"backward propagation"* procedure is implemented to minimize the loss function $\mathcal{L}_{total}$. Accordingly, the gradient of the loss function $\nabla\mathcal{L}_{total}(\theta)$ is computed for updating the network in each iteration.

$$\frac{\partial\mathcal{L}_{total}}{\partial\theta} = \sum_{k=1}^{256} \gamma_k \frac{\partial}{\partial\theta}\mathcal{L}^{[k]} \tag{4.6}$$

In this thesis, the popular optimization algorithm called Adaptive Moment Estimation (ADAM) with default setting is employed to train the proposed model. The details of the proposed models are presented in Table 4.2.

4.3.2. CNN_{MOC}

The authors in [7] have introduced a CNN model (CNN_{DDLA}) to reveal the secret key from the de-synchronized countermeasure. Similar

to MLP_{DDLA}, their model needs to be trained repeatedly to determine the correct key. To mitigate this disadvantage, this part introduces a multi-output model based on CNN architecture (CNN_{MOC}), which can break de-synchronized countermeasures in a single training process. The proposed CNN_{MOC} consists of an input layer, share layers, and an output layer, as depicted in Fig.4.4.b. The shared layer consists of two blocks. Each block includes a 1D convolutional (*conv1d*) layer, an average pooling (*pool*) layer, a batch normalization (*norm*) layer, and a rectified linear unit (*relu*) layer. These layers are placed in order as follows *conv1d-norm-pool-relu*. In the training phase, with the equivalent weights used for all branches, the loss function of each branch and for the whole network is calculated by the formula (4.5) (4.4), respectively, same as in MLP_{MOC}.

Similar to MLP_{MOC}, the CNN_{MOC} model performs *forward propagation* to calculate the output (256 nodes) from the input layer and *backward propagation* to update the learning metrics. However, unlike in MLP_{MOC} where each node has a separate weight vector, nodes in CNN_{MOC} share weights. Each node performs convolutions on the data with the convolution filter being formed by the weights by Equation (3.6). Before feeding up to a nonlinear activation function, the output of (3.6) is normalized. In this case, two *conv1d* layers corresponding to two blocks are designed as described in [7] with the size 32×1 and 16×1, respectively.

The details of the proposed models are presented in Table 4.2. For simplicity, the simplest model based on CNN_{DDLA} model is chosen, except for the output layer. It is worth noting that the attacker is able to apply other hyperparameters to our proposed architecture to enhance the success rate of SCA attacks.

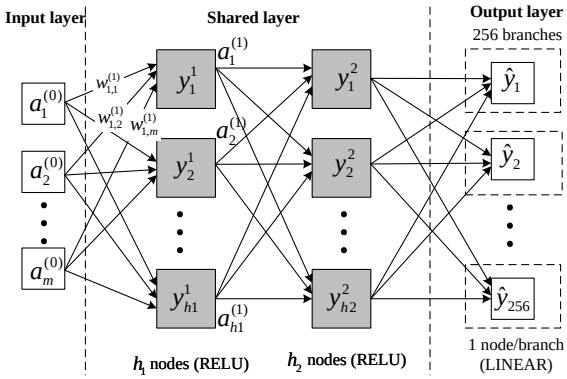

Figure 4.5: Structure of proposed multi-output regression neural network.

4.4. Proposed multi-output regression neural networks

4.4.1. MLP_{MOR}

Considering to regression problem, a MOR-based model is introduced in this part. Similar to the MOC model, the proposed network consists of an input layer, shared layers, followed by 256 branches corresponding to K hypothesis keys. However, a regression model is used instead of a classification model. Therefore, the output of each branch has only one node, as depicted in Fig. 4.5.

To simultaneously achieve K output values corresponding to K key hypotheses, the $\mathrm{MLP}_{\mathrm{MOR}}$ model performs *forward propagation* procedure to calculate the outputs from the input layer. The calculation of each node of the shared layer in the proposed model is the same as the calculation of the MOC model. Once the prediction for a given input $\boldsymbol{t}_i$ is computed, the separate loss on each branch can be calculated by using Mean Squared Error (MSE) for each key guess k as follows:

$$\mathcal{L}^{[k]}(\theta) = \frac{1}{N_s}\sum_{i=1}^{N_s}\left(y_i - \hat{y}_i\right)^2 \tag{4.7}$$

Similar to MOC models, all tasks (branches) of the proposed archi-

Table 4.3: Hyperparameter of the proposed MOR model based on MLP architecture.

Model		MLP_{MOR}
Input size	ASCAD	700
	CHES-CTF	2200
Shared layer (Activation)		300×100/ 500×300/ 700×500 (Relu)
Branch		256
Output layer/branch		1-Linear
Batch size		50/100/500/1000
Learning rate		0.001
Regularzation	Layer1	0.01/0.005/0.002/0
	Layer2	0.02/0.01/0.004/0

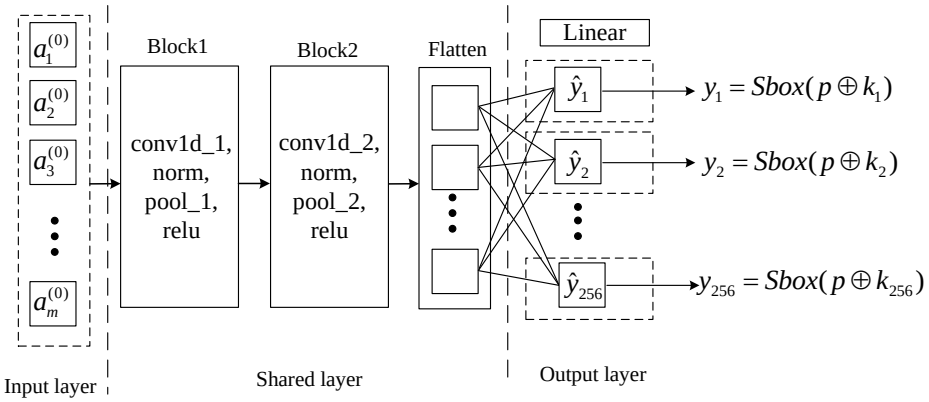

Figure 4.6: Proposed multi-output regression neural network using CNN architecture.

tecture are considered equally important since each key hypothesis has the same priority. To optimize the model, the task-specific losses are simply added together to produce a single scalar loss value as described in [93]. Therefore, our proposed model is optimized by performing *backward propagation* with the sum of losses ($\mathcal{L}_{total}$) using formula (4.4).

The details of the proposed networks are presented in Table 4.3. It is worth noting that the MLP_{MOR} model is performed with three variants of the shared layer, including the size of a shared layer is 300×100, 500×300, and 700×500, for comparison with MLP_{DDLA}.

Table 4.4: Hyperparameter of the proposed MOR model based on CNN architecture

Model		CNN_{MOR}
Input size	CW-shifted	480
Shared layer (Activation)		Conv1d_1(N* 32×1 filters), Pool_1(2×1), Norm, Relu Conv1d_2(N* 16×1 filters), Pool_2(4×1), Norm, Relu
Number of filters (N)	Conv1d_1	4/8/16
	Conv1d_2	4/8/16
Branch		256
Output layer/branch		1-Linear
Batch size		50/100/500/1000
Learning rate		0.001

4.4.2. CNN_{MOR}

This part introduces a multi-output regression model based on CNN architecture (CNN_{MOR}), which can break de-synchronized countermeasure in a single training process. The CNN_{MOR} model consists of an input layer, share layers, and an output layer, as depicted in Fig.4.6. The shared layer consists of two blocks as same as CNN_{MOC}. The novelty of the proposed model is that each branch contains only one node for the output layer. It means that each branch of the model produces real values for input data. These outputs are then compared to the ground-truth values (ID labels). Therefore, the separate loss of each branch and the total loss are calculated as the same in MLP_{MOR} by formula (4.7) and formula (4.4), respectively.

For simplicity, the proposed CNN_{MOR} starts with the architecture based on CNN_{DDLA} model, except for the output layer. However, a different number of filters, as well as different batch sizes, will be used to fine-tune the proposed models. The details of the proposed models are presented in Table 4.4. It is worth noting that the attacker is able to

apply other hyperparameters to the proposed architecture to enhance the success rate of SCA attacks.

4.5. Validation experiments

All experiments were performed by Keras framework on a personal computer with Intel Core i5-9500 CPU, DDR4 24GB memory. It means that the complexity of our proposal is acceptable and can be implemented on a personal computer easily.

4.5.1. Attack on unprotected data

In this part, different experiments are performed to demonstrate the proposed models work well on unprotected datasets. Since the efficiency of the LSB label has been demonstrated in Timon's work [7], the experiments of the proposed model use the Identity label on Dataset 4. Dataset 3 is selected for MLP_{DDLA}. However, only 3,000 power traces are used in these datasets for all experiments. In this part, the MLP_{MORy} model is used where $y = 1, 2, 3$ denotes the size of the hidden layer corresponding to the size of 300×100, 500×300, and 700×500, respectively. The results are presented in Fig. 4.7.

Overall, all models achieve successful attacks. However, there are different results in the first ten epochs of each model. Concretely, it is difficult to distinguish the correct key on the loss metric of MLP_{MOR1}. When the size of the shared layer (SoSL) increases, the gap between the correct key and incorrect keys is clearer, especially in the case of MLP_{MOR3}, the correct key can be taken successfully at epoch 10. This result indicates that the size of input data has a strong impact on the multi-output model. In addition, the result also shows that a better

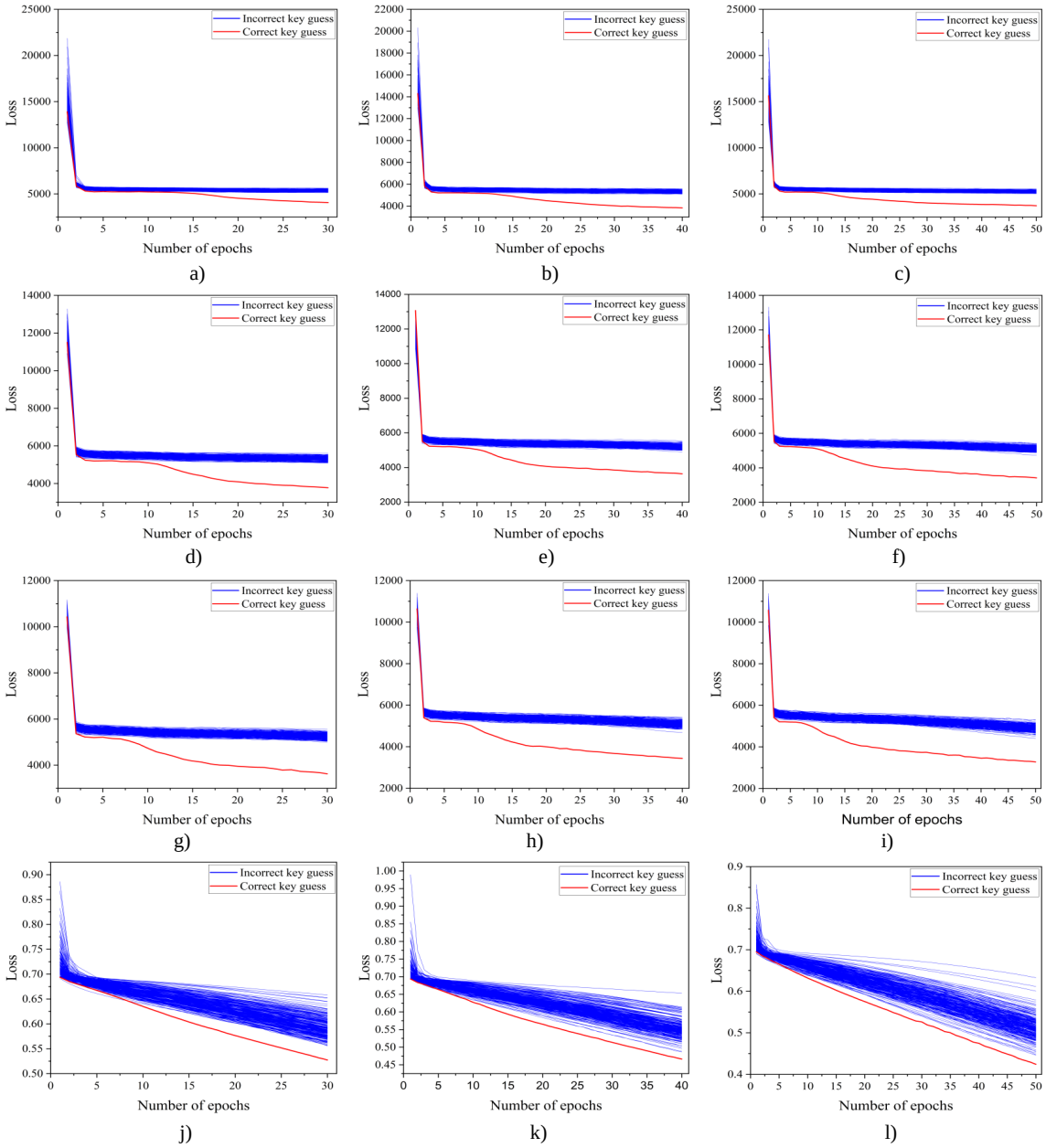

Figure 4.7: Results of loss metric of MLP_{MOR} on unprotected dataset with different numbers of epochs. Left column: 30 epochs, Center column: 40 epochs, Right column: 50 epochs; Fig (a),(b),(c) are results of MLP_{MOR1}; Fig (d),(e),(f) are results of MLP_{MOR2}; Fig (g),(h),(i) are results of MLP_{MOR3}; Fig (j),(k),(l) are results of MLP_{DDLA}.

result can be achieved by increasing the SoSL.

Regarding execution time, the presented attacks above are repeated 30 times. The execution time is then averaged and presented in Fig. 4.8. The results show that the attack time of MLP_{DDLA} is the highest in all

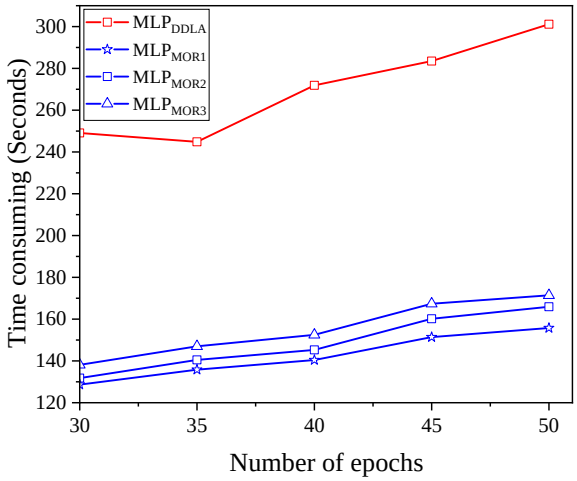

Figure 4.8: Experimental results of proposed multi-output models on unprotected data (ChipWhisper) using different size of shared layer

numbers of epochs. Fortunately, employing a multi-output architecture reduces the attack time significantly (approximately 1.76 times). Furthermore, the results suggest that the attack time is adversely affected by the SoSL, with a longer attack time observed as higher SoSL values are applied.

4.5.2. Attack on protected data

Masking countermeasure

a) MLP_{MOC}

All experiments in this part are performed on Dataset 1. It is noted that the labels corresponding to each key hypothesis can be taken out simultaneously or separately depending on the output of the model. Therefore, the reconstructed datasets can be used for training both multi-output and single-output models.

Firstly, the performance of the proposed network with different values for SoSL is investigated. The models with three values of the shared layer size (50, 200, and 400) are denoted as SoSL-50, SoSL-200, and SoSL-

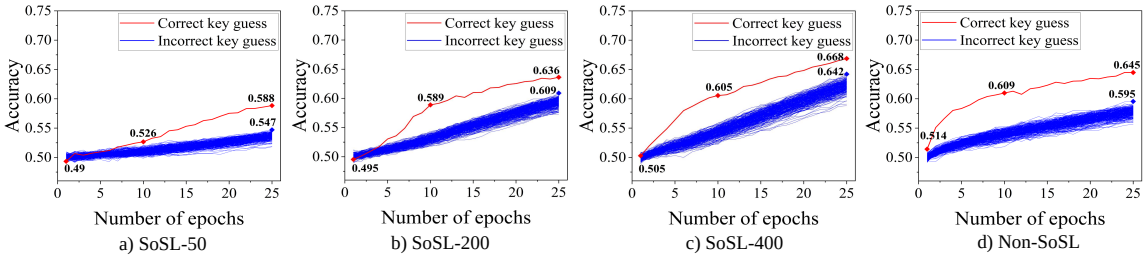

Figure 4.9: The experimental results of MLP_{MOC} models on masking countermeasure using different SoSL on each branch.

400, respectively. As depicted in Fig. 4.9, there is an increasing trend of accuracy along with the increment of SoSL. Accordingly, the model of SoSL-50 achieves poor discrimination in the first ten epochs compared to other models. In contrast, the model of SoSL-400 can discriminate the correct key from incorrect ones very early in Fig. 4.9c. Concretely, at the 25^{th} epoch, the training accuracy of the branch of the correct key (red) increases from 0.588 for SoSL-50 to 0.636 and 0.668 for SoSL-200 and SoSL-400, respectively. However, at this epoch, the gaps between the accuracy of the correct key and the highest accuracy of incorrect keys (blue) decrease from 0.041 to 0.027 and 0.026, respectively. Hence, using a high SoSL may cause an over-fitting problem, which leads to failing attacks if we prolong the training time in too many epochs.

Switching to the proposed network of non-shared layer, so-called Non-SoSL, further experiments are carried out on Dataset 1. As a result, the Non-SoSL model achieves higher performance than all others at all epochs, as shown in Fig. 4.9.d. Especially, the Non-SoSL model provides a clear distinction between the correct key and incorrect keys very soon at the first epoch and a big gap (0.05) at the 25^{th} epoch. This result has clarified that the shared layer has a negative impact on the convergence of the proposed model due to common use for all outputs. Only the

output of the correct key contributes a stable update to the shared layer; other outputs make the weights of shared layers chaotic. The reported results have clarified the efficiency of the proposed methods for masking protected devices.

In the second experiment, the execution time of the proposed network is evaluated and compared to MLP_{DDLA} and MLP_{PL} [40]. To achieve reliable results, the previous experiments are repeated 50 times on the same dataset. In addition, this experiment also performs the attack using MLP_{DDLA} 50 times. The computation time is then averaged and presented in Fig. 4.10. Firstly, the impact of SoSL on the execution time of the proposed network is evaluated. As depicted in Fig. 4.10.a, the computation time of SoSL-50 is lowest (78.439 seconds), and SoSL-400 is highest (130.35 seconds). As stated before, the main purpose of the shared layer is to reduce the parameter for all branches. However, the results of the execution time for SoSL-400 and Non-SoSL indicate that the models using high SoSL will be more time-consuming than the model without using a shared layer (130.35 compared to 109.657 seconds). Despite the lowest execution time, SoSL-50 achieves a lower accuracy than all other models. By this observation, the SoSL-200 model is specified as the best choice of a model using a shared layer to achieve better results of both accuracy and execution time compared to other models for the last experiments.

Unlike MLP_{PL}, the multi-loss-based model in this work provides the training metric separately on each output. It leads to lower complexity than MLP_{PL} using a custom function. To demonstrate this assumption, the reported results in [40] are used to make the comparison. Accordingly, MLP_{PL} reduces the execution time approximately 2.81 times

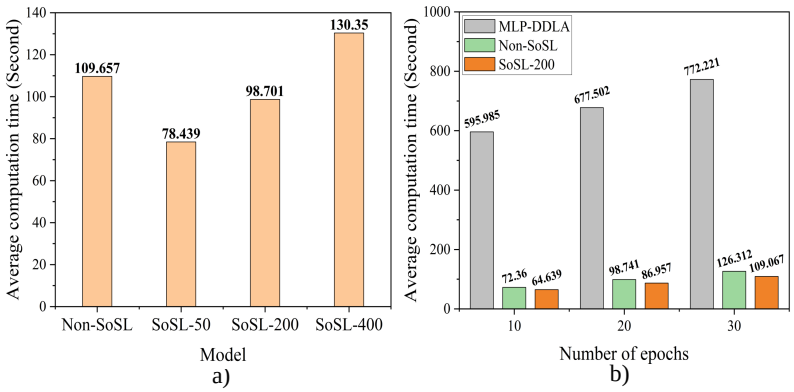

Figure 4.10: The experimental results on masking countermeasure. a, b) Accuracy of proposed model with and without shared layer, respectively; c) Comparison of attack time.

compared to MLP_{DDLA} (from 1950.9 to 693.8 seconds), whereas Non-SoSL decrease the execution time approximately 5.62 times compared to MLP_{DDLA} (from 1052.2 to 182.1 seconds). It is noted that the comparisons above are made in the same conditions, such as the number of traces, dimensional input, and the number of training epochs. These results have clarified our assumption and demonstrated that the proposed model outperforms both MLP_{DDLA} and MLP_{PL} on masking-protected devices.

Finally, the attack time comparison of SoSL-200, Non-SoSL, and MLP_{DDLA} is conducted. Fig. 4.10.b presents the execution time of attacks using selected models on different numbers of epochs. Overall, the execution time of MLP_{DDLA} is the highest in all cases. Interestingly, the execution time of the proposed network decreases dramatically about eight times and nine times (from 595.98 to 72.36 and 64.639 seconds), corresponding to Non-SoSL and SoSL-200 over ten epochs, respectively. Similar results can be seen in the case of training 30 epochs. The execution time of Non-SoSL and SoSL-200 decreased significantly about six and seven times (from 772.221 to 126.312 and 109.067) compared to MLP_{DDLA}, re-

spectively. These results clarify that the proposed model outperforms MLP_{DDLA} in the computation time.

b) MLP_{MOR}

This experiment aims to break the masking dataset, namely CHES-CTF 2018, which has not been investigated by non-profiled DLSCA yet. To compare with previous work, Dataset5 using the LSB label is used for training DDLA_{MLP} model, and Dataset6 using the identity label is used for performing regression tasks based on the proposed MLP_{MOR} models. Firstly, we perform the non-profiled attack based on a classification neural network using MLP architecture as described in [7]. The original architecture is kept except for the batch size in the set $\{1000, 500, 100, 50\}$ is selected to find out the suitable value. In addition, the regularization $L1/L2$ of hidden layer are also applied to achieve better results as described in [23]. The grid search values of regularization are selected as shown in Table 4.3.

The attack results are shown on the first row of Fig. 4.11. Despite changing the batch size, DDLA_{MLP} can not reveal the correct key from CHES-CTF-2018 dataset. Similar results are also achieved with all sets of values of regularization. These results indicate that the classification model is not useful in this case and motivates us to consider the regression models. By investigating various attacks with different sizes of the hidden layers, MLP_{MOR3} with the $L1/L2$ values of the hidden layer1 and hidden layer2 set to 0.01 and 0.02, respectively, provides the best results. The attack results are shown on the second row of Fig. 4.11. It can be seen in Fig. 4.11e,f, MOR-MLP3 can not reveal the correct key. However, when the batch size goes down to 100 and 50, the correct key is clearly discriminated from the incorrect ones. These results have demon-

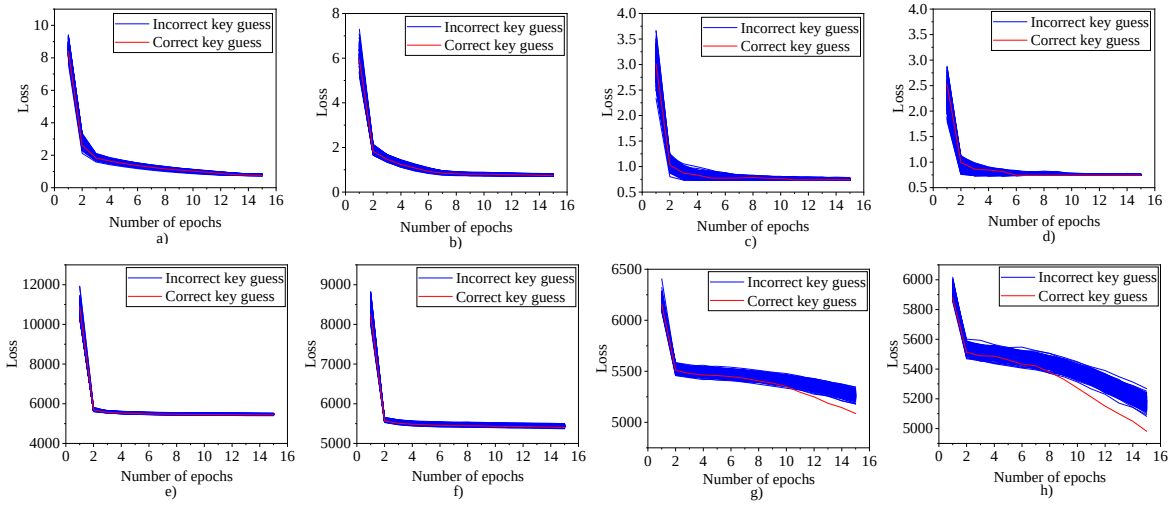

Figure 4.11: Experimental results on CHES-CTF 2018 dataset. a,e) Batch size= 1000; b,f) Batch size= 500; c,g) Batch size= 100; d,h) Batch size= 50; first row) DDLA_{MLP}; second row) MOR-MLP3;

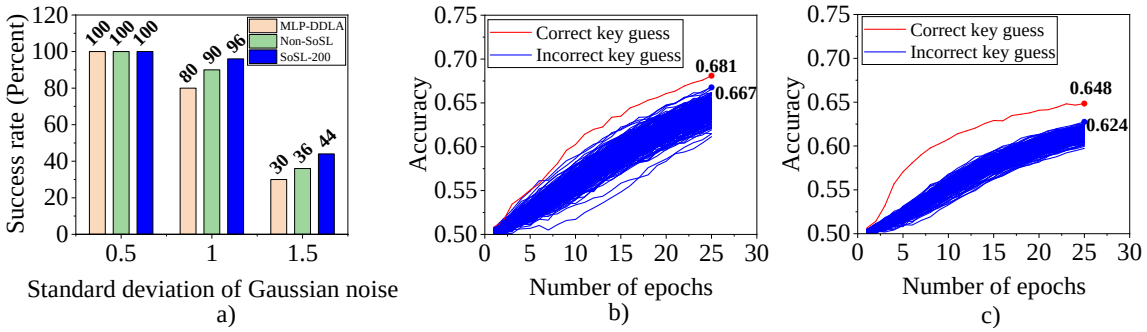

Figure 4.12: The experimental results on combined masking and noise generation. a) Success rate of MLP_{DDLA} and MLP_{MOC} models on different levels of noise using 20,000 power traces; b) SoSL-200 on 20,000 power traces, $\sigma = 1.5$; c) SoSL-200 on 50,000 power traces, $\sigma = 1.5$;

strated that the proposed MOR architecture works well on CHES-CTF 2018 dataset. In addition, the results have clarified that identity label works well in non-profiled DLSCA context using MLP architecture.

Noise generation countermeasures

a) MLP_{MOC}

In this part, the noise injection countermeasure described in Section 1.2.3 is applied to reconstruct new datasets called DatasetX-N1,

DatasetX-N2, and DatasetX-N3 (correspond to $\sigma = 0.5$, 1.0 and 1.5, respectively) are reconstructed as the same technique as DatasetX, where X = 1, 2, 3. In this case, we consider MLP_{DDLA} is more reliable than $\text{MLP}_{\text{max-shared}}$ on noisy data. Therefore, only the comparison between Non-SoSL, SoSL-200, and MLP_{DDLA} is performed. By repeating the attacks using Non-SoSL, SoSL-200, and MLP_{DDLA} 50 times, we calculate the percentage of successful attacks over total attacks. The comparison of the success rate between MLP_{DDLA}, Non-SoSL, and SoSL-200 is shown in Fig. 4.12.a. Evidently, all models achieve good performance (100%) with the presence of a small level of additive noise ($\sigma = 0.5$). However, in the case of higher noise ($\sigma = 1.0$), the number of successful attacks drops from the 100% to 80% and 90% corresponding to MLP_{DDLA} and SoSL models, respectively. Interestingly, the success rate of SoSL-200 only decreases slightly from 100% to 96%. A similar trend can be seen at the higher level of Gaussian noise ($\sigma = 1.5$). The success rate goes down significantly because the models provide poor discrimination, as illustrated in Fig. 4.12.b, in the case of SoSL-200 (0.681 and 0.667). However, our network still achieves better results than MLP_{DDLA} (44% and 36% compared to 30%). We perform further attacks using SoSL-200 on a larger size of the dataset (Dataset3-N3). A clear gap between correct and incorrect keys (0.648 and 0.624) can be seen in Fig. 4.12.c. More interesting, the success rate is 100%. It indicates that by using the reasonable value of SoSL, the proposed network can mitigate the effect of the additive noise better. In addition, the attacker can perform DDLA attacks with reasonable epochs, a larger number of traces, or more hyperparameters, which will, in turn, improve the success rate.

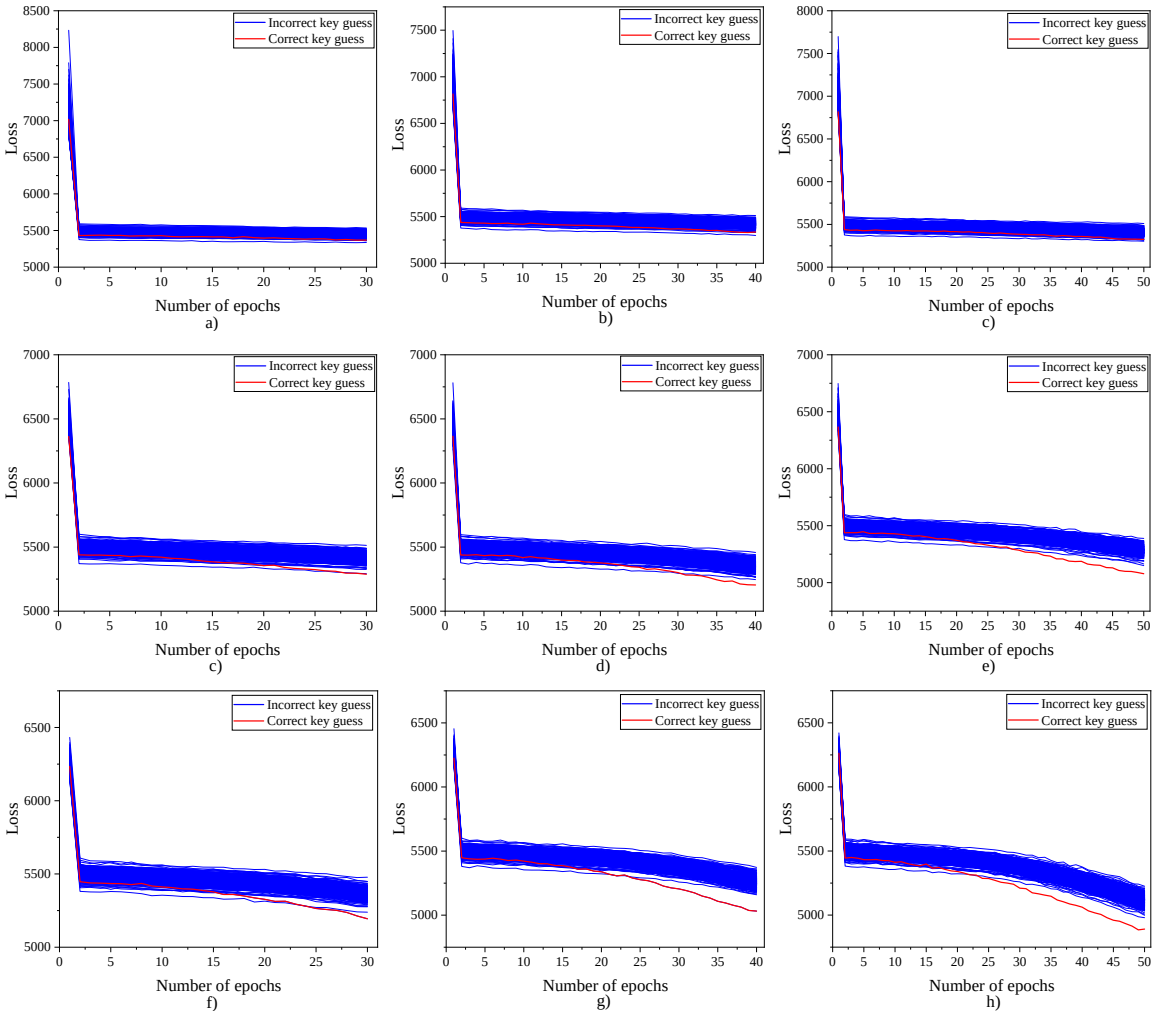

Figure 4.13: Results of loss metric of proposed models on ASCAD with different numbers of epochs. Left column: 30 epochs, Center column: 40 epochs, Right column: 50 epochs; Fig.(a),(b),(c) are results of MLP_{MOR1}; Fig.(d),(e),(f) are results of MLP_{MOR2}; Fig.(g),(h),(i) are results of MLP_{MOR3}.

b) MLP_{MOR}

To evaluate the efficiency of MLP_{MOR} models, ASCAD data is selected. Different levels of noise are added to the original ASCAD data. Then, the datasets called Dataset1-X and Dataset2-X, where X is N1, N2 and N3 corresponding to $\sigma = 0.5, 1.0$ and 1.5, respectively.

To select the best model for the ASCAD data, various experiments are performed on different models corresponding to different sizes of hidden

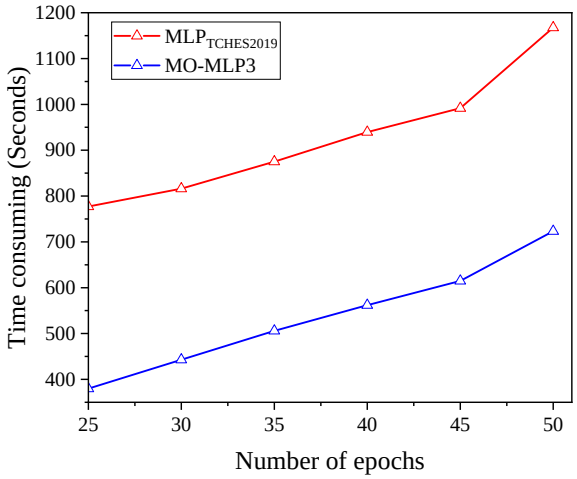

Figure 4.14: The attack time comparison of the proposed MLP_{MOR3} model and DDLA_{MLP} on the ASCAD-dataset.

layers on Dataset2-N1. For each model, the number of epochs is changed from 30 to 50 with step 10. As depicted in Fig. 4.13, MLP_{MOR1} can not reveal the correct key with all epochs. In addition, the results show that the MLP_{MOR1} model can not learn. Moving on to the bigger size of the hidden layer, better results can be clearly seen. The correct key can be revealed at epoch 30. If the training is prolonged, a clear discrimination between correct and incorrect keys can be achieved, especially in the case of MLP_{MOR3}. It is worth noting that the batch size for all models is set to 50, and the regularization values are zero in this case. These results have demonstrated the proposed models using identity label work well on the ASCAD data. To clarify the enhancement of performance compared to previous work, additional attacks are performed using MLP_{DDLA} with the same number of epochs, and the execution time is then recorded. As shown in Fig. 4.14, the blue line and the red line show the execution time of MLP_{MOR3} and MLP_{DDLA}, respectively. The results have demonstrated that MLP_{MOR3} outperforms MLP_{DDLA} in terms of execution time.

Regarding reliability, the attacks are repeated 100 times, and then

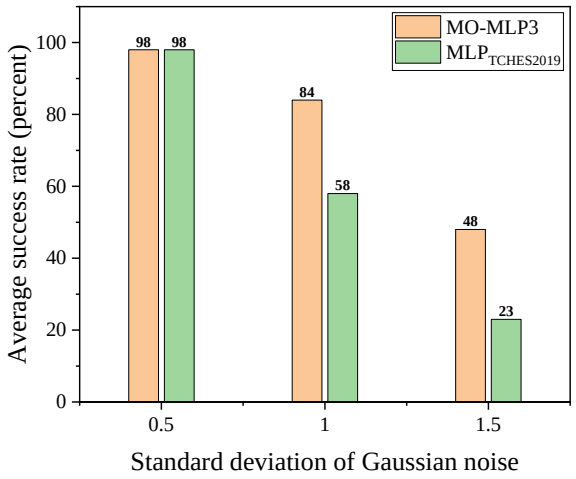

Figure 4.15: Experimental results of proposed multi-output model on full ASCAD-dataset using MO-MLP3

calculate the percentage of successful attacks over total attacks for each level of noise. The results of the first case are presented in Fig. 4.15. Evidently, all models achieve good performance (98%) with the presence of a small level of additive noise ($\sigma = 0.5$). In the case of higher noise ($\sigma = 1.0$), the success rate of MLP_{MOR3} only decreases slightly from 98% to 84%, whereas it can be seen as a dramatic decline of MLP_{ref} (from 98% to 58%). A similar trend can be seen at the higher level of Gaussian noise ($\sigma = 1.5$). Despite the fact that the success rate goes down significantly. The proposed network still achieves better results than MLP_{DDLA} (48% compared to 23%). It indicates that by using the reasonable size of the hidden layer, the proposed network can better overcome the effect of additive noise.

De-synchronized countermeasure

Finally, a popular protected scheme, namely de-synchronization, is considered. To simulate this countermeasure, we randomly shift each power trace of Dataset3 and Dataset4 in a maximum of 20 samples as described in Section 1.2.3. Consequently, two new datasets called

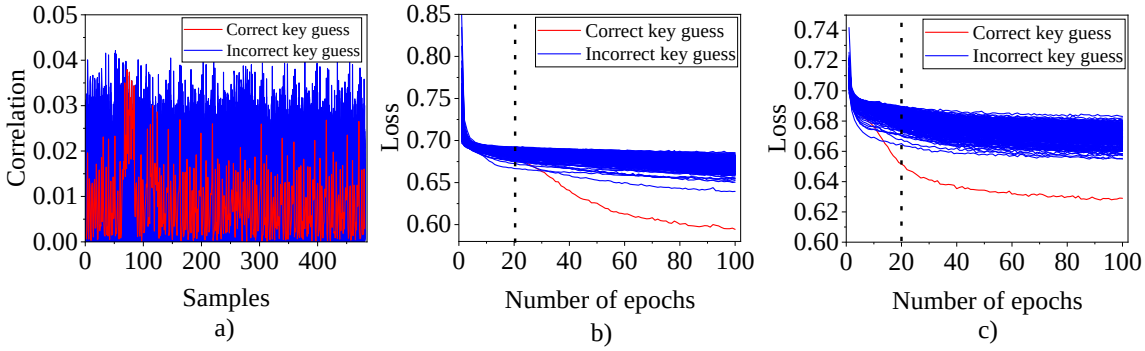

Figure 4.16: Attack results on de-synchronized power traces using CPA, CNN_{DDLA}, and CNN_{MOC}. a) CPA; b) CNN_{DDLA}; c) CNN_{MOC}.

Dataset3-sh20 and Dataset4-sh20 are reconstructed for training CNN_{MOC} and CNN_{MOR}, respectively.

a) CNN_{MOC}

In this experiment, the loss metric of a training process is exploited to reveal the correct key. Firstly, a CPA attack is performed on Dataset3-sh20 to validate the efficiency of the de-synchronized countermeasure. As depicted in Fig. 4.16.a, the secret key can not be revealed. In contrast, a good result in detecting the correct key can be seen in Fig. 4.16.b and Fig. 4.16.c. These results demonstrate that the CNN model can break the de-synchronization countermeasure based on the translation-invariance property. However, the attack time of CNN_{MOC} is shorter by approximately 30 times compared to CNN_{DDLA} (703.65 seconds compared to 20792.43 seconds). In addition, CNN_{MOC} provides a clear distinction at a very early epoch compared to that of CNN_{DDLA}.

To clarify this assumption, various attacks are performed using fewer epochs. Firstly, the number of epochs reduces from 100 to 50. The attacks are repeated 100 times. The success rate and the average time are then taken and shown in Table 4.5. These results have clarified that the proposed model outperforms previous work regarding attack time.

Table 4.5: Attack time comparison of CNN_{MOC} and TCHES2019 on de-synchronized power traces using 50 epochs.

Model	No. of epochs	Success rate (%)	Attack time (hours)
CNN-DDLA [7]	50	100	**3.064**
CNN-MOC	50	100	**0.098**

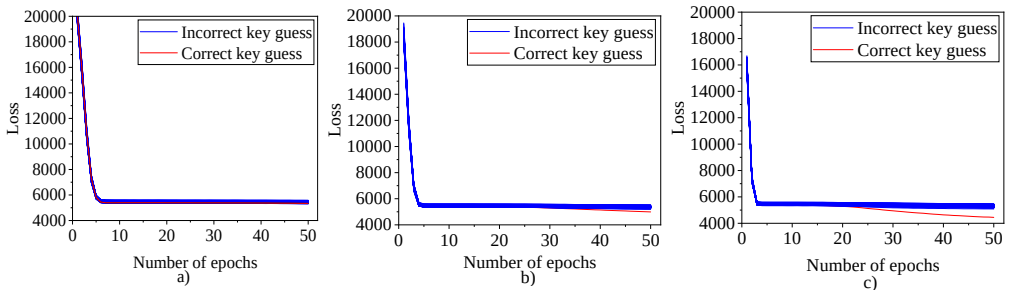

Figure 4.17: Attack results on de-synchronized power traces using the CNN_{MOR} models. a) 4 filters; b) 8 filters; c) 16 filters

b) CNN_{MOR}

In this experiment, different CNN_{MOR} models corresponding to a different number of filters are used to find out the best MOR-CNN model for Dataset4-sh20. Let denote CNN_{MORx} for the variant of the proposed model, where $x = 4, 8, 16$ represents the number of filters.

As discussed in CNN_{MOC}'s experiments, the CNN model outperforms the CPA attack and can break the de-synchronization countermeasure based on the translation-invariance property. Therefore, it is assumed that the CNN-based multi-output regression architectures can provide better performance in DDLA attacks.

Indeed, we perform different DLSCA attacks on Dataset4 using the proposed MOR-CNN models. To select the best hyperparameter for CNN_{MOR}, a grid search with a different number of filters and batch size is performed. As depicted in Fig. 4.17, all CNN_{MOR} models provide good discrimination of the correct key and the incorrect ones. Interestingly,

the results are better when the number of filters increases. Compared to CNN_{DDLA}, CNN_{MOR} does not achieve a clear distinction, especially in the case of CNN_{MOR4} as depicted in Fig 4.17.a. This observation raises a question about the reliability of our proposed model compared to CNN_{DDLA}. To answer this question, we decided to repeat the experiments corresponding to each model 100 times to find out the success rate of attacks. The results are summarized in Table 4.6. Surprisingly, despite poor discrimination, CNN_{MOR4} provides a high success rate compared to CNN_{DDLA} (about 84 % compared to 100 %). More interestingly, the SR of the remaining CNN_{MOR} models is 100%. These results demonstrate the reliability of the proposed architecture compared to CNN_{DDLA}.

Apart from the success rate, the execution time of all experiments is also recorded to make the comparison of the proposed model and CNN_{DDLA}. As shown in Table 4.6, for recovering one key bye, CNN_{DDLA} requires approximately 3.064 hours, whereas the attack time of CNN_{MOR4} and $\text{CNN}_{\text{MOR16}}$ are only 0.075 and 0.103 hours, respectively. In other words, our proposed MOR architectures perform the attacks up to approximately 40 times faster than previous work using DDLA. These results have clarified our assumption that MOR could increase the performance of DDLA by simultaneously predicting 256 key hypotheses. More importantly, the identity labeling technique has also been proven to work effectively in this case.

4.5.3. Results comparison

This part provides the comparison of the results between the proposals and other common techniques using different data in a non-profiled context. The attack results are summarized in Table 4.7. Overall, the

Table 4.6: Attack time comparison of proposed models and TCHES2019 on desynchronized power traces.

Model	No. of epochs	Success rate (%)	Attack time (hours)
CNN-DDLA [7]	50	100	**3.064**
CNN-MOR4	50	84	**0.098**
CNN-MOR8	50	100	**0.087**
CNN-MOR16	50	100	**0.103**

Table 4.7: The comparison of attack results on masking countermeasure using different models.

Model	Data	No. of traces	No. of epochs	Attack time (second)	Results (*)	Power model
MLP_{DDLA}[1] [7]	ASCAD [12]	20,000	30	772.2	S	LSB
MLP_{MOC}[1]		20,000	30	**109.1**	S	LSB-vector
MLP_{DDLA}[2] [7]		20,000	50	1950.9	S	LSB
MLP_{PL}[2] [40]		20,000	50	693.8	S	LSB
MLP_{SL}[2] [40]		20,000	50	**14.5**	S	LSB
1-order CPA[1]		1,200	-	-	F	HW
2-order CPA[1] [9]		1,200	-	1188.4	S	HW
BP-CPA[1] [94]		**1,200**	-	446.9	S	HW
MLP_{DDLA}[1] [7]	CHES2018-CTF [95]	40,000	16	N/A	F	LSB
MLP_{MOR}[1]		40,000	16	N/A	S	ID
CNN_{DDLA}[2] [7]	CW-shifted	10,000	50	7069.1	S	LSB
CNN_{PL}[2] [40]		10,000	50	3983.3	S	LSB
CNN_{SL}[2] [40]		10,000	50	**31.9**	S	LSB
CNN_{DDLA}[1] [7]		10,000	50	11,030.4	S	LSB
CNN_{MOC}[1]		10,000	50	354.8	S	LSB-vector
CNN_{MOR}[1]		10,000	50	**270**	S	ID
BP-CPA[1] [94]		10,000	-	-	F	HW

[1] This work, Keras, Intel Corei5-9500, 24GB RAM; (*) S: Success; F: Failed

[2] Keras, NVIDIA GeForce GTX 1080Ti GPU, Intel Corei7-8700K, 48GB RAM.

proposed models based on MOL architecture achieve good results on different protected schemes. Concretely, the attack results on the masking dataset show that statistic-based techniques could reveal the secret key with only 1200 power traces. Unfortunately, the drawbacks of these

attacks are high memory usage required and the square order of computational complexity. By applying the BP-CPA in chapter 2, the execution time of 2-order attacks reduces significantly (approximately 2.6 times compared to conventional 2-order CPA). However, a pre-processing technique must be applied to attack high-order leakage data. Therefore, BP-CPA still take a long time and can not break other countermeasure, such as de-synchronized power traces as shown in the last row of Table 4.7.

In terms of DL-based techniques, DL-based attacks could reveal the secret key without any pre-processing techniques. Moreover, the attack time reduces dramatically. Specifically, the MLP_{SL} in [40] achieve the fastest attack (about 14.5 seconds) since it exploits the max-shared layer. However, the models in [40] improves the execution time without enhancing success rate of the attacks. It is noted that, it is not fair to compare the execution time directly between MLP_{SL} and MLP_{MOC}. However, we can compared the number of reductions of attack time in the same conditions (the computer systems, the number of traces, and the number of epochs). The next fastest technique is our proposed MLP_{MOC} models (about 109.6 seconds), as shown in the second row of the Table 4.7. Interestingly, MPL_{MOC} not only reduces the execution time, but also enhance the success rate of the attacks as described in Chapter 4. In other words, the attack results of MLP_{MOC} are more confident than MLP_{SL}. Therefore, MLP_{MOC} is more suitable in hardware security evaluation. The same conclusion can be made for CNN_{SL} and CNN_{MOC}. Based on Table 4.7, it can be seen that the main drawback of DL techniques is the requirement of a huge number of power traces compared to statistic-based techniques. In addition, the performance of the DL-based attacks

depends on the computational resources.

Moving on to another masking dataset, the proposed MLP_{MOR} is also the best candidate for performing attacks on CHES2018-CTF data. More important, MLP_{MOR} performs the attacks using the ID labeling technique. It means that the evaluation process can be implemented easily by using the intermediate values (Sbox output) directly. The results in Table 4.7 also show that MLP_{DDLA} can not reveal the secret key with the LSB labeling technique.

Regarding the de-synchronized countermeasure, only DL-based attacks can steal the secret key successfully. It is simply because CPA attacks require re-aligned power traces, whereas CNN can work well on these traces based on the translation-invariance property. In addition, by applying MOC architecture, CNN_{MOC} reduces the attack time from 5.7 hours to approximately 0.2 hours compared to CNN_{DDLA}. Another good choice is CNN_{MOR}. Similar to MLP_{MPR}, CNN_{MOR} can perform the attack using the ID labels instead of HW, HD, or LSB power models. It leads to an easy evaluation process without knowledge about power consumption models.

4.6. Disadvantages and resistance against MO-DLSCA attacks

Despite performing the SCA attack faster, proposed techniques based on MOL architectures have some drawbacks.

Firstly, the attack results could be unstable for each attack due to the multiple sources of randomness in the deep learning training process [96]. On the other hand, some metrics must be determined manually. The number of epochs, for example, is fixed for all attacks. It leads to the attack time is not optimized. As illustrated in Fig. 4.16, it is easy to see

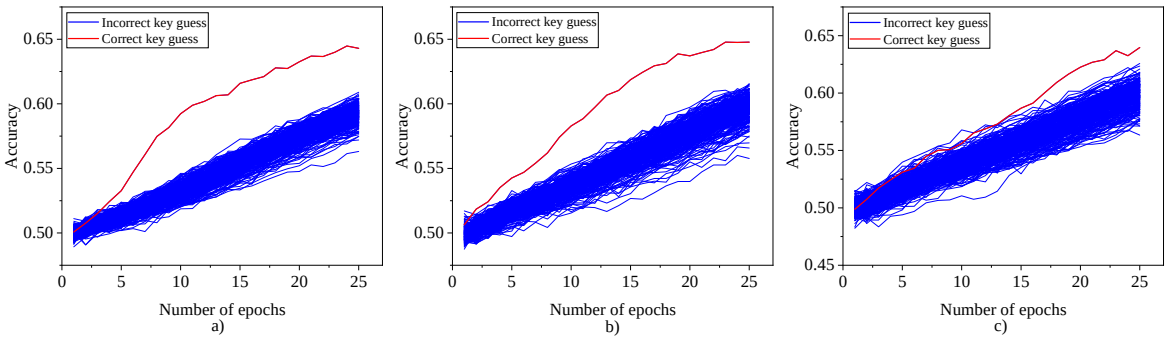

Figure 4.18: Attack results on different numbers of power traces using the MLP_{MOC} models. a) 20,000 traces; b) 10,000 traces; c) 7,000 traces

that the secret key can be discriminated clearly at epoch 20, whereas the proposed techniques determine the correct key in a fixed number of epochs (100 epochs). Obviously, it leads to a high-cost and time-consuming evaluation process. To mitigate this issue, *"early stopping"* technique should be employed in MO-DLSCA attacks to determine the correct key at a suitable number of training epochs.

Secondly, the number of measurements is very high compared to statistic-based attacks. As presented previously, BP-CPA requires only 1200 power traces to break the ASCAD data, whereas MO-DLSCA needs 20,000 measurements to perform the attacks. The main reason is that MO-DLSCA can not determine the minimum power traces to attack successfully. The experiments on ASCAD data presented in this chapter could be performed successfully with less than 20,000 measurements, as depicted in Fig. 4.18. Therefore, this is the issue that will be investigated in the future works of this thesis.

Finally, the DL model can naturally perform the combined function as described in Section. Regarding de-synchronized, CNN-based models also can deal with the leakage sample in any position of a power trace based on the translation-invariance property. However, MO-DLSCA is

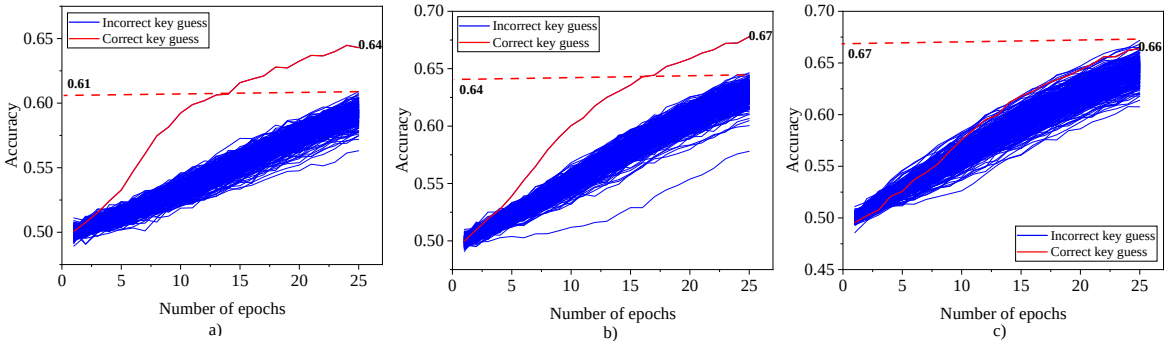

Figure 4.19: Attack results on different levels of additive noise using the MLP_{MOC} models. a) $\sigma = 0$; b) $\sigma = 1.0$; c) $\sigma = 1.5$

sensitive to the presence of additive noise compared to statistic-based SCA. Indeed, as demonstrated in [97], the authors showed that injecting noise at the input layer increases the network performance. However, in the case of non-profiled attacks, various outputs are obtained simultaneously. They consist of one output value and the other corresponding to the correct key and incorrect keys, respectively. Therefore, noise injection of the input layer increases the classification performance of all outputs, including incorrect keys. Consequently, it is difficult to discriminate between the correct and incorrect keys in this case.

Based on the disadvantages of MO-DLSCA, a suggestion for a resistance method for the cryptographic device against MO-DLSCA could be described as follows:

- Using a noise generation countermeasure for the hardware designs. It can be achieved by applying a parallel computation of a function of the cryptographic algorithm as indicated in [98]. To clarify the efficiency of noise generation countermeasure, an experiment was implemented on the noisy dataset. The attack results are illustrated in Fig. 4.19. It can be seen that when the noise is zero, the accuracy of the branch corresponding to the correct key is the highest. Other branches that

correspond to incorrect keys are low. However, when the noise increases, the accuracy of all key increase. These results demonstrate that additive noise leads to improve classification performance for the whole network. It is worst when the noise is high enough. The additive noise makes the model learn too well for all noisy data. As a result, the correct key can not be detected from the incorrect key, as illustrated in Fig. 4.19.c.

The reported results have demonstrated the efficiency of noise generation countermeasure against MLP_{MOC}. Similar results also can be achieved on MLP_{MOR}. However, these results are only obtained based on simulation datasets. The real implementation of hardware (software) of this countermeasure against MO-DLSCA will be investigated in the future works of this thesis.

4.7. Summary

In this chapter, different DL models based on the multi-output neural network are proposed. The proposals have mitigated the main drawback of DDLA by predicting the correct key after only one training process. The experimental results have indicated that the proposed models remarkably outperform the DDLA attack in terms of execution time and success rate. Specifically, the proposed MOC model reduces the execution time up to nine times compared to the MLP_{DDLA} in the case of masking countermeasure applied. Regarding MLP_{MOR}, the attack results have clarified that the identity labeling technique can be used for non-profiled DLSCA successfully. Regarding the combined countermeasure, both MOR and MOC models achieve a better success rate than that of MLP_{DDLA} by at least 20%. The experimental results have also clarified that CNN_{MOC} can break the de-synchronization countermeasure.

More interestingly, the proposed model performs SCA attacks faster, up to 40 times compared to CNN_{DDLA}. However, by using a fixed number of epochs, the attack results are not optimized. In addition, noise generation countermeasure is also investigated as a potential candidate for preventing MO-DLSCA. The results of this chapter are published in [C4], [J2], and [J4].

CONCLUSIONS AND SUGGESTIONS FOR FUTURE STUDIES

This section summarizes the contributions of the thesis and presents some open problems for future studies.

A. Conclusions

Side-channel attacks have become a realistic threat to implementations of cryptographic algorithms and received great attention from the hardware security research community. Research on SCA attacks is crucial for detecting and preventing potential hardware security problems. In addition, efficient SCA attacks play a vital role in the certification process. It helps to reduce the time to the market of products. In this thesis, effective SCA methods are proposed for reducing the computation time and enhancing the success rate of SCA security testing in different scenarios, such as high dimensional data, imbalanced datasets, and the presence of SCA countermeasures. The major contributions of the thesis can be summarized as follows.

- Two low complexity correlation power analysis (CPA) techniques called P-CPA and BP-CPA are proposed based on the correlation distribution and power trace biasing technique. Especially, the execution time of SCA security testing is reduced by approximately 2 and 2.6 times (P-CPA and BP-CPA, respectively). This contribution

is presented in [C1], [J1].

- Investigating the single-output DLSCA attacks on breaking different countermeasures without any pre-processing requirements, such as masking, noise generation, and de-synchronized. These countermeasures can not be broken by statistic-based methods, such as conventional CPA, B-CPA, and BP-CPA. Considering the remaining issues of non-profiled DLSCA, a dimensional reduction technique for DLSCA is introduced using the P-CPA method. In addition, a novel labeling technique, the so-called Significant Hamming Weight (SHW), is proposed for solving imbalanced dataset problems. The experimental results on MLP and CNN architectures have clarified that the data input dimension can reduce significantly. In addition, by applying SHW, the power traces needed for the training process reduces approximately 30% compared to LSB or 9-HW labeling technique. This contribution is presented in [C2,C3], [J3,J5] and [P1].

- A multi-output classification and multi-output regression neural networks are proposed to mitigate the drawbacks of the single-output DLSCA. The execution time of SCA security evaluation decreases significantly on different countermeasures, such as masking (9 times) and de-synchronized power trace (40 times). In addition, the success rate of SCA attacks increases by at least 20% compared to single-output architecture in the case of the presence of Gaussian noise. This contribution is presented in [C4],[J2, J4].

B. Limitations

Besides numerous effectiveness in attacking SCA data. The proposals still contain several limitations, as shown below:

Firstly, in the case of BP-CPA, the number of traces for the attack is limited because of the complexity of second-order leakage data processing. In addition, this process requires high memory usage. Therefore, in the case of a high number of second-order power traces, it is difficult to apply pre-processing techniques. In addition, other power consumption models have not been investigated. In this case, only HW model is investigated and applied to biasing power trace techniques. In means that the efficiency of proposed techniques has been clarified only on software implementations of cryptographic algorithms.

Regarding the leakage sample selection, the number of POI is determined manually. As stated previously, the number of POI is selected based on the practical attack results. In some cases, the number of POI is too small to cover the correct sample t_{ct}. Consequently, it leads to a failed attack. In other cases, the number of POI is too large. Hence, the attack time is not optimized. This issue is also investigated in the future work of this study. Furthermore, the random delay has not been investigated by BP-CPA. This countermeasure will cause a misaligned problem. Therefore, the POI extractor will not effectively work because it requires each operation of the cryptographic algorithm should be located at the same position in each power trace to find out the correlation.

Secondly, the proposed DLSCA based on multi-output architecture

is not optimized for the attack time. Another main drawback has also been indicated in Chapter 4. Accordingly, the number of measurements needed for MO-DLSCA is very high compared to statistic-based attacks.

Finally, despite several ideas for countermeasures against proposed attacks, non of the suggestions have been implemented and verified in practice.

C. Future Studies

From the analysis above, several possible open problems require further investigations in order to enhance the performance of SCA evaluation as follows:

- Firstly, future works will be directed toward investigating the suggestions of SCA countermeasures against proposed attacks in the real scenario.

- Secondly, investigate BP-CPA techniques on different SCA platforms, especially in hardware implementation.

- Finally, other advanced DL architectures, such as LSTM, and RNN, will be investigated in the SCA domain. More importantly, an online DLSCA method will be employed to reduce the attack time and determine the minimum measurements needed for DLSCA.

PUBLICATIONS

*** Published papers:**

[J1] **Ngoc-Tuan Do**, V. Hoang, and C. Pham, "Low Complexity Correlation Power Analysis by Combining Power Trace Biasing and Correlation Distribution Techniques," *IEEE Access, (SCIE, Q1, IF= 3.476)*, vol. PP, pp. 17578-17589, 2022,
doi: 10.1109/access.2022.3150833.

[J2] Van-Phuc Hoang, **Ngoc-Tuan Do**, Van Sang Doan, "Efficient Non-profiled Side Channel Attack Using Multi-output Classification Neural Network," 2022. *IEEE Embedded systems letter, (SCIE, Q2, IF= 1.524)*, doi: 10.1109/LES.2022.3213443.

[J3] **Ngoc-Tuan Do**, Van-Phuc Hoang, Van Sang Doan, Cong-Kha Pham, "On the Performance of Non-Profiled Side Channel Attacks Based on Deep Learning Techniques," 2022. *IET Information Security, (SCIE, Q2, IF= 1.3)*, doi: 10.1049/ise2.12102.

[J4] **Ngoc-Tuan Do**, Van-Phuc Hoang, Van Sang Doan, "A novel non-profiled side channel attack based on multi-output regression neural network," *J Cryptogr Eng (2023), (SCIE, Q2, IF= 1.585)*, https://doi.org/10.1007/s13389-023-00314-4.

[J5] Van-Phuc Hoang, **Ngoc-Tuan Do**, Van Sang Doan, "Performance

Analysis of Deep Learning Based Non-profiled Side Channel Attacks Using Significant Hamming Weight Labeling," 2023. *Mobile Networks and Application Journal, (SCIE, Q2, IF= 3.077).* doi:10.1007/s11036-023-02128-4

[C1] **Ngoc-Tuan Do**, Hoang, VP. "An Efficient Side Channel Attack Technique with Improved Correlation Power Analysis," *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, 2020*, vol 334. Springer, Cham. https://doi.org/10.1007/978-3-030-63083-6_22 (Scopus).

[C2] **Ngoc-Tuan Do**, V. -P. Hoang and V. -S. Doan, "Performance Analysis of Non-Profiled Side Channel Attacks Based on Convolutional Neural Networks," *APCCAS 2020*, 2020, pp. 66-69, doi: 10.1109/APCCAS50809.2020.9301673 (Scopus).

[C3] **Ngoc-Tuan Do**, V. -P. Hoang and V. -S. Doan, "Performance Analysis of Non-profiled Side Channel Attack Based on Multi-Layer Perceptron Using Significant Hamming Weight Labeling," *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, 2022*, vol 334. Springer, Cham. https://doi.org/10.1007/978-3-031-08878-0_17 (Scopus).

[C4] **Ngoc-Tuan Do**, P. C. Le, V. P. Hoang, V. S. Doan and C. K. Pham, "MO-DLSCA: Deep Learning Based Non-profiled Side Channel Analysis Using Multi-output Neural Networks," 2022 International Conference on Advanced Technologies for Communications (ATC), 2022, pp. 245-250, doi: 10.1109/ATC55345.2022.9943024.

***Patents:**

[P1] :

“ Phương pháp thực hiện phân tích kênh bên không lập mẫu sử dụng mạng nơ-ron tích chập đối với thuật toán mã mật AES 128 bit”

“Non-profiled side-channel analysis method using convolutional neural network on AES-128 algorithm”

Registration date: 30/11/2021

Number of registration: 1-2021-07694

Published date: 25/02/2022, Intellectual Property Office of Viet Nam

Author: Hoàng Văn Phúc, **Đỗ Ngọc Tuấn,** Đoàn Văn Sáng

BIBLIOGRAPHY

[1] *"Introduction" in Power Analysis Attacks: Revealing the Secrets of Smart Cards.* Boston, MA: Springer US, 2007, pp. 1–13.

[2] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Advances in Cryptology — CRYPTO' 99*, M. Wiener, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 388–397.

[3] T.-H. Le, C. Canovas, and J. Clédière, "An overview of side channel analysis attacks," in *Proceedings of the 2008 ACM symposium on Information, computer and communications security - ASIACCS '08.* ACM Press, 2008.

[4] S. Chari, J. R. Rao, and P. Rohatgi, "Template attacks," in *Cryptographic Hardware and Embedded Systems - CHES 2002*, B. S. Kaliski, ç. K. Koç, and C. Paar, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 13–28.

[5] T. Kubota, K. Yoshida, M. Shiozaki, and T. Fujino, "Deep learning side-channel attack against hardware implementations of aes," *Microprocessors and Microsystems*, p. 103383, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0141933120305408

[6] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," in *Cryptographic Hardware and Embedded Systems - CHES 2004*, M. Joye and J.-J. Quisquater, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 16–29.

[7] B. Timon, "Non-profiled deep learning-based side-channel attacks with sensitivity analysis," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2019, no. 2, pp. 107–131, Feb. 2019. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/7387

[8] T. S. Messerges, "Using second-order power analysis to attack DPA resistant software," in *Cryptographic Hardware and Embedded Systems — CHES 2000*. Springer Berlin Heidelberg, 2000, pp. 238–251.

[9] E. Oswald, S. Mangard, C. Herbst, and S. Tillich, "Practical second-order dpa attacks for masked smart card implementations of block ciphers," in *Topics in Cryptology – CT-RSA 2006*, D. Pointcheval, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 192–207.

[10] K. M. Abdellatif, "Towards efficient alignment for electromagnetic side channel attacks," in *2019 31st International Conference on Microelectronics (ICM)*. IEEE, dec 2019.

[11] A. Jia, W. Yang, and G. Zhang, "Side channel leakage alignment based on longest common subsequence," in *2020 IEEE 14th In-*

ternational Conference on Big Data Science and Engineering (BigDataSE). IEEE, dec 2020.

[12] R. Benadjila, E. Prouff, R. Strullu, E. Cagli, and C. Dumas, "Deep learning for side-channel analysis and introduction to ascad database," *Journal of Cryptographic Engineering*, vol. 10, 06 2020.

[13] P. C. Kocher, "Timing attacks on implementations of diffie-hellman, RSA, DSS, and other systems," in *Advances in Cryptology — CRYPTO '96*. Springer Berlin Heidelberg, 1996, pp. 104–113.

[14] J.-J. Quisquater and D. Samyde, "ElectroMagnetic analysis (EMA): Measures and counter-measures for smart cards," in *Smart Card Programming and Security*. Springer Berlin Heidelberg, 2001, pp. 200–210.

[15] D. Genkin, A. Shamir, and E. Tromer, "Acoustic cryptanalysis," *Journal of Cryptology*, vol. 30, no. 2, pp. 392–443, feb 2016.

[16] S. Skorobogatov, "Using optical emission analysis for estimating contribution to power analysis," in *2009 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*. IEEE, sep 2009.

[17] J. Brouchier, N. Dabbous, T. Kean, C. Marsh, and D. Naccache, "Thermocommunication," 2009, david.naccache@ens.fr 14242 received 29 Dec 2008. [Online]. Available: http://eprint.iacr.org/2009/002

[18] *Power Consumption*. Boston, MA: Springer US, 2007, pp. 27–60. [Online]. Available: https://doi.org/10.1007/978-0-387-38162-6_3

[19] P. N. Fahn and P. K. Pearson, "IPA: A new class of power attacks," in *Cryptographic Hardware and Embedded Systems*. Springer Berlin Heidelberg, 1999, pp. 173–186.

[20] *Differential Power Analysis*. Boston, MA: Springer US, 2007, pp. 119–165. [Online]. Available: https://doi.org/10.1007/978-0-387-38162-6_6

[21] A. Alipour, A. Papadimitriou, V. Beroulle, E. Aerabi, and D. Hély, "On the performance of non-profiled differential deep learning attacks against an aes encryption algorithm protected using a correlated noise generation based hiding countermeasure," in *2020 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2020, pp. 614–617.

[22] Y.-S. Won, D.-G. Han, D. Jap, S. Bhasin, and J.-Y. Park, "Non-profiled side-channel attack based on deep learning using picture trace," *IEEE Access*, vol. 9, pp. 22 480–22 492, 2021.

[23] K. Kuroda, Y. Fukuda, K. Yoshida, and T. Fujino, "Practical aspects on non-profiled deep-learning side-channel attacks against AES software implementation with two types of masking countermeasures including RSM," *Journal of Cryptographic Engineering*, mar 2023.

[24] D. Bae and J. Ha, "Performance metric for differential deep learning analysis," *J. Internet Serv. Inf. Secur.*, vol. 11, no. 2, pp.

22–33, 2021. [Online]. Available: https://doi.org/10.22667/JISIS.2021.05.31.022

[25] L. Zhang, X. Xing, J. Fan, Z. Wang, and S. Wang, "Multilabel deep learning-based side-channel attack," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, no. 6, pp. 1207–1216, jun 2021.

[26] F. Hu, H. Wang, and J. Wang, "Multi-leak deep-learning side-channel analysis," *IEEE Access*, vol. 10, pp. 22 610–22 621, 2022.

[27] H. Maghrebi, "Deep learning based side-channel attack: a new profiling methodology based on multi-label classification," *IACR Cryptol. ePrint Arch.*, vol. 2020, p. 436, 2020.

[28] D. Kwon, H. Kim, and S. Hong, "Non-profiled deep learning-based side-channel preprocessing with autoencoders," *IEEE Access*, vol. 9, pp. 57 692–57 703, 2021.

[29] C. O'Flynn and Z. Chen, "ChipWhisperer: An open-source platform for hardware embedded security research," in *Constructive Side-Channel Analysis and Secure Design*. Springer International Publishing, 2014, pp. 243–260.

[30] H. Guntur, J. Ishii, and A. Satoh, "Side-channel AttacK user reference architecture board SAKURA-g," in *2014 IEEE 3rd Global Conference on Consumer Electronics (GCCE)*. IEEE, oct 2014.

[31] A. Gohr, S. Jacob, and W. Schindler, "Ches 2018 side channel contest ctf - solution of the aes challenges," Cryptology ePrint Archive, Paper 2019/094, 2019.

[32] J.-S. Coron and I. Kizhvatov, "An efficient method for random delay generation in embedded software," in *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2009, pp. 156–170.

[33] E. Prouff, M. Rivain, and R. Bevan, "Statistical analysis of second order differential power analysis," *IEEE Transactions on Computers*, vol. 58, no. 6, pp. 799–811, jun 2009.

[34] T.-H. Le, J. Clédière, C. Canovas, B. Robisson, C. Servière, and J.-L. Lacoume, "A proposition for correlation power analysis enhancement," in *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2006, pp. 174–186.

[35] Y. Kim, T. Sugawara, N. Homma, T. Aoki, and A. Satoh, "Biasing power traces to improve correlation in power analysis attacks," *First International Workshop on Constructive Side-Channel Analysis and Secure Design (COSADE 2010)*, 01 2010.

[36] W. Hu, L. Wu, A. Wang, X. Xie, Z. Zhu, and S. Luo, "Adaptive chosen-plaintext correlation power analysis," in *2014 Tenth International Conference on Computational Intelligence and Security*, 2014, pp. 494–498.

[37] W. Unger, L. Babinkostova, M. Borowczak, and R. Erbes, "Side-channel leakage assessment metrics: A case study of GIFT block

ciphers," in *2021 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE, jul 2021.

[38] M. Xiangliang, L. Bing, W. Hong, W. Di, Z. Lizhen, H. Kezhen, and D. Xiaoyi, "Non-profiled deep-learning-based power analysis of the SM4 and DES algorithms," *Chinese Journal of Electronics*, vol. 30, no. 3, pp. 500–507, may 2021.

[39] C. O'Flynn and Z. David Chen, "Side channel power analysis of an aes-256 bootloader," in *2015 IEEE 28th Canadian Conference on Electrical and Computer Engineering (CCECE)*, 2015, pp. 750–755.

[40] D. Kwon, S. Hong, and H. Kim, "Optimizing implementations of non-profiled deep learning-based side-channel attacks," *IEEE Access*, vol. 10, pp. 5957–5967, 2022.

[41] N. Q. Tran and H. Q. Nguyen, "Efficient cnn-based profiled side channel attacks," *Journal of Computer Science and Cybernetics*, vol. 37, no. 1, pp. 1–22, 2021.

[42] Y.-S. Won, X. Hou, D. Jap, J. Breier, and S. Bhasin, "Back to the basics: Seamless integration of side-channel pre-processing in deep neural networks," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 3215–3227, 2021.

[43] M.-L. Akkar, R. Bevan, P. Dischamp, and D. Moyart, "Power analysis, what is now possible..." in *Advances in Cryptology — ASIACRYPT 2000*. Springer Berlin Heidelberg, 2000, pp. 489–502.

[44] T. Messerges, E. Dabbish, and R. Sloan, "Examining smart-card security under the threat of power analysis attacks," *IEEE Transactions on Computers*, vol. 51, no. 5, pp. 541–552, may 2002.

[45] R. Bevan and E. Knudsen, "Ways to enhance differential power analysis," in *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2003, pp. 327–342.

[46] J.-S. Coron, P. Kocher, and D. Naccache, "Statistics and secret leakage," in *Financial Cryptography*. Springer Berlin Heidelberg, 2001, pp. 157–173.

[47] R. Mayer-Sommer, "Smartly analyzing the simplicity and the power of simple power analysis on smartcards," in *Cryptographic Hardware and Embedded Systems — CHES 2000*. Springer Berlin Heidelberg, 2000, pp. 78–92.

[48] X.-D. Zeng, S. Chao, and F. Wong, "Optimization of bagging classifiers based on SBCB algorithm," in *2010 International Conference on Machine Learning and Cybernetics*. IEEE, jul 2010.

[49] C. Rechberger and E. Oswald, "Practical template attacks," in *Information Security Applications*. Springer Berlin Heidelberg, 2005, pp. 440–456.

[50] Y. Souissi, M. Nassar, S. Guilley, J.-L. Danger, and F. Flament, "First principal components analysis: A new side channel distinguisher," in *Information Security and Cryptology - ICISC 2010*. Springer Berlin Heidelberg, 2011, pp. 407–419.

[51] C. Ou, S.-K. Lam, D. Sun, X. Zhou, K. Qiao, and Q. Wang, "Snr-centric power trace extractors for side-channel attacks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, no. 4, pp. 620–632, 2021.

[52] B.-A. Dao, T.-T. Hoang, A.-T. Le, A. Tsukamoto, K. Suzaki, and C.-K. Pham, "Exploiting the back-gate biasing technique as a countermeasure against power analysis attacks," *IEEE Access*, vol. 9, pp. 24 768–24 786, 2021.

[53] W. Finnoff, F. Hergert, and H. G. Zimmermann, "Improving model selection by nonconvergent methods," *Neural Networks*, vol. 6, no. 6, pp. 771–783, jan 1993.

[54] L. Prechelt, "Early stopping — but when?" in *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2012, pp. 53–67.

[55] K. Papagiannopoulos, O. Glamočanin, M. Azouaoui, D. Ros, F. Regazzoni, and M. Stojilović, "The side-channel metrics cheat sheet," *ACM Computing Surveys*, oct 2022.

[56] M. Kerkhof, L. Wu, G. Perin, and S. Picek, "Focus is key to success: A focal loss function for deep learning-based side-channel analysis," in *Constructive Side-Channel Analysis and Secure Design*. Springer International Publishing, 2022, pp. 29–48.

[57] L. Wouters, B. Gierlichs, and B. Preneel, "On the susceptibility of texas instruments SimpleLink platform microcontrollers to non-invasive physical attacks," in *Constructive Side-Channel Analysis*

and Secure Design. Springer International Publishing, 2022, pp. 143–163.

[58] C. O'Flynn and Z. D. Chen, "Side channel power analysis of an AES-256 bootloader," in *2015 IEEE 28th Canadian Conference on Electrical and Computer Engineering (CCECE)*. IEEE, may 2015.

[59] C. Clavier, J.-L. Danger, G. Duc, M. A. Elaabid, B. Gérard, S. Guilley, A. Heuser, M. Kasper, Y. Li, V. Lomné, D. Nakatsu, K. Ohta, K. Sakiyama, L. Sauvage, W. Schindler, M. Stöttinger, N. Veyrat-Charvillon, M. Walle, and A. Wurcker, "Practical improvements of side-channel attacks on AES: feedback from the 2nd DPA contest," *Journal of Cryptographic Engineering*, vol. 4, no. 4, pp. 259–274, mar 2014.

[60] S. Chari, C. S. Jutla, J. R. Rao, and P. Rohatgi, "Towards sound approaches to counteract power-analysis attacks," in *Advances in Cryptology — CRYPTO' 99*. Springer Berlin Heidelberg, 1999, pp. 398–412.

[61] E. Prouff and M. Rivain, "Masking against side-channel attacks: A formal security proof," in *Advances in Cryptology – EUROCRYPT 2013*. Springer Berlin Heidelberg, 2013, pp. 142–159.

[62] J. Yang, J. Han, F. Dai, W. Wang, and X. Zeng, "A power analysis attack resistant multicore platform with effective randomization techniques," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 6, pp. 1423–1434, jun 2020.

[63] "Simple power analysis," in *Power Analysis Attacks*. Springer US, pp. 101–118.

[64] T. Güneysu and A. Moradi, "Generic side-channel countermeasures for reconfigurable devices," in *Cryptographic Hardware and Embedded Systems – CHES 2011*. Springer Berlin Heidelberg, 2011, pp. 33–48.

[65] N. Veyrat-Charvillon, M. Medwed, S. Kerckhof, and F.-X. Standaert, "Shuffling against side-channel attacks: A comprehensive study with cautionary note," in *Advances in Cryptology – ASIACRYPT 2012*, X. Wang and K. Sako, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 740–757.

[66] C. Herbst, E. Oswald, and S. Mangard, "An AES smart card implementation resistant to power analysis attacks," in *RoboCup 2005: Robot Soccer World Cup IX*. Springer Berlin Heidelberg, 2006, pp. 239–252.

[67] D. Jayasinghe, A. Ignjatovic, and S. Parameswaran, "Rftc: Runtime frequency tuning countermeasure using fpga dynamic reconfiguration to mitigate power analysis attacks," in *2019 56th ACM/IEEE Design Automation Conference (DAC)*, 2019, pp. 1–6.

[68] ——, "SCRIP: Secure random clock execution on soft processor systems to mitigate power-based side channel attacks," in *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, nov 2019.

[69] B. Hettwer, K. Das, S. Leger, S. Gehrer, and T. Guneysu, "Lightweight side-channel protection using dynamic clock randomization," in *2020 30th International Conference on Field-Programmable Logic and Applications (FPL)*. IEEE, aug 2020.

[70] K. Tiri, M. Akmal, and I. Verbauwhede, "A dynamic and differential cmos logic with signal independent power consumption to withstand differential power analysis on smart cards," in *Proceedings of the 28th European Solid-State Circuits Conference*, 2002, pp. 403–406.

[71] M. Joye and J.-J. Quisquater, Eds., *Cryptographic Hardware and Embedded Systems - CHES 2004*. Springer Berlin Heidelberg, 2004.

[72] T. Popp and S. Mangard, "Masked dual-rail pre-charge logic: DPA-resistance without routing constraints," in *Cryptographic Hardware and Embedded Systems – CHES 2005*. Springer Berlin Heidelberg, 2005, pp. 172–186.

[73] M. Kar, A. Singh, S. Mathew, A. Rajan, V. De, and S. Mukhopadhyay, "8.1 improved power-side-channel-attack resistance of an AES-128 core via a security-aware integrated buck voltage regulator," in *2017 IEEE International Solid-State Circuits Conference (ISSCC)*. IEEE, feb 2017.

[74] "FIP 1403 DRAFT Security Requirements for Cryptographic Modules (Revised Draft)." [Online]. Available: https://csrc.nist.gov/CSRC/media/Publications/fips/140/3/archive/2009-12-11/documents/fips140-3-draft-2009.pdf

[75] G. Goodwill, B. Jun, J. Jaffe, and P. Rohatgi, "A testing methodology for side channel resistance," 2011.

[76] D. Das, S. Maity, S. B. Nasir, S. Ghosh, A. Raychowdhury, and S. Sen, "High efficiency power side-channel attack immunity using noise injection in attenuated signature domain," in *2017 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. IEEE, may 2017.

[77] D.-H. Bui, D. Puschini, S. Bacles-Min, E. Beigne, and X.-T. Tran, "AES datapath optimization strategies for low-power low-energy multisecurity-level internet-of-things applications," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 12, pp. 3281–3290, dec 2017.

[78] K.-H. Pham, T.-H. Tran, T.-P. Nguyen, and C.-K. Pham, "An Efficient Masking Method for AES Using Tower Fields," in *2022 IEEE Ninth International Conference on Communications and Electronics (ICCE)*. IEEE, jul 2022.

[79] D. Xu, Y. Shi, I. W. Tsang, Y.-S. Ong, C. Gong, and X. Shen, "Survey on multi-output learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 7, pp. 2409–2429, 2020.

[80] V.-s. Doan, T. Huynh-the, and V.-p. Hoang, "MoDANet : Multitask Deep Network for Joint Automatic Modulation Classification and Direction of Arrival Estimation," vol. 7798, no. c, pp. 1–5, 2021.

[81] O. Reyes and S. Ventura, "Performing Multi-Target Regression via a Parameter Sharing-Based Deep Network," *International Journal of Neural Systems (2019).*, 2019.

[82] T. N. Quy and N. H. Quang, "A novel points of interest selection method for svm-based profiled attacks," *Journal of Science and Technology on Information security*, vol. 2, no. 12, pp. 45–58, 2020.

[83] N. Q. Tran, H. Q. Nguyen, and V.-P. Hoang, "Combined vmd-gso based points of interest selection method for profiled side channel attacks," in *International Conference on Industrial Networks and Intelligent Systems*. Springer, 2021, pp. 483–503.

[84] M.-L. Akkar and C. Giraud, "An implementation of des and aes, secure against some attacks," in *Cryptographic Hardware and Embedded Systems — CHES 2001*, Ç. K. Koç, D. Naccache, and C. Paar, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 309–318.

[85] E. Prouff and M. Rivain, "A generic method for secure sbox implementation," in *Information Security Applications*, S. Kim, M. Yung, and H.-W. Lee, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 227–244.

[86] Y. Zheng, Y. Zhou, Z. Yu, C. Hu, and H. Zhang, "How to compare selections of points of interest for side-channel distinguishers in practice?" in *Information and Communications Security*. Springer International Publishing, 2015, pp. 200–214.

[87] *Statistical Characteristics of Power Traces.* Boston, MA: Springer US, 2007, pp. 61–99. [Online]. Available: https://doi.org/10.1007/978-0-387-38162-6_4

[88] S. Mangard, "Hardware countermeasures against dpa – a statistical analysis of their effectiveness," in *Topics in Cryptology – CT-RSA 2004*, T. Okamoto, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 222–235.

[89] K. M. Abdellatif, D. Couroussé, O. Potin, and P. Jaillon, "Filtering-based CPA," in *Proceedings of the Fourth Workshop on Cryptography and Security in Computing Systems.* ACM, jan 2017.

[90] S. Picek, A. Heuser, A. Jovic, S. Bhasin, and F. Regazzoni, "The curse of class imbalance and conflicting metrics with machine learning for side-channel evaluations," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 209–237, nov 2018.

[91] E. Cagli, C. Dumas, and E. Prouff, "Convolutional neural networks with data augmentation against jitter-based countermeasures," in *Lecture Notes in Computer Science.* Springer International Publishing, 2017, pp. 45–68.

[92] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014.

[93] R. Groenendijk, S. Karaoglu, T. Gevers, and T. Mensink, "Multi-loss weighting with coefficient of variations," in *2021 IEEE Winter*

Conference on Applications of Computer Vision (WACV). IEEE, jan 2021.

[94] N.-T. Do, V.-P. Hoang, and C.-K. Pham, "Low complexity correlation power analysis by combining power trace biasing and correlation distribution techniques," *IEEE Access*, vol. 10, pp. 17 578–17 589, 2022.

[95] G. Perin, Ł. Chmielewski, and S. Picek, "Strength in numbers: Improving generalization with ensembles in machine learning-based profiled side-channel analysis," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 337–364, aug 2020.

[96] L. Wu, G. Perin, and S. Picek, "On the evaluation of deep learning-based side-channel analysis," in *Constructive Side-Channel Analysis and Secure Design*. Springer International Publishing, 2022, pp. 49–71.

[97] M. SABRI and T. KURITA, "Effect of additive noise for multi-layered perceptron with AutoEncoders," *IEICE Transactions on Information and Systems*, vol. E100.D, no. 7, pp. 1494–1504, 2017.

[98] N. Kamoun, L. Bossuet, and A. Ghazel, "Correlated power noise generator as a low cost dpa countermeasures to secure hardware aes cipher," in *2009 3rd International Conference on Signals, Circuits and Systems (SCS)*, 2009, pp. 1–6.