UCLouvain
Institute for Information
and Communication Technologies,
Electronics and Applied Mathematics

Information Extraction and Exploitation for Efficient Side-Channel Security Evaluations

Yuanyuan Zhou

Thesis submitted in partial fulfillment
of the requirements for the degree of
Docteur en Sciences de l'Ingénieur

Dissertation committee:

Prof. Christophe De Vleeschouwer (UCLouvain, chair)
Prof. François-Xavier Standaert (UCLouvain, advisor)
Dr. Benedikt Gierlichs (KU Leuven)
Dr. François Koeune (UCLouvain)
Dr. Eleonora Cagli (CEA Tech)

February 2023, DELFT, The Netherlands.

Contents

Contents i

List of Figures v

Acknowledgments ix

Abstract xi

1 Introduction 1
1.1 The context and motivation 1
1.2 Contributions and thesis organization 2

2 Background 9
2.1 Advanced Encryption Standard (AES) 9
2.2 Rivest–Shamir–Adleman (RSA) 11
2.3 Elliptic Curve Cryptography (ECC) 12
2.3.1 Elliptic curve 12
2.3.2 Elliptic curve cryptography 13
2.3.3 Elliptic curve scalar multiplication (ECSM) 14
2.4 Side-channel attacks 15
2.4.1 Side-channel attacks procedure 16
2.4.2 Side-channel attacks alignment 18
2.4.3 Side-channel attacks evaluation metrics 19

I Multivariate Side-Channel Security Evaluations 21

3 Worst-Case Horizontal Attacks on Scalar Multiplications 23
3.1 Systematic approach 27

3.1.1 Generic scalar multiplication architecture 27
3.1.2 Information extraction 29
3.1.3 Information combination 30
3.1.4 ECDH vs. ECDSA 31
3.2 Experimental results on Cortex-M4 32
3.2.1 Target implementation 32
3.2.2 Device and setup 35
3.2.3 Identifying and extracting the information 35
3.2.4 Information combination 37
3.3 Experimental results on Cortex-A8 39
3.3.1 Device and setup 39
3.3.2 Preprocessing 40
3.3.3 Information combination 41
3.4 Conclusion 43

4 Scatter Needs a Missing Case 45
4.1 Preliminaries and Background 47
4.1.1 Scatter transform with Chi^2/MIA distinguishers . . . 47
4.1.2 On-the-fly linear regression 50
4.1.3 Selection of parameters 51
4.2 First-order experiments 51
4.2.1 Setting #1: a simulated shuffled implementation . . . 51
4.2.2 Setting #2: a concrete jittery implementation 54
4.3 Higher-order Scatter 55
4.3.1 The need of a combination function 55
4.3.2 Second-order simulated experiments 58
4.4 Conclusion 59

II Deep Learning Side-Channel Security Evaluations 61

5 Alignment Matters for Deep Learning Side-Channel Analysis 63
5.1 Preliminaries 65
5.1.1 Target implementations 65
5.1.2 Alignment method 65
5.1.3 Deep Residual Network 66
5.1.4 Accuracy, Loss and key rank 68
5.1.5 Deep learning DPA attack 68
5.2 Warming up: results on CWL (AES-128 S-box) 68

5.2.1 Performance comparison with the state-of-the-art . . 69
5.3 Experimental results on DUT1 (AES-128 key transfer) 70
5.3.1 Performance comparison with the state-of-the-art . . 73
5.4 Experimental results on DUT2 (AES S-box) 74
5.4.1 Performance comparison with the state-of-the-art . . 77
5.5 Comparison with ASCAD_CNN_Best model using ASCAD data set 78
5.6 Conclusions 79

6 Fully Convolutional Networks to Attack Scalar Multiplication 81
6.1 Background 83
6.1.1 Fully Convolutional Networks 83
6.2 Experimental Results on BeagleBone Black Implementation running at 1GHz 85
6.2.1 Neural Network Architecture 85
6.2.2 BBB Results 85
6.3 Conclusion and Future Work 90

7 Towards More Efficient S-box Pooling Profiled Attacks 93
7.1 Background and Preliminaries 95
7.1.1 Template attack 96
7.1.2 Stochastic model attack 98
7.2 Methodology 98
7.2.1 S-box Pooling profiled attack 99
7.2.2 Knowledge of POIs assumption 100
7.2.3 Metrics and Selection of parameters 101
7.3 Experimental results 101
7.3.1 Common settings 102
7.3.2 Setting #1: an unmasked sequential AES-128 S-boxes implementation 103
7.3.3 Setting #2: a masked sequential AES-128 S-boxes implementation 106
7.3.4 Setting #3: a masked parallel AES-128 S-boxes implementation 108
7.4 Conclusion 111

Appendices
Appendix 7.A Full attack results on DUT1 112
Appendix 7.B Full attack results on DUT2 112

Appendix 7.C Full attack results on DUT3 115

III Partial Key Exposure 117

8 Extended Partial Key Exposure Attack on CRT-RSA with Additive Exponent Blinding 119
8.1 Preliminaries 122
8.1.1 Notations 122
8.1.2 Coppersmith's method 122
8.1.3 PKE attack on unblinded CRT exponents 124
8.2 Extended PKE attack on additively blinded CRT exponents . 124
8.2.1 Step 1a: Computing (k', l') from MSB using factoring. 127
8.2.2 Step 1b: Computing $(k', l'_1, ..., l'_z)$ from MSB using GCD. 130
8.2.3 Step 1c: Computing (k', l') with known LSBs. 131
8.2.4 Step 2: Factoring N with computed k'. 132
8.3 Experimental results 134
8.4 A use case of EPKE on real SCA partial key leakage 141
8.4.1 Deep learning profiled attack 142
8.4.2 Knowledge of POIs assumption 142
8.4.3 Metrics 143
8.4.4 Montgomery Ladder Exponentiation implementation 143
8.4.5 EPKE attack on SCA partial key leakage verification strategy 144
8.4.6 Attack results. 146
8.5 Conclusion and Future Work 149

9 Conclusions and Perspective 151
9.1 Multivariate side-channel security evaluations 151
9.2 Deep learning side-channel security evaluations 153
9.3 Partial Key Exposure 156

Bibliography 159

List of Figures

2.1 Illustration of AES Round Function 11
2.2 Illustration of ECDH Key Agreement 14

3.1 Conditional probability tree for a 3 bits scalar $k = (k_0, k_1, k_2)$. 25
3.2 Level view of a regular scalar multiplication. First level (top): scalar bit handling. Second: elliptic curve arithmetic. Third: Field arithmetic. Fourth: register operations. Fifth: leakages on register operations 28
3.3 Complexity of the lattice-based attack against ECDSA (for NIST P-256 curve). Number of nonces (y axis) needed when having d bits of partial information per nonce (x axis). 33
3.4 First-order success rate of the 123-bit recovery on the Cortex-M4 depending on the chunk size d for different $N = 600$. . . 38
3.5 First-order success rate of the 123-bit recovery on the Cortex-M4 depending on the number of poi N for different values of d. 39
3.6 Three EM traces before and after alignment 41

4.1 Scatter transform with MIA and Chi^2 distinguishers: high-level view. 48
4.2 Success rate on simulated shuffled implementations, with d the window size, n_i the number of POIs per window and various SNR values. 53
4.3 100 overlapped EM traces before alignment (a) and after alignment (b), and target S-box estimated SNR (c). 56
4.4 Attacks against a real implementation with strong jitter. . . . 57

4.5 Success rate on second-order simulated shuffled implementations, with d the window size, n_i the number of POIs per window and various SNR values. 60

5.1 Structure of ResNet. 67
5.2 Rank of correct subkey candidates of all 16 S-boxes with CWL data set. 69
5.3 Performance comparison of 4 NN models using CWL data set. 70
5.4 256 overlapped aligned and misaligned EM traces of DUT1. 71
5.5 SOST traces of aligned and misaligned EM traces of DUT1. . 72
5.6 ResNet Model Accuracy with DUT1 data sets. 72
5.7 Percentage of correctly identified classes with DUT1 data sets. 73
5.8 Performance comparison of 4 NN models using DUT1 data sets. 73
5.9 256 overlapped aligned and misaligned EM traces of DUT2. 75
5.10 SOST traces of aligned and misaligned EM traces of DUT2. . 75
5.11 Rank of correct subkey candidate with DUT2 data sets. . . . 76
5.12 Performance comparison of 4 NN models using DUT2 data sets. 77
5.13 Performance comparison of ResNet and ASCAD_CNN_Best models using ASCAD data set. 78

6.1 Structure of FCN. 84
6.2 FCN Model Accuracy and Loss with POI selection. 87
6.3 FCN Model Accuracy and Loss without POI selection. 89

7.1 Profiled attacks schematic diagram. 97
7.2 DUT1 Guessing Entropy results, X-axis: Number of attack traces, Y-axis: Guessing Entropy. 105
7.3 DUT2 Guessing Entropy results, X-axis: Number of attack traces, Y-axis: Guessing Entropy. 108
7.4 DUT3 Guessing Entropy results, X-axis: Number of attack traces, Y-axis: Guessing Entropy. 110
7.5 Full DUT1 Guessing Entropy results. 113
7.6 Full DUT2 Guessing Entropy results. 114
7.7 Full DUT3 Guessing Entropy results. 116

8.1 Step 1a Factoring Time for 1024-bit key. 137
8.2 Step 1a Factoring Time for 2048-bit key. 138
8.3 Step 1a Factoring Time for 3072-bit key. 138

8.4 Step 1b Estimated versus assumed probability that $l'_1, \ldots, l'_z$ are coprime. 139
8.5 Step 1b Histogram of f values with two l'_z values for 1024-bit key. 140
8.6 Step 1b Histogram of f values with two l'_z values for 2048-bit key. 140
8.7 Step 1b Histogram of f values with two l'_z values for 3072-bit key. 141
8.8 100 Overlapped Power Traces of One Exponent Bit Processing. 146
8.9 Training and Validation Accuracy. 147

Acknowledgments

FIRST of all, I am very grateful to Prof. François-Xavier Standaert (my advisor) for all his kind help and support. Otherwise, my PhD study would not start at all. We started our first co-work thanks to Prof. Yu Yu in 2012, right before I joined Brightsight. In 2014 we discussed the possibility of doing part-time PhD study under his supervision. Because I have received my Master's degree from a Chinese military university, UCLouvain did not acknowledge it. With the help of Prof. François-Xavier Standaert, Mrs Sylvie Baudine and Brightsight (especially Mr Lex Schoonen). Finally, I managed to enrol in UCLouvain.

I also must thank the owners of Brightsight, Mr Jan Pieters and Mr Dirk-Jan Out, for the financial support they paid me even though I spent time on PhD study.

In the past years, Viviane Sauvage and other PhD colleagues helped me as much as possible. Also, I would like to express my gratitude to Yu Yu, Romain Poussier, Joop van de Pol, Rob Bekkers, and Sébastien Duval for their great help.

I sincerely thank the jury members for reading my thesis and for the invaluable discussions and suggestions during and after the private defence. I am thankful to Prof. Christophe De Vleeschouwer for being the president and Dr. François Koeune for being the secretary.

Last but not least, I can never finish this 8-year long journey without the support and encouragement of my wife Sophia and my daughter Joyce. They are the source of my passion and persistence.

This work has been funded in parts by H2020 project 731591 (acronym REASSURE) and the ERC project 724725 (acronym SWORD) and the UCLouvain ARC project NANOSEC.

Abstract

SIDE-channel analysis (SCA) is one pillar of security evaluations of cryptographic algorithm implementations. Due to different widely used SCA countermeasures and new evaluation requirements of using more side-channel traces and applying deep learning techniques, more efficient side-channel information extraction and exploitation techniques are essential for security evaluations. In this thesis, I investigate such techniques in the context of SCA security evaluations.

The first part is committed to traditional multivariate techniques. Concretely, I contribute to i) a worst-case horizontal attack evaluation of elliptic curve cryptosystems (ECC) by extracting information as much as possible, ii) comparing the state-of-the-art Scatter technique against misalignment countermeasures with the linear regression technique on symmetric cryptography from the information extraction perspective.

The second part focuses on deep learning-based information extraction. More precisely, I first study the impact of alignments on deep learning SCA evaluations. Second, I simplify the previous worst-case horizontal attack on ECC thanks to the deep learning technique. Third, I propose more efficient profiling attacks via S-box pooling on symmetric cryptography algorithms and deep learning SCA.

The third part devotes to information exploitation after the SCA information extraction. Concretely, I put forward the bounds of required SCA key information extraction by extending the partial key exposure attack to CRT-RSA with additive exponent blinding countermeasure.

Chapter 1

Introduction

SCA has been pioneered by Kocher in 1996 [Koc96] and is different from classical cryptanalysis techniques. The principle of SCA is to analyse the side-channel signals (timing, power consumption, electromagnetic radiation, etc.) during the execution of the target cryptographic algorithms to reveal the secret key. Since then, SCA has been studied very intensively by academic and industrial fields from both attack and defence perspectives. To assess the security strength of cryptographic devices, as required by the certification bodies, SCA is the de facto penetration test for security evaluations.

1.1 The context and motivation

As security evaluators focusing on SCA tests, our daily work is to perform different types of SCA attacks on various cryptographic implementations to gain sufficient assurance of the security strength of those implementations. On the one hand, generally speaking, a standard SCA test is expected to finish within a one-month time-frame according to gaining so-called AVA_VAN.5 security assurance[1]. On the other hand, (i) more and more SCA countermeasures (*e.g.*, time jitters, random delays, variable internal clock, shuffling, boolean or arithmetic masking, additive or multiplicative blinding, dummy operations) are present in cryptographic devices; (ii) the certification bodies require more side-channel traces (min-

[1]https://www.sogis.eu/documents/cc/domains/sc/JIL-Application-of-Attack-Potential-to-Smartcards-v3-1.pdf

imum 1,000,000, even up to 10,000,000 in some cases); (iii) the certification bodies mandate deep learning techniques for SCA tests.

Taking those into account, we come to a natural question: "***can we more efficiently extract and/or exploit (leakage) information from the side-channel traces in a security evaluation context?***" SCA tests of security evaluations are often grey-box penetration testing. Because usually, the evaluators know the hardware and/or software design information of the targeted cryptographic implementation but do not know or have control of the involved randomness used for the masking/blinding SCA countermeasures. In security evaluations, it is mandatory by the certification bodies that the evaluation labs must review the hardware design and corresponding source code of the target cryptographic algorithm implementations to gain high security assurance. However, the involved randomness used for the masking/blinding SCA countermeasures is only generated by the internal true or pseudo random number generators, those generated random numbers are not readable or controllable from the cryptographic algorithm interface functions. It is critical in such a context that the evaluators can extract and exploit the potential side-channel leakage as much as possible within a limited time frame to gain sufficient security assurance of the target implementations.

This thesis is motivated by addressing a few specific problems of the abovementioned challenge. For information extraction, due to the intrinsic of the evaluated implementations, only multivariate side-channel leakage is potentially present in the security evaluation context. Therefore, this thesis first focused on multivariate information extraction using conventional and deep learning multivariate attacks on symmetric and asymmetric cryptographic algorithms. And then, it is focused on information exploitation from partial key exposure (PKE) attack on CRT-RSA (Chinese-Remainder-Theorem - Rivest–Shamir–Adleman) algorithm viewpoint.

1.2 Contributions and thesis organization

This thesis consists of three parts, the first part contains two chapters (Chapter 3 and Chapter 4) focusing on side-channel leakage information extraction with conventional multivariate attacks in a security evaluation context. The second part contains three chapters (Chapter 5, Chapter 6 and Chapter 7) focusing on side-channel leakage information extraction with deep learning multivariate attacks in a security evaluation context. The

last part consists of one chapter (Chapter 8), whose focus is side-channel leakage information exploitation in a security evaluation context. The main contributions in this thesis are as follows.

In Chapter 3 (based on the work published at CHES'17 [PZS17]) we further, investigate and systematize the security evaluation of scalar multiplication implementations against HDPA (Horizontal Differential Power Analysis). First of all, such attacks can potentially exploit most of the informative samples provided by a leaking implementation, and are, therefore, a natural candidate for approaching their worst-case security level, which we aim for. Second, most of the HDPA literature is based on the correlation distinguisher and assumes an a priori leakage model. Yet, given our goal to approach the worst-case security level of scalar multiplication implementations, it is natural to study efficient solutions allowing a better characterization of the leakages. Given its broad applicability in the context of block cipher implementations, linear regression appears as an excellent candidate for this purpose [SLP05], and we, therefore, study its applicability in our asymmetric setting. Third, and quite importantly, only a few practical experiments have been reported on the application of HDPA against actual implementations of scalar multiplication algorithms. We contribute to this issue by providing the results of experiments against two (one is more challenging than the other considering the noise level of the side-channel signals) targets: the first one is a low-frequency ARM Cortex M4 microcontroller (without interrupts), and the second runs a Linux operating system in the background and runs at high frequency. While successful attacks against this second target have been published for block ciphers [BGRV15, LMPT15], no public report discusses their vulnerabilities in the ECC (Elliptic Curve Cryptography) case. We also illustrate the application of the framework to both ECDH and ECDSA.

In Chapter 4 (based on the work published at COSADE'20 [ZDS20]), we start by investigating the basic potential of Scatter transform with Chi^2 and MIA distinguishers for efficient exploitation of first-order multivariate leakages. For this purpose, our seed observation is that the COSADE 2018 paper mostly compared Scatter transform with Chi^2 and MIA distinguishers with univariate CPA (Correlation Power Analysis)-based attacks. In this context, it appears natural that Scatter transform with Chi^2 and MIA distinguishers resists better to misaligned traces since the misalignment will typically spread the informative samples over multiple time dimensions (*i.e.*, a multivariate distribution). We, therefore, compare the efficiency of Scatter transform with Chi^2 and MIA distinguishers with a

more natural competitor for first-order multivariate attacks, namely the on-the-fly regression-based distinguisher described in [DPRS11]. We performed experiments against a simulated shuffled implementation and a concrete jittery implementation, both similar to the settings investigated in [TGWC18]. Our results suggest that the on-the-fly regression always outperforms Scatter transform with Chi^2 and MIA distinguishers in these contexts. We follow by studying the applicability of Scatter transform to masked implementations where computations are performed on secret-shared data. In this respect, we first show that Scatter's basic (univariate) probabilistic transform is inherently unable to characterize the higher-order multivariate statistical leakages of a masked implementation. Hence, the only possible option to deal with such cases is (as usual) to generalize Scatter transform to multivariate distributions, either by estimating these distributions directly or by combining the leakage samples in a non-linear manner (see for example [PRB09, SVO+10]). Note that the latter implies that Scatter transform cannot avoid the combinatorial explosion of the number of samples to test in order to detect Points-of-Interest (POIs). Our experiments next confirm the findings of Thiebeauld *et al.* that non-linear combination functions are beneficial for the efficiency of higher-order Scatter transform from a data complexity viewpoint [TVW19]. Since we are then back to a situation where on-the-fly regression is applicable, we finally compare both distinguishers (linear regression distinguisher and Scatter transform with MIA distinguisher) and show empirically that, as in the first-order context, Scatter transform with Chi^2 and MIA distinguishers is outperformed by linear regression in the simulated cases we studied. Overall, we cannot preclude another useful application of Scatter. But in the absence of theoretical or empirical arguments highlighting its interest over other established distinguishers, we conclude that it currently lacks a use case.

In Chapter 5 (based on the work published at JCEN'19 [ZS20]), we mitigate a tempting shortcoming in the interpretation of these past results, namely the fact that side-channel attacks based on deep learning do not benefit from re-synchronization. We insist that such a shortcoming is not induced by previous authors, in particular the CHES 2017 ones. We only consider it as a natural question to confirm whether or not such algorithms sometimes benefit from some sort of preprocessing. We believe the question is of importance since a general negative answer would significantly simplify the life of evaluation laboratories. For this purpose, we describe experimental case studies based on two protected implementa-

tions (one targeting a key transportation layer, the other targeting an AES-128 S-box), the measurements of which are affected by misalignments and hardware interrupts. In both cases, we show that the application of a re-synchronization preprocessing before the application of a deep learning algorithm allows for reducing the data complexity of the attacks. In the second case, we even consider a "compressive" alignment (*i.e.*, reducing the number of samples in the preprocessed traces because we can use a shorter interval after the alignment), which was not only necessary from the data complexity point of view but also highly beneficial from the time complexity point of view (to maintain a reasonable execution time for the deep learning attacks). From an information-theoretic viewpoint, the latter can only reduce the total amount of information in the traces, hence showing clear experimental evidence that re-synchronization can make the traces easier to exploit even for deep learning algorithms. As an additional result, we also introduce the use of Residual Networks (ResNet) as an alternative to the traditional Convolutional Neural Networks (CNN) and Multi-Layer Perceptrons (MLP) that have been previously considered in the literature. We compare its performance with the other three state-of-the-art neural network models in a side-channel context, namely the ASCAD_CNN model [PSB+18], the SPACE_CNN model [MPP16] and the SCANet model [PSK+18]. The results with all our 6 data sets with or without alignment demonstrate that this ResNet model quite systematically outperforms the other three models.

In Chapter 6 (based on the work published at SITB'19 [ZS19]), we are motivated to use deep learning techniques to simplify the complicated steps of the previous work in Chapter 3. For this purpose, we investigate the challenging case of a regular Montgomery ladder scalar multiplication implementation based on a BeagleBone Black Board running at 1 GHz. We show that a good part of the single-trace attack in Chapter 3 can be automated. It is relevant to security evaluation labs since it provides them with an easier way to conduct worst-case security evaluations of ECC cryptosystems. Yet, our results also show that some alignment and preprocessing (*i.e.*, information identification) remains necessary for the deep learning attacks to succeed against such challenging targets. Indeed, similar to the RSA case shown in the CHES'19 paper [CCC+19a] by Carbone *et al.*, they used the signal-to-noise ratio (SNR) for POI selection. Generally speaking, for asymmetric cryptography implementations, the number of sample points of target operations is usually pretty large, and POI selection is essential for deep learning attack performance. To the best of our knowledge,

this is the first public report of the deep learning-based attack on ECC implementations.

In Chapter 7 (based on the work published at ACNS'22 AIHWS workshop [ZS22]), we aim to find a more efficient way to decrease both the number of required profiling traces and the required profiling time while preserving the attack performance. The general idea of our work is S-box pooling. The first step is to reconstruct a larger set of profiling traces by pooling the profiling traces corresponding to each S-box. The second step is to execute profiling only once on this new profiling traces set. The last step is to attack all the S-boxes to disclose all the subkeys (*e.g.*, 16 in the AES-128 case) using the attack traces. First, from a data complexity point of view, the proposed S-box pooling approach can decrease the required amount of profiling traces by a factor of 8 or 16 while preserving the attack performance as shown in Table 7.2. In other words, the evaluators can measure eight or even sixteen times fewer profiling traces. The extra benefit will be reducing the preprocessing time because 8 or 16 times fewer traces need to be re-synchronized. Second, from the time complexity perspective, the proposed S-box pooling approach can decrease the required profiling time by a factor of 8 or 16 (considering the AES-128 case) while preserving the attack performance.

In Chapter 8 (based on the work published at Asiacrypt'22 [ZvdPYS22]), first, we extend the PKE attack on CRT-RSA with unblinded exponents (d_p and d_q) [MNS22] to that with additively blinded ones, *i.e.*, $d'_p = d_p + r_p(p-1)$ and $d'_q = d_q + r_q(q-1)$ for unknown random blinding factors r_p and r_q. Instead of restricting r_p and r_q to 32 bits or any lengths within the reach of brute force [FKM+06], we allow the blinding factors up to 128 bits or even more. Our extended PKE works ideally when the products of public exponent e and the blinding factors are roughly $N^{\frac{1}{12}}$. In this setting, the attackers/evaluators need only a $\frac{1}{3}$-fraction of MSBs or LSBs of d'_p and d'_q to recover the entire private key. Second, we formally analyze the asymptotic time complexity of the proposed PKE attack. In the LSB case, it runs in polynomial time under the standard Coppersmith-type heuristic assumption. In the MSB case, it either runs in sub-exponential time with a reduced input size (*i.e.*, the number to factor in this case is of size $e^2 \cdot r_p \cdot r_q$, significantly smaller than the RSA modulus N), or runs in probabilistic polynomial time (PPT) under a novel assumption that certain values corresponding to the blinded exponents are coprime with a certain probability, but with the additional cost that, for one of the expo-

nents, multiple observations of the blinded execution are needed. We implement the proposed PKE attack for the most common key sizes (1024-bit, 2048-bit, and 3072-bit) and blinding factor lengths (32-bit, 64-bit, and 128-bit), and our extensive experiments not only confirm the effectiveness of the Coppersmith-type heuristic (in the LSB case), but also that of our own assumption and the feasibility of the sub-exponential time complexity (in the MSB case). In particular, the MSB attack succeeds with the probability that randomly distributed numbers are coprime (using multiple observations of the blinded exponents), or completes mostly in seconds or minutes for most parameter settings (using single observation of the blinded exponents). In the most challenging case occasionally observed for 2048-bit (resp., 3072-bit) key with 128-bit additive blinding exponent, it finishes in less than 1.5 (resp., 50) hours. To the best of our knowledge, this is the first PKE on CRT work that experimentally verifies the effectiveness using 128-bit additive exponent blinding factors. Third, the extended PKE attack can be combined with various forms of side-channel attacks, *e.g.*, cold boot attacks [HSH+08, WCN+17], cache-timing micro-architectural attacks [YGH16, AGTB19], and timing, power or electromagnetic analysis attacks on CRT-RSA [Koc96, Sch00, Hla09, CCC+19a, ZBHV21], to significantly enhance the applicability of these attacks (by reducing the goal of full key recovery to obtaining only a $\frac{1}{3}$-fraction of leakage). Note that this PKE attack can tolerate some errors of the partial key leakage, *i.e.*, combining the error-free MSBs or LSBs of d'_p from one observation of the partial key leakage, and those of d'_q from another one. To this end, we demonstrate an application of the extended PKE attacks on deep learning-based partial side-channel leakage from a typical real-world target, *i.e.*, a Montgomery Ladder exponentiation CRT implementation with 2048-bit key and 64-bit additive exponent blinding on a 45 nm secure microcontroller with RSA co-processor.

Chapter 2

Background

In this chapter, we provide common background to the subsequent chapters. More specifically, we first present some background details on symmetric cryptography with a focus on AES in Section 2.1. Second, in Section 2.2 we describe some preliminaries of RSA asymmetric cryptography. Next, we explain some basic concepts of elliptic-curve-based asymmetric cryptography in Section 2.3. Finally, in Section 2.4 we briefly introduce side-channel attacks in the context of security evaluations, namely, the procedure, the alignment and the metrics.

In this thesis, unless stated otherwise, we use capital letters for random variables and small caps for their realizations. We use sans serif font for functions (*e.g.* F) and calligraphic fonts for sets (*e.g.* $\mathcal{A}$). We use capital bold letters for matrices (*e.g.* $\mathbf{M}$) and small bold caps for vectors (*e.g.* $\mathbf{v}$). We denote the conditional probability of a random variable A given B with $\Pr[A|B]$.

2.1 Advanced Encryption Standard (AES)

Modern cryptography contains symmetric and asymmetric cryptography algorithms, both implementations targeted for SCA tests in security evaluations. To this end, AES and Triple-DES (Data Encryption Standard) are the most common targeted symmetric algorithms in security evaluations. Both AES and Triple-DES are block cipher algorithms. A block cipher is a cryptographic primitive that consists of a pair of encryption and decryption functions. The encryption function $\mathsf{Enc} : \{0,1\}^n \times \{0,1\}^{|k|} \rightarrow$

$\{0,1\}^n, n, |k| \in \mathbb{N}$ uses a secret key $k \in \{0,1\}^{|k|}$ to transform a fixed length plaintext $p \in \{0,1\}^n$ into a fixed length ciphertext $c \in \{0,1\}^n$. The corresponding decryption function is defined as $\mathsf{Dec} : \{0,1\}^n \times \{0,1\}^{|k|} \rightarrow \{0,1\}^n$ that transforms a ciphertext c back into its corresponding plaintext p. We denote $\mathsf{Enc}_k(p)$ (resp. $\mathsf{Dec}_k(c)$) as the encryption (resp., decryption) of p (resp., c) under the key k. The block cipher is a bijective function, *i.e.*, for $\forall k \in \{0,1\}^{|k|}, \forall p \in \{0,1\}^n, \mathsf{Dec}_k(\mathsf{Enc}_k(p)) = p$.

In general, to achieve the security of computational indistinguishability from a random permutation in $\{0,1\}^n$ when the key is unknown, the design of block ciphers follows the principles of *diffusion* and *confusion* proposed by Shannon [Sha49]. *Diffusion* is to make each bit of plaintext and key influence the ciphertext bits as much as possible, while *confusion* is to make the relationship among plaintext, key and ciphertext too complicated to exploit by statistical analysis. Most block ciphers make use of iterative substitutions for *confusion* and iterative permutations for *diffusion*. Iterated block ciphers are based on the observation that many iterations of a transformation that is weak by itself can result in a strong cipher. Such a transformation is called a *round* and is constructed with different components. The intermediate values in the round transformation are usually called *states*. Substitution Permutation Network (SPN) is one of the most commonly-used round structures. AES is based on SPN. The SPN-based round transformation consists of *key addition*, *substitution* and *permutation* layers. The *key addition* layer is to insert the round key derived from the master key via a *key scheduling* algorithm in the encryption or decryption process. The *substitution* layer implements a complex non-linear transformation called S-box. Its security strength determines the security strength of the block cipher because it is the only non-linear component in the SPN structure. The *permutation* layer is responsible for diffusion by iterative permutation of the *state* bits and combining the substitution of state bits.

In 2001, NIST (National Institute of Standards and Technology) standardized AES as the successor of the (Triple-)DES symmetric algorithm. Its block size is 128 bits and its key size can be 128 bits, 192 bits, or 256 bits. The number of rounds is respectively 10, 12, or 14 depending on the key size. The states and round keys are represented as 4-by-4 matrices containing bytes in column-major order. Figure 2.1[1] displays the AES round transformation composed of four different byte-oriented operations:

- `SubBytes` (SB) is the non-linear *substitution* layer of AES, and the S-

[1] Inspired by [Jea16].

box is defined as an 8-bit multiplicative inverse in the Galois field $\mathsf{GF}(2^8)$ combined with an affine transformation.

- `ShiftRows` (SR) is the first part of the *permutation* layer of AES, and it is a simple byte-wise left circular shift operation on the last three rows without changing the first row. The second, third and fourth rows are respectively shifted by one, two and three positions.
- `MixColumns` (MC) is the second part of the *permutation* layer of AES, and it operates on the columns of the state with a reversible linear transformation: a matrix multiplication of every column. More precisely, each column is represented by a four coefficients polynomial in $\mathsf{GF}(2^8)$ and then multiplied by the polynomial $3x^3 + x^2 + x + 2$ mod $x^4 + 1$.
- `AddRoundKey` (ARK) is the *key addition* layer of AES, which mixes the state and the round key via a simple bit-wise exclusive-or (XOR) operation.

The first round is slightly different by executing an initial `AddRoundKey` operation using the master key as the round key before the `SubBytes` operation. The last round does not need the `MixColumns` operation.

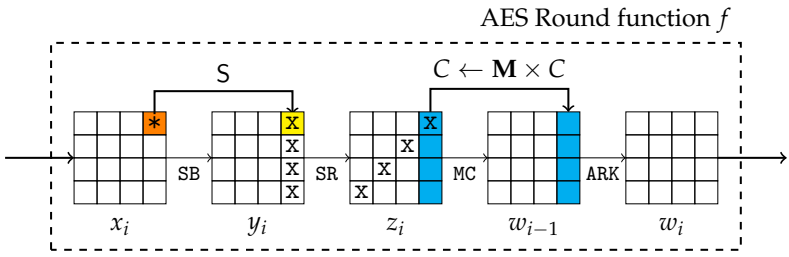


Figure 2.1: Illustration of AES Round Function

2.2 Rivest–Shamir–Adleman (RSA)

RSA [RSA78] is one of the most widely used asymmetric cryptographic algorithms. Its foundation is the hardness of integer factorization problems. RSA can be used for both encryption/decryption and digital signature schemes. Its private key is used for signature generation or ciphertext

decryption, and its public key is used for signature verification or plaintext encryption.

We denote an RSA public key as (N, e), where $N = pq$ is the modulus and $e = N^{\alpha}$ is the public exponent, and p and q are two randomly generated primes during the RSA key generation. In practice, p and q usually have the same bit-length, so they are bounded as $p, q = \Theta(\sqrt{N})$. $\phi(N) = (p-1)(q-1)$ is the Euler's totient function of N, and it fulfils $gcd(e, \phi(N)) = 1$, in which gcd corresponds to the Greatest Common Divisor. The private key $d = e^{-1} \mod \phi(N)$. Given a message C, the signature generation or decryption of C is to simply perform the private key modular exponentiation $M = C^d \mod N$. The counterpart signature verification or encryption of M is to execute the public key modular exponentiation $C = M^e \mod N$.

RSA can make use of the Chinese Remainder Theorem (CRT) optimization to speed up exponentiation operations (up to four times faster), which is especially favoured by power-constrained embedded systems, and it is called CRT-RSA. Its public key is still (N, e), while the CRT private exponents are marked as d_p and d_q ($ed_p = 1 \mod (p-1)$, $ed_q = 1 \mod (q-1)$), without loss of generality, which are assumed to be full-size, *i.e.*, $d_p, d_q = \Theta(\sqrt{N})$. The CRT private key exponentiation is executed as

$$M = (C^{d_q} \bmod q) + q \cdot \Big((q^{-1} \bmod p) \cdot \Big((C^{d_p} \bmod p) - (C^{d_q} \bmod q) \Big) \bmod p \Big)$$

, where C is the ciphertext to be decrypted, and M is the plaintext. The CRT public key exponentiation is the same as the non-CRT one, *i.e.*, $C = M^e \mod N$.

2.3 Elliptic Curve Cryptography (ECC)

2.3.1 Elliptic curve

Let $\mathbb{F}_p$ be a finite field with a characteristic bigger than 3. It is defined by $\mathcal{E}(\mathbb{F}_p)$ as the set of points $(x, y) \in \mathbb{F}_p^2$ ((x, y) called affine coordinates) that satisfy the Weierstrass equation $y^2 = x^3 + ax + b$, $(a, b) \in \mathbb{F}_p^2$ with discriminant $\Delta = -16(4a^3 + 27b^2) \neq 0$. $\mathcal{E}(\mathbb{F}_p)$ along with a point at infinity $\mathcal{O}$ form an Abelian additive group.

The addition over $\mathcal{E}(\mathbb{F}_p)$ requires field additions, subtractions, multiplications and one inversion. It is denoted by $+$ the addition of two points

P and Q, *i.e.*, $R = P + Q$, and by $[k]P$ the k-times repeated additions $P + P + ... + P$ with $k \in \mathbb{N}$ (called scalar multiplication or point multiplication). The number of points on an elliptic curve is called the order and denoted by $\texttt{ord}(\mathcal{E}(\mathbb{F}_p))$.

For $\forall P \in \mathcal{E}(\mathbb{F}_p)$, the set of its multiples generates a cyclic subgroup of $\mathcal{E}(\mathbb{F}_p)$ denoted by $\langle P \rangle$. The order of this subgroup and equivalently the order of P is denoted by $\texttt{ord}(P)$ and defined as the smallest integer n satisfying $[n]P = \mathcal{O}$. The generated cyclic subgroup with a large enough order can be used for cryptographic applications, and a given generator of such a subgroup is called a *base point*.

2.3.2 Elliptic curve cryptography

In the 1980s, Miller [Mil85] and Koblitz [Kob87] independently introduced ECC. Like other asymmetric cryptography algorithms based on a hard problem, it is based on the intractability of the Elliptic Curve Discrete Logarithm Problem (ECDLP). Given $Q \in \langle G \rangle$ for a point G with order $\texttt{ord}(G)$, the ECDLP is to find the integer $l \in [0, \texttt{ord(G)}-1]$ such that $Q = [l]G$. The ECDLP is a special case of the discrete logarithm problem (DLP) in which the cyclic group $\mathcal{G}$ is represented by the group $\langle G \rangle$ of points on an elliptic curve.

ECC is often used for ECDH (Elliptic Curve Diffie-Hellman) key exchange agreement and for ECDSA (Elliptic Curve Digital Signature Algorithm) signature generation and verification.

ECDH. As illustrated in Figure 2.2, it is a key agreement protocol to establish a shared secret key between Alice and Bob via an insecure (could be eavesdropped by a third party) channel. Initially, the domain parameters must be agreed upon. Also, each of them has his/her own ECC key pair consisting of a private key d (a randomly selected integer in the interval $[1, \texttt{ord(G)}-1]$ and a public key represented by a point Q (= $[d]G$). Alice's key pair is denoted as (d_A, Q_A), and Bob's is (d_B, Q_B). They know each other's public keys before the execution of the protocol.

Alice computes point $(x_k, y_k) = [d_A]Q_B$ and Bob computes point $(x_k, y_k) = [d_B]Q_A$. The shared secret is the x-coordinate of the resulting point x_k. The shared secret calculated by both parties is equal. Their private keys are secure unless another party can solve the ECDLP. In addition, no party other than Alice or Bob can compute the shared secret, unless that party can solve the ECDHP (Elliptic Curve Diffie-Hellman Problem). The ECDHP is to compute the point $[a][b]G$ given $[a]G, [b]G$, with unknown

$a, b \in [0, \texttt{ord(G)}-1]$ for a point G with order $\texttt{ord}(G)$.

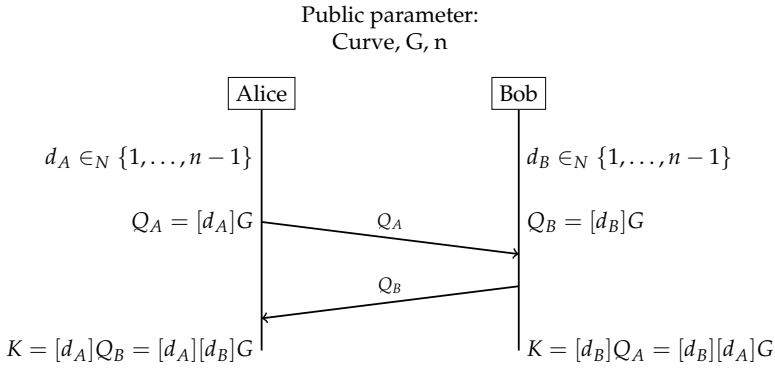


Figure 2.2: Illustration of ECDH Key Agreement

ECDSA. For the ECDSA signature generation scheme, the order n of the base point G must be prime. Similar to the ECDH case, Alice's ECC key pair is (d_A, Q_A), and Bob's is (d_B, Q_B). Alice wants to send a signed message m to Bob, she follows the steps below:

1. Compute the message digest $e = \texttt{H}(m)$ using a cryptographic hash function H with the output converted to an integer.
2. Let z be the L_n leftmost bits of e, where L_n is the bit length of n.
3. Generate a cryptographically secure random integer $k \in [1, n-1]$.
4. Calculate the curve point $Q(x, y) = [k]G$.
5. Compute $r = Q_x \mod n$. If $r = 0$, go back to step 3.
6. Calculate $s = k^{-1} \times (z + r \times d_A) \mod n$. If $s = 0$, go back to step 3.
7. The pair (r, s) is the ECDSA signature.

2.3.3 Elliptic curve scalar multiplication (ECSM)

Most elliptic curve cryptosystems require computing a scalar multiplication $[k]P$ from a number $k \in [1, \texttt{ord(P)}-1]$ and a curve point P, where $\texttt{ord}(P)$ is the order of the subgroup generated by P. In each case, k is a sensitive variable unknown to the attacker which can be a private key (*e.g.* for ECDH key exchange) or directly linked to it (*e.g.* for ECDSA). As a result, scalar multiplication represents an important source of potential side-channel leakages. To thwart the most basic side-channel attacks, scalar

multiplication algorithms avoid conditional branching and have a regular execution independently of the bits of k. In the following, we will consider the (left to right) Montgomery ladder [JY02] as described by Algorithm 1. We now view the n-bit scalar as a binary vector $\mathbf{k} = (k_0, ..., k_{n-1})$ (where k_0 is the most significant bit).

Algorithm 1 Montgomery ladder.

Input: $P, \mathbf{k} = (k_0, ..., k_{n-1})$
Output: $[k]P$
  $R_0 \leftarrow \mathcal{O}$
  $R_1 \leftarrow P$
  for $i = 0$ to $n - 1$ **do**
    $R_{1-k_i} \leftarrow R_{1-k_i} + R_{k_i}$
    $R_{k_i} \leftarrow [2]R_{k_i}$
  end for
  return R_0

Jacobian Coordinates

In general, field inversions are costly compared to additions, subtractions and multiplications. Moving from affine coordinates to Jacobian coordinates allows for avoiding the inversion when performing an addition or a doubling over $\mathcal{E}(\mathbb{F}_p)$. The Jacobian plane $\mathcal{J}$ over $\mathbb{F}_p^3$ is defined as $\{(X, Y, Z) \in \mathbb{F}_p^3 \; s.t. \; \forall \lambda \in \mathbb{F}_p, (X, Y, Z) = (\lambda^2 X, \lambda^3 Y, \lambda Z)\}$. The set of points $\mathcal{E}_{\mathcal{J}}(\mathbb{F}_p)$ defined by the equation $Y^2 = X^3 + aXZ^4 + bZ^6$ defines an elliptic curve over the Jacobian plane. The Jacobian point (X, Y, Z), $Z \neq 0$ corresponds to the affine point $(X/Z^2, Y/Z^3)$. The point at infinity in affine coordinates corresponds to the point $(\lambda^2, \lambda^3, 0)$ in Jacobian coordinates.

Given two points P and Q in $\mathcal{E}_{\mathcal{J}}(\mathbb{F}_p)$ with $P \neq \pm Q$, the formulas for the addition $P + Q$ and doubling $P + P$ are respectively given by Algorithms 2 and 3. It is worth noting that an addition over $\mathcal{E}_{\mathcal{J}}(\mathbb{F}_p)$ (resp., a doubling) requires 16 field multiplications (resp., 10).

2.4 Side-channel attacks

Until Kocher's seminal work [Koc96], it had been believed that the security of embedded cryptosystems is equivalent to the mathematical security of the involved cryptographic algorithm. Because a cryptanalyst only

Algorithm 2 Addition over $\mathcal{E}_{\mathcal{J}}(\mathbb{F}_p)$.

Input: $P = (X_1, Y_1, Z_1), Q = (X_2, Y_2, Z_2), P \neq \pm Q$
Output: $P + Q = (X_3, Y_3, Z_3)$
$\mathsf{Z}_1 \leftarrow Z_1^2, \mathsf{Z}_2 \leftarrow Z_2^2, U_1 \leftarrow X_1\mathsf{Z}_2, U_2 \leftarrow X_2\mathsf{Z}_1, H \leftarrow U_1 - U_2, S_1 \leftarrow Y_1\mathsf{Z}_2Z_2, S_2 \leftarrow Y_2\mathsf{Z}_1Z_1, R \leftarrow S_1 - S_2, \mathsf{H} \leftarrow H^2, G \leftarrow \mathsf{H}H, V \leftarrow U_1\mathsf{H}$
$X_3 \leftarrow R^2 + G - 2V$
$Y_3 \leftarrow R(V - X_3) - S_1G$
$Z_3 \leftarrow Z_1Z_2H$
return (X_3, Y_3, Z_3)

Algorithm 3 Doubling over $\mathcal{E}_{\mathcal{J}}(\mathbb{F}_p)$.

Input: $P = (X_1, Y_1, Z_1)$
Output: $P + P = (X_2, Y_2, Z_2)$
$\mathsf{X} \leftarrow X_1^2, \mathsf{Y} \leftarrow Y_1^2, \mathsf{Z} \leftarrow Z_1^2, M \leftarrow 3\mathsf{X} + a\mathsf{Z}^2, T \leftarrow \mathsf{Y}^2, S \leftarrow 4X_1Y$
$X_2 \leftarrow M^2 - 2S$
$Y_2 \leftarrow M(S - X_2) - 8T$
$Z_2 \leftarrow 2Y_1Z_1$
return (X_2, Y_2, Z_2)

has access to some inputs and/or outputs of the targeted cryptographic algorithm and potentially knows the design of the target algorithm (considering Kerckhoff's principle) to retrieve the secret key, no internal state can be observed during the execution of the target algorithm. Kocher has broken this adversary's model to observe the internal variables via side-channel(s) during the execution of cryptographic algorithms. In this context, a side-channel is an unintended communication channel that leaks some information from a device through physical observation, such as execution time [Koc96], power consumption [KJJ99], electromagnetic radiation emanation [QS01], photonic emissions [SNK+12], or acoustic emanation [GST14]. Side-channel leakage poses a new threat dimension for cryptographic algorithms if the leakage is correlated with the internal variable of the algorithm processing.

2.4.1 Side-channel attacks procedure

Concretely, SCA is a pillar for security evaluations of information security products[2], and profiled attacks are a de facto standard penetration test for

[2]https://www.sogis.eu/uk/supporting_doc_en.html

cryptographic algorithms. For security evaluations, based on the rating policy to gain so-called AVA_VAN.5 security assurance[3], such a standard profiled attacks-based penetration test typically costs 3 to 4 weeks in total for an experienced security evaluator. It is grey-box testing although the security evaluators often can get access to the detailed design information (hardware and/or software) of the implemented symmetric algorithms being evaluated. The outline of time division (in total 3 to 4 weeks) for such a standard profiled attacks-based penetration test starting from scratch is below,

- Stage 1 ($1 \sim 2$ weeks): Understand the target implementation and narrow down the target interval as much as possible through SPA (Simple Power Analysis) /SEMA (Simple EletroMagnetic Analysis) and CPA /CEMA (Correlation EletroMagnetic Analysis) (or similar techniques). It also includes the scripting time of the measurement and measurement/analysis time. It is vital to narrow down the target interval as much as possible because the sampling rate for measuring EM traces can be very high (5 GHz to 10 GHz). Hence, the number of sample points within the interesting interval can be the bottleneck of the subsequent profiled attacks. Another critical step in this stage is to choose the appropriate EM signal after surface scans of the chip. It is easier when the location of the symmetric algorithm co-processor is known to the evaluators, but if it is not known, this task can become very time-consuming.

- Stage 2 ($0.5 \sim 1$ week): Measure at least 1,000,000 (recently some certification bodies demand 5,000,000 or up to 10,000,000 because of more and more security countermeasures in the modern secure microcontrollers) profiling traces and at least 150,000 attack traces (50,000 for each of 3 different attack keys).

- Stage 3 ($1 \sim 1.5$ weeks): Preprocess the measured traces (*e.g.*, align them [PZS17]) and conduct template and deep learning profiled attacks on all S-boxes (*e.g.*, 16 in the AES-128 case) in an attempt to recover those three different attack keys (3 sets of attack traces as mentioned in Stage 2). It also includes the reporting time. More precisely, due to the implemented countermeasures such as jitters and random delays, often the evaluators have to re-synchronize (align)

[3]https://www.sogis.eu/documents/cc/domains/sc/JIL-Application-of-Attack-Potential-to-Smartcards-v3-1.pdf

the traces step by step to get closer and closer to where the S-box operations are supposed to take place [ZS20]. It commonly costs 1 or 2 days considering the number of traces and the number of sample points of every trace. Afterwards, the evaluators need to perform both template and deep learning attacks on each S-box one by one. Currently, the best practice is to execute profiling and attack each S-box one by one [GLP06]. It naturally requires lots of time taking into account the number of sample points (normally a few thousand sample points for one S-box considering the EM traces) and the amount of the traces. With deep learning evaluations now being mandated by certification bodies, this issue further amplifies. Because usually, compared to the classical template attacks, the training of the neural networks requires much more time owing to the many hyperparameters to be tuned.

2.4.2 Side-channel attacks alignment

Due to the complex architecture (and potential countermeasures) of the DUTs (Device Under Test) of security evaluations, mostly the measured power or EM traces are very misaligned. Good alignments are essential for security evaluations [ZS20]. To get better-aligned traces for evaluations, the evaluators often use the same method as in [PZS17], which exploits correlation to synchronize the side-channel traces focusing on the leakage part of targeted sensitive data (*e.g.*, the AES-128 key for the transportation case or the AES-128 S-box output for the profiled attack case). Because it is easy to implement and its affordable computation cost, it is decided to use this correlation-based alignment. Other alignment methods proposed in [BBGV16] are natural alternatives. This method works in three steps.

- First, a searching interval A that contains the operation to be synchronized is manually selected among all the traces.
- Second, a smaller reference interval B_q specific to each trace q is also manually chosen. That is, visually checking many traces to search for a common distinguishable pattern in the interval A, this common pattern probably does not exist for all the traces so some traces will be discarded during the alignments. The common distinguishable pattern is chosen as B_q.
- For each trace, we finally find the portion to be synchronized by using the second window B_q to search over the whole interval A. The

right portion is selected as the one having the maximum correlation with the reference interval. If the correlation is lower than a given threshold (chosen by the attacker/evaluator), the trace is assumed not good enough and discarded.

Note that concretely, this method was usually applied multiple times for each DUT by targeting different time intervals (for both the searching interval and reference interval). During the measurement for security evaluations, it is frequent that no very distinguishable feature close to the target interval can be used to trigger the oscilloscope. So we have to align the traces step by step to get close to the target interval, and then within the target interval, we do more local alignments. Roughly, we choose new intervals when the misalignment is getting worse, and we repeat this process until the target interval is well aligned. It allowed us to recursively improve the alignment for the part of the traces which corresponds to the target leakage. Before we measure the traces for profiling attacks, we used SPA/SEMA and CPA/CEMA techniques to narrow down the potential target interval.

2.4.3 Side-channel attacks evaluation metrics

SCA evaluation metrics are used to exploit the information after a side-channel attack has been performed to extract information, *i.e.*, to estimate the remaining computational effort required to recover the entire key for a given attack. In this thesis, we consider two commonly used evaluation metrics: *Guessing Entropy* (GE) [KB07] and *Success Rate* (SR) [SKS09]. In [Riv08], Rivain has already demonstrated that the GE can be computed from the SR at all orders.

GE. It is defined as the average rank of the subkey (with index i for a specific SCA attack) within its probability vector $\mathbf{p}_i$. In this regard, it estimates the average number of candidates to test after an attack to recover the correct key. It corresponds to the average remaining computational cost for a fixed attack strategy. More formally, given a set of n_a (*e.g.*, 100) independent attack result probability vectors ${\mathbf{p}_i}^j, i \in [0, n_s - 1], j \in [0, n_a - 1]$ (n_s corresponds to the number of subkeys of the target key, *e.g.*, 16 for an AES-128 key) containing the probabilities of the subkey i for the attack j extracted from an SCA attack, and the $\texttt{rank}(\mathbf{p}_i, k_i)$ function is defined to return the rank of the correct subkey k_i within the vector $\mathbf{p}_i$, we have the

following formula to compute *GE*.

$$\mathsf{GE} = \frac{\sum_{j=0}^{n_a-1} \mathtt{rank}(\mathbf{p}_i{}^j, k_i)}{n_a} \tag{2.1}$$

SR. The success rate of order o or oth-order SR ($o \in [1, 2^a]$ and a is the bit-length of a subkey), is defined as the probability that the correct subkey k_i is ranked among the top o ones for a probability vector $\mathbf{p}_i$ extracted from an SCA attack. In this regard, it estimates the probability of success for both a fixed attack strategy and computational cost. Similar to the GE case, given a set of n_a independent attack result probability vectors $\mathbf{p}_i{}^j$ from an SCA attack, and the $\mathbb{1}^o(\mathbf{p}_i, k_i)$ function is defined to return 1 if the correct subkey k_i is ranked among the top o ones and to return 0 otherwise. The formula to compute oth-order SR is as follows.

$$o\mathit{th} - \mathit{order}\ \mathsf{SR} = \frac{\sum_{j=0}^{n_a-1} \mathbb{1}^o(\mathbf{p}_i{}^j, k_i)}{n_a} \tag{2.2}$$

In this thesis, we only consider the most commonly used first-order *SR*.

Part I

Multivariate Side-Channel Security Evaluations

Chapter 3

Worst-Case Horizontal Attacks on Scalar Multiplications

THE wide number and variety of side-channel attacks against scalar multiplication algorithms makes their security evaluations complex, in particular in case of time constraints making exhaustive analyses impossible. In this chapter, we present a systematic way to evaluate the security of such implementations against horizontal attacks. As horizontal attacks allow extracting most of the information in the leakage traces of scalar multiplications, they are suitable to avoid risks of overestimated security levels. For this purpose, we additionally propose to use linear regression in order to accurately characterize the leakage function and therefore approach worst-case security evaluations. We then show how to apply our tools in the contexts of ECDSA and ECDH implementations, and validate them against two targets: a Cortex-M4 and a Cortex-A8 microcontrollers.

State of the art. The secure implementation of ECC is an important ingredient in modern information systems. In this chapter, we are concerned with side-channel attacks against scalar multiplication implementations which have been the focus of continuous interest over the last twenty years. We informally divide these attacks in two main categories: attacks using a *Divide and Conquer* (DC) approach and attacks using an *Extend and Prune* (EP) approach – which we next survey.

Attacks that belong to the first category aim at recovering the scalar bits independently and are therefore simple to analyze. They associate a probability or a score to each scalar bit. The scalar is trivially recovered if all the correct bits have the highest probability. If it is not the case, computational power can be used to mitigate the lack of side-channel information thanks to key enumeration [VGRS12,MOOS15,PSG16]. If the key is beyond computational reach (*e.g.*, beyond 2^{60}), rank estimation (which requires the knowledge of the key and is therefore only accessible to evaluators, not to concrete adversaries) allows estimating the computational security level of a leaking implementation [VGS13, BLvV15, GGP+15, MOOS15].

Attacks using an EP approach recover the scalar bits in a recursive manner. Recovering the i-th bit requires to first recover all the previous ones. For an n-bit scalar, EP attacks can be seen as a probability tree with 2^n leaves where each level corresponds to a different bit. Figure 3.1 illustrates such a probability tree for $n = 3$, where each node corresponds to the conditional probability of a bit given that the previous ones are correctly recovered. In this context, a first attack strategy is to only look at the most probable tree path. We refer to this method as first-order Success Rate (1-O SR). This strategy fails if not enough side-channel information is available for at least one of the bits. In such a case, the aforementioned enumeration algorithms cannot be applied due to the conditional dependencies. Yet, a recent study [LvVW14] describes a method to apply key enumeration/rank estimation on EP attacks.[1]

According to these two classes, we now describe the state of the art on side-channel attacks against scalar multiplications. For each of them, we describe its overall modus operandi, we show its associated class and we finally exhibit a countermeasure (if existing).

The first attacks discovered on scalar multiplication, such as timing and SPA [Koc96, KJJR11], aimed at finding different patterns that depend on a scalar bit value. Such differences occur if the scalar multiplication or the elliptic curve operations are irregular, which makes the timing pattern of the leaking implementations dependent on all the scalar bits. As a consequence, an adversary has to recover the scalar in a recursive manner and these attacks belong to the EP class. Using a regular scalar multiplication

[1] Instead of only looking at the most probable path, it assumes that the tree can be divided into three parts. The first part corresponds to the bits which the adversary is certain to have recovered correctly. The second part contains the (computationally feasible) paths on which the adversary has partial information. The final part is the exhaustive remaining part of the tree on which the adversary has no information. Enumeration is done on this sub-tree using a Pollard-like method.

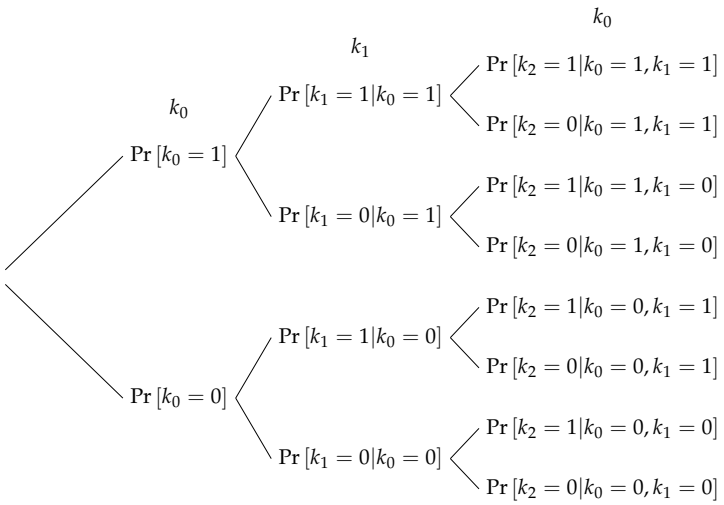


Figure 3.1: Conditional probability tree for a 3 bits scalar $k = (k_0, k_1, k_2)$.

as described in Section 2.3 naturally thwarts them.

Later on, (vertical) DPA [KJJR11] and CPA [BCO04] were applied to scalar multiplications [Cor99,MDS99]. Such methods aim at recovering the scalar bits iteratively using several side-channel leakages traces of a scalar multiplication with a fixed scalar and several known inputs. Thanks to the inputs knowledge, the scalar bits are recovered by guessing one internal value that depends on the input and the guessed bits. Independently of the distinguisher (*e.g.* correlation, difference of means, ...), we refer to all attacks using this framework as DPA. Here again, guessing the internal value associated with a scalar bit requires that all the previous bits have been correctly recovered. Hence, DPA also belongs to the EP class. Since this method requires leakages on several executions with a fixed scalar, scalar randomization techniques [Cor99, CJ01] are efficient countermeasures. Protocols using a scalar nonce such as ECDSA [PUB00] are naturally protected against DPA.

Next, Template Attacks (TA) [CRR02] were introduced as a powerful tool to extract all the information available in some leakage traces. In the context of scalar multiplication, it has been first introduced against ECDSA [MO08]. From the prior knowledge of the input, the attack computes 2^d templates in order to recover the first d bits of the scalar nonce. The actual bit of the nonce is then found with templates matching. The

secret key is finally recovered using lattice techniques [NS03, BvdPSY14] from several partially known nonces and their associated signatures. Interestingly, since such TA do not recover the whole scalar, they neither belong to the EP nor DC classes.

Alternatively, Online Template Attacks (OTA) [BCP+14, DPN+16] were introduced to recover the full scalar. This method interleaves the templates building and the attack phases, thus requiring more online access to the target device. As the iterative template building requires the knowledge of the current internal state which depends on the previous bits, OTA belongs to the EP class. Since both TA and OTA require the knowledge of the input, point randomization techniques [Cor99, JT01] are effective. Using the fact that $(X, Y, Z) = (\lambda^2 X, \lambda^3 Y, \lambda Z)$ in Jacobian coordinates (see section 2.3.3), an option to randomize the input is to change its coordinates at the beginning of each scalar multiplication using a random λ.

HDPA [CFG+10, BJPW13a] are another powerful alternative to TA. From the posterior knowledge of the input, the scalar bits are recovered by guessing several internal values that depend on both the input and the guessed bits (instead of only one in the case of DPA). As guessing the internal values associated with a particular bit requires the knowledge of the previous ones, HDPA belongs to the EP class. As for TA, point randomization is effective against HDPA.

Finally, Horizontal Collision Attacks (HCA) [Wal01, HKT15, DGH+16, BJPW13a, BJPW13b, CFG+12] have been proposed in order to bypass the point randomization countermeasures. These attacks require the scalar multiplication to exhibit different operand input/output collisions that depend on the scalar bit values. Being able to detect such collisions from a trace allows recovering the scalar bits. Up to our knowledge, only one collision attack defeats the Montgomery scalar multiplication implemented as described in Section 2.3.3 by Algorithm 1 [HKT15]. The collision is able to find whether two consecutive scalar bits are the same or not. Given this information, the scalar is finally recovered using a modified version of the Baby-Step/Giant-Step algorithm. If they target a regular scalar multiplication, HCA belong to the DC class, since the position of the collisions in the leakage traces depends only on the current scalar bit value in this case. In case of irregular algorithms, this position depends on the previous bits and HCA then belong to the EP class. In general, countermeasures aiming at increasing the noise level (*e.g.*, shuffling, random delays, ...) are the only effective ones against the attack presented in [HKT15]. In this respect, one drawback of these attacks (from the adversary's viewpoint) is that they

only exploit a very small part of the available information compared to HDPA and TA (which implies that the amount of noise needed to hide the collisions is more limited).

We additionally mention that all the aforementioned attacks (except the TA) ignore the leakages due to the direct manipulation of the scalar bits during the scalar multiplication (such as discussed in [HMH+12, Che14, NCOS16]). Exploiting these leakages is an orthogonal concern to this study and is therefore out of the scope of our investigations.

Based on this broad state of the art, we further investigate and systematize the security evaluation of scalar multiplication implementations against HDPA. The rest of the chapter is organized as follows. Section 3.1 describes the generic view we consider for regular scalar multiplication along with the systematic security evaluation method. Finally, Section 3.2 and Section 3.3 respectively show the experimental results for the application of this framework against our two targets.

Note that the Montgomery ladder algorithm (see Algorithm 1 in Section 2.3.3) targeted in this chapter is typically reflective of the state of the art implementations of ECC secure against SPA. In the following, we further considered implementations protected with scalar randomization in order to avoid DPA attacks. If scalar randomization is present, it requires the knowledge of scalar blinding factor or targeting some curves with sparse group order [FRV14] during the profiling phase. So our focus is on single-trace attacks which naturally goes with our worst-case information extraction motivation.

3.1 Systematic approach

In this section we describe a systematic method for the worst-case security analysis of scalar multiplications. We first give an abstract view of regular scalar multiplications. We then use this abstraction to specify the amount of informative points in our leakage traces. We finally show how these informative points can be extracted and combined to attack the scalar bits, and show how this method applies on two ECC primitives, namely ECDH and ECDSA.

3.1.1 Generic scalar multiplication architecture

As explained in Section 2.3.3, elliptic curve cryptosystems require to compute a scalar multiplication. We also previously discussed the regularity

requirements of the scalar multiplication implementations, which implies that they can be described as a fixed and predictable sequence of operations. In this context, all operations that affect the internal state depending on the scalar bit value contain sensitive information. In order to quantify this information, we will next describe the scalar multiplication based on different levels, depicted in Figure 3.2.

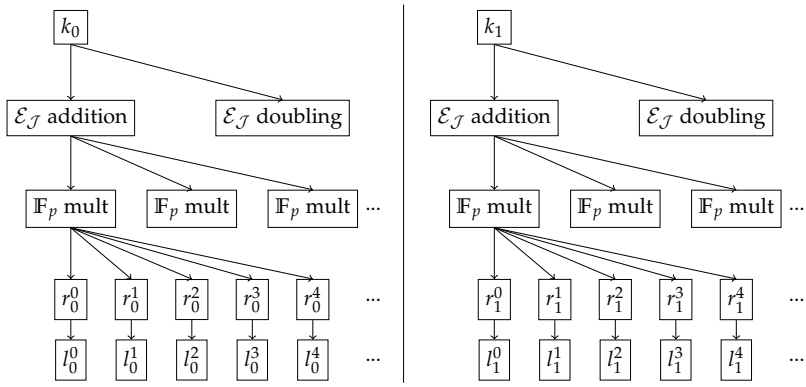


Figure 3.2: Level view of a regular scalar multiplication. First level (top): scalar bit handling. Second: elliptic curve arithmetic. Third: Field arithmetic. Fourth: register operations. Fifth: leakages on register operations

At the top level, a regular binary scalar multiplication is an iterative processing of the scalar bits. Each bit handling is itself composed of a fixed number of additions and doublings. Then, each addition (resp., doubling) contains a fixed number of field operations (such as field additions, subtractions and multiplications). Finally, each field operation is composed of a fixed number of register operations (such as register additions, subtractions and multiplications). As a result, and for an n-bit scalar, the sequence of register operations that can be exploited by an adversary can be divided into n parts that depend on the scalar bit index. Independently of the kind of operation, we assume that each part contains N register operations. We therefore have that a regular binary scalar multiplication leads to n sequences of N sensitive operations of which the results are stored in registers. We denote as $\mathbf{r}_i = (r_i^j)$, $j \in [0, N-1]$ the N intermediate computation results occurring during the manipulation of the i-th scalar bit.[2]

[2]Note that this is an abstract view. In practice, an operation can have more than one reg-

Eventually, each of these computations will lead to side-channel leakages denoted as $\mathbf{l}_i = (l_i^j)$, $j \in [0, N-1]$.

3.1.2 Information extraction

From the previous abstract view of the scalar multiplication and its associated leakages, the next step is to extract the information. Given a leakage l on a register r, we compute the probability $\Pr[l|r = x], x \in [0, 2^{|r|} - 1]$ that the observed leakage is caused by the manipulation of the value x by r (where $|r|$ denotes the size of the register in bits). While using templates would be optimal, their computational complexity is exponential in $|r|$ as it requires to estimate $2^{|r|}$ Probability Density Functions (PDF) per register (*e.g.*, $2^{|r|}$ means and variances for Gaussian templates). Therefore, and as a more efficient alternative, we use linear regression [SLP05] with a linear basis containing the $|r|$ bits of the registers. The latter decreases the profiling complexity from $O(2^{|r|})$ to $O(|r|)$, which is particularly interesting in the case of a $r = 32$-bit architecture (as we will investigate). Yet, this admittedly comes at the cost of a potential information loss in case the leakage function has informative non-linear terms (which we will briefly discuss in Sections 3.2 and 3.3).

We denote by $\lceil \mathbf{k} \rceil_i$ the first i bits of $\mathbf{k}$. We denote by $r_i^j(P, \lceil \mathbf{k} \rceil_i)$ the internal value processed by the register operation r_i^j when the input point is P and the first i bits of $\mathbf{k}$ are equal to $\lceil \mathbf{k} \rceil_i$. Similarly, we denote by $l_i^j(P, \lceil \mathbf{k} \rceil_i)$ the leakage corresponding to its manipulation. Using N_{prof} leakages from random inputs (P^q) and random scalars $(\mathbf{k}^q)$, $q \in [0, N_{prof} - 1]$, we compute the $N_{prof} \times 33$ matrix $\mathbf{A}$ where the q-th line is the binary representation of $r_i^j(P^q, \lceil \mathbf{k}^q \rceil_i)$ appended with a constant 1. From $\mathbf{A}$ and the vector $\mathbf{l}_i^j = (l_i^j(P^q, \lceil \mathbf{k}^q \rceil_i))$, $q \in [0, N_{prof} - 1]$ containing the N_{prof} leakages, we compute the linear basis $\mathbf{b}$ that characterizes l_i^j using Equation (3.1) (where $\mathbf{A}^\top$ denotes the transpose of $\mathbf{A}$):

$$\mathbf{b} = (\mathbf{A}^\top \mathbf{A})^{-1} \mathbf{A}^\top \mathbf{l}_i^j. \tag{3.1}$$

Assuming that the noise follows a Gaussian distribution, the variance σ^2 is computed using a second set of traces (in order to avoid overfitting) as shown by Equation (3.2). Note that we assume independence between

ister input/output. In such case, one can count this operation as corresponding to several registers or only use one of them.

the noise and the value processed by the register [LPB+15].

$$\sigma^2 = (\mathbf{l}_i^j - \mathbf{Ab})^\top \cdot (\mathbf{l}_i^j - \mathbf{Ab}). \tag{3.2}$$

From $\mathbf{b}$ and σ^2, the probability that a leakage l_i^j comes from the manipulation of x by r_i^j is then given by Equation (3.3), where $\mathcal{N}_{\mu,\sigma^2}(\mathrm{l})$ denotes the evaluation in l of the normal density function with mean μ and variance σ^2, and $(x)_2$ denotes the binary decomposition of x appended with 1:

$$\Pr\left[l_i^j \,\middle|\, x\right] = \mathcal{N}_{(x)_2 \cdot b, \sigma^2}(l_i^j). \tag{3.3}$$

3.1.3 Information combination

Now that we are able to identify and extract the information, we aim at combining it to attack the scalar bits. As shown above, HDPA belong to the EP class. In order to exploit the information on a specific bit, a hypothesis on the values of the previous ones must be made.

From the knowledge of input P and the knowledge of the first $i-1$ bits, the likelihood that the bit i is equal to 1 (resp., 0) is computed by guessing all the register values $r_i^j(P, \lceil \mathbf{k} \rceil_{i-1} || 1)$ (resp. $r_i^j(P, \lceil \mathbf{k} \rceil_{i-1} || 0)$ and combining their likelihoods. Given the leakages $\mathbf{l}_i$ on the i-th bit, Equation (3.4) gives the probability that the leakages come from the first $i-1$ bits being equal to $\lceil \mathbf{k} \rceil_{i-1}$ and the i-th bit being equal to $v \in \{0, 1\}$ (where $||$ denotes the concatenation):

$$\Pr\left[\mathbf{l}_i \,\middle|\, \lceil \mathbf{k} \rceil_i = \lceil \mathbf{k} \rceil_{i-1} || v\right] = \prod_{j=0}^{N-1} \Pr\left[l_i^j \,\middle|\, r_i^j(P, \lceil \mathbf{k} \rceil_{i-1} || v)\right] \cdot \tag{3.4}$$

This formula assumes that the leakages l_i^j are independent (which simplifies the profiling phase as we only consider univariate samples). While this assumption may not be perfectly correct, we verified experimentally that considering multivariate samples did not improve the efficiency of our attacks in the next section, and therefore assume it to be sufficiently correct for simplicity. Extending this framework to attack k by chunks of d bits is straightforward. Given a d bits hypothesis vector $\mathbf{v}$, Equation (3.5) extends the previous equation to attack d bits at a time, where $\mathbf{v}_{|\mathbf{j}}$ denotes the j first

elements of $\mathbf{v}$:

$$\Pr\left[\mathbf{l}_i, ..., \mathbf{l}_{i+d-1} \;\middle|\; \lceil \mathbf{k} \rceil_{i+d-1} = \lceil \mathbf{k} \rceil_{i-1} || \mathbf{v}\right] = \prod_{j=0}^{d-1} \Pr\left[\mathbf{l}_{i+j} \;\middle|\; \lceil \mathbf{k} \rceil_{i-1} || \mathbf{v}_{|\mathbf{j}}\right] . \tag{3.5}$$

We will refer to $\Pr\left[\mathbf{l}_i, ..., \mathbf{l}_{i+d-1} \;\middle|\; \lceil \mathbf{k} \rceil_{i+d-1} = \lceil \mathbf{k} \rceil_{i-1} || \mathbf{v}\right]$ as $\Pr_{\lceil \mathbf{k} \rceil_{i-1} || \mathbf{v}}$ when there is no ambiguity, it is only for readability reasons.

3.1.4 ECDH vs. ECDSA

In the rest of this chapter we always assume that $\mathbf{k}$ is attacked by chunks of d bits. For simplicity, we assume that $n_d = \frac{n}{d} \in \mathbb{N}$ and we rewrite $\mathbf{k} = (\mathbf{k}_0, ..., \mathbf{k}_{n_d - 1})$ being viewed as binary vectors of d elements.

ECDH.

In order to attack ECDH, the attacker has to recover the full scalar in one trace. He starts by attacking the first d bits using Equation (3.5). Using a leakage from a known input, he computes the likelihoods $\Pr_{\mathbf{v}}$ for all the 2^d scalar hypotheses $\mathbf{v} \in \{0,1\}^d$. He then selects the hypothesis $\mathbf{k}_0^* = \mathsf{argmax}_{\mathbf{v}}(\Pr_{\mathbf{v}})$ that maximizes the likelihood as being the correct guess. The following scalar guesses are selected iteratively according the previous results as $\mathbf{k}_i^* = \mathsf{argmax}_{\mathbf{v}}(\Pr_{\mathbf{k}_0^* || ... || \mathbf{k}_{i-1}^* || \mathbf{v}})$, with $\mathbf{v} \in \{0,1\}^d$. The iterative process ends when the adversary has obtained a full scalar hypothesis $\mathbf{k}^* = (\mathbf{k}_0^*, ..., \mathbf{k}_{n_d-1}^*)$.

The attack trivially succeeds if $\mathbf{k}^* = \mathbf{k}$, which corresponds to a 1-O SR. If this is not the case, one can use computational power to mitigate the lack of information by enumerating through the other paths [LvVW14]. In this study we only look at the 1-O SR as our focus is on optimal information extraction (rather than exploitation).

Eventually, the chunk size d is a parameter chosen by the adversary. It increases the attack complexity exponentially in d, but allows increasing linearly the amount of leakage samples exploited by the adversary.

ECDSA.

As for ECDH, a potential strategy to attack the ECDSA scalar nonce is to recover all its bits. Another option is to use the algebraic relation between the nonce and the secret key. Using lattice techniques, one can attack the

secret key by recovering partial on the first d bits of several nonces. Note that, in a security evaluation context, this lattice attack step is not required. Security evaluators only need to correctly recover a few bits of the nonces, and then assess how many attack traces are needed for such a key length according to the number of correctly recovered bits. This strategy fails if at least one of the nonces' partial information is not recovered properly. As a consequence, the attacker has to make sure that the d-bit partial information of each nonce is correct. As the attack on the first d bits does not suffer from the conditional dependencies, one can turn the previously estimated likelihoods into true probabilities by applying Bayes' theorem, as in Equation (3.6). From these probabilities, the adversary decides to ignore all the results having a probability lower than a given threshold to maximize the success of the lattice attack.

$$\Pr\left[\mathbf{k}_0 = \mathbf{v} \,\middle|\, \mathbf{l}_0, ..., \mathbf{l}_{d-1}\right] = \frac{\Pr_{\mathbf{v}}}{\sum_{\mathbf{v}^* \in \{0,1\}^d} \Pr_{\mathbf{v}^*}}. \tag{3.6}$$

To give more insight on the side-channel requirements of a lattice-based attack against ECDSA, Figure 3.3 shows how many nonces are needed (y axis) in function of the partial information d (x axis). As we can see, the number of required nonces decreases exponentially when d increases. It confirms the need of being able to extract most of the information and to discard the wrong results in a meaningful way. We use the fplll [dt16] library v4.0.4 with block sizes of 20 and 30 (only for 4 bits leaked case) to perform the experiments.

3.2 Experimental results on Cortex-M4

In this section we apply previous systematic approach to attack ECC implementations in a 32-bit Cortex-M4 microcontroller. We first describe our implementation, device and the measurement setup. We then follow our evaluation framework step by step and discuss experimental results.

3.2.1 Target implementation

We implemented the finite field and elliptic curve arithmetic in assembly on both chips. We chose the NIST P-256 curve [PUB00]. Our attack framework is independent of the choice of curve, as its only requirement is the regularity of the implementation. We thus focused on achieving constant

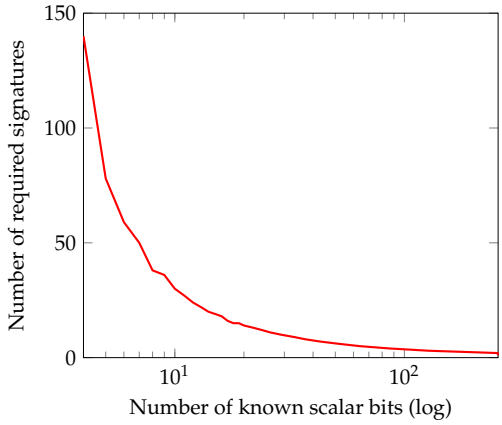


Figure 3.3: Complexity of the lattice-based attack against ECDSA (for NIST P-256 curve). Number of nonces (y axis) needed when having d bits of partial information per nonce (x axis).

time without optimizations. We implemented the Montgomery ladder as described in Section 2.3.3 using Jacobian coordinates. We used the addition and doubling formulas from Algorithm 2 and 3. The whole scalar multiplication runs in approximately 17,000,000 clock cycles.

Field additions and subtractions are implemented in a straightforward manner using carry additions and subtractions. Field multiplications are done using the Long Integer Multiplication (LIM) followed by a modular reduction. LIM is easily implemented using the 32 bits unsigned multiplication umull and umaal (with accumulate) assembly instructions producing a 64 bits result. Using the fact that the field characteristic is a generalized Mersenne prime number, the modular reduction is done as described in [PUB00].

Algorithm 4 shows how we perform LIM in assembly from two 256 bits numbers represented as eight 32-bit words $a = (a_0, ..., a_7)$ and $b = (b_0, ..., b_7)$, where a_0 and b_0 denote the least significant words and mod denotes the modular operator. Note that a and b are stored in RAM, and r_i denotes registers.

Algorithm 4 Assembly long integer multiplication.

Input: $a = (a_0, ..., a_7), b = (b_0, ..., b_7)$
Output: $c = a \times b = (c_0, ..., c_{15})$

$c \leftarrow 0$
$\mathrm{r}_2 \leftarrow \mathsf{load}(a_0)$
$\mathrm{r}_3 \leftarrow \mathsf{load}(b_0)$
$\mathrm{r}_0, \mathrm{r}_1 \leftarrow \mathsf{umull}(\mathrm{r}_2, \mathrm{r}_3)$
$c_0 \leftarrow \mathsf{store}(\mathrm{r}_0)$
for $i = 1$ to 7 **do**
  $\mathrm{r}_3 \leftarrow \mathsf{load}(b_i)$
  $\mathrm{r}_{1-i \bmod 2} \leftarrow 0$
  $\mathrm{r}_{i \bmod 2}, \mathrm{r}_{1-i \bmod 2} \leftarrow \mathsf{umaal}(r_2, r_3)$
  $c_i \leftarrow \mathsf{store}(\mathrm{r}_{i \bmod 2})$
end for
$c_8 \leftarrow \mathsf{store}(\mathrm{r}_0)$
for $i = 1$ to 7 **do**
  $\mathrm{r}_2 \leftarrow \mathsf{load}(a_i)$
  $\mathrm{r}_3 \leftarrow \mathsf{load}(b_0)$
  $\mathrm{r}_0 \leftarrow 0$
  $\mathrm{r}_1 \leftarrow 0$
  $\mathrm{r}_0, \mathrm{r}_1 \leftarrow \mathsf{umull}(\mathrm{r}_2, \mathrm{r}_3)$
  $\mathrm{r}_3 \leftarrow \mathsf{load}(c_i)$
  $\mathrm{r}_0 \leftarrow \mathsf{adds}(\mathrm{r}_0, r_3)$
  $c_i \leftarrow \mathsf{store}(\mathrm{r}_0)$
  for $j = 1$ to 7 **do**
    $\mathrm{r}_3 \leftarrow \mathsf{load}(b_j)$
    $\mathrm{r}_{1-j \bmod 2} \leftarrow 0$
    $\mathrm{r}_{j \bmod 2}, \mathrm{r}_{1-j \bmod 2} \leftarrow \mathsf{umaal}(r_2, r_3)$
    $\mathrm{r}_3 \leftarrow \mathsf{load}(c_{i+j})$
    $\mathrm{r}_{j \bmod 2} \leftarrow \mathsf{adcs}(\mathrm{r}_{j \bmod 2}, \mathrm{r}_3)$
    $c_{i+j} \leftarrow \mathsf{store}(\mathrm{r}_{j \bmod 2})$
  end for
  $\mathrm{r}_0 \leftarrow \mathsf{adcs}(\mathrm{r}_0, 0)$
  $c_{8+i} \leftarrow \mathsf{store}(\mathrm{r}_0)$
end for
return c

3.2.2 Device and setup

Our first target is a 32-bit ARM® Cortex-M4 microcontroller from the Atmel SAM4C-EK evaluation kit.[3] It proceeds most instructions in constant time and does not have any programs running in parallel that could disturb the signal. Moreover, this microcontroller runs at 100 MHz which makes it a relatively easy target for power acquisition.

We monitored the voltage variation using a 4.7 Ω resistor inserted in the supply circuit of the chip. We performed the trace acquisition using a Lecroy WaveRunner HRO 66 ZI oscilloscope running at 200 megasamples per second. For each scalar multiplication execution, we triggered the measurement at the beginning of the execution and recorded the processing of the first 123 bits of the scalar (due to the memory size of the used oscilloscope). Each trace contains 40,000,000 time samples and weighs 320,000,000 mega-bytes (written in double precision format because of the 12-bit ADC of the used oscilloscope).

Since the device does not suffer from any interruptions, the power traces are directly used without any preprocessing.

3.2.3 Identifying and extracting the information

Among all the register operations $\mathbf{r}_i$ available for a given bit, we target only the higher 32-bit result of each umull and umaal instructions. This choice allows our attack to remain implementation-independent (since those intermediate results will indeed appear in most implementations). As shown by the doubling and addition formulas in Section 2.3.3, an addition plus a doubling consist in 25 field multiplications (the curve uses $a = -3$, thus the multiplication by a is done using subtractions). Each field multiplication itself consists in 64 32-bits register multiplications. As a result, we attack the implementation using $N = 25 \times 64 = 1,600$ leakage samples per scalar bit.

In order to efficiently identify the time positions of the corresponding registers r_i^j, we use the unprofiled correlation and partial SNR described below. They exploit a set of $N_{poi} = 8,000$ traces $\mathbf{l}$ acquired using random known inputs (P^q) and scalars $(\mathbf{k}^q)$, $q \in [0, N_{poi} - 1]$. We denote by $\mathbf{l}[t]$ vector of size N_{poi} containing the N_{poi} leakages of the t-th time sample of

[3] http://infocenter.arm.com/help/topic/com.arm.doc.ddi0439b/DDI0439B_cortex_m4_r0p0_trm.pdf
http://www.atmel.com/tools/SAM4C-EK.aspx

each trace.

Unprofiled correlation.

Given the N_{poi} internal values $\mathsf{r}_i^j = r_i^j(P^q, \lceil \mathbf{k}^q \rceil_i)$ and a leakage model M, we apply the Pearson's correlation ρ on each time sample. That is, we compute $\rho(\mathsf{M}(\mathsf{r}_i^j), \mathbf{l}[t])$, $t \in [0; 40,000,000]$. We used the Hamming weight function for M. The time sample showing the highest correlation is selected as the time sample of r_i^j. The disadvantage of using unprofilied correlation for the POI research is the requirement of a leakage model. However, it allows using the information from the full 32 bits of the internal values r_i^j.

Partial Signal to Noise Ratio (SNR).

Computing the standard SNR is not suitable to identify a specific register as any bijective relation between two registers will make them impossible to distinguish. Moreover, a 32-bit SNR would require more than 2^{32} leakages samples in the case of our 32-bit devices. Using partial SNR allows avoiding these issues at the cost of a controlled information loss. For this purpose, the 32-bit values of r_i^j are first truncated to x bits. Each trace is then labeled according to its truncated value and split into 2^x sets $\mathcal{S}_i$. For each time sample, the partial SNR is finally computed as $\frac{\mathsf{var}(\mathsf{mean}(\mathcal{S}_i))}{\mathsf{mean}(\mathsf{var}(\mathcal{S}_i))}$ where var and mean respectively denote the sample variance and mean functions. The time sample showing the highest SNR ratio is selected as the time sample of r_i^j. While partial SNR does not rely on a leakage model, it suffers from algorithmic noise because of the truncation process (which reduces the information exploited). That is, since the remaining $32 - x$ bits are not taken into account when creating the sets $\mathcal{S}_i$, the actual signal is reduced. However, ignoring the value of this remaining bits allows avoiding the bijection issue. In order to illustrate the bijection issue, one can take the AES as an example. Computing the full SNR related to the plaintext m XORed with the key k doesn't allow differentiating whether we are before or after the S-box computation. This means an SNR spike will be exhibited for both $x \oplus k$ and $\mathsf{S}(m \oplus k)$, where S denotes the AES S-box. However, computing a partial SNR on *e.g.* only 4 bits of $x \oplus k$ instead of the 8 bits will break the bijection between the input and the output of the S-Box. As a result, it will allow discriminating between the input and the output of the S-box in the partial SNR spike.

Optimizations.

Applying one of these two methods on the full trace for all r_i^j's is very time consuming as we have $123 \times 1,600$ registers to characterize over 40,000,000 time samples. However, we know that the time order of the register is $(r_0^0, ..., r_0^{N-1}, r_1^0, ...r_i^j, ...r_{n-1}^{N-1})$. Using that knowledge, we can first search r_0^0 among the first W time samples. Using correlation, we select r_0^0's position by computing a p-value with a threshold of 5 [DS16]. If r_0^0 has not been found, we move the window to the next W time samples and repeat this process until r_0^0 is found. Once this temporal location is found, we search r_0^1 similarly, by setting the initial offset of the window at r_0^0. This process is iterated until all the registers are found. We set the window value W to 20,000, chosen to be slightly higher than $\frac{40,000,000}{123 \times 25}$ (the time samples divided by the number of field multiplications).

Extraction.

Once the temporal locations of all the registers are found, we apply linear regression on each of them to extract the information as described in section 3.1.2, using a set of $N_{prof} = 10,000$ traces. Note that for this simple device, the leakage model was found to be close to Hamming weight and the linear leakage model is not expected to cause a significant information loss. Yet, the formal analysis of this statement with leakage certification tools is an interesting scope for further research [DSV14, DSP17].

3.2.4 Information combination

Using the maximum likelihood approach of Section 3.1.3, we attack the first 123 bits of the Cortex-M4 implementation with the 1-O SR described in Section 3.1.4. We used a new set of traces that has not been used to identify nor to extract the information for this purpose. As a first experiment, we compute the success rate of recovering the 123 bits using all the information. Secondly, we simulate an implementation with less information by reducing the number N of register per scalar bit. In that case, we study how using computational power by increasing the chunk's size d can mitigate the lack of information. Finally, we study how the number N of informative registers impacts the success rate of the attack.

Our first experiment is to look at the success rate using all the information. In that case we have $N = 1,600$ registers per scalar bit. We achieved a success rate of 0.85 using a chunk's size of 1 bit. It shows that such a device

would require much more algorithmic noise to be protected against worst case attacks (*e.g.* using random delays, shuffling, ...).

As using all the information allows recovering all the 123 bits with a high success rate, we next simulate a less informative implementation by using $N = 600$. In such a case, increasing the chunk size d is an option to increase success rate. As stated in Section 3.1.4, the number of points of interest increases linearly with the size of d at the cost of an exponential time complexity increase. Figure 3.4 shows the impact of the chunk size on the success rate. As we can see, the success rate increases linearly with d. We also see that the slope of the curve is lower than 1. This is explained by the fact that the number of points of interest for the bits of indexes $(0, 1, ..., d-1)$ of each chunk is equal to $(dN, (d-1)N, ..., N)$. That is, only the first chunk's bit fully benefits from increasing d, while the last one does not get any improvement.

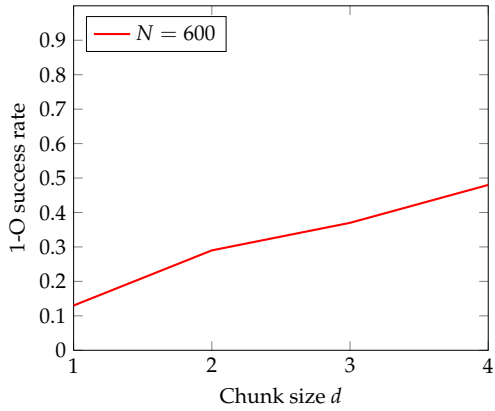


Figure 3.4: First-order success rate of the 123-bit recovery on the Cortex-M4 depending on the chunk size d for different $N = 600$.

As a last experiment, we study the impact of the number of target register N on the success rate. Figure 3.5 shows the evolution of the success rate in function of N for different values of d. Independently of d, the success rate increases exponentially with N.

From these experiments, we conclude that as general in side-channel analysis, the information extraction phase of the attacks is the most critical one, since it is the one causing the exponential security loss. Computational power can then be used as a useful (sometimes necessary) complementary ingredient to mitigate a possible lack of information.

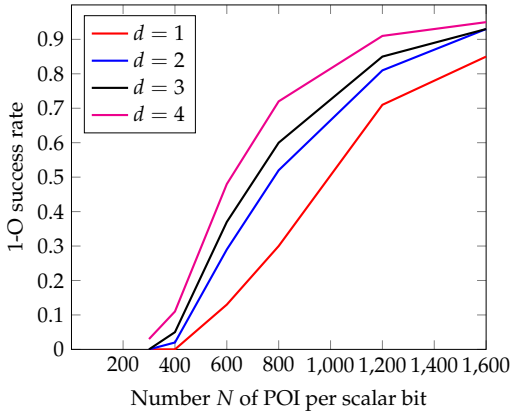


Figure 3.5: First-order success rate of the 123-bit recovery on the Cortex-M4 depending on the number of poi N for different values of d.

3.3 Experimental results on Cortex-A8

In this section we show the attack results on the Cortex-A8 microcontroller. As in the previous section, we first describe the device and measurement setup. We then show the application of the framework against this target along with the results. The scalar multiplication is implemented the same way in Jacobian coordinates as described in Section 3.2.1.

3.3.1 Device and setup

Our second target is a 32-bit AM335x 1GHz ARM® Cortex-A8 linux-based single board computer.[4] As opposed to the previous target, this one is way more challenging [BGRV15, LMPT15]. As it is running a full version of Ubuntu 14.04 and more than 100 processes are running in the background while we execute our assembly Montgomery ladder scalar multiplication implementation via SSH from the host PC. The CPU has instruction cache and 13-stage ARM pipeline, all those factors introduce a lot of noise and interruptions. Moreover, the high (1 GHz) frequency also add more obstacles in terms of side-channel measurements.

We measured the EM emission using a Langer HV100-27 magnetic near

[4] http://infocenter.arm.com/help/topic/com.arm.doc.ddi0344k/DDI0344K_cortex_a8_r3p2_trm.pdf
https://beagleboard.org/black

field probe. As in [BGRV15, LMPT15], we got the best EM signal when the probe is around the capacitor *C65*. During the measurements, we set the CPU frequency to the highest 1 GHz and the CPU frequency governor to *'Performance'*. We measured the EM traces using a Lecroy WaveRunner 620Zi oscilloscope at a sampling rate of 10 GS/s. For each scalar multiplication execution, we also triggered the measurement at the beginning of the execution and recorded the processing of the first 4 bits of the scalar, so that each trace contains 2,000,000 sample points. As mentioned in [BGRV15,LMPT15], the traces contain long interruptions that randomly appear due to the Linux system. We eliminated these interruptions by running the program with the "`nohup`" command (it is short for "No Hangups" and this supplemental command tells the Linux system not to stop a command once it has started.) via SSH without any elevating techniques. While this technique removes the big interruptions, the traces still contain many smaller ones.

3.3.2 Preprocessing

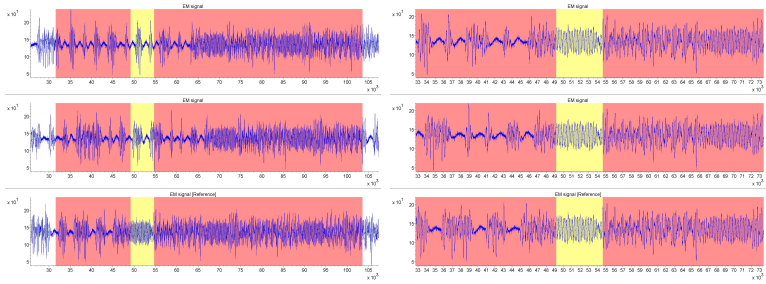
In order to deal with the small interruptions, the traces have to be preprocessed. The overall synchronization iterates over three steps. The first one consists in synchronizing the traces around a particular field multiplication. The second step is to cut the traces around the synchronized area into slices. Finally, each slice is added to the set of preprocessed traces by concatenation. These three steps are repeated for each field multiplication.

Synchronization.

While the last two steps of the preprocessing are straightforward, the synchronization part deserves more insight. We use a correlation-based alignment method discussed in Section 2.4.2 to synchronize the EM traces per field multiplication operation. This method works in three steps that are depicted in Figure 3.6. The left (resp., right) part of the figure shows the traces before (resp., after) synchronization.

- Firstly, a searching interval A that contains the operation to be synchronized is selected among all the traces. This is shown by the red window.
- Secondly, a second smaller reference interval B_q specific to each trace q is chosen, shown by the yellow window on the three traces in Figure 3.6(a).

- For each trace, we finally find the portion to be synchronized by using the second window B_q to search over the whole interval A. The right portion is selected as the one having the maximum correlation with the reference interval. If the correlation is lower than a given threshold (arbitrarily chosen by the attacker), the trace is assumed not good enough and is thus discarded.



(a) Before alignment (b) After alignment

Figure 3.6: Three EM traces before and after alignment

Once the traces are synchronized, we identify and extract the information the same way as in the previous section on the Cortex-M4 with a different number of traces. We used $N_{poi} = N_{prof} = 100,000$ traces to both identify and extract the information. As will be clear next, a linear leakage model was sufficient to obtain positive results even for this more challenging device. However, as in the previous section, analyzing the quality of this model with leakage certification tools would certainly be worth further investigations.

3.3.3 Information combination

As for the experiment on the Cortex-M4, we use the maximum likelihood approach of Section 3.1.3 to attack the first 4 bits of the Cortex-A8 implementation. This time, we assume we are attacking an ECDSA secret key and use the probabilistic approach of Section 3.1.4 on ECDSA. We used a new set of 2,200 traces that has not been used to identify nor to extract the information.

Our first experiment simply looks at the 1-O SR. In that case, we recovered the 4 bits with a success rate of 0.8155. As shown by Figure 3.3, we

Threshold	Scalar 1-O SR	Key 1-O SR	# discarded result	# remaining result
0.5	0.8174349612	$3.9 \cdot 10^{-13}$	9	2,191
0.75	0.8462296698	$5.5 \cdot 10^{-13}$	171	2,029
0.9	0.8680042239	$2.5 \cdot 10^{-9}$	306	1,894
0.99	0.9025578563	$5.8 \cdot 10^{-7}$	558	1,642
0.9999	0.956596	0.002	1,025	1,175
0.99999	0.980366	0.062	1,235	965
0.9999999	0.991708	0.312	1,597	613
0.999999999	0.9942363112	0.445	1,853	347
0.9999999999	1	1	1,958	242

Table 3.1: Evolution of the ECDSA scalar and key recovery success rate in function of the threshold.

know that 140 ECDSA nonces are required to recover the secret key with 4 bits of partial information. As no error on the partial information is tolerated, the success rate of the key recovery is equal to $0.8155^{140} \approx 3.9 \cdot 10^{-13}$. This confirms the strong need of a sound way to discriminate the wrong results.

Motivated by this first experiment, we next studied how we can automatically remove the wrong attack results. This is achieved by setting a probability threshold under which some attack traces will be discarded. That is, an attack trace is considered as invalid if the probability given by formula 3.6 for the most likely partial nonce after the attack is below a given threshold. Intuitively, the higher the threshold is, the more confident we are in having a successful partial nonce recovery, which comes at the cost of increasing the number of discarded attack traces. Table 3.1 shows how the success rate evolves in function of the probability threshold over the 2,200 attack traces. As we can see, using a threshold of 0.5 does not discard much results and thus does not increase the probability to recover the ECDSA secret key. However, the first-order success rate increases when setting a higher probability threshold. We finally achieve a perfect success rate using a threshold of 0.9999999999. Using this value, 1,958 of the attack results were discarded, thus keeping 242 of them. As only 140 correct scalars are needed to recover the ECDSA secret key, we achieve a success rate of 1.

3.4 Conclusion

This chapter provides a generic evaluation framework for HDPA against scalar multiplication algorithms, instantiates it with state of the art tools for preprocessing, POI detection, profiling and information extraction, and applies it to concrete implementations reflective of the variety of targets that can be used for embedded cryptographic applications. In view of the limited experimental reports available on the topic, we hope these implementation and systematization efforts can be used to clarify the practicality of such advanced attacks in practice, even against challenging targets running at high clock frequencies, and therefore argue for their integration as a part of current certification practices.

From a designer's point of view, our results also highlight that implementations of scalar multiplications on commercial platforms with scalar randomization activated are generally at risk, in view of the huge amount of informative samples such implementations provide. Straightforward solutions to improve this situation include performance optimizations (since implementations with less cycles inevitably leak less to the horizontal adversary given a fixed sampling rate) and the addition of noise. Yet, our analysis of an ARM Cortex-A8 running at 1 GHz (a presumably noisy target) suggests that this may not be enough. In this respect, the systematic implementation of point randomization seems strictly necessary to reach high security levels, the evaluation of which (with a systematic evaluation framework as we described in this chapter) is an interesting scope for further research, in order to better assess its worst-case security.

Chapter 4

Scatter Needs a Missing Case

SCATTER is a multivariate transform proposed in combination with the Chi^2 and MIA distinguishers at COSADE 2018. Its primary motivation is to inherently deal with the misalignment and synchronization issues that may decrease the efficiency of concrete side-channel attacks. In this chapter, we first show empirically that when compared to natural competitors for first-order multivariate attacks (*e.g.*, exploiting linear regression on-the-fly), it does not bring improvements in the (simulated and actual) implementation settings studied by its authors. We then show that the same holds in the higher-order case: in most practically-relevant settings, Scatter transform works best when combined with a combination function mixing the leakage samples in a non-linear manner, bringing it back to a situation where it does not improve standard distinguishers.

Side-channel attacks are an important threat to the security of modern embedded devices [MOP07]. Masking [CJRR99, GP99] and shuffling [HOM06, VMKS12] are among the most investigated solutions to mitigate these attacks.

Informally, masking can be viewed as a data randomization which aims at forcing the adversary to estimate higher-order statistical moments of the leakage distributions; similarly, shuffling can be viewed as a time randomization which aims at forcing the adversary to deal with information spread in multivariate distributions. As a result, evaluating a masked and/or shuffled implementation boils down to a quest for simple and effi-

cient tools enabling the analysis of higher-order and multivariate statistical distributions. The literature typically divides such distinguishers as profiled ones, like Template Attacks (TAs) [CRR02], where the adversary can use a device he controls to build a leakage model, and non-profiled ones, like CPA [BCO04], where the adversary uses a hypothetical model based on engineering intuition.

The Scatter transform was introduced at COSADE 2018 [TGWC18]. Roughly, it is a multivariate pre-processing to use in combination with "generic-emulating" distinguishers [WOS14], such as Mutual Information Analysis (MIA) [GBTP08] or the Chi^2 test [MRSS18]. Its main motivation comes from the observation that the efficiency of concrete side-channel attacks can be significantly reduced in case of misaligned traces, which may be due to jitter in the measurements or to dedicated countermeasures such as shuffling (or random delays [CK10]). In this regard, the wavelet scattering transform exploited in [DDFP20] might be an alternative to handling misalignment in side-channel security evaluations. Scatter transform is claimed to efficiently deal with such synchronization issues, while having potential for improving higher-order side-channel attacks (*e.g.*, against masked implementations) [TVW19]. Preliminary experiments showed good features in these directions, but a comparison with competing distinguishers is missing.

In this chapter, we complete this research in two directions. We start by investigating the basic potential of Scatter transform with Chi^2 and MIA distinguishers for an efficient exploitation of first-order multivariate leakages. For this purpose, our seed observation is that the COSADE 2018 paper mostly compared Scatter transform with Chi^2 and MIA distinguishers with univariate CPA-based attacks. In this context, it appears natural that Scatter transform with Chi^2 and MIA distinguishers resists better to misaligned traces, since the misalignment will typically spread the informative samples over multiple time dimensions (*i.e.*, a multivariate distribution). We therefore compare the efficiency of Scatter transform with Chi^2 and MIA distinguishers with a more natural competitor for first-order multivariate attacks, namely the on-the-fly regression-based distinguisher described in [DPRS11]. We performed experiments against a simulated shuffled implementation and a concrete jittery implementation, both similar to the settings investigated in [TGWC18]. Our results suggest that the on-the-fly regression always outperforms Scatter in these contexts.

We follow by studying the applicability of Scatter transform to masked implementations where computations are performed on secret-shared data.

In this respect, we first show that Scatter's basic (univariate) probabilistic transform is inherently unable to characterize the higher-order multivariate statistical leakages of a masked implementation. Hence, the only possible option to deal with such cases is (as usual) to generalize Scatter transform to multivariate distributions, either by estimating these distributions directly or by combining the leakage samples in a non-linear manner (see for example [PRB09, SVO+10]). Note that the latter implies that Scatter transform cannot avoid the combinatorial explosion of the number of samples to test in order to detect POIs.

Our experiments next confirm the findings of Thiebeauld *et al.* that non-linear combination functions are beneficial for the efficiency of higher-order Scatter transform from a data complexity viewpoint [TVW19]. Since we are then back to a situation where on-the-fly regression is applicable, we finally compare both distinguishers and show empirically that, as in the first-order context, Scatter transform with MIA distinguisher is outperformed by linear regression in the simulated cases we studied.

Overall, we cannot preclude another useful application of Scatter. But in the absence of theoretical or empirical arguments highlighting its interest over other established distinguishers, we conclude that it currently does not improve over existing solutions.

Note that our study is limited to the investigation of Scatter in combination with side-channel distinguishers (as it was proposed so far). One possible scope for further investigation is the study of this probabilistic transform as a pre-processing before leakage detection (*e.g.*, with the Chi^2 test [MRSS18]).

4.1 Preliminaries and Background

4.1.1 Scatter transform with Chi^2/MIA distinguishers

The Scatter transform applied to first-order leakages and combined with the Chi^2 and MIA distinguishers is illustrated on the top of Figure 4.1. The basic idea is to consider each d-dimension trace as d (one-dimension) samples, and to estimate the distribution of these d samples thanks to histograms. The histograms are then partitioned according to the key guess and a hypothetical leakage model (*e.g.*, the Hamming weight of an S-box output). The Chi^2 or MIA distinguishers are finally used to search for the correct key guess. More precisely:

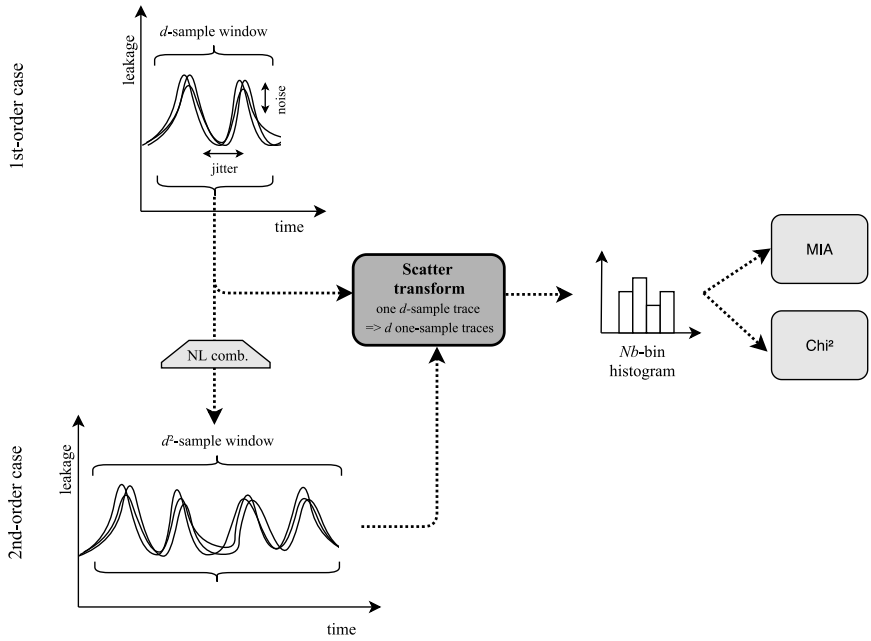


Figure 4.1: Scatter transform with MIA and Chi^2 distinguishers: high-level view.

1. We estimate histograms based on the amplitude of the sample points within a window of size d. For each measured trace, we convert the d sample points to an N_b-bin histogram. For an 8-bit oscilloscope, the max N_b is 256.
2. The "histogram traces" are then partitioned based on the key guess and hypothetical leakage model. In this chapter we consider the Hamming weight leakages of an AES-128 S-box output same as the Scatter paper. As a result, we obtain $N_b \times 9 \times 256$ partitioned histogram traces (*i.e.*, 9 Hamming weights, 256 key candidates).
3. We compute the distributions $\mathsf{pdf}_{g,h}[u]$ using the partitioned histogram traces, for each key guess g and corresponding Hamming weight hypothetical leakage h, where u denotes the histogram value:

$$\mathsf{pdf}_{g,h}[u] = \frac{\mathsf{Acc}_{g,h}[u]}{\sum_{u'=0}^{N_b-1} \mathsf{Acc}_{g,h}[u']}, \tag{4.1}$$

in which $\mathsf{Acc}_{g,h}[u]$ is the total number of occurrences of value u for a key guess g and its corresponding Hamming weight hypothetical leakage h.

4. The correct key guess $g_{correct}$ is distinguished by applying a generic-emulating side-channel distinguisher to the estimated distributions $\mathsf{pdf}_{g,h}[u]$.

Both the Chi^2 and MIA distinguishers can be used in combination with the Scatter transform in order to search for the correct key candidate.

The Chi^2 distinguisher is based on Pearson's $\mathcal{X}^2$-test to perform a partition-based DPA [SGV08]. When successful, the partition based on the correct key guess should lead to the highest confidence level to reject the null hypothesis. In the Scatter context, it estimates how much a distribution differs from a general distribution (*e.g.*, in our case study, the mean distribution of all 9 Hamming weight leakage distributions for a key guess g and a value u) — the correct key guess being expected to show the most significant difference. The Chi^2 value is computed according to the following formula:

$$\mathcal{X}^2_{g,h} = \sum_{u=0}^{N_b-1} \frac{(\mathsf{pdf}_{g,h}[u] - \frac{1}{9} \cdot \sum_{h'=0}^{8} \mathsf{pdf}_{g,h'}[u])^2}{\frac{1}{9} \cdot \sum_{h'=0}^{8} \mathsf{pdf}_{g,h'}[u]}. \tag{4.2}$$

For each key guess, there are 9 Chi^2 scores corresponding to 9 Hamming weights. The logarithm sum of all 9 scores is then used as the final score.

The MIA distinguisher was introduced by Gierlichs *et al.* [GBTP08]. It is based on estimating the mutual information between a hypothetical leakage model and the actual leakages. Under a correct partitioning (*i.e.*, the correct key guess), it is expected that the largest mutual information should be observed for the correct key candidate to distinguish the correct key guess from the wrong ones. The MIA value is computed according to the following formula:

$$\mathrm{MI}_g = \sum_{h=0}^{8} \mathsf{P}(\mathsf{Y} = \mathsf{h}) \cdot (\bigtriangledown_1(g,h) - \bigtriangledown_2(h)), \tag{4.3}$$

in which:

$$\mathsf{P}(\mathsf{Y}=\mathsf{h}) = \frac{1}{n \cdot 9} \cdot \sum_{u=0}^{N_b-1} \mathsf{Acc}_{g,h}[u], \tag{4.4}$$

with n the number of traces collected and:

$$\bigtriangledown_1(g,h) = \sum_{u=0}^{N_b-1} \mathsf{pdf}_{g,h}[u] \cdot (\mathsf{pdf}_{g,h}[u]), \tag{4.5}$$

$$\bigtriangledown_2(h) = \sum_{u=0}^{N_b-1} (\frac{1}{9} \cdot \sum_{h'=0}^{8} \mathsf{pdf}_{g,h'}[u]) \cdot \log(\frac{1}{9} \cdot \sum_{h'=0}^{8} \mathsf{pdf}_{g,h'}[u]). \tag{4.6}$$

4.1.2 On-the-fly linear regression

The use of linear regression for (profiled) side-channel attacks was introduced by Schindler *et al.* [SLP05]. It is also mentioned that it is applicable to the non-profiled setting, but no details nor experiments. It was then extended to non-profiled key-recovery attacks in [DPRS11] in detail. We next denote this non-profiled extension as LRA.

Let us denote the leakage measurement as $\boldsymbol{L}$. The target m-bit intermediate value v (*e.g.*, the S-box output in our case) is first decomposed according to some basis. In the following, we will use the usual (linear) basis made of the 8 bits of v $(v[m-1], v[m-2], \ldots, v[0])$. LRA then simply tests the linear relation between the actual leakages and their approximation with this basis, thanks to the coefficient of determination R^2. More precisely:

1. We first compute $(v_{\hat{g}}[m-1], v_{\hat{g}}[m-2], \ldots, v_{\hat{g}}[0])$ for each key guess $\hat{g}$ and each input plaintext & measurement $\boldsymbol{L}_i, i = 0, 1, \ldots, n-1$.
2. We then estimate the linear regression model between the measurement $\boldsymbol{L}$ and the following approximation:

$$\boldsymbol{L}_{\text{app}} = \beta_{\hat{g},0} + \beta_{\hat{g},1} \cdot v_{\hat{g}}[0] + \ldots + \beta_{\hat{g},m} \cdot v_{\hat{g}}[m-1]\,, \tag{4.7}$$

   using ordinary least square method to estimate the parameter $\beta_{\hat{g},j}$.
3. We finally compute the coefficient of determination $R^2_{\hat{g}}$ for each key guess. The correct key guess g_{correct} is supposed to show the highest R^2 value.

4.1.3 Selection of parameters

The efficiency of the three aforementioned distinguishers is quite dependent on the good selection of their parameters: number of bins for the Chi^2 and MIA distinguishers, size of the basis for LRA. As already mentioned, our experiments are based on LRA with a linear 9-element basis (the eight S-box output bits and a constant), which is a standard choice for this distinguisher [SLP05]. For the Chi^2 and MIA distinguishers, choosing the optimal number of bins is usually tricky. We selected 9 and 25 bins in our experiments: 9 since it naturally corresponds to Hamming weight leakages, 25 to assess the impact of more bins. We note that this choice is expected to be slightly detrimental to the LRA distinguisher (since under a Hamming weight assumption, a 2-element basis with the Hamming weight of the S-box output should be even faster to estimate).

4.2 First-order experiments

We first investigate a simulated shuffled implementation, since this was the case study put forward in the COSADE 2018 paper on Scatter. We continue by targeting a real device of which the measurements are affected by a strong jitter, preventing the good alignment of the traces around the leaking part.

4.2.1 Setting #1: a simulated shuffled implementation

Shuffling is a widely-used side-channel countermeasure [HOM06, VMKS12]. Its main principle is to execute sensitive operations in a random order so that their leakages are spread over a multivariate distribution. As a result, each single point in time can correspond to the execution of various operations.

Shuffled implementations are the typical context in which Scatter's multivariate transform was claimed to be a useful tool at COSADE 2018.

Implementation settings.

The main parameter influencing the security of a shuffled implementation is the number of parallel operations which are randomized. We next consider a default size of 16 (corresponding to the AES-128 case) and additionally experimented with a permutation of size 64, which could correspond

to the execution of 48 dummy S-boxes. In our default setup, a single POI is leaking (corresponding to the target S-box execution) but we also considered a case with four POIs (which does not reflect a concrete AES-128 implementation and was just aimed to understand the impact of a denser leakage in the Scatter window). Finally, we used an SNR of 10, 1 and 0.1, reflecting low-noise, medium-noise and high-noise contexts [Man04]. The way we generated simulated traces is similar to the Scatter paper. For a window size d (*i.e.*, the shuffling size in our experiments), we:

1. Choose the number of informative points (n_i),
2. Pick up their location in the d possible positions uniformly at random,
3. Put random leakages (of the same shape, *i.e.*, $\alpha \times HW(data) + \beta$ according to [TGWC18], all points share the same random α and β parameters) in all the other points,
4. Add Gaussian noise to the entire trace based on the chosen SNR.

Our simulations focus on Hamming weight leakages for the first-round first S-box of an AES-128 encryption, namely $\mathsf{HW}(\mathsf{Sbox}(p[0] \oplus k[0]))$, where $p[0]$ and $k[0]$ correspond to the first bytes of the 16-byte AES-128 input and key, respectively.

For each simulation setting, we estimated the Success Rate (SR) of the different attacks under investigation based on 100 independent experiments [SMY09].

Attack results.

The results of our experiments are in Figure 4.2. We analyzed a wide range of parameters reflecting the various settings in which the Scatter transform could be exploited. As previously mentioned, we also evaluated this transform with both the Chi^2 and MIA distinguishers, using 9 and 25 bins. Those are systematically compared with the LRA distinguisher (9-element basis). Note that, all sample points were used as the input for the LRA distinguisher without POIs selection.

In general, these experiments carry the expected intuitions regarding the impact of our different parameters: decreasing the SNR makes the attacks more difficult (*i.e.*, when moving from the left of the figure to the right of the figure); increasing d (*i.e.*, the permutation size) makes the attacks more difficult (*i.e.*, when moving from lines 1 and 2 to lines 3 and 4);

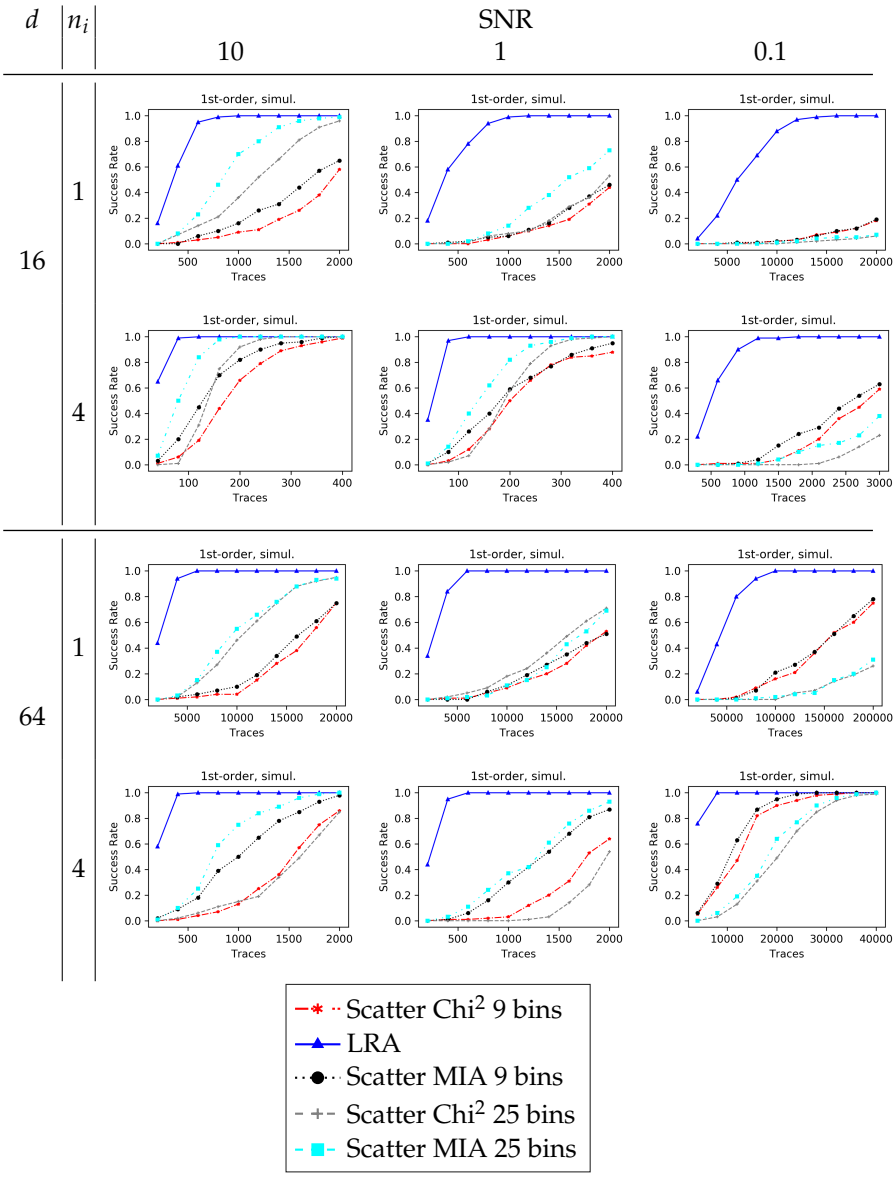


Figure 4.2: Success rate on simulated shuffled implementations, with d the window size, n_i the number of POIs per window and various SNR values.

increasing the number of POIs makes the attack easier (*i.e.*, when moving from line 1 to line 2 and from line 3 to line 4). More specifically related to Scatter:

- LRA always outperforms Scatter with both the Chi^2 and MIA distinguishers, no matter the permutation size, noise level and number of POIs;
- The performance gap between LRA and Scatter transform with Chi^2 and MIA distinguishers is getting bigger as the attacks become more difficult (*i.e.*, when the permutation size increases, the noise level increases and the number of POIs decreases);
- Scatter with the MIA distinguisher performs slightly better than Scatter with the Chi^2 distinguisher (which is in line with the COSADE 2018 results).
- As for the impact of the number of bins for Scatter transform with Chi^2 and MIA distinguishers: more bins generally show better results with lower noise and less bins generally works better with higher noise. The latter is in line with the findings of [GBTP08].

4.2.2 Setting #2: a concrete jittery implementation

We now extend our investigations to a real device, namely a software AES-128 implementation using a secure processor. Due to the variable internal clock, the inserted random instructions during AES-128 calculations, and the interrupts caused by the running Android-like operating system (OS), the measured traces are very jittery and we cannot really align the traces at the leaking time interval. We study how well Scatter transform with Chi^2 and MIA distinguishers can handle this challenging scenario.

Implementation settings.

The target secure processor is a Cortex-M4 chip running at 50 MHz next to a Qualcomm MSM8998 general processor which is running an Android-like OS. The secure processor is used for cryptography calculations and it communicates with the MSM8998 processor via UART (Universal Asynchronous Receiver/Transmitter) interface. The AES-128 implementation is unprotected except for the random instructions inserted by the OS during the AES-128 execution. Interrupts are additionally caused by the running

Android-like OS and make the measured traces more noisy and hard-to-align. We measured 100,000 ElectroMagnetic (EM) traces on top of the secure processor using an SGS Brightsight homemade EM probe (3 turns of 0.05 mm wires) with an external RF Bay LNA-1800 low noise amplifier, with a LeCroy Waverunner 620Zi oscilloscope, at a sampling rate of 5 GHz.

During the measurements, we triggered the oscilloscope at the end of the entire AES-128 encryption command processing. The raw EM traces are noisy and hardly show distinct patterns that can be used for alignment, as can be seen in Figure 4.3(a). We therefore used a simple correlation-based pre-processing (alignment) as described in Section 2.4.2 in order to better synchronize these EM traces.

After performing such an alignment, we were able to determine where the AES-128 computations occur by means of SEMA and CEMA, as shown in Figure 4.3(b).

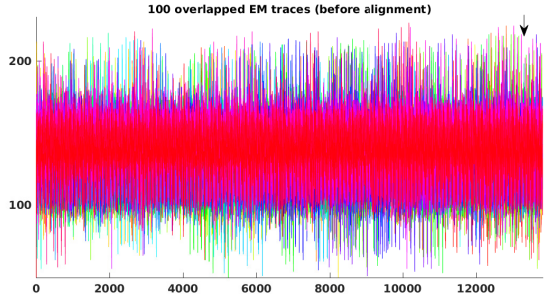
Attack results.

Our comparisons are based on 99,902 aligned EM traces focusing on the leaking part (the other traces were discarded). As a first note, none of the investigated distinguishers directly succeeded in recovering key bytes by exploiting the leakage in the time domain. We then applied a Fast Fourier Transform (FFT) in order to convert the traces into the frequency domain and to mitigate the impact of misalignment. After this pre-processing, LRA was able to recover all 16 key bytes of an AES-128 state, but Scatter transform was not (neither with the Chi^2 nor with the MIA distinguishers). A potential explanation for the bad performance of Scatter transform with Chi^2 and MIA distinguishers is: after FFT it deals with real numbers, histogram estimation with real numbers does not work well. These results are illustrated in Figure 4.4 where the success rate is estimated based on 100 independent experiments.

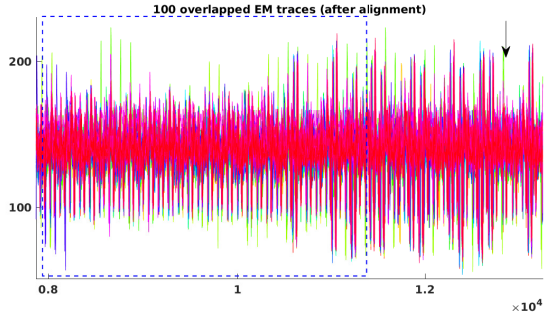
4.3 Higher-order Scatter

4.3.1 The need of a combination function

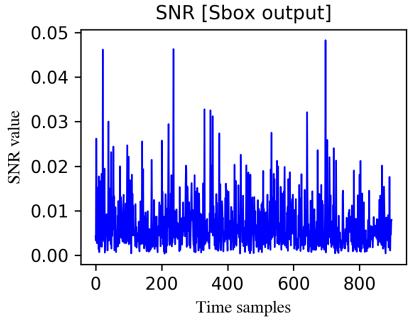
We start with a simple negative result highlighting the need to generalize the Scatter transform before application to higher-order side-channel attacks. For this purpose, let us imagine that the two "clock cycles" represented at the top of Figure 4.1 correspond to the two shares of a masked



(a)



(b)



(c)

Figure 4.3: 100 overlapped EM traces before alignment (a) and after alignment (b), and target S-box estimated SNR (c).

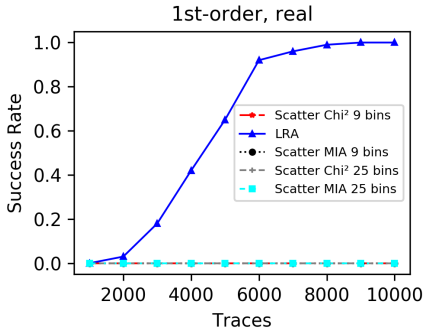


Figure 4.4: Attacks against a real implementation with strong jitter.

sensitive variable x. Let us further consider that this sensitive variable is one bit and can be written as $x = x_1 \oplus x_2$ with x_1 picked up uniformly at random. Let us finally assume that the adversary can obtain the leakage of the two shares x_1 and x_2, denoted as l_1 and l_2: under Hamming weight leakages, we have $l_1 = x_1$ and $l_2 = x_2$, meaning that the adversary can directly observe the shares. In this context, 1st-order probing security is guaranteed because the observation of either l_1 or l_2 does not reveal anything about x. By contrast, a second-order probing attack is trivial since $l_1 \oplus l_2 = x$. More interestingly, a second-order statistical attack is also successful since the distribution of (l_1, l_2) when $x = 0$ is (0,0) with probability $\frac{1}{2}$ and (1,1) with probability $\frac{1}{2}$, while this distribution becomes (0,1) with probability $\frac{1}{2}$ and (1,0) with probability $\frac{1}{2}$ when $x = 1$ (which has a different variance).

If we now apply the Scatter transform, each bivariate trace (l_1, l_2) is split into two univariate traces l_1 and l_2, and histograms are built from these two traces. As a result, the two traces (0,0) and (1,1) that correspond to the case $x = 0$ are turned into four traces 0, 0, 1, 1. Their histogram gives 0 with probability $\frac{1}{2}$ and 1 probability $\frac{1}{2}$. Similarly, the two traces (0,1) and (1,0) that correspond to the case $x = 1$ are turned into four traces 0, 1, 1, 0, leading to exactly the same histogram. So directly applying the first-order Scatter transform to a masked implementation cancels the differences between these distributions that can be used to mount a successful second-order attack. The same example generalizes to any number of shares and probing/statistical security order.

As usual in side-channel analysis, the solution to prevent this issue is

to generalize the transform to higher-orders. There are essentially two solutions for this purpose: either one considers all the pairs (and triples, quadruples, ...) of samples and applies a multivariate (*e.g.*, Chi^2 or MIA) distinguisher to it, or one uses a combination function (*e.g.*, the normalized product in the context of Hamming weight leakages [PRB09,SVO+10]) and applies a (univariate in the case of LRA or multivariate in the case of Scatter transform) distinguisher to its output. As discussed for example in [BGP+11, MRSS18], directly considering all the pairs (and triples, quadruples, ...) of samples and applying a multivariate distinguisher is usually more expensive, due to the curse of dimensionality when estimating multivariate distributions in a non-parametric manner. Our experiments showed the same trend and so do the experiments of Thiebeauld *et al.* in [TVW19].

As a result, we next consider higher-order attacks based on a combination function illustrated at the bottom of Figure 4.1. That is, in the second-order case we will concretely investigate, we start by extending the original d-sample window to a d^2-sample window containing all the normalized product samples and then apply the Scatter transform combined with the Chi^2 or MIA distinguishers, or LRA. Note that this solution suffers from the usual drawback that the cost of finding the POIs in the traces grows exponentially in the number of shares (and exactly the same would hold for the first aforementioned solution where a multivariate distinguisher is applied to all the tuples of samples).[1]

4.3.2 Second-order simulated experiments

Implementation settings.

We now complete the previous first-order experiments with second-order simulations. We consider a 2-share implementation where the adversary obtains the two Hamming weights corresponding to the two shares of a target S-box's output. Based on the previous observation that Scatter transform with Chi^2 and MIA distinguishers tends to behave better in less challenging scenarios (and in order to limit the cost of our simulations, which increases with the security levels), we selected the following parameters:

[1] In the report on the second-order application of Scatter [TVW19], an optional projection of the histogram traces is considered. In our experiments, this projection (just as the direct bivariate attacks) did not exhibit any improvement. This seems natural in a simulated setting where the normalized product combination function is known to be optimal [PRB09]. So we next ignore this optional projection.

a permutation of size $d = 4$ with a single POI (*i.e.*, $n_i = 1$) and an SNR of 10, 1 and 0.25. For completeness, we also report results with $d = 16$, 4 POIs (*i.e.*, $n_i = 4$) and an SNR of 10 and 1. We focused on Scatter transform with the MIA distinguisher that was the best in class for our first-order experiments. (This is also similar to what has been done in [TVW19]).

Attack results.

The results of our second-order experiments based on a normalized product combination function are displayed in Figure 4.5.

Observations are essentially similar to the first-order case: again, LRA systematically outperforms attacks based on Scatter transform with MIA distinguisher and the more "challenging" the implementation (*e.g.*, the lower the SNR), the bigger the gap. This can be explained by the fact that the product combining pre-processing generates traces that can be exploited in a very similar way as an unprotected implementation (up to the noise level that is amplified by the product operation). Similarly, histogram estimation with real numbers after the product combining pre-processing could be another factor impacting the performance of Scatter.

4.4 Conclusion

Exploiting the side-channel leakages of an implementation protected with jitter, shuffling and/or masking is a challenging problem. The Scatter transform was introduced at COSADE 2018 as a novel transform aimed to make such attacks more efficient. In this chapter, we nailed down its specificity and compare it with a natural competitor for unprofiled multivariate side-channel analysis.

It turns out a standard on-the-fly application of linear regression leads to more efficient attacks in many practically-relevant contexts, including all the ones initially put forward by the Scatter authors. Our conclusions are specific to our implementations of Scatter transform with Chi^2 and MIA distinguishers. It is therefore an interesting open problem to determine whether this transform can sometimes be a useful ingredient in a side-channel security evaluation. The exhibition of a simulated case where such a gain can be observed appears as a natural next to answer this question. Without such a case, evaluators are left with the conclusion that it currently does not improve over existing solutions.

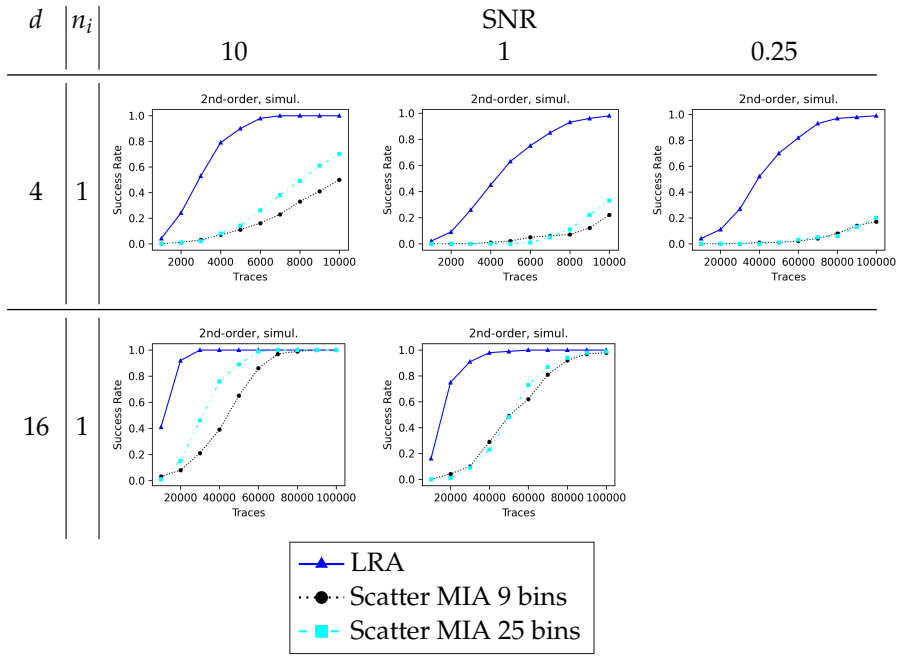


Figure 4.5: Success rate on second-order simulated shuffled implementations, with d the window size, n_i the number of POIs per window and various SNR values.

Part II

Deep Learning Side-Channel Security Evaluations

Chapter 5

Alignment Matters for Deep Learning Side-Channel Analysis

PAST research has shown an increasing interest from the side-channel community regarding the use of machine learning / deep learning techniques as a powerful way to exploit physical leakages with limited knowledge of the target implementations: see for example [CCC+19b, HGG18, HZ12, HGM+11, LPB+15, MPP16, PH18, PHJ+19, PHJL17, PSK+18, PSB+18, RQL19]. In general, one potential advantage of these techniques compared to more conventional statistical tools (*e.g.*, Gaussian templates [CRR02] and linear regression [SLP05]) is that they quite efficiently deal with large dimensionalities (which may prevent the need of estimating large covariance matrices, or to rely on dimensionality reduction [APSQ06]). This intuition has been recently put forward by Cagli *et al.* at CHES 2017, they demonstrate that (*e.g.*, deep) learning algorithms are good candidates to exploit the leakage of implementations protected with jitter-based countermeasures [CDP17].

We consider the question whether synchronization / alignment methods are still useful / necessary in the context of side-channel attacks exploiting deep learning algorithms. While earlier works have shown that such methods / algorithms have a remarkable tolerance to misaligned measurements, we answer positively and describe experimental case studies of side-channel attacks against a key transportation layer and an AES-128

S-box where such a pre-processing remains beneficial (and sometimes necessary) to perform efficient key recoveries.

Our results also introduce generalized Residual Networks as a powerful alternative to other deep learning tools (*e.g.*, Convolutional Neural Networks and Multi-Layer Perceptrons) that have been considered so far in the field of side-channel analysis. In our experimental case studies, it outperforms the other three published state-of-the-art neural network models for the data sets with and without alignment, and it even outperforms the published optimized CNN model with the public ASCADv1 fixed-key[1] data set. Conclusions are naturally implementation-specific and could differ with other datasets, other values for the hyper-parameters, other machine learning models and with other alignment techniques.

Cautionary note. We acknowledge that these conclusions may be affected by the level of profiling of the implementations. In theory, for an infinite amount of profiling on raw traces, re-synchronization may become useless for deep learning as for any well-specified multivariate side-channel attack. A similar statement holds for the choice of parameter for the ResNet we exploited. In this respect, our goal is only to show that for a realistic amount of profiling and standard use of a popular deep learning algorithm, re-synchronization may help. More generally, our conclusion is admittedly based on an experimental basis. So the conclusions of this chapter could be different with other datasets, other values for hyper-parameters, other machine learning models or with other alignment techniques.

The rest of this chapter is organized as follows. Section 5.1 introduces the alignment method that we used, the necessary background on deep Residual Networks and the target implementations that we investigated. Section 5.2 describes the verification of the ResNet model we adopted regarding its capability in a side-channel context by analyzing AES-128 traces obtained from the ChipWhisperer Lite board and the comparison with the state-of-the-art neural network models regarding performance. Section 5.3 and Section 5.4 respectively show the experimental results for the application of this ResNet model with the impact of misalignment against our two main targets, and the comparison of their performances. Finally, in Section 5.5 we further demonstrate the good generalization of our ResNet model by comparing its performances with the published optimized ASCAD_CNN model using all 16 S-boxes data from their published ASCADv1 fixed-key data set.

[1] https://github.com/ANSSI-FR/ASCAD/tree/master/ATMEGA_AES_v1/ATM_AES_v1_fixed_key

5.1 Preliminaries

5.1.1 Target implementations

We investigated two main target implementations in this chapter, and additionally used the simple case of an AES-128 software implementation on the ChipWhisperer Lite board for preliminary assessments. In this warming up case, we capture 90,000 profiling power traces with a 16-byte randomized AES input and key, and 10,000 attack power traces with a fixed random AES-128 key and randomized AES input.

For our first Device Under Test (DUT1), we target the AES-128 key during its transportation. The latter is important for security evaluations, since it frequently happens that the keys encrypted in some Non-volatile Memory (NVM) has to be transported to the cryptographic co-processor when invoking the corresponding cryptographic encryption/decryption operations. In case the key is decrypted/masked/unmasked during this transportation, it may lead to additional sources of leakages that could be exploited by an adversary. Concretely, our DUT is a modern 32-bit secure microcontroller with a built-in secure AES coprocessor and the AES-128 key is encrypted and stored in an EEPROM. The encrypted AES-128 key needs to be decrypted and masked during its transfer from EEPROM to the AES coprocessor to execute AES-128 encryption. We acquire 80,000 EM traces for profiling and 20,000 traces for the attack.

Finally, our second DUT is a more standard case of an AES co-processor for another secure microcontroller resistant to first-order leakage, where we target the S-box output of the first block cipher round. For this one, we measure 500,000 EM traces for profiling and 30,000 traces for attack.

We note that the different ratios between the numbers of profiling and attack traces can be connected to the fact that it is in general more complex to profile a leaking device than to attack it once a model is well estimated [SKS09]. As a result, the more secure an implementation (*e.g.*, due to masking or other countermeasures), the larger this ratio can be.

5.1.2 Alignment method

Due to the complex architecture (and potential countermeasures) of our DUT, their raw measured EM traces are very misaligned. To get better aligned traces for our investigations, we use the same method as in [PZS17] (described in Section 2.4.2), which exploits correlation in order to synchro-

nize the EM traces focusing on the leakage part of targeted sensitive data (*e.g.*, the AES-128 key for the transportation case in Section 5.3 and the AES-128 S-box output for the case in Section 5.4).

Note that concretely, this method was usually applied multiple times for each DUT by targeting different time intervals (for both the searching interval and reference interval). During the measurement, no very distinguishable feature close to the target interval can be used to trigger the oscilloscope. So we have to align the traces step by step to get close to the target interval, and then within the target interval we do more local alignments. Roughly, we choose new intervals when the misalignment is getting larger, and we repeat this process until the target interval is well aligned. This allowed us to recursively improve the alignment for the part of the traces that corresponds to the target leakage. Before we measure the traces for profiling attacks, we used SPA/SEMA and CPA/CEMA techniques to narrow down the potential target interval.

For DUT1, we first try to find the leakage of key data using a CPA by measuring the whole AES-128 encryption command execution with a randomized AES-128 key data per execution, so the interval is rather large. We try different alignments and calculate the correlation of key data after each alignment. For DUT2, we cannot find the leakage of the S-box output because it is a masked implementation, but we figure out which interval is likely related to AES-128 encryption by SPA and SEMA techniques. The length of the target interval is changing when the input length is increasing. In order to recursively align the traces to the target interval, we accordingly choose the distinguishable feature of most traces step by step. After we recursively aligned the traces to the target interval, the segment around the target interval of each trace is then cut from the original traces and used for profiling attacks. In general, the required time of this recursive alignment process depends on the number of traces, the number of sample points per trace and the size of the chosen interval. For DUT1 it takes less than one hour and for DUT2 it takes about 7 hours using a PC with an Intel® Core™ i7-6850K CPU 3.60GHz and 64 GB RAM.

5.1.3 Deep Residual Network

Since they have been proposed/applied in the computer vision field with great successes [HZRS15], deep ResNets [HZRS16b] have been widely applied in different fields such as machine translation [WSC+16], speech synthesis [vdODZ+16], speech recognition [XDH+16] and AlphaGo [SSS+17].

Thanks to the open source deep learning libraries Keras [C+15] and Tensorflow [AAB+15], it is straightforward to build a model for performing profiling attacks in a side-channel context. We refer to these original papers for the details of the method and next list the parameters that we used in our experiments.

The core design idea of ResNets is to extend neural networks to very deep structures without degradation problems thanks to a so-called *shortcut connection*. ResNet is a stack of residual blocks: each residual block consists of several layers and a shortcut connection, the shortcut connection connects the input and output of that residual block. As depicted in Figure 5.1, the latter is inserted in each residual block such that the gradient flows directly through the bottom layers. Each residual block consists of three basic blocks and each basic block is composed of three layers: a convolutional layer γ followed by a batch normalization layer β [IS15] and a ReLU activation layer σ [NH10]. After stacking three residual blocks, a global average pooling layer δ is adopted (instead of a fully connected layer), in order to reduce the number of weights to be trained. Finally a softmax layer s is adopted to generate the class label of the input side-channel trace. In summary, the ResNet model can be written as:

$$\texttt{ResNet} = s \circ \delta \circ [\sigma \circ [\beta \circ \gamma \circ [\sigma \circ \beta \circ \gamma]^{n_1-1} \oplus 1]]^{n_2}, \tag{5.1}$$

where n_2 denotes the number of residual blocks (we set it to 3 for all our experiments) and n_1 is the number of basic blocks per residual block (we also set it to 3 in this chapter). We further use 128, 256 and 256 filters for these three residual blocks. That is, all the convolutional layers within one residual block are using the same amount of filters. Following the original ResNet paper [HZRS15], we also did not use a dropout layer.

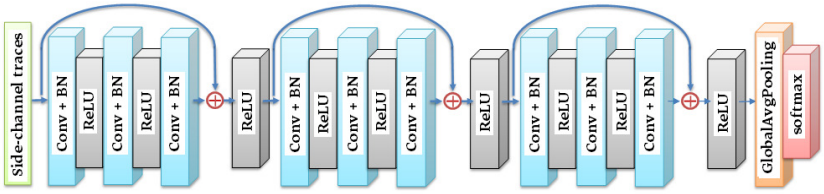


Figure 5.1: Structure of ResNet.

5.1.4 Accuracy, Loss and key rank

Accuracy [GBC16] and *Loss* are twin metrics that are widely used in the machine learning community to monitor and evaluate neural network models. Training accuracy is the successful classification rate over the training data and training loss is the error rate over the training data. After each epoch, the trained model is applied to the validation data to calculate validation accuracy and validation loss. These two values indicate how good the trained model is at predicting outputs for inputs it has never seen before. Validation accuracy increases initially and saturates as the model starts to overfit. We also use the key rank (*i.e.*, guessing entropy) as classical metric for side-channel security evaluations [SMY09].

5.1.5 Deep learning DPA attack

Different from the classical profiled attacks such as template attacks, deep learning attack makes no assumption of the leakage characteristic. It exploits the features (sample points with regard to side-channel traces) to classify the labels (sensitive data in the SCA context) using neural networks. The training process of neural networks (corresponds to the profiling) aims to construct a classifier function $F(.) : \mathbb{R}^d \rightarrow \mathbb{R}^{|S|}$. This function maps the input trace $\mathbf{l} \in R^d$ to the output vector $\mathbf{p} \in R^{|S|}$ of scores. During the training, for each training batch, the backpropagation method [Kel60, GBC16] is used to update the trainable parameters of the neural network model aiming at minimizing the loss, which is calculated to quantize the classification error over each training batch. In the attack phase, the built trained model (*i.e.*, $F(.)$ with all the final updated trainable parameters) is used to classify each attack trace to obtain its score vector $\mathbf{p}[s_{m,g}]$. Afterwards, the final score vector of each key candidate $\mathbf{p}[g]$ is calculated using all the attack traces (similar to Equation (7.3)). The key candidate $g^* = \text{argmax}\ \mathbf{p}[g]$ is considered the right subkey.

5.2 Warming up: results on CWL (AES-128 S-box)

In order to verify that the ResNet model is applicable in a side-channel context, we used it to analyze an unprotected AES-128 implementation on the ChipWhisperer Lite board, and fed the Hamming weight of the first-round S-box outputs into the ResNet model as described in 5.1.3. We only used 9 Hamming weight classes for this warming-up case because it is

known that its S-box output leakage follows the Hamming weight model. For other datasets used in this chapter, we used 256 classes of S-box output in our experiments because we did not know the leakage model. We then performed key recovery by classifying the Hamming weight of the S-box outputs of the attack traces and translating this into key information. We followed the best practice of deep learning to use balanced data per class (also for our other experiments), and we captured 90,000 profiling traces as mentioned in Section 5.1.1. Due to the uneven distribution of the 9 Hamming weight classes, in the end we have only 320 profiling traces per Hamming weight class. We use 20% profiling traces as validation data to improve the training of the weights and 3,000 sample points (each trace contains in total 3,000 sample points) per trace to feed into the ResNet model. Figure 5.2 displays the rank of the correct subkey candidates of all 16 S-boxes. As can be seen, with a few attack traces the correct subkeys of S-box 8 and S-box 16 can be disclosed. All the other S-boxes show similar results except that S-box 15 needs a few hundred traces to recover the subkey. For the CWL experiments, we use a batch size of 32 and 100 epochs, "Adadelta" optimizer with an initial learning rate of 1.0, adaptively reducing the learning rate with a factor of 10 if the validation loss is not decreased within 5 consecutive epochs.

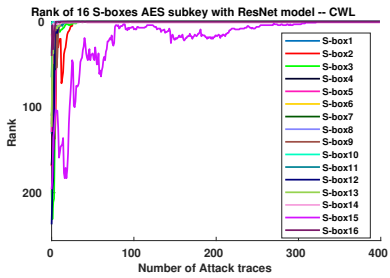
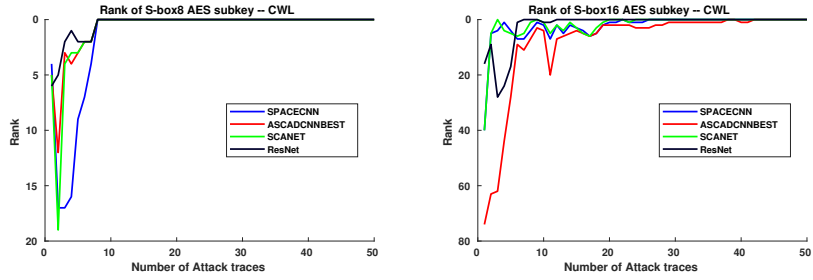


Figure 5.2: Rank of correct subkey candidates of all 16 S-boxes with CWL data set.

5.2.1 Performance comparison with the state-of-the-art

To compare the performance of our ResNet model with other state-of-the-art neural networks in a side-channel context, we performed the similar attacks using other three neural network models published in [MPP16, PSK+18, PSB+18].

Those three models are the best ones according to their experiments and we are using their default hyper-parameters. (We did not have enough information to replicate the CHES 2017 CNN model [CDP17]). We adapt those models to 9 Hamming weight classes. Figure 5.3 compares the rank of correct subkey candidates of S-box 8 and S-box 16 respectively using our ResNet model and the other 3 models. Only two S-boxes are targeted as a demonstration. It can be observed that our ResNet model compares positively to the others (in terms of convergence).



(a) S-box 8 (b) S-box 16

Figure 5.3: Performance comparison of 4 NN models using CWL data set.

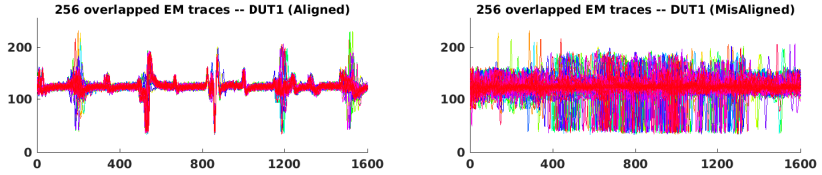
5.3 Experimental results on DUT1 (AES-128 key transfer)

We start with an analysis of a key transportation layer which is a less usual target in the academic literature, but a quite important one in industry. The fact that key transportation leaks information on the key is indeed a critical weakness.

In this context, our profiling traces are obtained by randomizing all the 16 bytes of AES-128 key and fixing the 16-byte AES input to execute AES-128 encryption (the fixed input data is not detrimental since we are targeting the key transportation). For the attack traces, it is pretty different: we are targeting the value of the first byte of a 16-byte AES-128 key considering the other bytes are leaking in a similar way. We randomize the first byte from $0x00$ to $0xFF$, so that we have 256 classes of attack traces for it and set the other 15 bytes to fixed random values. That is a realistic scenario that the attackers normally encounter: when an attacker wants to attack a

fixed AES-128 key, he needs to attack the AES-128 key bytes one by one while the other bytes are fixed random values. From a security evaluation viewpoint, we want to simulate the attack scenario as realistic as possible. The key bytes cannot be changed and we cannot attack all the 2^{128} possible keys (for 16-byte AES-128 key). We still want to evaluate how well we can correctly identify all 256 possible values of one key byte (consider it as an example of all 16 bytes), so normally we choose one example byte and vary it from $00x00$ to $0xFF$, but the other 15 bytes are fixed random values just like what the attacker will face for attacking a 16-byte key . We use 256 classes of profiling traces to train the ResNet model, and 20% of profiling traces as validation data to improve the training. We then use the trained model in order to classify all the 256 classes of attack traces to see how many classes can be correctly identified. From attack perspective, it is a profiling-based SPA attack. So we consider the percentage of correctly identified classes during the attack phase. In this SPA context, it is worth noting that one should consider summing the log probabilities of several attack traces where averaging is possible. For the DUT1 experiments, we use a batch size of 32 and 40 epochs, "Adadelta" optimizer with an initial learning rate of 1.0, adaptively reducing the learning rate with a factor of 10 if the validation loss is not decreased within 10 consecutive epochs.

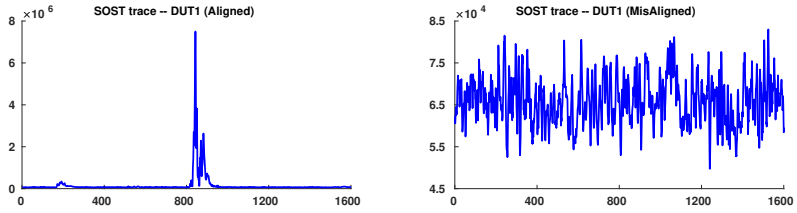
For this target, in order to figure out the impact of misalignment with regards to the deep learning attacks, we conduct the deep learning attacks using the same ResNet model with two data sets: aligned traces (applying our synchronization method 6 times) and misaligned ones (applying our synchronization method 4 times). This impact of the alignment is visually illustrated by Figure 5.4.



(a) Aligned (b) MisAligned

Figure 5.4: 256 overlapped aligned and misaligned EM traces of DUT1.

We further calculate the SOST (Sum Of Squared pairwise T-differences) [GLP06] trace per data set of DUT1 as shown in Figure 5.5 for better showing the impact of misalignment.



(a) Aligned (b) MisAligned

Figure 5.5: SOST traces of aligned and misaligned EM traces of DUT1.

Both aligned and misaligned traces have 1,600 sample points to be fed into the ResNet model, and we are using the same number of profiling and attack traces and the same batch size for both cases. The training accuracy and validation accuracy (as described in Section 5.1.4) of the ResNet model are given in Figure 5.6. From the profiling perspective, the training accuracy using aligned traces is converging much faster and higher than the one using misaligned traces, although we are using the same ResNet model. Besides, from the attack perspective, as shown in Figure 5.7, the aligned traces also show much better percentage of correctly identified classes results. With aligned traces, the percentage of correctly identified classes already reaches 100% after 7 epochs and afterward the percentage of correctly identified classes is stabilized at 100%. On the other hand, with the misaligned traces, the percentage of correctly identified classes reaches 100% after 18 epochs.

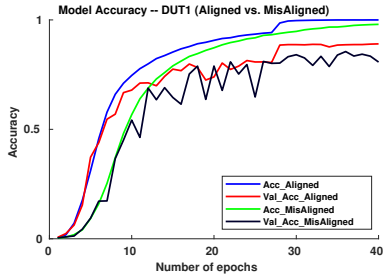


Figure 5.6: ResNet Model Accuracy with DUT1 data sets.

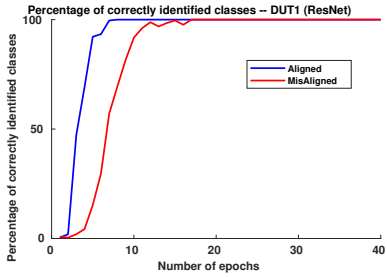
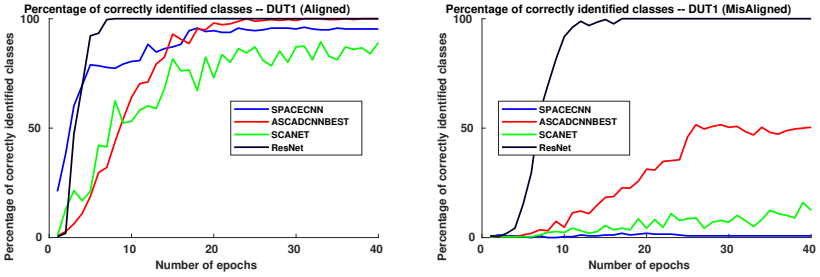


Figure 5.7: Percentage of correctly identified classes with DUT1 data sets.

5.3.1 Performance comparison with the state-of-the-art

Similarly, Figure 5.8 shows the percentage of correctly identified classes using our ResNet model and the other three models with both the aligned and misaligned DUT1 data sets. As can be seen for both aligned and misaligned traces, our ResNet model converges faster and gives better results from the attack perspective (less epochs of training are needed). In particular, considering the misaligned traces, our ResNet model outperforms the others a lot. It further confirms that the generalization of the ResNet model can bring an interesting alternative to the other models.



(a) Using aligned traces.

(b) Using misaligned traces.

Figure 5.8: Performance comparison of 4 NN models using DUT1 data sets.

5.4 Experimental results on DUT2 (AES S-box)

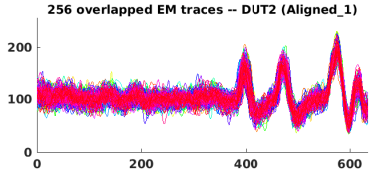
We now consider a more standard attack scenario where an adversary exploits the leakage of an S-box execution in the first round of the AES-128 block cipher. In this case, for the profiling traces, we randomize all the 16 bytes of AES-128 key and AES-128 input data to execute an AES-128 encryption. And for the attack traces, we fixed a random 16-byte AES-128 key and randomize the input data per trace. Similarly to the previous section, we train the ResNet model using 256 classes corresponding to the S-box outputs with the profiling traces, and we also use 20% profiling traces as validation data. For the DUT2 experiments, we use a batch size of 32 and 30 epochs, “Adadelta" optimizer with an initial learning rate of 1.0, adaptively reducing the learning rate with a factor of 10 if the validation loss is not decreased within 10 consecutive epochs.

To evaluate the impact of misalignment in this attack scenario, the same ResNet model is again used to perform deep learning attacks with aligned traces and misaligned ones, this time considering three different data sets. In the first one, next denoted as the *Aligned_1* data set (applying our synchronization method 9 times) and shown in the top graph of Figure 5.9, the traces are well aligned at the leakage part marked red and every trace has 634 sample points. The second data set (next denoted as the *misAligned_1* data set and shown in the middle graph of Figure 5.9) corresponds to the most misaligned traces. It is obtained by applying the alignment method only three times to the raw traces. In this case, to include the leakage part in the traces, every trace consists of 11,598 sample points. Finally, for the last data set (denoted as the *misAligned_2* data set and shown in the bottom graph of Figure 5.9), we apply only one alignment step less than what we do for the *Aligned_1* data set. As a result, the raw traces are aligned very close to the leakage part, but still the traces are not aligned at the leakage interval, and every trace contains 744 sample points.

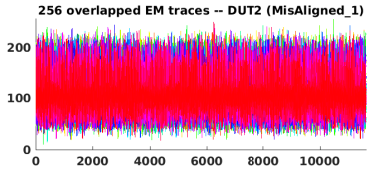
To further demonstrate the impact of misalignment, we also calculate the SOST trace per data set of DUT2 as shown in Figure 5.10.

Based on this setup, first we launch the deep learning attacks using the *Aligned_1* data set. All the 16 S-boxes are getting similar results as depicted in the top graph of Figure 5.11.

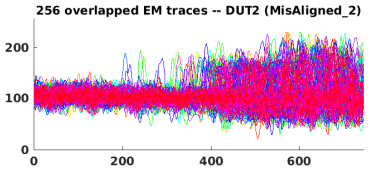
We then performed the same attacks as in the previous section against the *misAligned_1* data set. In this case, it took about one week per S-box for 30 epochs with a single NVIDIA GTX 1080Ti GPU card, hence highlight-



(a) Aligned_1

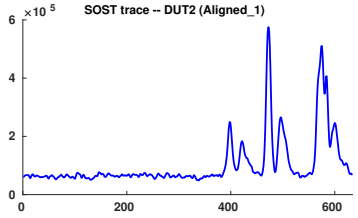


(b) MisAligned_1

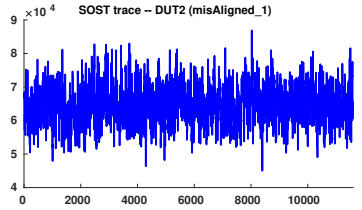


(c) MisAligned_2

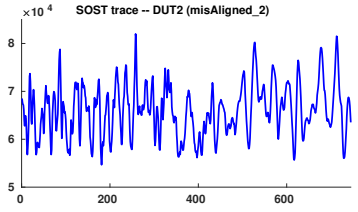
Figure 5.9: 256 overlapped aligned and misaligned EM traces of DUT2.



(a) Aligned_1

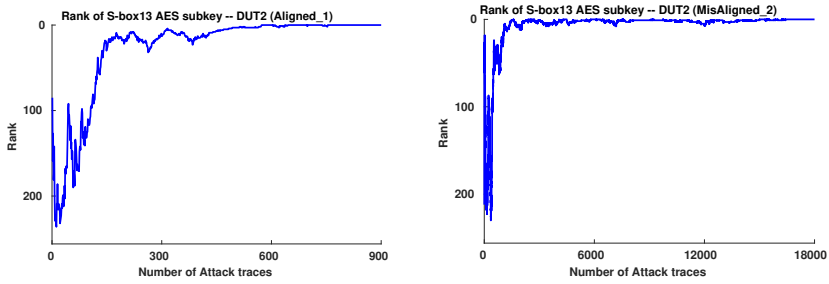


(b) MisAligned_1



(c) MisAligned_2

Figure 5.10: SOST traces of aligned and misaligned EM traces of DUT2.



(a) Using *Aligned_1* aligned traces. (b) Using *misAligned_2* misaligned traces.

Figure 5.11: Rank of correct subkey candidate with DUT2 data sets.

ing the importance of alignment also for time complexity reasons. The attacks took about 4 hours per S-box for 30 epochs for the *Aligned_1* data set. Even after running it for a few weeks with different hyper-parameters (*e.g.*, optimizer, number of residual blocks, number of filters, batch size, number of epochs, adding dropout layer [SHK+14] right before the last softmax layer in Figure 5.1), the rank of the correct subkey candidate remained very low and it was not possible to recover the subkeys. This result moderates the intuition that in a deep learning context, the best practice is to use the raw data without dimensional reduction, and the previous observation in [MPP16, PSB+18] that dimensional reduction methods lead to worse results in deep learning SCA context.

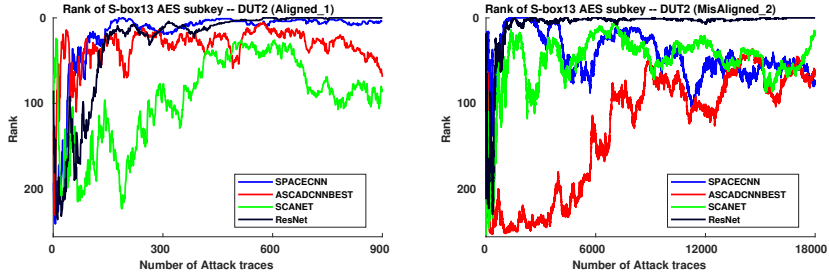
Following, we further conduct the deep learning attacks on the *misAligned_2* data set. For this purpose, we first use the same ResNet model and the same amount of profiling traces as what we used in the *misAligned_1* data set case, which is 1,000 profiling traces per class of S-box output value including 20% of validation traces. In this case, the rank of the correct subkey remained stuck at high values. We next tweak the hyper-parameters of the ResNet model and still get similar results. Eventually, we increase the number of profiling traces to 1,200 and add a dropout layer (with a dropout factor of 0.3) right before the last softmax layer. To speed up the tests, we use dual NVIDIA GTX 1080Ti GPU cards. The best rank results are shown in the bottom graph of Figure 5.11. We also perform similar attacks (using the same ResNet model with the same amount of profiling traces and same dropout factor) on the other S-boxes using the same *misAligned_2* data set, this time with worse results.

From the profiling perspective, no big difference could be observed between the training accuracy using aligned traces from the *Aligned_1* data set and the one using misaligned traces from the *misAligned_2* data set.

However, from the attack perspective, as can be seen from Figure 5.11, the aligned traces show much better key rank results. With aligned traces, the correct subkey is steadily ranked at the top position with more than 750 attack traces. The required number of attack traces for the misaligned case is about 16,294 (with more tweaks of the hyper-parameters and more profiling traces). We note again the larger ratio between the number of profiling and attack traces, that typically reflects a more challenging target.

5.4.1 Performance comparison with the state-of-the-art

To further check the performance of our ResNet model, Figure 5.12 displays the rank of correct subkey candidates using our ResNet model and the other 3 models with both *Aligned_1* and *misAligned_2* data sets. For both aligned and misaligned traces, again, our ResNet model can recover the correct subkey byte faster than the others. Especially for the misaligned traces, our ResNet model outperforms the others a lot.



(a) Using aligned traces.

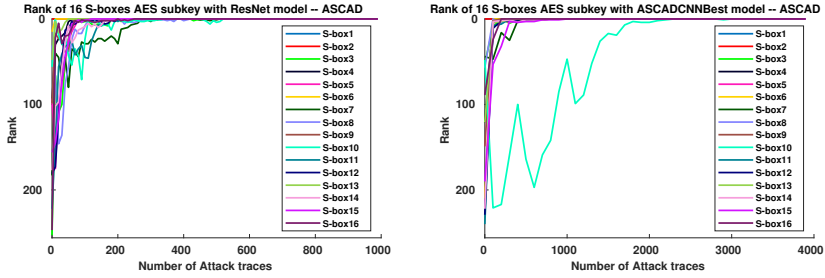
(b) Using misaligned traces.

Figure 5.12: Performance comparison of 4 NN models using DUT2 data sets.

5.5 Comparison with ASCAD_CNN_Best model using ASCAD data set

To further demonstrate the generalization of our ResNet model, we compare the rank results of 16 S-boxes based on the published ASCADv1 fixed-key[2] data set (700 sample points per S-box) using both the ResNet model and the published ASCAD_CNN_Best model. For both models, we use batch size of 100 and 200 epochs (the best choice for ASCAD_CNN_Best). For the ResNet model, we use "Adadelta" optimizer with an initial learning rate of 1.0, adaptively reducing the learning rate with a factor of 10 if the validation loss is not decreased within 10 consecutive epochs. For ASCAD_CNN_Best model, we use "RMSProp" optimizer with an initial learning rate of 10^{-5} and keep it fixed during the training (the best choice for ASCAD_CNN_Best).

Figure 5.13 shows the rank of correct subkey candidates of all 16 S-boxes using both models. For both models, all 16 subkey bytes can successfully be recovered using less than 4,000 traces. The first two S-boxes are not masked and the correct key candidates of them can be recovered with only 1 trace. For our ResNet model, all the subkey bytes can be recovered using less than 600 traces. For the ASCAD_CNN_Best model, 15 subkey bytes can be recovered using around 1,000 traces but S-box 10 requires 3,000 traces to get stable results. It further confirms the good generalization of our ResNet model.



(a) Using ResNet model.

(b) Using ASCAD_CNN_Best model.

Figure 5.13: Performance comparison of ResNet and ASCAD_CNN_Best models using ASCAD data set.

[2] https://github.com/ANSSI-FR/ASCAD/tree/master/ATMEGA_AES_v1/ATM_AES_v1_fixed_key

5.6 Conclusions

Our results confirm that neural network models are powerful tools for black box leakage analysis and demonstrate the good generalization of the proposed ResNet model compared with the other state-of-the-art NN models in a side-channel context. Yet, even with such powerful tools, preprocessing the leakage traces with alignment/re-synchronization methods (and possibly filtering, ...) can be highly beneficial to the attacks/evaluations' success. This is true from the data complexity point-of-view, and data complexity is generally accounted for the most important complexity measure in side-channel analysis. But our second DUT shows that it is also true from the time complexity point-of-view. Indeed, alignment plays the role of data compressing in this sense because a much shorter interval will be used. It then allows for significantly reducing the number of samples in the traces fed to the ResNet, which may reduce its execution time from weeks to days (or even hours). The latter conclusions naturally become increasingly relevant for devices protected with a variety of countermeasures. As stated previously, our conclusions are specific to the investigated datasets. In particular, deep ResNets only outperform other tested models in our case study (and are not claimed to be universally better, as popularized by the no-free-lunch theorem). We believe our conclusion regarding the interest and sometimes need of preprocessing to be more general (since consistently observed with different models and datasets).

Chapter 6

Fully Convolutional Networks to Attack Scalar Multiplication

We aim to simplify the worst-case horizontal attack on scalar multiplication discussed in Chapter 3 by making use of deep learning techniques, and to automate the critical steps of this previous work, namely the information identification, information extraction and information combination steps. For this purpose, we gradually increase the number of automated steps, targeting a very challenging assembly-level regular Montgomery ladder scalar multiplication implementation on a BeagleBone Black (BBB) Board running at 1 GHz. Our results demonstrate that the latter two steps can be simplified using deep learning techniques and lead to similar results as previous work. By contrast, the first step still requires some additional engineering. By showing that POIs selection can play an important (sometimes necessary) role for attacking asymmetric cryptography algorithms using deep learning techniques, we bring a more contrasted view on the advantage and limitations of such techniques. To the best of our knowledge, this is the first public report of deep learning based attack on ECC implementations. Besides, we propose the use of Fully Convolutional Networks (FCN) as an alternative (deep) learning tool for side-channel analysis.

Scalar multiplication is the cornerstone of ECC cryptosystems, thus its implementation security is critical for concrete deployment. As surveyed

in Chapter 3, in the past decades, two categories of attacks targeting the scalar multiplication have been presented in the literature: *Divide and Conquer* (DC) ones and *Extend and Prune* (EP) ones. The DC approach is trying to recover the scalar bits independently, so each scalar bit is associated with a probability of score. The EP approach is trying to recursively recover the scalar bits, so that the attacker has to recover all the $i-1$ first bits to recover the i-th bit.

In this chapter, we are concerned with a very powerful type of EP attacks, denoted as HDPA [BJPW13a, CFG+10]. As discussed in Chapter 3, these attacks are in the same time very interesting for security evaluations (since they are able to extract a lot of information from leakage traces) and cumbersome to mount. In order to systematize the analysis, we proposed in Chapter 3 a principled approach based on three steps – information identification, information extraction and information combination. Concretely, we performed the information identification with a correlation attack, and the next two steps with linear regression [SLP05], which implies significant engineering efforts (in terms of time complexity, amount of data to collect and memory complexity). It also requires a good knowledge of the implementation and side-channel analysis techniques.

On the other hand, in the last few years, the development of machine learning/deep learning in the side-channel context [HZ12, LBM15, MPP16, HGM+11, LPB+15, CDP17, HGG18, CCC+19a, RQL19, Tim19, PH18, PHJL17, PSB+18, PHJ+19] is booming as a powerful alternative to conventional profiling techniques. In this chapter, we are motivated to use deep learning techniques to simplify the complicated steps of the previous work in Chapter 3. For this reason, we study the challenging case of a regular Montgomery ladder scalar multiplication implementation based on a BeagleBone Black Board running at 1 GHz. It turns out that a good part of our previous single-trace attack in Chapter 3 can be automated, which provides security evaluation labs with an easier way to perform worst-case security evaluations of ECC cryptosystems. Nevertheless, our results suggest that the alignment and preprocessing (*i.e.*, information identification) step is still needed for successful deep learning attacks on such challenging targets. It is in line with the observation of the RSA case discussed in the latest CHES'19 paper [CCC+19a]. Their deep learning attacks on RSA modular exponentiations used SNR-based POI selection. Overall, the target operations for asymmetric cryptography implementations usually contain a lot of sample points. Therefore, the POI selection is crucial for the deep learning attack performance.

The rest of this chapter is organized as follows. Section 6.1 introduces the necessary background and the FCN neural network model that we used. Section 6.2 shows the target implementation, the required preprocessing of the EM traces and deep learning experimental results. Finally, we conclude and offer some directions for future research in Section 6.3.

6.1 Background

For most of ECC cryptosystems, the scalar k is sensitive data because it is either directly used as a private key (*e.g.*, ECDH key exchange) or used as an ephemeral key (*e.g.*, ECDSA) related to the private key. As in our previous work presented in Chapter 3, we also target the representative SPA/SEMA-secure Montgomery ladder scalar multiplication implementation (see Algorithm 1) in this chapter. The results can be naturally applied to implementations using scalar randomization countermeasures (similarly, with the knowledge of random blinding factor or targeting some curves with sparse group order [FRV14] during the profiling phase) against DPA attacks.

In this chapter, we also convert the affine coordinates to Jacobian coordinates to avoid expensive field inversion operations of point addition and point doubling as illustrated in Algorithm 2 and Algorithm 3. The cost of each point addition and each point doubling is $16\texttt{MUL} + 1\texttt{ADD} + 6\texttt{SUB}$ and $10\texttt{MUL} + 9\texttt{ADD} + 4\texttt{SUB}$, respectively.

6.1.1 Fully Convolutional Networks

FCNs have been first introduced for semantic segmentation on images [LSD15]. FCN is a kind of CNN without fully connected layers, it recovers the pixel-level classification information from abstract features. We refer the readers to the original paper [LSD15] for more details. The core difference between traditional Convolutional Neural Networks (CNN, such as AlexNet, VGGNet) and FCN is, traditional CNNs are suitable for image-level tasks but FCN is extended to suit for pixel-level tasks. We consider sample points of side-channel traces as pixels and there is no previous work using FCN in side-channel context, so we want to introduce FCN into side-channel attacks. They have then been widely used for different applications, due to their compelling quality and efficiency. Classification is of course one of its application scenarios.

The design idea of FCN is simple: as depicted in Figure 6.1, it is a stack of several basic blocks. Each basic block is composed of three layers: a convolutional layer γ followed by a batch normalization layer β [IS15] and a ReLU activation layer σ [NH10]. After stacking five basic blocks, a global average pooling layer δ is adopted (instead of a fully connected layer), in order to largely reduce the number of weights to be trained. Finally a softmax layer s is adopted to generate the class label of the input side-channel trace. We adopt the same strategy as in ResNet [HZRS16a] to exclude any pooling operation for each basic block, in order to prevent overfitting. Batch normalization is applied to speed up the convergence and help improve generalization.

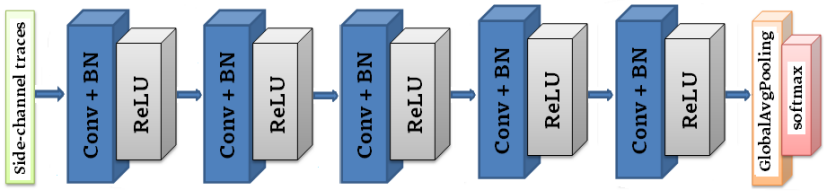


Figure 6.1: Structure of FCN.

In summary, the FCN model can be written as:

$$\texttt{FCN} = s \circ \delta \circ [\sigma \circ \beta \circ \gamma]^{n_1}, \tag{6.1}$$

where n_1 denotes the number of basic blocks (we set it to 5 for all our experiments). We further use 32, 128, 128, 256 and 256 filters for these five basic blocks. The kernel size of the first and last convolutional layer is 8 and 3, respectively. The other three convolutional layers are using the same kernel size of 5. Regarding our choice of hyperparameters, we followed the best practice of other deep learning application fields to do a small grid-search, that is, varying the number of basic blocks from 3 to 7, and then varying the optimizer among "Adadelta", "Adam", "Nadam", "RMSProp" and "SGD". For the number of filters we reuse the typical values used in most of computer vision applications, and for the kernel size we also use small values since they showed better performance in other deep learning fields. We must mention that it is not an ideal way to decide the network structure and other hyperparameters, one of our future work direction is to use latest Neural Architecture Search (NAS) [PGZ+18, CZH19] techniques to automatically design effective neural network architecture for our data

set.

Similarly, we use the *Accuracy* and *Loss* metrics described in Section 5.1.4 to monitor and evaluate neural network models. For comparison with the previous work in Chapter 3, we also use the same first-order Success Rate (1-O SR) metric, which is the probability of recovering the target n-bit scalar computed over repeated experiments.

6.2 Experimental Results on BeagleBone Black Implementation running at 1GHz

Concerning simplifying the complex steps while preserving similar attack performance in Chapter 3, in this chapter, we target the same Montgomery ladder scalar multiplication implemented on the BeagleBone Black board running at 1 GHz. More precisely, we reuse the same set of traces after the preprocessing (correlation-based alignment) as described in Chapter 3. To this end, we use 100,000 preprocessed EM traces for profiling and 2,200 preprocessed traces to attack (similar to our previous work). 20% of the profiling traces are used as validation data to improve the training of the weights. Each preprocessed EM trace contains 518,491 sample points.

6.2.1 Neural Network Architecture

Our experiments are implemented using Keras [C+15] and Tensorflow [AAB+15] with Nvidia GTX 1080Ti GPU. We employ the aforementioned FCN neural network model in Section 5.1.4. During our experiments, we use dropout of 0.2 for the last two basic blocks (after the ReLU activation layer) and of 0 for the rest [SHK+14]. All layers are randomly initialized with uniform initialization [GB10].

We use batch size of 1 due to the large number of sample points per trace and Adadelta optimizer with an initial learning rate of 1.0, adaptively reducing the learning rate with a factor of 10 if the validation accuracy is not increased within 30 consecutive epochs. The training will be stopped after 150 epochs or if the learning rate is getting lower than 10^{-3}.

6.2.2 BBB Results

As mentioned before, we are motivated to simplify the three critical steps of systematic approach proposed in previous work [PZS17] using deep

learning techniques. We take a two-step progressive strategy to conduct our experiments. We start with only simplifying the most sophisticated (information extraction and information combination) steps of the previous systematic approach in Chapter 3. We next try to automate all three steps (*i.e.*, including the information identification).

Results with Information Identification.

In this first experiment, we use the same information identification step as [PZS17] to identify the POI to feed into the neural networks. This step consists of three parts: unprofiled correlation, partial SNR and optimizations.

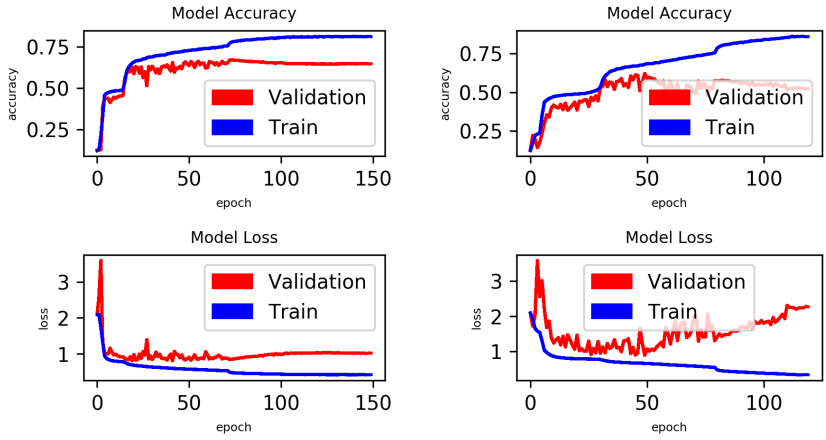
We only focus on the higher 32-bit result of each umull and umaal instructions for all the available register operations $\mathbf{r}_i$ of each scalar bit.

We apply the unprofiled correlation attack and the partial SNR approach from [PZS17] to efficiently identify the time positions of the corresponding registers r_i^j. The POI search is performed using the preprocessed $N_{poi} = 100,000$ traces $\mathbf{l}$ acquired using random known inputs (P^q) and scalars $(\mathbf{k}^q)$, $q \in [0, N_{poi} - 1]$. The N_{poi} leakages of the t-th time sample of each trace is denoted as a $\mathbf{l}[t]$ vector of size N_{poi}.

The unprofiled correlation approach boils down to compute Pearson's correlation coefficient ρ between each time sample for the N_{poi} internal values $\mathrm{r}_i^j = r_i^j(P^q, \lceil \mathbf{k}^q \rceil_i)$ and a Hamming weight leakage model HW. For our POI search, we compute $\rho(\mathsf{HW}(\mathrm{r}_i^j), \mathbf{l}[t])$, $t \in [0; 518,490]$.

To calculate partial SNR, the 32-bit values of r_i^j are first truncated to x bits. Then each trace is labeled with its truncated value and split into 2^x sets $\mathcal{S}_i$. The partial SNR of each time sample equals to $\frac{\mathsf{var}(\mathsf{mean}(\mathcal{S}_i))}{\mathsf{mean}(\mathsf{var}(\mathcal{S}_i))}$, in which, var and mean are the sample variance and mean functions. The time sample showing the highest SNR ratio is chosen as the time sample of r_i^j.

Applying each of these two methods on the full trace for all r_i^j's is computationally intensive. So using the fact that the time order of the registers is $(r_0^0, ..., r_0^{N-1}, r_1^0, ...r_i^j, ...r_{n-1}^{N-1})$, we first search r_0^0 among the first W time samples. Using correlation, we decide r_0^0's position by computing a p-value with a threshold of 5 [DS16]. We repeatedly move the window to the next W time samples until r_0^0 is found. The search of r_0^1 is similar with setting the initial offset of the window to r_0^0. We iterate this process to find all the registers. We use a window value W of 20,000 for our POI search.



(a) 80,000 training traces.

(b) 40,000 training traces.

Figure 6.2: FCN Model Accuracy and Loss with POI selection.

As in Chapter 3, we finally attack the first 4 bits of the scalar. The number of POIs is 3,492 so we feed those POI sample points into the neural network. We label each trace using the value of the first 4 bits of its scalar. The training accuracy/loss and validation accuracy/loss (as described in Section 5.1.4) of the FCN model are given in the left graph of Figure 6.2 when using 80,000 training traces and 20,000 validation traces out of 100,000 profiling traces. The training takes about 23 hours, it could be halved because the validation accuracy is getting stable after 70 epochs as can be seen from the accuracy graph.

After the training, we use the trained model to recover the 4 scalar bits of all 2,200 attack traces. We calculate the 1-O SR and use the same threshold mechanism as in Chapter 3 to discard the wrong attack results. That is, by setting a probability threshold under which some attack traces will be discarded. A higher threshold provides more confidence to have a successful partial nonce recovery, but it comes with the cost of increasing the number of discarded attack traces.

Concretely, we summarize in Table 6.1 the evolution of success rate in function of the probability threshold over the 2,200 attack traces. The scalar 1-O SR is 0.6692 when the threshold is set to 0.5. Yet we need 140 ECDSA nonces to recover the secret key with 4 bits of partial information using

lattice attacks [PZS17]. Since no error on the partial information is tolerated in a lattice attack context, the success rate of the key recovery is calculated as $0.6692^{140} \approx 3.7 \cdot 10^{-25}$. So we have to discard wrong attack results.

As can be seen, the first-order success rate is increasing with higher probability thresholds. We achieve a success rate of 1 using a threshold of 0.99. In that case, we discard 2,055 of the attack results and still remain 145 of them, which is remarkably similar to the results of our previous work in Chapter 3.

Threshold	Scalar 1-O SR	Key 1-O SR	# discarded result	# remaining result
0.5	0.6691519105	$3.7 \cdot 10^{-25}$	764	1,436
0.75	0.7562225476	$1.0 \cdot 10^{-17}$	1,167	1,033
0.9	0.8251572327	$2.1 \cdot 10^{-12}$	1,564	656
0.95	0.8950495049	$1.8 \cdot 10^{-7}$	1,748	452
0.96	0.9090909091	$1.6 \cdot 10^{-6}$	1,800	400
0.97	0.9247910863	$1.7 \cdot 10^{-5}$	1,868	332
0.98	0.9503546099	0.0008	1,957	243
0.985	0.9629629630	0.0051	2,012	188
0.99	1	1	2,055	145

Table 6.1: Evolution of the ECDSA scalar and key recovery success rate in function of the threshold.

We further investigated the impact of reducing the number of training (resp. validation) traces to 40,000 (resp. 10,000). The right graph of Figure 6.2 illustrates the training and validation accuracy/loss. The validation accuracy is worse than using 80,000 training traces. A similar observation holds for the scalar 1-O SR and key 1-O SR. Both suggest that the model is still improving for the amount of collected traces.

Results without Information Identification.

We finally investigated the full automation of the three steps of the previous systematic approach presented in Chapter 3.

For this purpose, we skipped the POI search and directly feed all the 518,491 sample points per trace into the neural networks. Due to the huge number of sample points per trace, the training is very time consuming.

We use the same neural network architecture as in Section 6.2.1 to conduct the experiments, but in this case we only use 100 epochs because each epoch takes about 2.5 hours. Figure 6.3 displays the training accuracy/loss and validation accuracy/loss. The training accuracy and validation accuracy are pretty low, leading to unsuccessful attacks. The big drop of validation accuracy (spike of validation loss resp.) is caused by the dropout mechanism that we used during the training, and at that stage the weights of neurons are not converged yet. Later there is no big drop when the training is getting stabilized.

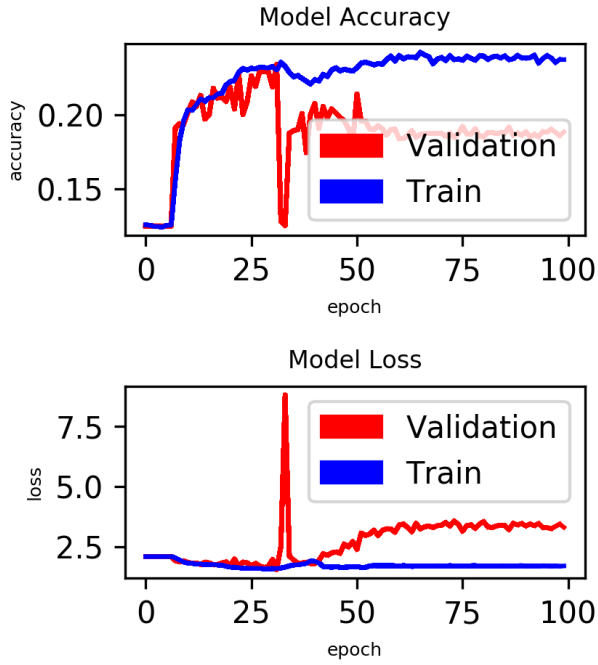


Figure 6.3: FCN Model Accuracy and Loss without POI selection.

Regarding the failure of the attack, we give our tentative explanation. First, as we mentioned before, FCN model is designed for extracting pixel-level classification information from the raw data, taking into account that there are more than half million sample points (pixels) per trace, the tuning of neuron weights are depending on about 150 times more sample points (pixels) compared to the previous case of training with POIs. From deep

learning point of view, generally it requires more data and more epochs to make the training converge, which we can not afford due to our computation power, we will consider it in our future work. Secondly, the neural network model is directly reused from the previous experiment, which we did small grid-search based on the POIs dataset to choose the number of blocks and optimizer of FCN model. The adaptation of network architecture is definitely necessary considering the results. NAS-type of techniques [PGZ+18, CZH19] are in scope.

These results suggest that despite their versatility, deep learning based attacks benefit from a good selection of points of interest in case of challenging targets (in particular, it allows fitting the model with much less traces). The investigation of more suitable deep learning techniques for this kind of application is an interesting scope for further research.

6.3 Conclusion and Future Work

Motivated by simplifying a previous systematic approach of worst-case HDPA attack on secure scalar multiplication implementations, we use deep learning techniques to automate the critical steps of such attacks. Our experimental results of an ARM Cortex-A8 running at 1 GHz (a presumably noisy target) suggest that the sophisticated information extraction and information combination steps can be performed in a quite black box manner using deep learning techniques. On the other hand, they also suggest the need of dimensionality reduction/detection of points of interest before launching the attack (or at least, our results show that such a preliminary step makes deep learning attacks significantly more efficient), which is a generic problem to tackle for attacking asymmetric cryptography implementations using deep learning techniques considering the large number of sample points per trace. Simple SNR-based POI selection could be a cheaper and faster alternative as demonstrated in single-trace attack on RSA case [CCC+19a] using deep learning techniques.

For the developers of scalar multiplication implementations, our results confirm that scalar randomization activated implementations are generally at risk, considering both the huge amount of informative samples such implementations offer and the simplicity deep learning techniques provide. To mitigate this issue, point randomization is probably the most suitable solution. Evaluating that kind of implementations is an interesting track for further investigations. Another potential direction is to target register

manipulations of point addition and point doubling operations from the elliptic curve arithmetic level.

Chapter 7

Towards More Efficient S-box Pooling Profiled Attacks

NOWADAYS, profiled attacks are the standard penetration tests for security evaluations. Often the security evaluators have to perform profiled attacks on each S-box to quantify the security strength of the target symmetric cryptographic algorithm implementations more accurately. The required time to conduct such profiled attacks is very long due to the number of profiling traces (for many certification bodies, at least 1,000,000 are mandated). It is getting even more time-consuming after introducing deep learning profiled attacks. Furthermore, some certification bodies instruct up to 5,000,000 or 10,000,000 profiling traces because modern embedded secure IC products have more and more countermeasures against side-channel attacks. It is a challenge to simultaneously decrease the number of required profiling traces and the required profiling time while retaining the attack performance for profiled attacks. In this chapter, we propose a simple yet remarkably effective pooling approach to address this problem for security evaluations. That is, pooling over the S-boxes to build a large profiling set and perform the profiling on this large set once. Intensive experiments are conducted with this pooling approach using different profiling tools (template attack and its pooled variant, stochastic model and deep learning) on three different AES-128 implementations (a sequential S-box software AES-128 implementation without masking, a se-

quential S-box software AES-128 implementation with first-order masking and a parallel S-box hardware AES-128 implementation with first-order masking). The experimental results have shown that the proposed pooling approach can lead to similar attack performance while decreasing both the required number of profiling traces and the required profiling time by a factor of 8 or even 16. Note that the proposed S-box pooling is likely not applicable if the S-boxes leak differently.

In Kocher's seminal work [Koc96], SCA were proposed to extract secret keys of cryptographic algorithms implementations via the timing side channel. Since then, SCA have drawn plenty of attention in the community. On the one hand, it is extended to different side-channels, *e.g.*, power consumption [KJJ99], electromagnetic radiation [QS01]. On the other hand, different distinguishers and new approaches are also adapted to SCA. Amongst them, profiled attacks are considered the most powerful SCA after Chari *et al.* published the novel template attacks [CRR02]. Various research works have focused on improving profiled attacks from different perspectives, *e.g.*, for efficiency purposes [SLP05, CK13], or portability and robustness [EG12, CK14, BCH+20, WO15].

It is worth noting that there are two types of profiled attacks depending on what we are profiling, either profiling the key itself or profiling some intermediate data that depends on it (*e.g.*, S-box output). Generally speaking, profiling the key directly is more studied in the asymmetric cases [CCC+19a, ZBHV21, ZS19], profiling key-dependent values (*e.g.*, S-box output) is more studied in the symmetric cases, and key transportation [OP11, CK13, CK18, ZS20, WdHG+20] is less often studied (though applies in all cases).

As introduced in Section 2.4.1, profiled attacks in security evaluations consist of three stages. To simplify the task of the evaluators, there is not much we can do considering Stage 1 and Stage 2. Also, we cannot skip or shorten the necessary re-synchronization preprocessing for Stage 3 [ZS20]. And the same target implementations in [ZS20] are considered in this chapter. Hence, the only option left is to *find a more efficient way to decrease both the number of required profiling traces and the required profiling time while preserving the attack performance*. Somewhat surprisingly, there have been limited attempts to characterize and improve such practical challenges. In this chapter, therefore, we aim to study it for popular profiled distinguishers and various target implementations.

The general idea of our work is S-box pooling. The first step is to reconstruct a larger set of profiling traces by pooling the profiling traces corre-

sponding to each S-box. The second step is to execute profiling only once on this new profiling traces set. The last step is to attack all the S-boxes to disclose all the subkeys (*e.g.*, 16 in the AES-128 case) using the attack traces.

We note that a recent and independent work investigated a similar technique (S-box pooling, which they denote as cross-subkey training) from a different perspective [HWW21]. Namely, their focus is to improve the attack performances when the number of profiling traces is overly limited. They use the accuracy metric while we use the guessing entropy metric for attack performance comparison. In their work, only deep learning profiled attacks are considered so accuracy could be a suitable metric. They have observed significant accuracy improvement with cross-subkey training compared to the case where the amount of profiling traces is overly limited. It can be explained by the fact that sufficient profiling traces are available by using cross-subkey training and all 16 S-boxes are expected to leak in a very similar way because of the unmasked sequential software AES-128 S-box implementation. Both factors lead to the better generalization of the deep learning model they are using. We relatively aim to optimize the profiling complexity when this number is sufficient. We also cover more target devices (they only target an unprotected AES-128 software implementation: we additionally cover a masked AES-128 software implementation and a masked hardware one) and distinguishers (they only consider one deep learning distinguisher: we additionally cover other state-of-the-art profiling tools). Overall, their conclusions are complementary to ours and show that S-box pooling can also improve attack performances when the number of profiling traces is limited.

The rest of this chapter is organized as follows. Section 7.1 introduces the necessary background on profiled attacks used in this chapter. Then, we describe our proposed S-box pooling approach and methodology in Section 7.2. Finally, Section 7.3 demonstrates the effectiveness of this pooling approach based on the intensive experimental results on three different AES-128 implementations, for different profiling tools.

7.1 Background and Preliminaries

Since Chari *et al.* have introduced template attacks in their pioneering work [CRR02], profiled attacks have gradually become the commonly recognized most powerful side-channel attacks. Profiled attacks consist of two phases, *i.e.*, the profiling phase and the attack phase as shown in Fig-

ure 7.1. During the profiling phase, an attacker/evaluator uses a profiling device (and has control of the key or at least knows the key) to model the leakage characteristic of the target key-dependent sensitive data (typically the S-box output of symmetric algorithms) with the side-channel traces of the target implementation. The outcome of the profiling phase is the built leakage characteristic models for every possible target sensitive data value, *e.g.*, 256 values of the AES S-box output [1]. During the attack phase, the attacker/evaluator uses the victim device to measure the side-channel traces of the target implementation. And then, he/she matches the traces with the previously built leakage characteristic models of the target sensitive data. For each attack trace, the adversary calculates the target sensitive data based on the known input and the guessed key unit (*e.g.*, one key byte). A score is computed for each possible guessed key unit value (256 possible values for one byte of AES key) per each trace. In the end, for each hypothesised key unit value, the scores for all the attack traces are combined using *e.g.*, the maximum likelihood method to get a combined score of that specific guessed key unit value. The combined score of each possible hypothesised key unit value is compared to find the highest one. The hypothesised key unit value with the highest score is considered the recovered key unit. This attack process is repeated for all key units to reveal the complete key.

In this chapter, we use four different state-of-the-art profiled attacks, namely, template attack (TA) and its pooled variant (TA_p), stochastic attack (SA) and deep learning attack (DL), to experimentally investigate the efficiency of our proposed pooling profiling traces approach.

7.1.1 Template attack

From an information-theoretic viewpoint, TA is believed to be the most powerful type of SCA [CRR02] when (1) the noise of side-channel traces follows the Gaussian distribution and (2) an unlimited number of traces are available [LPB+15, HRG14]. It makes use of a multivariate normal distribution to model the probability density function of each possible target sensitive data given a leakage observation, so it is parameterized as

$$p(L = l|S = s) = \frac{1}{\sqrt{(2\pi)^d|\sigma_s|}} e^{-\frac{1}{2}(l-\mu_s)^\top \sigma_s^{-1}(l-\mu_s)}. \tag{7.1}$$

[1] That is, the identity model is used for labeling as what we do in this chapter. The Hamming Weight model can also be used for labeling.

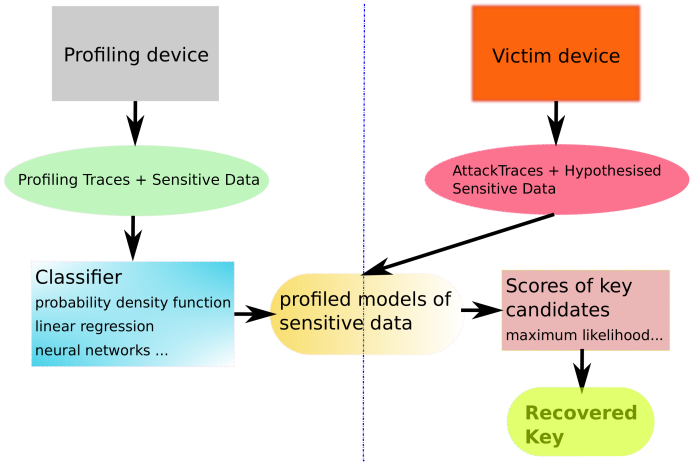


Figure 7.1: Profiled attacks schematic diagram.

In this equation, d is the number of sample points. $|\sigma_s|$ denotes the determinant of the covariance matrix and $\top$ indicates the transpose. In practice, usually, some POIs are detected first for dimension reduction purpose.

In the attack phase, the probability $p(L = l|K = g)$ for each key candidate is set by $p(L = l|S = s)$ given a known input of each attack trace. The classification of each key guess is then computed based on Bayes' Theorem as follows:

$$p(K = g|L = l) = \frac{p(L = l|K = g) \cdot p(K = g)}{p(L = l)}. \tag{7.2}$$

Based on an assumption that all attack traces are independent, to make use of all available attack traces for each key guess, a final score of each key guess is calculated as below,

$$p_g = p(g|L) = \frac{\prod_{m=1}^{M} p(L = l_m|K = g) \cdot p(K = g)}{\prod_{m=1}^{M} p(L = l_m)}. \tag{7.3}$$

In practice, the p_g is usually calculated using the sum of the log-posterior to avoid the potential arithmetic underflow problem. The highest p_g indicates the correct key candidate g^*.

The pooled variant of the TA, next denoted TA_p, is proposed in [CK13]. It uses only a single pooled covariance matrix σ to reduce the profiling complexity, which is dominated by the estimation of the covariance. The rest is the same as normal TA.

7.1.2 Stochastic model attack

SA [SLP05] relies on linear regression to build the leakage characteristic models of sensitive data. It assumes that the side-channel leakage observation of the target sensitive data s at time t consists of two parts $l_t(s) = h_t(s) + R_t$, where $h_t(s)$ is the key-dependent part and the latter one is a non-key dependent noise term with zero mean. Similar to TA, the profiling also contains two parts, the linear approximation $\hat{h}_t$ of h_t and the estimation of noise-related covariance matrix σ. The estimation of $\hat{h}_t$ for each time instantiation is done in a chosen suitable u-dimensional vector subspace $\mathcal{F}_{u;t}$. In this chapter, we choose the subspace $\mathcal{F}_9$ by utilizing the bitwise coefficients of the AES S-box output. Afterwards, $d1$ POIs are chosen based on the estimated $\hat{h}_t$ to compute the covariance matrix σ. The profiling results in a Gaussian multivariate density $\hat{f} : \mathbb{R}^{d1} \rightarrow \mathbb{R}$.

During the attack phase, we only consider the maximum likelihood principle to recover the key following [GLP06, SLP05]. More specifically, the correct key guess g^* is the one that maximizes:

$$p_g = p(g|L) = \prod_{m=1}^{M} \hat{f}(\mathbf{l_t}(s_{m,k}) - \hat{\mathbf{h}}_\mathbf{t}(s_{m,g})). \tag{7.4}$$

In this equation, k indicates the unknown correct subkey to be revealed.

7.2 Methodology

As discussed above, the problem to be solved in this chapter is: to simultaneously decrease both the required number of profiling traces and the required profiling time while the performance of the online attack is essentially unchanged.

In the following, we first introduce the proposed S-box pooling approach and how it resolves this challenge for security evaluators. We then describe the methodology, which we will use to systematically investigate the soundness of the proposed pooling approach in a profiled attacks context. It includes the metric for comparing profiling performance, the met-

rics used for training neural networks, the knowledge of POIs assumption, the POI selection approach, and the choice of parameters for different profiled attacks in our experiments. For simplicity, we will only discuss the AES case. But the principle can be easily extended to other symmetric cryptographic algorithms.

7.2.1 S-box Pooling profiled attack

The core idea of this pooling approach is: to extract the profiling traces of each S-box and to stack the extracted profiling traces for all S-boxes in order to build a new large set of profiling traces. In theory, this approach is based on improving the signal-to-noise ratio of the profiling traces by increasing the amount of the profiling traces. Instead of directly measuring a lot of profiling traces, it is to gather enough profiling traces by extracting and stacking all available S-box calculation segments of each side-channel trace. For instance, we measure 1,600 profiling traces of the first round of AES-128 encryption. So we have 1,600 traces for each S-box. By extracting and stacking then we build a new set of 25,600 (= 16×1,600) profiling traces since there are 16 S-boxes. In the ideal situation where the S-boxes are leaking according to the same model, this strategy will lead to a reduction of the number of profiling traces by a factor of 16 (for the AES-128). Furthermore, the profiling time will be reduced by the same factor (In this chapter, for performance comparison, no matter with and without S-box pooling, we only considered using the same deep learning model.) since it only needs to train a single model. Of course, in practice, the situation may not be ideal and it is the goal of our following investigations to clarify the extent to which S-box pooling is a good trade-off for concretely relevant case studies. For this purpose, we used the following 4-step method:

1. Step 1: Measure enough profiling traces (the number of profiling traces marked as N_{ori}) to be able to retrieve the AES-128 key, perform TA, TA_p, SA and DL profiled attacks on each S-box separately to recover all 16 subkeys. Make them the baseline attack performance for later use and comparison.
2. Step 2: Determine the minimum required number of profiling traces (denoted as N_{min}) to achieve similar profiling performance as the baseline using a binary search algorithm. That is, starting from $N_{ori}/2$ profiling traces to conduct profiled attacks to compare the profiling performance with the baseline and then using $N_{ori}/4$ or $3N_{ori}/4$ pro-

filing traces (and so on) until profiling performance similar to the baseline obtained using N_{min} profiling traces.

3. Step 3: Build four new sets of profiling traces by pooling the profiling traces of each S-box based on the predefined S-box pooling ratio (labelled as PR) of 16, 8, 4 and 2 with the determined N_{min}, in which the pooling ratio of 16 means pooling $N_{min}/16$ profiling traces of each S-box to build the new set of profiling traces. Meanwhile, construct another four new sets of profiling traces without pooling the profiling traces of each S-box, *i.e.*, directly taking N_{min}/PR profiling traces from the original measurement for each S-box.
4. Step 4: For each S-box pooling ratio PR, perform TA, TA_p, SA and DL profiled attacks on each S-box using the new built set of profiling traces with pooling and using the newly constructed set of profiling traces without pooling. Note that, with pooling, we only need profiling once. While without pooling, we need profiling 16 times because we have to execute profiling for each S-box one by one following the current best practice.

Eventually, compare the profiling performance with and without pooling to the baseline.

7.2.2 Knowledge of POIs assumption

There is an implicit assumption about the knowledge of the POIs to apply this pooling approach. That is, evaluators can figure out the rough timing interval of each S-box of one AES round calculation in the side-channel traces. Through SPA/CPA and SEMA/CEMA as mentioned in Stage 1 in Subsection 2.4.1, this is feasible for most of the evaluated security products in the security evaluation grey-box testing context. For instance, the evaluators can vary the input length and/or the key length, perform correlation analyses on the input and output data, make use of the design information such as the location of the AES co-processor in the glue logic area or temporarily switch off some countermeasures like jitters.

Concerning the POI selection for classical profiled attacks TA, TA_p and SA, we use the popular SOST [GLP06] for POI selection, since typically the masking countermeasure is implemented in the evaluated products. It does not require any secret information about the implementation, and it can be easily, efficiently computed. It is also commonly used by security evaluators for profiled attacks.

7.2.3 Metrics and Selection of parameters

In order to compare the profiling complexity of different tools, information-theoretic metrics like mutual information (MI) are natural candidates [SKS09]. Yet, it requires that all the investigated profiling methods give rise to probabilistic outcomes. In some cases, the tools we consider fairly output scores that do not directly embed such a probabilistic meaning. Therefore, we will preferably evaluate our strategy based on the guessing entropy (GE) metric [SMY09]. Note that both metrics have the same comparative value and are thus equally good for our purposes (but computing guessing entropy curves is generally more expensive than estimating information-theoretic metrics).

Regarding the metrics used to train the neural networks, we use the Negative Log-Likelihood (NLL) loss function [CDP17] for DL profiled attacks, because it is proved that minimizing the NLL loss is equivalent to maximizing the Perceived Information [RSV+11, BHM+19] and thus to minimize the online attack complexity, thanks to the recent work [MDP20].

For SA profiled attack, since we do not precisely know which model is followed by the target sensitive intermediate data, we choose a linear 9-element basis (the eight S-box output bits together with a constant), which is the standard choice for this distinguisher [SLP05].

7.3 Experimental results

To confirm the soundness of the proposed pooling approach in security evaluations, in this section, we apply the pooling approach to three different representatives of AES-128 implementations, from easy to hard. That is, we consider a sequential S-box software AES-128 without masking [OC14], a sequential S-box software AES-128 with first-order masking [PSB+18] and a parallel S-box hardware AES-128 with first-order masking. The first DUT (Device Under Test) in Subsection 7.3.2 represents a sort of ideal case: all the S-boxes are supposed to leak in almost the same way, because all the S-boxes are executed sequentially, and no countermeasures are involved. The aim is to verify how the proposed S-box pooling approach behaves in an almost ideal leakage context. In Subsection 7.3.3, the second DUT is going further: to investigate how efficient the proposed S-box pooling approach will be in a more realistic scenario. In this case, because all the S-boxes are still executed sequentially, there should be no interference regarding leakage characteristics between them. Finally, the

third DUT in Subsection 7.3.4 is the scenario closest to the security evaluations. Because it is a hardware AES co-processor implementation with random masking, which is just the case for most modern evaluated products. In this case, multiple S-boxes are executed in parallel, so the leakage characteristic of each S-box will be affected by the others being executed at the same time. The involved random masking of each S-box will also introduce more discrepancy in the leakage characteristic given a limited amount of profiling traces. It is therefore essential to know how much profiling efficiency the evaluators can gain in this realistic scenario. We label those three DUTs as DUT1, DUT2 and DUT3 in the rest of this chapter.

7.3.1 Common settings

For all the three DUTs, we compare the profiling performance with and without S-box pooling using four different state-of-the-art profiled attacks, namely, TA, TA_p, SA and DL profiled DPA attacks. For each DUT, the same device is used for the profiling and attacking phase. All these four profiled attacks are implemented in Python and PyTorch [PGM+19] version 1.7.1 with an NVIDIA GTX 1080Ti GPU. The following common settings are used for all the experiments in this chapter,

1. No matter whether the masking countermeasure is present or not, the target of all profiled attacks in this chapter is the first round S-box output of AES-128 encryption. That is, $Sbox(p[i] \oplus k[i])$, in which p denotes a 16-byte AES-128 input and k corresponds to a 16-byte AES-128 key, with i the index of the S-box. Note that we focus on the common scenario where the masked randomness is not given to evaluators. Our conclusions would apply identically in the more worst-case scenario where this randomness is known for profiling. But the concrete gains would be less significant (because we expect the profiling task to be significantly simplified in that case).
2. The used DL neural network model for DL attacks is a published MLP [Mag20]. The structure of this model is simple and shallow, while it showed pretty good performance for similar DUTs (similar to our DUT1 and DUT2) in that paper. It also showed great performance for all the three DUTs in our work. In addition, we use the *Adadelta* optimizer [Zei12] and adopt the adaptive learning rate policy *ReduceLROnPlateau* to gradually decrease the learning rate with a factor of 0.05 if the training stagnates. We use a batch size of 32 and

50 as the number of epochs for all the DL experiments. All the profiling traces and attack traces used for DL experiments are normalized using the *StandardScalar* function from the Scikit-learn [PVG+11] library by removing the mean and scaling to unit variance. For all the DL experiments, we isolate 15% profiling traces as a validation set. This is due to the fact that a validation data set is critical to DL performance as it provides a way to timely detect over-fitting [Guy97]. As aforementioned, all the DL experiments utilize the NLL loss to train the model. The details of the used DL model and the hyperparameters are described in Table 7.1.

3. As mentioned in Step 2 of Subsection 7.2.1, we need to determine the minimum number of required profiling traces N_{min} for each DUT before the performance comparison for different profiled attacks. To this end, we use DL attacks to determine the N_{min} for each DUT because only DL attacks can fully recover all the subkeys for all three DUTs as shown in Appendix 7.A, Appendix 7.B and Appendix 7.C.

Table 7.1: MLP model details

MLP
nb_epoch = 50 batch_size_training = 32
Dense(50, activation="relu", input_shape=(nb_samples,)) BatchNormalization() Dense(100, activation="relu") BatchNormalization() Dense(256, activation="softmax") compile(loss='categorical-crossentropy', optimizer='adadelta', metrics=['accuracy'])
learning_rate_policy = ReduceLROnPlateau(optimizer, 'min', factor=0.05, verbose=True)

7.3.2 Setting #1: an unmasked sequential AES-128 S-boxes implementation

DUT1 is the ChipWhisperer unmasked software AES-128 with 16 sequential S-boxes. Considering the sequential implementation of the S-box layer without masking, all the 16 S-boxes of this AES-128 implementation are supposed to leak in the same way (or with negligible discrepancy). The goal of using this DUT is to verify, in such an almost ideal context, whether the proposed pooling approach can use 16 times fewer profiling traces to

achieve similar profiling performance. In terms of decreasing the required number of profiling traces and the required profiling time, it is the optimum that we are seeking.

Implementation settings.

This software AES-128 implementation is based on an 8-bit ATXmega128D4-AU microcontroller. The 16 S-boxes are executed one by one. A set of 11,000 power consumption traces of 3,000 sample points each has been measured via the onboard ADC of the ChipWhisperer Lite board for the experiments. More specifically, 10,000 profiling traces have been measured with random key data and random input data, 1,000 attack traces have been measured with a fixed random key and random input data.

Based on the knowledge of POIs assumption discussed in Subsection 7.2.2, we first identified the time intervals of each S-box in the power traces utilizing SPA and CPA. In the end, we cut 50 sample points (*e.g.*, sample points 110∼160 for S-box 1, 206∼256 for S-box 2, ..., 1550∼1600 for S-box 16) for each S-box from the originally acquired power traces to build 16 new subsets traces, where each new subset corresponds to one S-box. These 16 subsets are as a whole denoted as nPR_1.

Following our designed methodology in Subsection 7.2.1, we first performed TA, TA_p, SA and DL attacks on these 16 new sets of traces as the baseline of attack performance. This baseline is marked nPR_1 in the subsequent GE results. Second, the minimum required number of profiling traces N_{min} was determined using a binary search algorithm. In this case, N_{min} is 6,700 based on the DL attack results as explained in Subsection 7.3.1.

Next, following the proposed pooling approach, we built four new sets of profiling traces according to the PR of 16, 8, 4 and 2. And they are denoted as PR_16, PR_8, PR_4 and PR_2, respectively. Also, another four sets of profiling traces without pooling were constructed by directly taking N_{min}/PR profiling traces for each S-box from the previously built sets nPR_1. And they are denoted as nPR_16, nPR_8, nPR_4 and nPR_2, respectively. Similar to nPR_1, each of nPR_16, nPR_8, nPR_4 and nPR_2 consists of 16 subsets, and each subset corresponds to one S-box. We then conducted TA, TA_p, SA and DL attacks on PR_16, PR_8, PR_4, PR_2, nPR_16, nPR_8, nPR_4 and nPR_2.

For the sets with pooling the profiling traces PR_16, PR_8, PR_4 and PR_2, we only perform profiling once followed by attacking all 16 S-boxes

to retrieve the subkeys.

On the contrary, for the sets without pooling the profiling traces nPR_16, nPR_8, nPR_4 and nPR_2, we need to perform profiling 16 times. Each profiling is for one S-box followed by attacking that single S-box.

Finally, we compare all the conducted profiled attacks results (with and without pooling profiling traces) with the baseline results.

Attack results.

As suggested by comparing the aforementioned profiled attacks results with the baseline GE results shown in Figure 7.2, S-box pooling can decrease both the number of profiling traces and profiling time by a factor of 16 considering the DL attack results. For readability, here we only present the results for three different sets, *i.e.*, the baseline GE results (nPR_1, 6,700 profiling traces), the GE results with pooling with PR of 16 (PR_16, 6,700 = 16×418 profiling traces) and the GE results without pooling with PR of 16 (nPR_16, 418 profiling traces). The complete comparison is in Appendix 7.A.

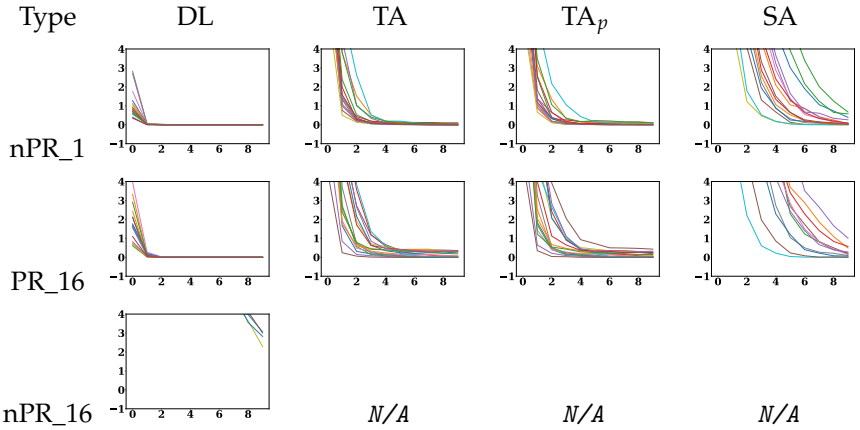


Figure 7.2: DUT1 Guessing Entropy results, X-axis: Number of attack traces, Y-axis: Guessing Entropy.

Considering the GE results of different profiled attacks on this DUT in the same setting (the same PR, with pooling or not), it is worth noting that DL shows the best attack performance. It suggests that the DL attack has a better generalization for this DUT. TA and its variant TA$_p$ show similar

but slightly worse attack performance while SA offers the worst attack performance for this DUT, which is in line with the observations in [GLP06][2]. For the nPR_16 setting, the GE results of TA, TA_p and SA attacks are not available due to very few profiling traces for each class of S-box output, which leads to singular results during the profiling.

More noticeably, this trend is the same when considering the impact of pooling profiling traces. As can be observed, the DL attack performance of the PR_16 case using pooling is already comparable to the baseline DL attack performance. While the DL attack performance of the nPR_16 case without pooling is way worse than the baseline DL attack performance. Considering the DL attack performance, the PR_8 case using pooling already outperforms the baseline one. Hence, these GE results show that, using pooling can decrease the required number of profiling traces by a factor of 16 without reducing the attack performance. That is the optimal result one can achieve, and it most likely holds because all 16 S-boxes are almost leaking in the same way. From a time complexity point of view, using a pooling ratio of 16 means using the same total amount of profiling traces as the baseline case, while only profiling once in the pooling case but profiling 16 times in the baseline case. It results in decreasing the required profiling time by a factor of 16 as well.

As mentioned before, this is an almost ideal case: no masking and sequential S-boxes. Next, we will further investigate the impact of the masking countermeasure on the performance of the proposed pooling approach.

7.3.3 Setting #2: a masked sequential AES-128 S-boxes implementation

DUT2 is the public data set ASCADv1 fixed-key, which contains the traces of a first-order masked software AES-128 with 14 sequential S-boxes. Because the first two S-boxes are not masked, only the last 14 masked S-boxes are evaluated in our experiments. It is still a sequential S-boxes implementation. So it is expected that those S-boxes will leak very similarly.

Implementation settings.

The ASCADv1 data set is based on an implementation using an 8-bit ATMega8515 microcontroller with a first-order masking countermeasure. 60,0-

[2]It is out of scope because the goal of this chapter is to verify the efficacy of the proposed pooling approach in terms of decreasing both the required number of profiling traces and the required profiling time.

00 EM (Electromagnetic Radiation) traces with a fixed AES-128 key were acquired using an EM coil at a sampling rate of 2 GHz, in which 50,000 are the profiling traces, and the remaining 10,000 traces are the attack traces. Each trace consists of 100,000 sample points.

Following the knowledge of POIs assumption, we narrowed down the time intervals of each S-box in the EM traces. In the end, 700 sample points (*e.g.*, sample points 45400∼46100 for S-box 3, 32910∼33610 for S-box 4, ..., 18330∼19030 for S-box 16) for each S-box were cut out from the acquired EM traces to build 14 new subsets traces. These 14 subsets are as a whole denoted as nPR_1.

TA, TA_p, SA and DL attacks were then conducted on nPR_1 as the baseline of attack performance. The minimum required number of profiling traces N_{min} of 37,000 was specified based on the DL attack results.

We further constructed four new sets of profiling traces according to the PR of 14, 7, 4 and 2. And they are denoted as PR_14, PR_7, PR_4 and PR_2, respectively. In addition, their counterparts without S-box pooling, *i.e.*, nPR_14, nPR_7, nPR_4 and nPR_2, were also constructed. TA, TA_p, SA and DL attacks were then performed on these eight sets of traces, and the results are compared with the baseline.

Attack results.

Figure 7.3 displays our results. For the same reason as in the previous case, only the results of nPR_1 (37,000 profiling traces), PR_14 (37,000 = 14×2,642 profiling traces) and nPR_14 (2,642 profiling traces) are shown here, Appendix 7.B provides the full results. Again, the DL attack results suggest that using pooling can decrease both the number of profiling traces and profiling time by a factor of 14.

Regarding the different performances of different profiles distinguishers, our observations are similar to DUT1. Namely, the DL attack shows slightly better results than TA and its variant TAp, while SA delivers the worst performance.

Also, the same trend is observed concerning the impact of pooling profiling traces: PR_14 already gives slightly better results than the baseline nPR_1 considering the DL attacks. The results without using pooling are significantly worse than the baseline ones. In short, from both the data complexity and time complexity viewpoints, using pooling can decrease the required number of profiling traces and the required profiling time by a factor of 14, which is the optimal result. Most likely, all those 14 masked

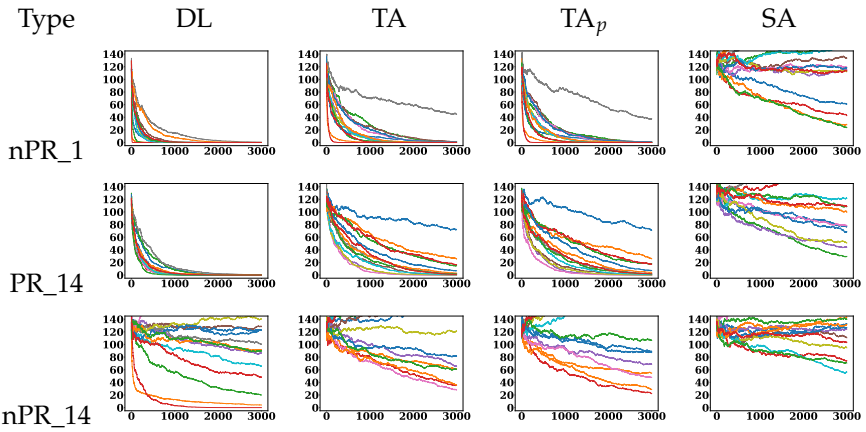


Figure 7.3: DUT2 Guessing Entropy results, X-axis: Number of attack traces, Y-axis: Guessing Entropy.

boxes leak in a very similar way.

In this sequential masked S-boxes case, using pooling comparably leads to optimal attack performance. It further confirms that S-box pooling can be used for protected implementations in security evaluations. To further assess the impact of other implementation factors on the performance of the pooling approach, a very realistic scenario is finally brought into our scope. Namely, many modern secure microcontrollers have a dedicated AES co-processor, in which random masking is adopted, and parallel S-boxes are sometimes implemented for performance purposes. Such an implementation will be assessed in our next target implementation.

7.3.4 Setting #3: a masked parallel AES-128 S-boxes implementation

DUT3 is a 90 nm secure microcontroller with an AES co-processor, which is equipped with first-order masked parallel S-boxes, *i.e.*, 4 S-boxes are executed at the same time. It is supposed that the leakage characteristic of each S-box within a group will be affected by the other 3 S-boxes in the same group executed in parallel. Furthermore, this DUT also has other built-in countermeasures such as hardware and software time jitters, power balancing and power smoothing.

Implementation settings.

A set of 520,000 EM traces of 100,000 sample points each has been measured via an EM coil using a LeCroy Waverunner 620Zi oscilloscope at a sampling rate of 10 GHz. 480,000 profiling traces have been measured with random key data and random input data. 40,000 attack traces have been measured with a fixed random key and random input data.

Under the knowledge of POIs assumption, the time intervals of each group of 4 parallel S-boxes were figured out in the EM traces using SPA/C-PA and SEMA techniques with the known AES co-processor location. For instance, the AES encryption was executed with different key lengths and input lengths to observe the difference they caused in the power and EM traces. Different numbers of AES rounds could be distinguished in the EM traces, and each AES round contains four groups of EM peaks, with each group corresponding to the execution of 4 parallel S-boxes. It has to be mentioned that, multiple alignments steps had to be done because the acquired EM traces were heavily misaligned. We used the same re-synchronization method as in Section 2.4.2, which exploits correlation to align each group of EM peaks in the traces.

During the measurement campaign, it was not possible to trigger the oscilloscope close to the first AES-128 round. Therefore, the EM traces were aligned several times to get close to the first AES-128 round step by step, followed by more local alignments within the first AES-128 round because there are 4 groups of EM peaks to be aligned one by one. Targeting different time intervals (for both the searching interval and reference interval), we applied this alignment method multiple times to the EM traces. In general, we chose new intervals when the misalignment was getting larger, and we repeated this process until the target interval was well aligned. In this way, we can gradually align each group of EM peaks corresponding to 4 parallel S-boxes executed.

After all the alignments, 500 sample points for each group of 4 parallel S-boxes were kept from the original EM traces to build 16 new subsets traces, and they are as a whole marked as nPR_1.

TA, TA_p, SA and DL attacks were then conducted on nPR_1 as the baseline of attack performance. The minimum required number of profiling traces N_{min} of 410,000 was further determined based on the DL attack results.

Similar to previous DUTs, we prepared eight new sets of profiling traces with or without pooling the profiling traces, *i.e.*, PR_16, PR_8, PR_4 and

PR_2, and nPR_16, nPR_8, nPR_4 and nPR_2. We performed TA, TA$_p$, SA and DL attacks on these eight sets of traces and compared the results with the baseline.

Attack results.

The comparison of results is shown in Figure 7.4. Once again, we only present the results of nPR_1 (410,000 profiling traces), PR_16 (410,000 = 16×25,625 profiling traces), PR_8 (820,000 = 16×51,250 profiling traces) and nPR_8 (51,250 profiling traces). The full results are in Appendix 7.C. This time the DL attack results demonstrate that using pooling can decrease both the number of profiling traces and profiling time by a factor of at least 8.

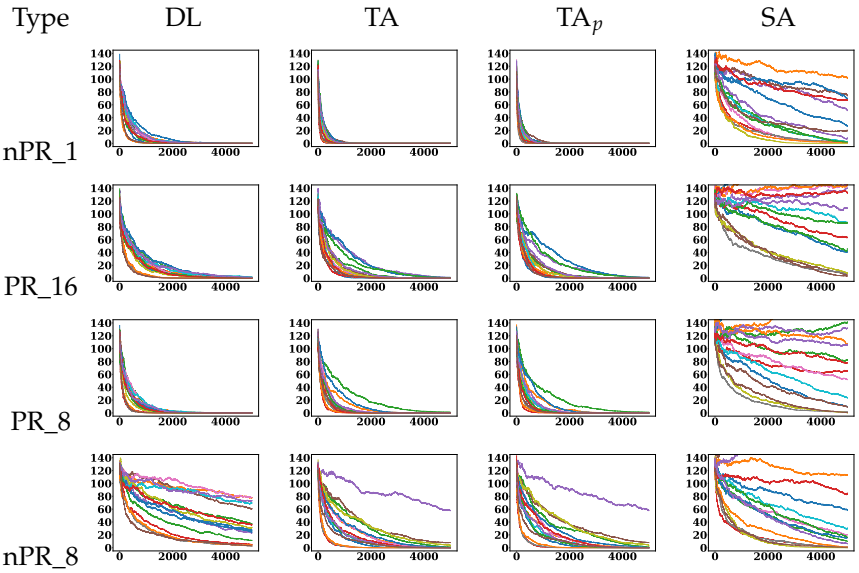


Figure 7.4: DUT3 Guessing Entropy results, X-axis: Number of attack traces, Y-axis: Guessing Entropy.

Different from the previous two DUTs, the TA and its variant TA$_p$ show better results than DL for all the 5 sets of profiling traces without pooling, *i.e.*, nPR_16, nPR_8, nPR_4, nPR_2 and the baseline nPR_1. The other way around is observed if the proposed S-box pooling approach is adopted: DL slightly outperforms TA and TA$_p$. Focusing on the pooling cases and the

DL results, using PR of 16 achieves marginally worse results than the baseline. While using PR of 8 leads to better results than the baseline. These observations put forward that even when evaluating a more challenging (and therefore more practically relevant) hardware implementation with masking, the S-box pooling approach remains effective. It does not lead to an optimal factor gain of 16 for the data and time complexity but still reduces these complexities to a significant factor 8. This sub-optimal gain is most likely caused by the parallel execution of 4 S-boxes. The leakage model of a single S-box is interfered by other 3 S-boxes among the same group.

To summarize, taking all the results for all the 3 DUTs as shown in Table 7.2 into account, it is concluded that, for unmasked and masked sequential S-boxes implementations, the gain of using pooling can reach the optimal. That is, decreasing both the required number of profiling traces and required profiling time by a factor of 16 (or 14). For the masked parallel S-boxes implementation we analyzed, the gain is slightly sub-optimal but still very noticeable for security evaluators. That is, decreasing both the required number of profiling traces and required profiling time by a factor of 8. It confirms that S-box pooling can be a beneficial new tool for security evaluators.

Table 7.2: Summary of the experiments of 3 DUTs

	PR	N_{ori}	N_{min}	Samples	Implementation	Data Complexity Gain	Time Complexity Gain
DUT1	16/8/4/2	10,000	6,700	50	SW Sequential	16 (*optimal*)	16 (*optimal*)
DUT2	14/7/4/2	50,000	37,000	700	Masked SW Sequential	14 (*optimal*)	14 (*optimal*)
DUT3	16/8/4/2	480,000	410,000	500	Masked HW Parallel	$\geqslant$8 (*sub-optimal*)	$\geqslant$8 (*sub-optimal*)

7.4 Conclusion

During the security evaluations of symmetric algorithm implementations, usually, evaluators have to repeat the profiling of each S-box to attack them separately. The burden is getting heavier when both classical profiled attacks (such as template attacks) and deep learning attacks are mandated by the certification bodies. Additionally, the requested number of profiling traces is also increasing for some certification bodies due to many countermeasures being implemented in the evaluated products. Whether there exists a way to perform such profiling efficiently is, therefore, an important and practically motivated research question. To this end, the community

so far mostly focused its efforts on improving the attack performance targeting one S-box. In this chapter, we analyze the complementary approach of trying to decrease the profiling (data and time) complexity thanks to S-box pooling.

Intensive experiments on three different AES-128 implementations were performed with 108 different configurations. The results demonstrate that, for representative unmasked software sequential S-boxes implementation and the masked software sequential S-boxes implementation, using this S-box pooling approach can decrease both the required number of profiling traces and required profiling time by a factor of 16 (or 14), which corresponds to the optimal gain. For the masked hardware parallel S-boxes implementation that we analyzed, the gain is still sub-optimal in terms of data complexity and time complexity. Nevertheless, we can decrease both the required number of profiling traces and required profiling time at least by a factor of 8 in this practically relevant setting. We used only the first round S-box computations for simplicity, indeed this S-box pooling can be extended to all S-box computations of all AES rounds. The goal of this chapter is to show that in some practically-relevant contexts, S-box pooling can lead to significant gains but there are admittedly implementations for which it may not be applicable, *e.g.*, if the S-boxes leak differently.

We believe these results show that the efficient exploitation of profiling measurements, for example, thanks to S-box pooling, can be a useful addition to the evaluators' toolbox. On the one hand, it is easy to adopt and integrate into existing toolchains. On the other hand, there are contexts in which this simple optimization can lead to significant gains (remembering that concretely, a factor 16 is highly relevant for evaluation tasks that can take days).

Appendix 7.A Full attack results on DUT1

In this section we present the full results on DUT1 without and with pooling.

Appendix 7.B Full attack results on DUT2

In this section we present the full results on DUT2 without and with pooling.

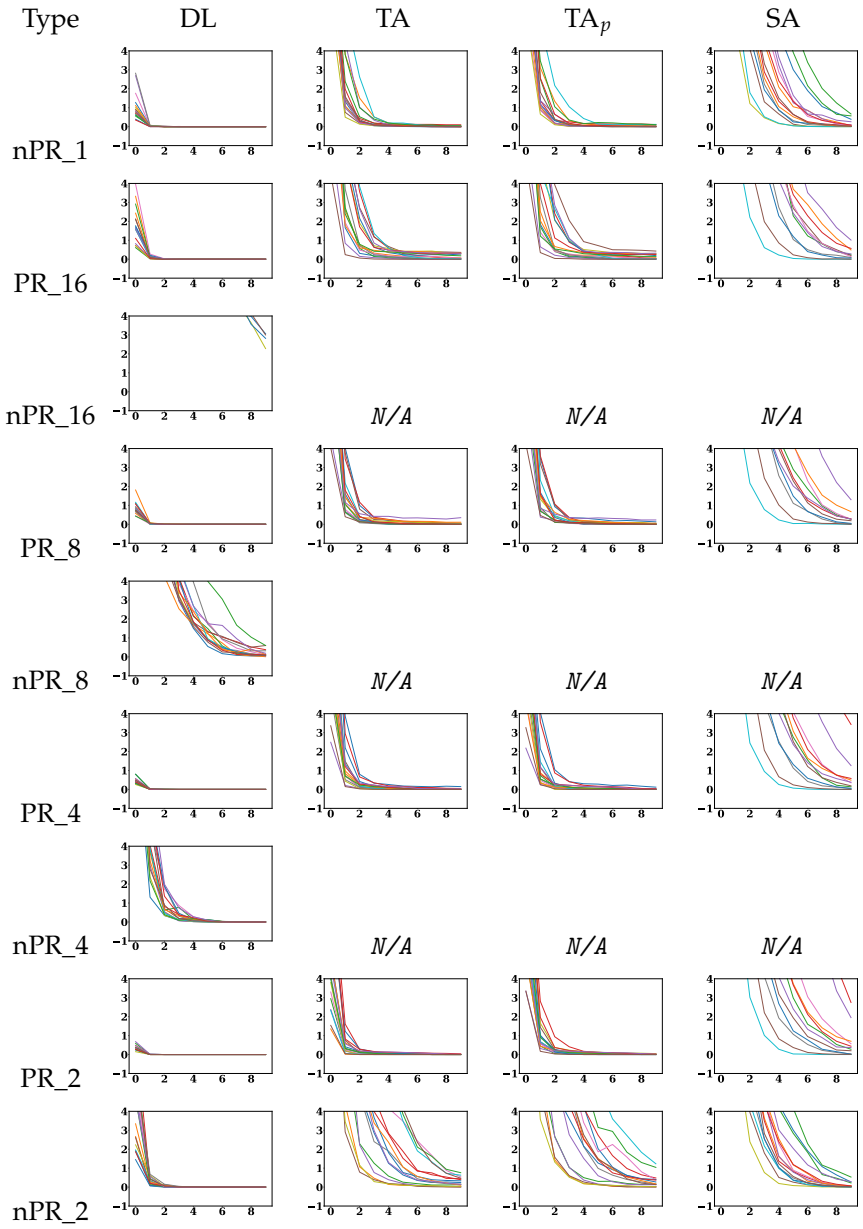


Figure 7.5: Full DUT1 Guessing Entropy results.

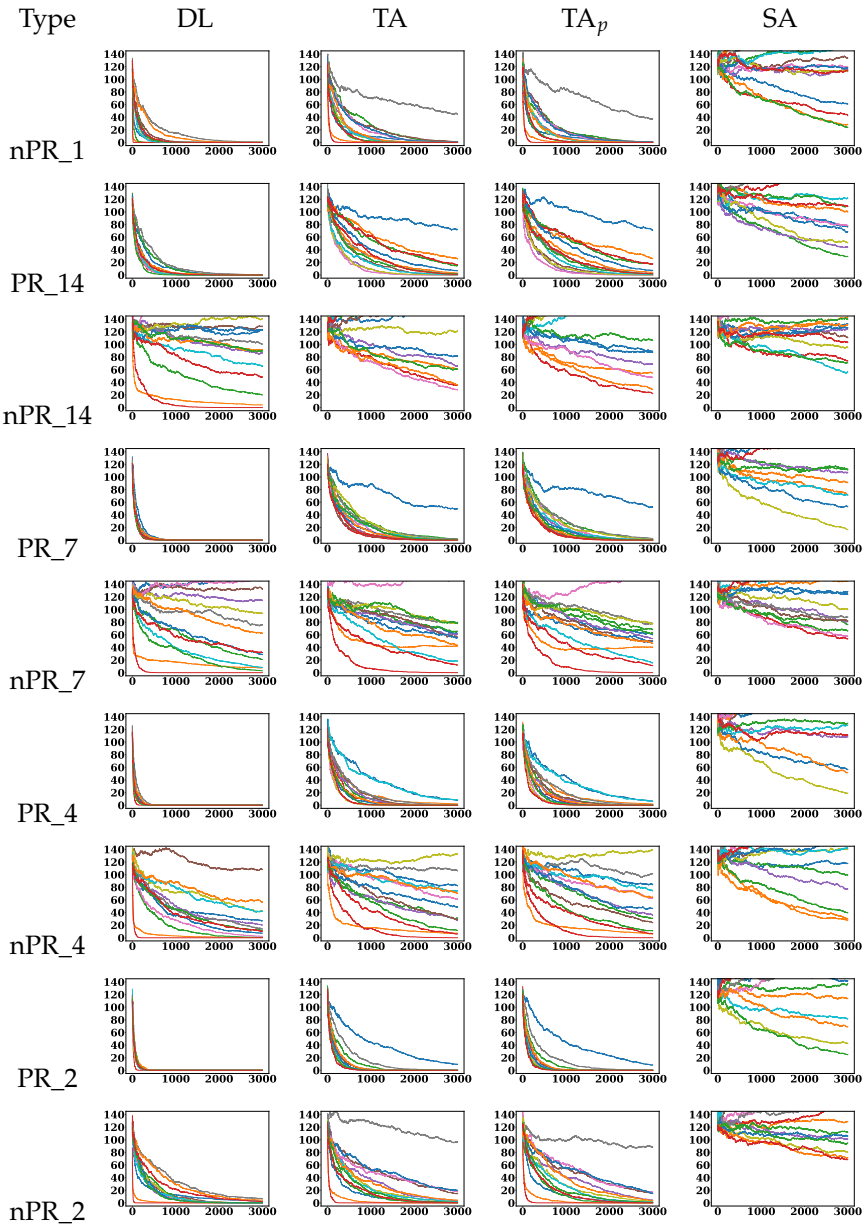


Figure 7.6: Full DUT2 Guessing Entropy results.

Appendix 7.C Full attack results on DUT3

In this section we present the full results on DUT3 without and with pooling.

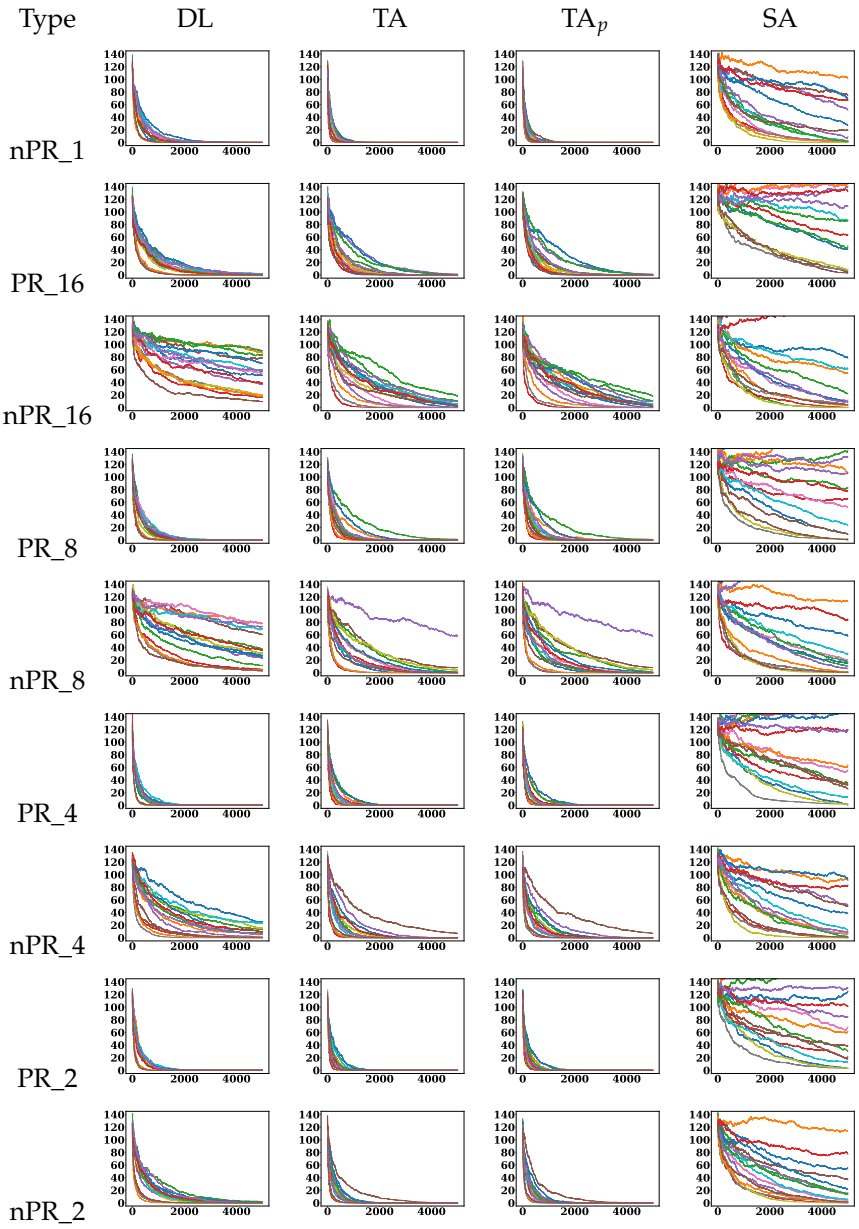


Figure 7.7: Full DUT3 Guessing Entropy results.

Part III

Partial Key Exposure

Chapter 8

Extended Partial Key Exposure Attack on CRT-RSA with Additive Exponent Blinding

At Eurocrypt 2022, May *et al.* proposed a PKE attack on CRT-RSA that efficiently factors N knowing only a $\frac{1}{3}$-fraction of either most significant bits (MSBs) or least significant bits (LSBs) of private exponents d_p and d_q for public exponent $e \approx N^{\frac{1}{12}}$. In practice, PKE attacks typically rely on the side-channel leakage of these exponents, while a side-channel resistant implementation of CRT-RSA often uses additively blinded exponents $d'_p = d_p + r_p(p-1)$ and $d'_q = d_q + r_q(q-1)$ with unknown random blinding factors r_p and r_q, which makes PKE attacks more challenging.

Motivated by the above, we extend the PKE attack of May *et al.* to CRT-RSA with additive exponent blinding. While admitting $r_p e \in (0, N^{\frac{1}{4}})$, our extended PKE works ideally when $r_p e \approx N^{\frac{1}{12}}$, in which case the entire private key can be recovered using only $\frac{1}{3}$ known MSBs or LSBs of the blinded CRT exponents d'_p and d'_q. Our extended PKE follows their novel two-step approach to first compute the key-dependent constant k' ($ed'_p = 1 + k'(p-1)$, $ed'_q = 1 + l'(q-1)$), and then to factor N by computing the root of a univariate polynomial modulo $k'p$. We extend their approach as

follows. For the MSB case, we propose two options for the first step of the attack, either by obtaining a single estimate $k'l'$ and calculating k' via factoring, or by obtaining multiple estimates $k'l'_1, \ldots, k'l'_z$ and calculating k' probabilistically via GCD.

For the LSB case, we extend their approach by constructing a different univariate polynomial in the second step of the LSB attack. A formal analysis shows that our LSB attack runs in polynomial time under the standard Coppersmith-type assumption, while our MSB attack either runs in sub-exponential time with a reduced input size (the problem is reduced to factor a number of size $e^2 r_p r_q \approx N^{\frac{1}{6}}$) or in probabilistic polynomial time under a novel heuristic assumption. Under the settings of the most common key sizes (1024-bit, 2048-bit, and 3072-bit) and blinding factor lengths (32-bit, 64-bit, and 128-bit), our experiments verify the validity of the Coppersmith-type assumption and our own assumption, as well as the feasibility of the factoring step.

To the best of our knowledge, this is the first PKE on CRT-RSA with experimentally verified effectiveness against 128-bit unknown exponent blinding factors. We also demonstrate an application of the proposed PKE attack using real partial side-channel key leakage targeting a Montgomery Ladder exponentiation CRT implementation.

As one of the longest-serving and the most widely used public-key cryptosystems, RSA can use the Chinese Remainder Theorem (CRT) optimization to speed up exponentiation operations, which is especially favored by power-constrained embedded systems. To mitigate side-channel attacks [Koc96], a real implementation of (CRT-)RSA often adopts blinding countermeasures such as message blinding, modulus blinding and exponent blinding [Koc96, Cor99, CFG+10, DSG21, MOP07], of which additive exponent blinding [Koc96] was referred to by the BSI as the "classical exponent blinding" [BSI13] and is widely deployed, *e.g.*, in open source cryptographic libraries MbedTLS [Mbe21], Libgcrypt [Lib21], and Botan [Bot22]. Throughout this chapter, we consider CRT-RSA with additive exponent blinding as the main target of our research.

At Eurocrypt'96, Coppersmith presented a novel lattice-based method to find small solutions of univariate modular polynomials with some applications to cryptanalysis of RSA [Cop96b], and he [Cop96a] extended this method to bivariate equations to factor RSA modulus N with half MSBs of one of its prime factors. Boneh *et al.* [BDF98] introduced PKE attacks to recover the full RSA private key using a few consecutive MSBs or LSBs

of the private key based on the Coppersmith method. Many subsequent works continued the research on PKE, but most do not take into account the widely deployed exponent blinding countermeasure [BM03,EJMdW05, Aon09,HS09,SV14,TK14,LZL14,TK15,TK19,TLP19,MNS21,MNS22], with only a couple of exceptions to the best of our knowledge [FKM+06,JL12]. Fouque *et al.* [FKM+06] investigated PKE attacks on an additively blinded private exponent with up to 32-bit blinding factors and very small e values for RSA without CRT. They showed that non-consecutive known bits — which could realistically result from SCA leakage of sliding-window exponentiation implementations — of the additively blinded private exponent could be used to recover the private exponent. However, their attack relies on the fact that both the public exponent e and the blinding factor are small to enable brute force, and requires multiple traces to gradually retrieve the entire private key. In particular, the retrieval requires several (50 in their experiments) instantiations of the recovered non-consecutive exponent bits (when $e = 3$) or 2-bit consecutive exponent bits (when $e = 2^{16} + 1$), each providing a $\frac{1}{64}$ to $\frac{1}{16}$ (depending on the public exponent, key length) non-consecutive or consecutive portion of the blinded exponents. Further, it works with very small e values (the cases of $e = 3$ and $2^{16} + 1$ were experimentally verified) and up to 32-bit blinding factors. Joye and Lepoint [JL12] also focused on PKE on RSA with additive exponent blinding. They constructed trivariate or bivariate polynomials based on Coppersmith's lattice techniques to recover the whole key with partial known blinded exponent MSBs/LSBs. Their attack requires in the best case more than half of the known MSBs or LSBs of the blinded private exponent, which was experimentally verified. In addition to the aforementioned works that assume some **partial** bits of the (blinded) private exponent(s) known with certainty, another line of works considers the case where **all** bits of the blinded private exponent(s) are classified, but where they can be wrongly classified with some error probability, *e.g.* [SI11,SW14,SW17,Bau12]. In summary, previous PKE attacks on (CRT-)RSA with additive exponent blinding mostly consider restricted cases of e and small blinding factors using special-structured partial key leakages from multiple observations, or need more than half of blinded private exponent from a single observation of the partial key leakage, or need a full recovery of the (blinded) exponent bits with errors.

In this chapter, we aim to answer the question: *how and to which extent can we extend the PKE attack to CRT-RSA with additive exponent blinding (ideally using as few as possible consecutive MSBs/LSBs of the private CRT ex-*

ponent from a single observation of the partial key leakage) both theoretically and empirically?

Several preliminaries and proofs in this chapter closely follow the corresponding versions in [MNS22]. In those cases, the contributions of May *et al.* are included here in full such that this chapter is self-contained and easier to read.

The rest of this chapter is organized as follows. Section 8.1 introduces the necessary background information about the Coppersmith method and the state-of-the-art PKE attack on unblinded CRT exponents published at Eurocrypt'22. Section 8.2 provides a mathematical proof of the extended PKE attack in the additive blinding scenario. Afterwards, we experimentally verify the effectiveness and time complexity of this approach in Section 8.3. Finally, in Section 8.4, an application of the extended PKE attack is demonstrated using partial side-channel key leakage of a typical real-world Montgomery Ladder exponentiation implementation with additive exponent blinding countermeasure.

8.1 Preliminaries

8.1.1 Notations

By 'log' we denote the base 2 logarithm, 'ln' denotes the natural logarithm, and $\zeta(.)$ denotes Riemann's zeta function. We use capital letters for random variables and small caps for their realizations. We use capital bold font letters for matrices (*e.g.*, $\mathbf{M}$) and small bold caps for vectors (*e.g.*, $\mathbf{v}$). We use $|\det \mathbf{M}|$ to denote the absolute value of the *determinant* of the matrix $\mathbf{M}$, which corresponds to the determinant of the lattice spanned by this matrix.

8.1.2 Coppersmith's method

Similar to many previous PKE works, we rely on Coppersmith's method to find small modular roots of multivariate polynomials [Cop97]. We first give a simple introduction to the multivariate Coppersmith's lattice-based method.

Let $f \in \mathbb{Z}[x_1, ..., x_j]$ be a j-variate polynomial over the integers with maximum degree δ in each variable separately, this polynomial has a small root $r = (r_1, ..., r_j)$ modular an integer M and let $U_i \in \mathbb{Z}$ denote some

known bounds. The goal is to find integers $|r_i| \leq U_i$ such that $f(r_1, ..., r_j) = 0$ in polynomial time.

To this end, a series of so-called *shift-polynomials* with a chosen sufficiently big positive integer $m \in \mathbb{N}$ and the indices $i_0, ..., i_j \in \mathbb{N}$ is constructed as below:

$$s_{[i_0,...,i_j]}(x_1, ..., x_j) = f^{i_0}(x_1, ..., x_j) \cdot x_1^{i_1} \cdot ... \cdot x_n^{i_j} \cdot M^{m-i_0}.$$

Those polynomials have the root r modulo M^m by construction. A subset of the constructed shift-polynomials $s_{[i_0,...,i_j]}(U_1x_1, ..., U_jx_j)$ is selected to generate an l-dimensional lattice $\mathcal{L}$ with their coefficient vectors such that the lattice $\mathcal{L}$ has a triangular basis matrix $\mathbf{B}$. Given a large enough chosen integer m and the determinant of $\mathcal{L}$ fulfilling the *enabling condition*, *i.e.*, $|\det \mathbf{B}| \leq M^{ml}$, then a collection of j polynomials $p_1(x_1, ..., x_j), ..., p_j(x_1, ..., x_j)$ can be computed in polynomial time. This is due to the fact that the coefficient vectors of the polynomials $p_j(U_1x_1, ..., U_jx_j)$ are elements of $\mathcal{L}$, which is generated by the coefficient vectors of the polynomials $s_i(U_1x_1, ..., U_jx_j)$. Therefore, the computation of $p_1(x_1, ..., x_j), ..., p_j(x_1, ..., x_j)$ can be done with the widely used LLL lattice basis reduction algorithm [LLL82]. Those computed polynomials have the root r not only modulo M^m but also over the integers as proved by Howgrave-Graham in [How97] as stated below.

Lemma 8.1 (Howgrave-Graham [How97])**.** *Let $h(x_1, ..., x_k) \in \mathbb{Z}[x_1, ..., x_k]$ be a polynomial in at most ω monomials. Suppose that $h(r_1, ..., r_k) \equiv 0 \mod M^m$ for some positive integer m. Also let $|r_i| < X_i$ for $1 \leq i \leq k$ and*

$$\|h(x_1X_1, ..., x_kX_k)\| < \frac{M^m}{\sqrt{\omega}}.$$

Then, $h(r_1, ..., r_k) = 0$ holds over the integers.

In the case of $j = 1$, *i.e.*, the polynomial f is univariate, the root r can be straightforwardly resolved from p_1 using standard techniques, *e.g.*, Newton's method. In the case of $j > 1$, if such (computed) polynomials generate an ideal $(p_1, ..., p_j)$ of zero-dimensional variety, we can use a Gröbner basis to resolve their root, it means that the RSA modulus N can be efficiently factored in our context. However, the existence of such a zero-dimensional variety relies on the standard Coppersmith-type heuristic assumption (see also [MNS22, Assumption 1]) as below:

Assumption 1. In the multivariate setting, Coppersmith's method yields polynomials that generate an ideal of zero-dimensional variety.

It is worth mentioning that it is essential to experimentally verify the validity of this assumption because it might fail in some cases, *e.g.*, in the small e regime of the TK attack [TK15] as pointed out in [MNS22].

8.1.3 PKE attack on unblinded CRT exponents

We first briefly recall the PKE attack on unblinded CRT private exponents in [MNS22]. The definition of PKE attacks on RSA is, that the RSA private key can be fully recovered in polynomial time given only a constant fraction of the secret exponent(s). This recent work put forward the state-of-the-art PKE attack on CRT exponents in small e regime from $\frac{1}{2}$ known LSBs of d_p (or d_q, see [BM03, Fig. 4]) to $\frac{1}{3}$ known LSBs (or MSBs) of both d_p and d_q (see [MNS22, Fig. 1]) when $e \approx N^{\frac{1}{12}}$. It is a novel two-step (both steps can finish in polynomial time) approach as below:

- **Step 1:** Compute CRT key constants k ($ed_p = 1 + k(p-1)$) and l ($ed_q = 1 + l(q-1)$) based on the known parts of d_p and d_q and the public key (N, e). If the MSBs of d_p and d_q are known, this step is trivial to solve a quadratic polynomial equation. In the LSB case (the LSBs are known), Coppersmith's lattice-based method [Cop97] is involved so this step relies on a standard Coppersmith-type heuristic assumption as mentioned above in 1. They experimentally verified this heuristic in addition to the mathematical proof.

- **Step 2:** Recover the unknown bits of d_p and subsequently the factor p of N using the previously calculated k. The authors considered it as an extension of Howgrave-Graham's *approximate divisor* algorithm [How01] to the case of *approximate divisor multiples* for some known multiple k of an unknown divisor p of N. They mathematically proved that this unknown divisor can be recovered in polynomial time.

8.2 Extended PKE attack on additively blinded CRT exponents

As discussed above, the problem to be solved in this chapter is: to which extent can one reveal the whole private key when only part of the additively blinded CRT exponent bits is disclosed. In the following, we de-

scribe the extended PKE attack on additively blinded CRT exponents based on [MNS22].

May *et al.* only considered the PKE attack on unblinded CRT private exponents in [MNS22], we extend their work to cover the additively blinded CRT private exponents, and it is called EPKE (Extended Partial Key Exposure) attack in the rest of this chapter.

Using the additive blinding factors r_p and r_q, the blinded CRT exponents are $d'_p = d_p + r_p(p-1)$ and $d'_q = d_q + r_q(q-1)$, where $r_p, r_q = N^\gamma$. Let $(d'^{(M)}_p, d'^{(M)}_q)$ be the MSBs of d'_p and d'_q, and $(d'^{(L)}_p, d'^{(L)}_q)$ be the LSBs. So, we have:

$$d'_p = d'^{(M)}_p 2^i + d'^{(L)}_p,$$
$$d'_q = d'^{(M)}_q 2^i + d'^{(L)}_q.$$

Hereafter, we label it as the MSB case if $(d'^{(M)}_p, d'^{(M)}_q)$ are known (*e.g.*, by side-channel attacks), or as the LSB case if $(d'^{(L)}_p, d'^{(L)}_q)$ are known. In addition to the Coppersmith-type assumption required for the LSB case in [MNS22], our algorithm in the MSB case has two options with different requirements:

- Given only parts of two blinded exponents d'_p and d'_q, a factoring step of a number roughly $\frac{1}{6}$ the size of N is required, leading to a sub-exponential time complexity.

- Given parts of one blinded exponent d'_p and multiple blinded exponents $d'_{q,i}$, the time complexity remains polynomial, with a success probability according to the additional heuristic Assumption 2 defined below (which will be experimentally verified).

Assumption 2. For z distinct blinded exponents $d'_{q,i} = d_q + r_{q,i}(q-1)$, the corresponding values $l'_i = l + r_{q,i}e$ are coprime with probability $1/\zeta(z)$.

According to [DE04, (1.2) and (3.1)], this assumption holds if the values $l'_i = l + r_{q,i}e$ are distributed uniformly. However, the assumption that these values are distributed uniformly is too strong. In any case, Assumption 2 is sufficient and is experimentally validated as described in Section 8.3.

Our main result of EPKE regarding factoring N is the extension of [MNS22, **Theorem 1**] as follows:

Theorem 8.2. *Let (N, e) be the public key and N be large enough with $e = N^{\alpha}$. Given the MSBs $(d_p'^{(M)}, d_{q,1}'^{(M)}, \ldots, d_{q,z}'^{(M)})$ or the LSBs $(d_p'^{(L)}, d_q'^{(L)})$ of the additively blinded CRT exponents (d_p', d_q') with blinding factors $r_p, r_{q,i} = N^{\gamma}$. If the unknown parts of d_p' and $d_{q(,i)}'$ are upper bounded by N^{δ} and $\delta \geq \gamma$ (which already holds for commonly used up to 128-bit additive blinding factors), where*

$$\gamma \leq \delta < \min\{\frac{1}{4} + \alpha + \gamma, \frac{1}{2} - 2\alpha - \gamma\}$$

for the MSB case, or

$$\delta < \min\{\frac{1}{4} + \alpha + \gamma, \frac{1}{2} - 2\alpha - 2\gamma\}$$

for the LSB case, then N can be factored

- *in polynomial time under Coppersmith's heuristic assumption (see Assumption 1) for the LSB case,*
- *in sub-exponential time* $\exp(c(\ln(N^{2\alpha+2\gamma}))^t(\ln\ln(N^{2\alpha+2\gamma}))^{1-t})$ *for the MSB case ($z = 1$) with constants c and t dependent on the underlying integer factorization algorithm, or*
- *in probabilistic polynomial time under Assumption 2 for the MSB case ($z > 1$).*

Proof outline. The additively blinded CRT exponents d_p', d_q' satisfy the equations

$$ed_p' = 1 + k'(p-1), \tag{8.1}$$

$$ed_q' = 1 + l'(q-1), \tag{8.2}$$

where $k' = k + r_p e$ and $l' = l + r_q e$ as we have

$$ed_p = 1 + k(p-1),$$
$$ed_q = 1 + l(q-1).$$

Similar to [MNS22], we use a two-step method to factor N.

Step 1: Given the unknown parts of d_p' and d_q' being upper bounded by $N^{\frac{1}{2}-2\alpha-\gamma}$ (MSB case) or $N^{\frac{1}{2}-2\alpha-2\gamma}$ (LSB case),

- in Section 8.2.1 we prove that k' can be calculated in sub-exponential time $\exp(c(\ln(N^{2\alpha+2\gamma}))^t(\ln\ln(N^{2\alpha+2\gamma}))^{1-t})$ using a single known MSBs of d'_q in the MSB case,
- in Section 8.2.2 we prove that k' can be calculated in probabilistic polynomial time using multiple known MSBs of d'_q in the MSB case under Assumption 2, and
- in Section 8.2.3 we prove that k' can be calculated in polynomial time under the Coppersmith-type heuristic Assumption 1 for multivariate polynomials in the LSB case.

Step 2: With the computed k' and the unknown MSBs or LSBs of d'_p bounded by $N^{\frac{1}{4}+\alpha+\gamma}$, we provide an algorithm for factoring N in polynomial time based on the novel result of May *et al.* of *approximate divisor multiples* [MNS22, **Theorem 3**].

Note that the proof outline for Theorem 8.2 and the proofs of Lemmas 8.3, 8.5 and 8.7 follow the work of May *et al.* [MNS22] and only differ in handling those two random blinding factors r_p and r_q.

8.2.1 Step 1a: Computing (k', l') from MSB using factoring.

In the MSB case, to compute k' using known MSBs of two blinded exponents d'_p and d'_q, the basic idea is to compute the product $k'l'$ followed by a factorization of this product to find the candidates of k' using the modular sum of $k' + l' \mod e$. The time complexity here is determined by the factorization of the product $k'l'$, which is sub-exponential according to [BS21, **Formula 3.10**]. The time complexities of the state-of-the-art fastest known integer factorization algorithms are of the form $\exp(c(\ln n)^t(\ln\ln n)^{1-t}))$ for some constants $c > 0$ and $0 < t < 1$, in which n is the number to be factored. For QS (Quadratic Sieve) and ECM (Elliptic-Curve Method) factorization algorithms, $t = \frac{1}{2}$; for NFS (Number Field Sieve), $t = \frac{1}{3}$. Note that, in practice, this sub-exponential time complexity is affordable because of the small size of e and commonly used small sizes (very often 64 bits or even 32 bits, at most 128 bits) of exponent blinding factors r_p and r_q. We also experimentally verify this in Section 8.3. Most of the experiments take seconds or minutes for the factorization step and occasionally need up to one and a half hours (resp., fifty hours) for 2048-bit (resp., 3072-bit) key with a regular PC.

Lemma 8.3 ((k', l') from MSB). *Let (N, e) be the public key and N be large enough with $e = N^{\alpha}$. Given the MSBs $(d_p'^{(M)}, d_q'^{(M)})$, if the unknown parts $(d_p'^{(L)}, d_q'^{(L)})$ are upper bounded by N^{δ}, where*

$$\gamma \leq \delta < \frac{1}{2} - 2\alpha - \gamma,$$

then (k', l') can be computed in time $\exp(c(\ln(N^{2\alpha+2\gamma}))^t(\ln\ln(N^{2\alpha+2\gamma}))^{1-t})$.

Proof. According to Equations (8.1) and (8.2), we have

$$k'p = k' - 1 + ed_p',$$
$$l'q = l' - 1 + ed_q'.$$

Multiplying those two equations, we get

$$k'l'N = (k'-1)(l'-1) + ed_p'(l'-1) + ed_q'(k'-1) + e^2 d_p' d_q'. \tag{8.3}$$

Let

$$\widetilde{A} = \frac{2^{2i} e^2 d_p'^{(M)} d_q'^{(M)}}{N}.$$

$\widetilde{A}$ can be easily computed with the known $(d_p'^{(M)}, d_q'^{(M)})$. In the following we demonstrate that $\lceil \widetilde{A} \rceil = k'l' - o(1)$ with $\delta < \frac{1}{2} - 2\alpha - \gamma$. Therefore, the estimation of the product $k'l'$ equals $\lceil \widetilde{A} \rceil$ given a sufficiently large N.

From Equation (8.3) we have

$$\begin{aligned} k'l'N - \widetilde{A}N &= (k'-1)(l'-1) + ed_p'(l'-1) + ed_q'(k'-1) + e^2 d_p' d_q' - \widetilde{A}N \\ &= (k'-1)(l'-1) + ed_p'(l'-1) + ed_q'(k'-1) \\ &\quad + 2^i e^2 (d_p'^{(M)} d_q'^{(L)} + d_p'^{(L)} d_q'^{(M)}) + e^2 d_p'^{(L)} d_q'^{(L)}. \end{aligned}$$

Considering

$$d_p', d_q' = \Theta(N^{\frac{1}{2}+\gamma}), d_p'^{M}, d_q'^{M} = \Theta(N^{\frac{1}{2}+\gamma-\delta}),$$

$$d_p'^{(L)}, d_q'^{(L)}, 2^i = \Theta(N^{\delta}), k', l' = \Theta(N^{\alpha+\gamma}),$$

we have

$$k'l'N - \widetilde{A}N = \mathcal{O}(N^{2\alpha+2\gamma}) + \mathcal{O}(N^{2\alpha+\frac{1}{2}+2\gamma}) + \mathcal{O}(N^{2\alpha+\frac{1}{2}+\gamma+\delta}) + \mathcal{O}(N^{2\alpha+2\delta}) = \mathcal{O}(N^{2\alpha+\frac{1}{2}+\gamma+\delta}),$$

and further we obtain

$$k'l' - \widetilde{A} = \mathcal{O}(N^{\delta+2\alpha+\gamma-\frac{1}{2}}) = o(1).$$

In conclusion, the product $k'l'$ can be calculated in polynomial time $\mathcal{O}(\log^2 N)$.

Further we can deduce from Equation (8.3) that

$$k' + l' \equiv 1 - k'l'(N-1) \mod e, \tag{8.4}$$

where the right-hand side can be computed with the obtained product $k'l'$ and the public key (N, e).

To compute k', the product $k'l'$ is factored using the state-of-the-art integer factorization algorithms [BS21] to get all its factors in sub-exponential time $\exp(c(\ln(N^{2\alpha+2\gamma}))^t(\ln\ln(N^{2\alpha+2\gamma}))^{1-t})$. Then, the search for a combination of those factors that satisfies Equation 8.4 will reveal two values corresponding to k' and l'. Putting all together, the time complexity of computing k' in the MSB case is $\exp(c(\ln(N^{2\alpha+2\gamma}))^t(\ln\ln(N^{2\alpha+2\gamma}))^{1-t})$. □

As opposed to the proof in [MNS22], it is not trivial to recover k' and l' from one product $k'l'$ in polynomial time. Trying to use Equation (8.4) to compute a univariate polynomial with k' and l' as roots will fail because, unlike $0 < k + l < 2e$, there is no small interval that $k' + l'$ will fall into to derive $k' + l'$ from $k' + l' \mod e$, which is due to the blinding factors r_p and r_q. Since $k = k' \mod e$ and $l = l' \mod e$, it may seem feasible to find k and l instead, but the method from [MNS22] will not work as it is non-trivial to derive the required product kl from $kl \mod e = k'l' \mod e$. Constructing a multivariate polynomial as in the LSB case to recover k and l using Coppersmith's method is also non-trivial, as the modulus e is not large enough compared to roots k and l. Regardless, the factoring method is fast enough in practice shown by the experimental results in Section 8.3.

In the end, two candidates for k' are found because the modular sum cannot tell which of those two possible values is k' and which is l'. The fact that two candidates are found for k' means that Step 2 may have to be

repeated for both candidates. However, it is not described in [MNS22] how the two values k and l can be distinguished in the MSB case, while they are two equivalent roots of a univariate polynomial (as opposed to the LSB case, where they together form one root of a bivariate polynomial).

8.2.2 Step 1b: Computing $(k', l'_1, ..., l'_z)$ from MSB using GCD.

This section describes how to compute, in the MSB case, k' using known MSBs of one blinded exponent d'_p and multiple blinded exponents $d'_{q,i}$ in heuristic probabilistic polynomial time. Rather than using factorization methods to determine the factors of $k'l'$, it is possible to use the fact that l' is different in every execution of the CRT exponentiation due to the random blinding factor. An attacker can recover partial information on two different exponentiation executions with blinded d_q, *i.e.*, $d'_{q,1}$ and $d'_{q,2}$, compute the products $k'l'_1$ and $k'l'_2$, and recover k' by computing the GCD (Greatest Common Divisor). It may be that l'_1 and l'_2 are not coprime, in which case the GCD will comprise $k' \cdot f$ where $f = \gcd(l'_1, l'_2)$. This happens with some probability, and can be solved by capturing additional exponentiation executions and successively computing the GCD with more products that include k'. Alternatively, the additional factor f is likely small (experimentally verified as shown in Figures 8.5-8.7) in practice and can be recovered using a small brute-force with only two instantiations. The attacker can guess f, derive the corresponding k', l'_1, and l'_2, and verify whether both pairs satisfy Equation (8.4).

Lemma 8.4 ($(k', l'_1, \ldots, l'_z)$ from MSB). *Let (N, e) be the public key and N be large enough with $e = N^\alpha$. Given the MSBs $(d'^{(M)}_p, d'^{(M)}_{q,1}, \ldots, d'^{(M)}_{q,z})$, if the unknown parts $(d'^{(L)}_p, d'^{(L)}_{q,1}, \ldots, d'^{(L)}_{q,z})$ are upper bounded by N^δ, where*

$$\gamma \leq \delta < \frac{1}{2} - 2\alpha - \gamma,$$

then $(k', l'_1, \ldots, l'_z)$ can be computed in time $\mathcal{O}(\log^2 N)$ with probability $1/\zeta(z)$ under Assumption 2.

Proof. In this case, to compute k', we use one observation $d'^{(M)}_p$ combined with z observations $d'^{(M)}_{q,1}, \ldots, d'^{(M)}_{q,z}$. To this end, we first calculate all the products $k'l'_1, \ldots, k'l'_z$, using the process described in the proof of Lemma 8.3 and then compute $k^* = \gcd(k'l'_1, \ldots, k'l'_z)$. According to Section 8.2.1, all

the products $k'l'_i$ can be calculated in polynomial time $\mathcal{O}(\log^2 N)$ using the already known MSBs ($d_p'^{(M)}$, $d_{q,1}'^{(M)}$, …, $d_{q,z}'^{(M)}$) and public key (N, e). Now, $k^* = k'$ if and only if $\gcd(l'_1, \ldots, l'_z) = 1$, which, according to Assumption 2 occurs with probability $1/\zeta(z)$. Upon obtaining k', it can be verified that this is correct by computing a corresponding l'_i and verifying that they satisfy Equation (8.4). The complexity of the GCD computation is $\mathcal{O}((\log z) \cdot M((2\alpha + 2\gamma)\log N)\log((2\alpha + 2\gamma)\log N)$ considering the state-of-the-art quasi-linear time recursive algorithm [SZ04, **Theorem 4**]. Putting all together, the time complexity of computing k' in the MSB case using multiple known MSBs is $\mathcal{O}(\log^2 N)$. □

8.2.3 Step 1c: Computing (k', l') with known LSBs.

As mentioned above, in the LSB case, the approach to computing k' using Coppersmith's lattice-based method relies on the standard heuristic Assumption 1. In Section 8.3, we will verify the efficiency of this heuristic method in practice.

Lemma 8.5 ((k', l') from LSB). *Let (N, e) be the public key and N be large enough with $e = N^\alpha$. Given the LSBs ($d_p'^{(L)}$, $d_q'^{(L)}$), if the unknown parts ($d_p'^{(M)}$, $d_q'^{(M)}$) are upper bounded by N^δ, where*

$$\delta < \frac{1}{2} - 2\alpha - 2\gamma,$$

then (k', l') can be computed in polynomial time under Assumption 1.

Proof. Replacing d'_p, d'_q in Equation (8.3) with $d_p'^{(M)}2^i + d_p'^{(L)}$, $d_q'^{(M)}2^i + d_q'^{(L)}$ respectively, we get

$$k'l'N \equiv (k'-1)(l'-1) + ed_p'^{(L)}(l'-1) + ed_q'^{(L)}(k'-1) + e^2 d_p'^{(L)} d_q'^{(L)} \mod 2^i e,$$

Denote

$$A \equiv -e^2 d_p'^{(L)} d_q'^{(L)} + ed_p'^{(L)} + ed_q'^{(L)} - 1,$$

then we obtain

$$k'l'(N-1) - k'(ed_q'^{(L)} - 1) - l'(ed_p'^{(L)} - 1) + A \equiv 0 \mod 2^i e. \tag{8.5}$$

A polynomial is constructed using Equation (8.5) as below:

$$f(x,y) = (N-1)xy - (ed_q'^{(L)} - 1)x - (ed_p'^{(L)} - 1)y + A,$$

which has the root (k', l') modulo $2^i e$ and all the coefficients can be easily calculated. To compute k' and l' using Coppersmith's lattice-based method, similar to [MNS22], we transform f into another polynomial g that has at least one *small* coefficient (*i.e.*, it does not grow as function of N). More precisely, we construct g by replacing the coefficient of x by $\gcd(ed_q'^{(L)} - 1, 2^i)$ and multiplying all other coefficients of f with

$$\left(\frac{ed_q'^{(L)} - 1}{\gcd(ed_q'^{(L)} - 1, 2^i)} \right)^{-1} \mod 2^i e.$$

We know that k', l' are upper bounded by $N^\gamma e$, and it is also known that, under Assumption 1, all roots (x_0, y_0) of g modulo $2^i e$ that satisfy $|x_0|, |y_0| < N^\gamma e$ can be solved in polynomial time if

$$(N^\gamma e)^2 < (2^i e)^{\frac{2}{3}}. \tag{8.6}$$

Because $e = N^\alpha$ and $2^i = \Theta(N^{\frac{1}{2}+\gamma-\delta})$, the inequality 8.6 is actually asymptotically equivalent to

$$\delta < \frac{1}{2} - 2\alpha - 2\gamma,$$

which completes the proof. □

8.2.4 Step 2: Factoring N with computed k'.

After computing k' as described in Sections 8.2.1 and 8.2.2 for the MSB case or in Section 8.2.3 for the LSB case, the second step is to factor N in polynomial time using the computed k' and the known part of d_p'. Please keep in mind that, in the MSB case when factoring is used, this step occasionally has to be performed twice because we do not know which of the two factors obtained in Step 1 corresponds to k' (as opposed to l'). Our Step 2 is similar to [MNS22] and based on their **Theorem 3** as below:

Theorem 8.6 (May-Nowakowski-Sarkar [MNS22])**.** *Suppose we are given a polynomial $f(x) = x + a$ and integers $k, N \in \mathbb{N}$, where $k = N^\mu$ for some $\mu \geq 0$.*

Let $p > N^\beta \in \mathbb{N}, \beta \in [0,1]$ be an unknown divisor of N. In time polynomial in $\log N, \log k$ and $\log a$, we can compute all integers x_0, satisfying

$$f(x_0) \equiv 0 \mod kp \quad \textit{and} \quad |x_0| \leq N^{\beta^2+\mu}.$$

To factor N with the computed k' and the known parts of (d'_p, d'_q), we have the following Lemma 8.7, which is a direct application of Theorem 8.6.

Lemma 8.7. *Let (N, e) be the public key and N be large enough with $e = N^\alpha$. Given the value k' and the MSBs $(d_p'^{(M)}, d_q'^{(M)})$ or the LSBs $(d_p'^{(L)}, d_q'^{(L)})$. If the unknown parts of (d'_p, d'_q) are upper bounded by N^δ for $\delta < \frac{1}{4} + \alpha + \gamma$, then N can be factored in time polynomial in $\log N, \log k'$ and $\log a$, where*

$$a = (ed_p'^{(M)} 2^i + k' - 1) \cdot (e^{-1} \mod k'N)$$

in the MSB case, or

$$a = \left(\frac{ed_p'^{(L)} + k' - 1}{\gcd(2^i e, k'N)} \right) \cdot \left(\left(\frac{2^i e}{\gcd(2^i e, k'N)} \right)^{-1} \mod \frac{k'N}{\gcd(2^i e, k'N)} \right)$$

in the LSB case.

Proof. In the MSB case, according to

$$ed'_p = 1 + k'(p-1) \quad \textit{and} \quad d'_p = d_p'^{(L)} + d_p'^{(M)} 2^i$$

we have

$$ed_p'^{(L)} + ed_p'^{(M)} 2^i + k' - 1 = k'p.$$

We construct a polynomial $f_{MSB}(x) = x + a \mod k'p$ with known coefficient

$$a = (ed_p'^{(M)} 2^i + k' - 1) \cdot (e^{-1} \mod k'N)$$

and root $x_0 = d_p'^{(L)}$.

Since $k' = \Theta(N^{\alpha+\gamma})$ and $p = \Theta(\sqrt{N})$, it is concluded from [MNS22, **Theorem 3**] that the unknown part $d_p'^{(L)}$ can be solved in polynomial time if $d_p'^{(L)} < N^{\frac{1}{4}+\alpha+\gamma}$. This condition is already fulfilled because $d_p'^{(L)} \leq N^\delta$. The last step to factor N is to get $p = \gcd(f_{MSB}(d_p'^{(L)}), N)$.

In the LSB case, similarly we construct a polynomial

$$f_{LSB}(x) = x + (ed_p'^{(L)} + k' - 1) \cdot ((2^i e)^{-1} \mod k'N),$$

however, it is slightly different from the MSB case that the modular multiplicative inverse $(2^i e)^{-1} \mod k'N$ does not always exist because $\gcd(2^i e, k'N) = 1$ does not always hold. To this end, we slightly change the polynomial to

$$f_{LSB}(x) = x + \left(\frac{ed_p'^{(L)} + k' - 1}{\gcd(2^i e, k'N)}\right) \cdot \left(\left(\frac{2^i e}{\gcd(2^i e, k'N)}\right)^{-1} \mod \frac{k'N}{\gcd(2^i e, k'N)}\right),$$

in this way, it is guaranteed that the modular multiplicative inverse $\left(\frac{2^i e}{\gcd(2^i e, k'N)}\right)^{-1} \mod \frac{k'N}{\gcd(2^i e, k'N)}$ exists. The rest is the same as the MSB case to factor N to get $p = \gcd\left(f_{LSB}(d_p'^{(M)}), N\right)$ and that finishes our proof. □

It is worth noting that our EPKE also does not work if $e \geq N^{\frac{1}{4}-\gamma}$, unless factoring is easy, according to [MNS22, **Corollary 1**].

8.3 Experimental results

As aforementioned, it is critical to experimentally verify the validity of Assumption 1 in the LSB case, as well as the validity of Assumption 2 and the sub-exponential time of factoring $k'l'$ in the MSB case. To assess the effectiveness of the EPKE attacks, we first conduct the EPKE attacks to recover the entire private exponent with $\frac{1}{3}$ known blinded MSBs or LSBs of d'_p and d'_q, with different key and additive blinding factor lengths. To this end, we consider the commonly used key length of 1024-bit, 2048-bit and 3072-bit, the commonly used additive blinding factor length of 32-bit, 64-bit and 128-bit, and both MSB and LSB cases. Rather than executing the EPKE attacks on real or simulated side-channel leakage, known keys are generated and a fraction of the known bits of the blinded private exponents are used directly. Since our bounds depend on both the length of e and the length of the additive blinding factor, we choose the length of e based on the length of the additive blinding factor such that $r_p^{\frac{2}{3}} e \approx N^{\frac{1}{12}}$ in the MSB case and such that $r_p e \approx N^{\frac{1}{12}}$ in the LSB case. In these cases, we can use only $\frac{1}{3}$-

Table 8.1: Summary of the EPKE experiments - MSB case with single $d_q'^{(M)}$

Len(N)	Len(e)	Len(r_p, r_q)	Len(UnknownLSB)	Step 1a Factoring time	Step 2 Lattice Dim.	LLL time
1024	64	32	336/**352**	<1 s	21	1s
1024	43	64	347/**362**	<1 s	21	1s
1024	17*	128	347/**401**	2s	21	2s
2048	149	32	665/**693**	42s	21	4s
2048	128	64	677/**704**	175s	21	4s
2048	85	128	697/**725**	340s	21	4s
3072	235	32	1008/**1034**	1787s	31	60s
3072	213	64	1014/**1045**	5993s	31	60s
3072	171	128	1032/**1066**	6651s	31	60s

Table 8.2: Summary of the EPKE experiments - MSB case with multiple $d_q'^{(M)}$

Len(N)	Len(e)	Len(r_p, r_q)	Len(UnknownLSB)	Step 1b Success Prob.
1024	64	32	336/**352**	0.67/0.89/0.93/0.97/0.98/0.99/0.98/0.98/1.00
1024	43	64	347/**362**	0.65/0.89/0.91/0.98/0.97/0.99/0.99/1.00/0.99
1024	17*	128	347/**401**	0.65/0.78/0.94/0.99/0.96/0.99/0.99/1.00/1.00
2048	149	32	665/**693**	0.67/0.79/0.89/0.95/0.99/1.00/0.98/0.99/1.00
2048	128	64	677/**704**	0.73/0.86/0.92/0.93/0.98/1.00/1.00/1.00/1.00
2048	85	128	697/**725**	0.73/0.86/0.92/0.95/0.99/1.00/0.99/1.00/1.00
3072	235	32	1008/**1034**	0.69/0.81/0.93/0.98/0.98/0.99/1.00/1.00/1.00
3072	213	64	1014/**1045**	0.72/0.82/0.94/0.98/0.98/0.99/1.00/1.00/1.00
3072	171	128	1032/**1066**	0.67/0.81/0.93/1.00/0.99/1.00/1.00/1.00/1.00

part of known bits to recover the full key. More precisely, $\alpha = \frac{1}{12} - \frac{2}{3}\gamma$ according to our bounds $\min\{\frac{1}{4} + \alpha + \gamma, \frac{1}{2} - 2\alpha - \gamma\}$ for the MSB case, and $\alpha = \frac{1}{12} - \gamma$ according to our bounds $\min\{\frac{1}{4} + \alpha + \gamma, \frac{1}{2} - 2\alpha - 2\gamma\}$ for the LSB case. If $\gamma = 0$, *i.e.*, without exponent blinding, our bounds are the same as [MNS22] for both MSB and LSB cases, *i.e.*, our bounds are the generalization of their work. The experimental EPKE attack results are summarized in Table 8.1 and Table 8.2 for the MSB case and Table 8.3 for the LSB case. The experiments are repeated 100 times for each setting by randomly generating a CRT key pair including e, so the time values in each table correspond to the average time over 100 experiments of each setting. We have implemented the experiments in SAGE 9.5 and YAFU [bbu22] 2.08 factorization toolkit (Ubuntu 20.04.4) with an Intel® Core™ i5-7500 CPU 3.40GHz.

The fourth column shows two values, the ones in bold font correspond

Table 8.3: Summary of the EPKE experiments - LSB case

Len(N)	Len(e)	Len(r_p, r_q)	Len(UnknownMSB)	Step 1c Lattice Dim.	LLL time	Step 2 Lattice Dim.	LLL time
1024	53	32	320/**341**	121	216s	21	<1 s
1024	21	64	320/**341**	121	202s	21	<1 s
1024	17*	128	191/**222**	121	461s	21	4s
2048	139	32	648/**682**	121	487s	21	4s
2048	107	64	647/**682**	121	470s	21	4s
2048	43	128	649/**682**	121	419s	21	4s
3072	224	32	978/**1024**	121	1104s	31	90s
3072	192	64	978/**1024**	121	1110s	31	90s
3072	128	128	976/**1024**	121	991s	31	87s

to the theoretical bounds, and the others correspond to the experimental results. In the MSB case, with a public exponent e of size $\alpha = \frac{1}{12} - \frac{2}{3}\gamma$, we nearly reach the asymptotic bound using a $\min\{\frac{1}{4} + \alpha + \gamma, \frac{1}{2} - 2\alpha - \gamma\}$-part MSBs of blinded CRT exponents (d'_p, d'_q) to recover the entire private key.

In the LSB case, with a public exponent e of size $\alpha = \frac{1}{12} - \gamma$, we succeeded in computing (k', l') in each performed experiment, affirming the validity of Assumption 1. We closely reach the asymptotic bound using a $\min\{\frac{1}{4} + \alpha + \gamma, \frac{1}{2} - 2\alpha - 2\gamma\}$-part LSBs of blinded CRT exponents (d'_p, d'_q) to recover the entire private key. We need more known bits to reveal the key compared with the MSB case.

It has to be mentioned that the fourth row (1024-bit key with 128-bit blinding factor) in both MSB and LSB cases is different from the other settings. Because the chosen e should be 1 (resp., $N^{-\frac{1}{24}}$) for the MSB (resp., LSB) case if we want to use $\frac{1}{3}$ known MSBs/LSBs of blinded CRT exponents (d'_p, d'_q) to recover the full private key. However, those two e values are not realistic, instead, we choose a very widely-used (*e.g.*, $2^{16} + 1$) size of e, *i.e.*, 17-bit, which complies with the NIST's recommendation of e and is also close to those two values. Our experiments suggest that in the MSB case the required known bits are close to the optimum value, *i.e.*, $\frac{1}{3}$ MSBs of blinded exponents. While in the LSB case we need more than $\frac{2}{3}$ LSBs of blinded exponents, because the used e value is too far away from the expected value $e = N^{-\frac{1}{24}}$.

Another critical point to be verified is the required sub-exponential factoring time to compute k' in Step 1a (see Section 8.2.1), as shown in Figures 8.1-8.3, clearly it can be observed that the required factoring time for all the settings is certainly affordable even with an average PC. Mostly, the

factoring finishes within seconds or minutes, but on very few occasions it requires between one hour and one and a half hours (resp., fourteen hours and fifty hours) for 2048-bit (resp., 3072-bit) key with 128-bit blinding. It is worth noting that the 1024-bit key with the 32-bit blinding factor case requires slightly more time than the 1024-bit key with the 64-bit blinding factor case, because the former case uses SAGE's internal factorization function while all other cases utilize YAFU's factorization functions.

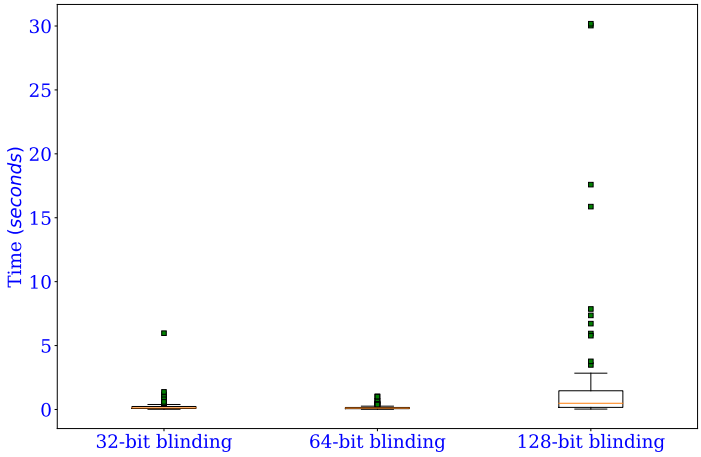


Figure 8.1: Step 1a Factoring Time for 1024-bit key.

Finally, the success probability of the probabilistic method to obtain k' in Step 1b (see Section 8.2.2) is shown in Figure 8.4. For all parameter sets in Tables 8.1-8.3, 100 RSA key pairs were generated, and for each key pair, 1,000 sets of values $(l'_1, \ldots, l'_{10})$ were generated based on blinded exponents. The number of values required to obtain a GCD of 1 was computed for each set, in order to determine the estimated success probability of the attack depending on z. Our experimental results in Table 8.2 also validate the estimated probability here as part of the full attack path, as the last column indicates the successful EPKE attack probability with $2 \sim 10$ different randomly generated l'_z values for each setting of key size and blinding factor length. As can be observed, the successful EPKE attack probability is already above 0.65 using only two l'_z values, and it reaches up to almost 1 using five l'_z values. Therefore, we experimentally verify the validity of As-

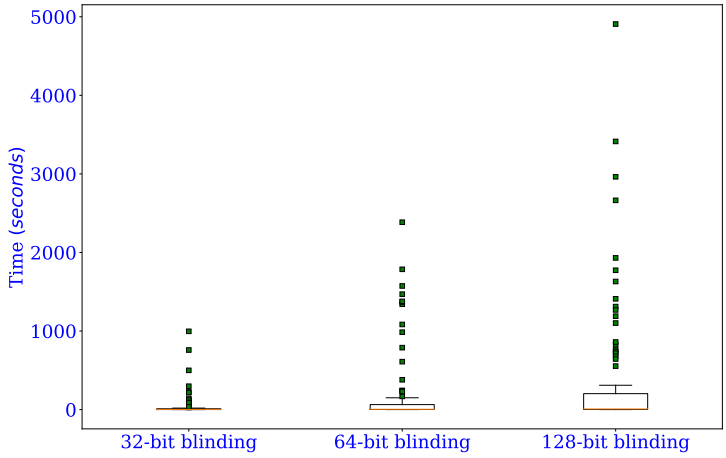


Figure 8.2: Step 1a Factoring Time for 2048-bit key.

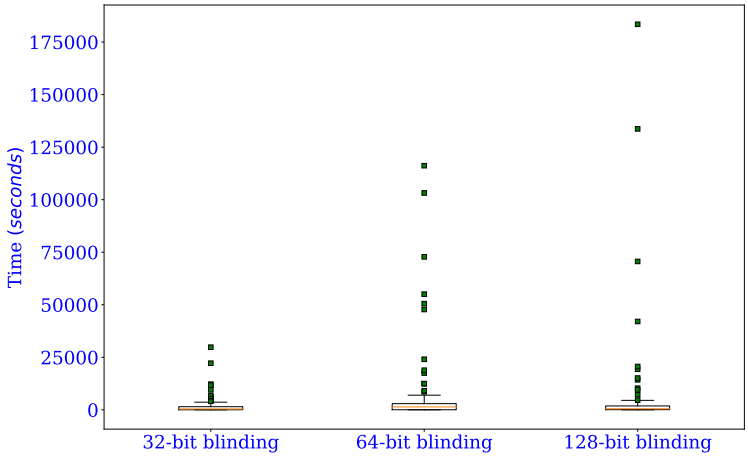


Figure 8.3: Step 1a Factoring Time for 3072-bit key.

sumption 2. In addition, as mentioned above, the f factor can be recovered using only two l'_z values with a small brute-force because it is expected to be small in practice. We also empirically confirm that the f factor is indeed pretty small as shown in Figures 8.5-8.7. Mostly it is smaller than 100 for different key sizes and exponent blinding factor lengths, the brute-force effort is affordable to compute k' with only two l'_z values to achieve the successful EPKE attack probability of 1.

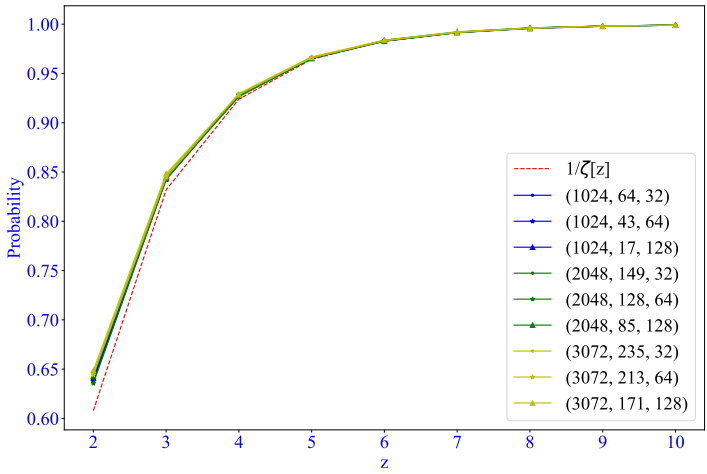


Figure 8.4: Step 1b Estimated versus assumed probability that $l'_1, \ldots, l'_z$ are coprime.

A somewhat surprising observation is, from a PKE attack perspective, comparing to without additive exponent blinding, CRT-RSA with additive exponent blinding of size N^γ and public exponents $e \leq N^{\frac{1}{12}-\gamma}$ becomes easier to attack with larger exponent blinding factors. For instance, CRT-RSA with 2048-bit N and widely used 17-bit public exponent e. Without exponent blinding, the PKE attack can efficiently break the scheme with roughly 48% (492 bits) of the LSBs of the CRT exponents. On the contrary, given a 153-bit exponent blinding, then roughly only 33% (389 bits) are required. However, from an SCA attack point of view, it is other way around. The additive exponent blinding limits an attacker to use only a single attack trace to get the partial key information instead of using multiple attack traces without exponent blinding.

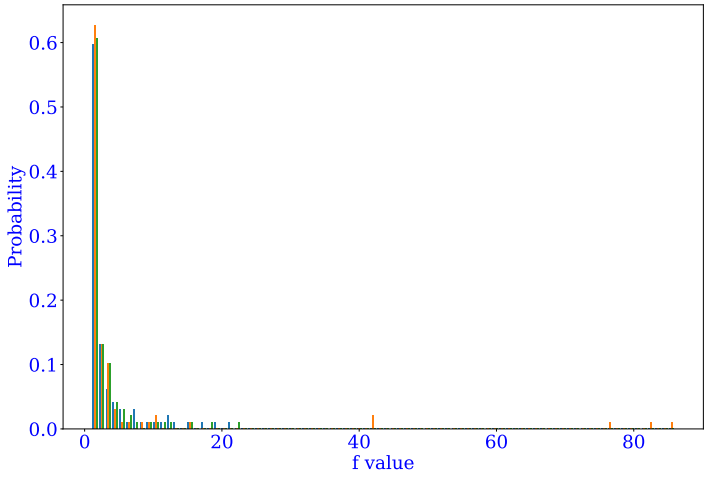


Figure 8.5: Step 1b Histogram of f values with two l'_z values for 1024-bit key.

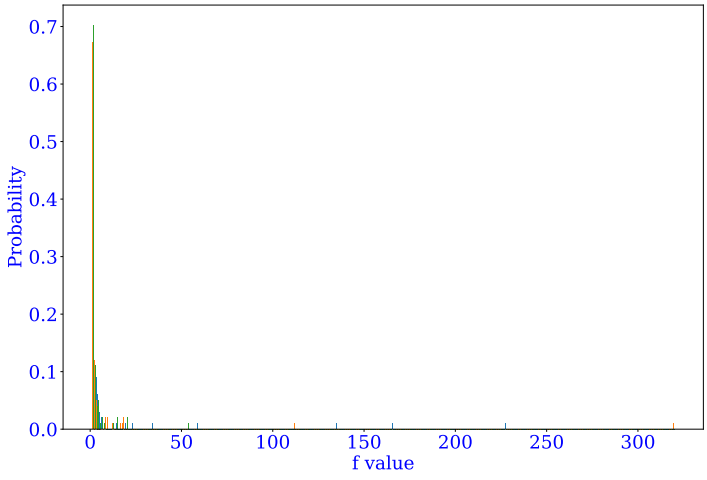


Figure 8.6: Step 1b Histogram of f values with two l'_z values for 2048-bit key.

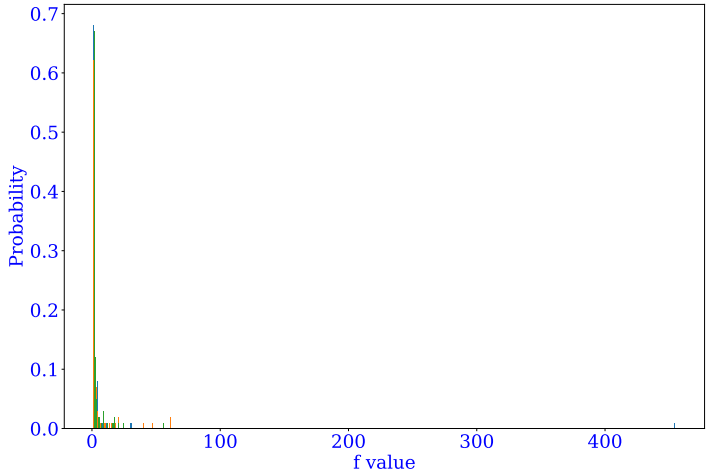


Figure 8.7: Step 1b Histogram of f values with two l'_z values for 3072-bit key.

In both cases, we verified the effectiveness of the proposed EPKE attacks using partially known blinded CRT exponents to disclose the entire private key. Next, we will demonstrate an application of this EPKE attack using the obtained partial side-channel key leakage via profiled attacks in a realistic context.

8.4 A use case of EPKE on real SCA partial key leakage

In the following, we first introduce our profiled attacks-based experimental verification methodology concerning the EPKE attacks for an attacker/evaluator to disclose the full private key based on partially recovered CRT exponents. It includes the metrics used for training neural networks and the knowledge of POIs assumption. In the end, we present the experimental results rendering this verification methodology.

8.4.1 Deep learning profiled attack

Since the seminal work of Kocher [Koc96], SCA has been a powerful and de facto tool to evaluate the physical security of various cryptographic implementations, especially on embedded devices. There is a rich body of literature about SCA on RSA implementations. Some of these works [Wal01, CFG+10, PC15, CCC+19a, ZBHV21] concentrated on RSA implementations with additive blinding. In particular, Carbone *et al.* [CCC+19a] conducted profiled attacks, which is considered the most powerful SCA, on an RSA implementation with all aforementioned blinding countermeasures on a CC EAL4+ certified IC. Zaid *et al.* [ZBHV21] improved the profiled attack by introducing a new ensembling loss function.

As introduced in Section 7.1, profiled attacks consist of two phases, *i.e.*, the profiling phase and the attack phase. In this chapter, we aim at the two possible values for one exponent bit, and the combination of probabilities is skipped because only single-trace attacks are in scope due to the exponent blinding countermeasure.

We use deep learning profiled attacks (DL) described in Section 5.1.5 to experimentally investigate the efficiency of our proposed extended PKE attack in the additive exponent blinding scenario. In this regard, the deep learning model is used to extract the required partial MSBs or LSBs of blinded CRT exponents as the input to our extended PKE attack to reveal the entire private key. Note that, in this chapter, only a single attack trace is used to decide the final probability of each attacked exponent bit. We use a published MLP [Mag20] neural network model for our DL profiled attacks. The structure of this model is very simple and shallow, but it showed pretty good performance. We will describe the parameter settings in Section 8.4.6.

8.4.2 Knowledge of POIs assumption

There is an implicit assumption about the knowledge of the POIs to apply the profiled attacks to CRT exponentiation implementations. That is, the attackers/evaluators can determine the rough timing interval of each exponent bit calculation in the side-channel traces. It is feasible for most of security products in a grey-box testing context via SPA/SEMA and CPA/CEMA (or similar techniques) as shown in [CCC+19a, DLO+19]. For instance, one can vary the key length, perform correlation analyses on the input and output data, make use of the design information such as the location of the RSA co-processor in the glue logic area or temporarily switch off some

side-channel countermeasures like jitters. The additive exponent blinding countermeasure will not make this harder, because it only adds more patterns corresponding to exponent bit calculations in the side-channel traces. If the patterns of one exponent bit calculation without exponent blinding can be identified in a side-channel trace, it is the same to identify the exponent bit calculation patterns given the presence of an additive exponent blinding countermeasure.

8.4.3 Metrics

We use the most common machine learning metric accuracy [GBC16] to monitor and evaluate our deep learning profiled attacks. Its definition is the successful classification rate obtained over a dataset. Accordingly, the training accuracy, the validation accuracy and the test accuracy correspond to the reached successful classification rates respectively over the training, the validation and the test sets. The training and validation accuracy metrics are used to monitor the performance of the neural network training, and we use the test accuracy to evaluate the trained model. The accuracy is suitable for our experiments because we focus on the successful classification rate of each exponent bit, and we use a balanced dataset (the number of profiling traces of each class is the same) to avoid the potential deceptive impact of the accuracy metric. Concerning the metrics used to train the neural networks, we use the NLL loss function [CDP17] for DL profiled attacks. It is a loss function calculated as $-\log y$ used in multi-class classification, where y is a prediction corresponding to the ground-truth label after the softmax [GBC16] activation function is applied. The loss for a mini-batch is computed by taking the mean or sum of all items in the batch. Because it is proved that minimizing the NLL loss is equivalent to maximizing the Perceived Information [RSV+11, BHM+19] and thus minimizing the online attack complexity, thanks to the recent work [MDP20].

8.4.4 Montgomery Ladder Exponentiation implementation

Our target is a Montgomery Ladder exponentiation CRT-RSA implementation on a modern 45 nm secure microcontroller equipped with an RSA co-processor running at 100 MHz. This is a typical real-world target from a side-channel attack viewpoint due to the implemented side-channel countermeasures, that is, SPA/DPA-resistant atomic Montgomery Ladder exponentiation with additive message and exponent blinding and multiplica-

tive modulus blinding. Moreover, the 32-bit CPU also has variable internal clock, random branch insertion, memory encryption and physical address scrambling countermeasures to enhance the side-channel resistance.

Algorithm 5 illustrates the implemented left-to-right Montgomery Ladder exponentiation [JY02]. It is a well-known and widely used SPA-resistant regular exponentiation algorithm without using dummy operations to defeat SPA/SEMA and safe-error attacks [YJ00]. We view the n-bit private exponent as a binary vector $\mathbf{d} = (d_{n-1}, ..., d_0)$ (where d_0 is the LSB). N is the modulus used for the exponentiation, and C is the message to be decrypted using the private exponent.

Algorithm 5 Montgomery Ladder.

Input: $C, N, \mathbf{d} = (d_{n-1}, ..., d_0)$
Output: $M = C^d \mod N$

```
1: R_0 ← 1
2: R_1 ← C
3: for i = n − 1 downto 0 do
4:     R_{¬d_i} ← R_{d_i} × R_{¬d_i} mod N
5:     R_{d_i} ← R_{d_i} × R_{d_i} mod N
6: end for
7: end for
8:
9: return R_0
```

8.4.5 EPKE attack on SCA partial key leakage verification strategy

To represent a realistic scenario of EPKE attack on CRT using SCA partial key leakage, we follow the verification strategy as below:

1. Determine the Montgomery Ladder exponentiation interval using the SPA technique as discussed in Section 8.4.2.
2. Measure a set of 5,000 profiling traces focusing on the exponentiation interval using a CRT key pair labelled as K_{Prof0}.
3. Randomly generate ten different CRT key pairs with 128-bit e values (considering a 64-bit additive blinding factor used by the target CRT implementation) to measure the attack traces for the MSB case, denote those ten attack key pairs as $K_{MSB1}, K_{MSB2}, ..., K_{MSB10}$. Similarly, for the LSB case, randomly generate another ten different CRT

key pairs $K_{LSB1}, K_{LSB2}, ..., K_{LSB10}$ with 107-bit e values to acquire the corresponding attack traces.

4. Measure a set of 10 attack traces focusing on the exponentiation interval using each of those 20 CRT attack key pairs labelled as K_{MSBi} and K_{LSBi}, $i \in [1, 10]$.
5. Train the neural network model using the profiling traces to save the best model with the highest test accuracy.
6. Perform the profiled attack on all the attack traces using the saved best neural network model to recover the MSBs or LSBs of CRT exponents. More precisely, recover 411 ($= 1024 + 64 - 677$, according to the sixth row in Table 8.1) MSBs of d'_p and d'_q for each attack trace with $K_{MSB1}, K_{MSB2}, ..., K_{MSB10}$. Similarly, recover 441 ($= 1024 + 64 - 647$, according to the sixth row in Table 8.3) LSBs of d'_p and d'_q for each attack trace with $K_{LSB1}, K_{LSB2}, ..., K_{LSB10}$.
7. Conduct the EPKE attack using the recovered 411 MSBs of d'_p and d'_q to disclose the full private key for $K_{MSB1}, K_{MSB2}, ..., K_{MSB10}$ and verify the required sub-exponential time. To this end, we benefit from the previously mentioned combination of the recovered MSBs of d'_p from one attack trace and the recovered MSBs of d'_q from another attack trace. For each MSB attack key K_{MSBi}, we have 100 different combinations, using those combinations to tolerate potential SCA errors. We do the same for the LSB attack keys. In this case we need to verify the validity of the Coppersmith-type heuristic assumption.

We acquired power consumption traces with a Lecroy WaveRunner 8254 oscilloscope at a sampling rate of 500 MS/s. For each exponentiation execution, we triggered the oscilloscope at its end and recorded the processing of the entire exponentiation. Each trace consists of 16,000,000 sample points. Each exponent bit processing corresponds to two Montgomery modular multiplications as shown in Algorithm 5, and it contains about 6,074 sample points illustrated in Figure 8.8.

While we recorded the entire exponentiation for this experiment, it is worth mentioning that an extra advantage of our EPKE attack is that the recovered MSBs or LSBs of d'_p and the MSBs or LSBs of d'_q can come from different observations of the partial key leakage. An attacker can, at his own convenience, capture and combine the error-free MSBs or LSBs of d'_p from one observation of the partial key leakage, and those of d'_q from another one, while the remaining parts can be arbitrarily erroneous or not

captured at all (*i.e.*, only capture one-third of the execution of one of the two CRT exponentiations).

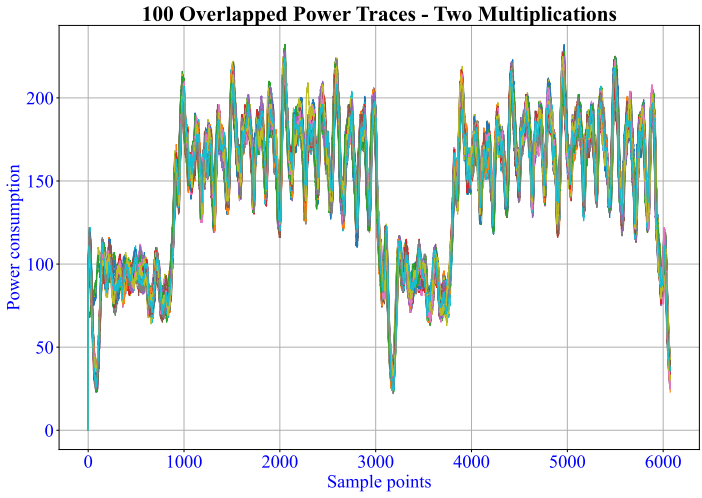


Figure 8.8: 100 Overlapped Power Traces of One Exponent Bit Processing.

8.4.6 Attack results.

We have implemented the DL profiled attacks in Python and PyTorch [PGM+19] version 1.10.1 with an NVIDIA GTX 1080Ti GPU. We use the *Adadelta* optimizer [Zei12] and utilize the adaptive learning rate policy *ReduceLROnPlateau* to gradually decrease the learning rate (with the default Adadelta optimizer initial learning rate of 1.0) with a factor of 0.05 if the training stagnates. We use a batch size of 512 and 100 as the number of epochs. All profiling and attack traces are normalized using the *StandardScalar* function from the Scikit-learn [PVG+11] library by removing the mean and scaling to unit variance. 20% of profiling traces compose a validation set. A validation data set is crucial to DL performance because it provides a way of instantly detecting over-fitting [Guy97]. The DL attacks utilize the trained model with the highest validation accuracy. As mentioned above, we use the NLL loss for the model training. Table 8.4 summarizes the details of the used DL model and the corresponding hyperparameters.

Table 8.4: MLP model details

MLP
nb_epoch = 100 batch_size_training = 512
Dense(50, activation="relu", input_shape=(nb_samples,)) BatchNormalization() Dense(100, activation="relu") BatchNormalization() Dense(2, activation="softmax") compile(loss='categorical-crossentropy', optimizer='adadelta', metrics=['accuracy'])
learning_rate_policy = ReduceLROnPlateau(optimizer, 'min', factor=0.05, verbose=True)

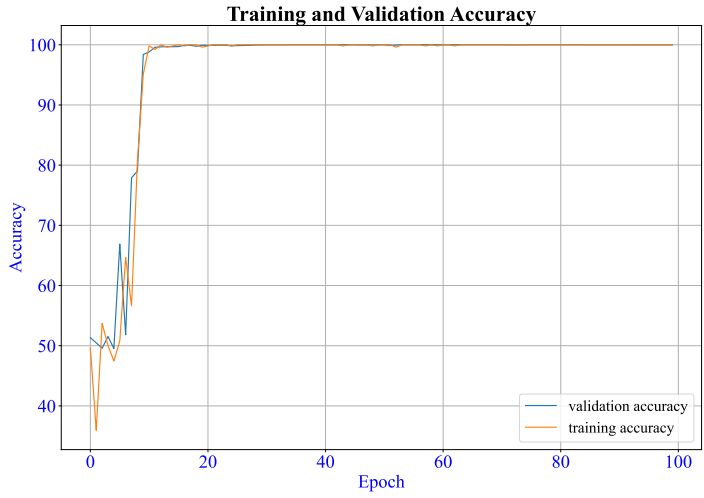


Figure 8.9: Training and Validation Accuracy.

We first train the model using the profiling traces to save the best model based on the highest validation accuracy depicted in Figure 8.9, in which the best model corresponds to a validation accuracy of 100%.

The saved best model is then used to retrieve the 411 (resp., 441) MSBs (resp., LSBs) of d'_p and d'_q for all the 10 MSB-case (resp., LSB-case) attack traces with attack key K_{MSBi} (resp., K_{LSBi}). For each attack key, there are 100 combinations of recovered MSBs (resp., LSBs) of d'_p from one attack trace and recovered MSBs (resp., LSBs) of d'_q from another attack trace. We conduct the EPKE attacks using those combinations to recover the prime factor p as described in Section 8.2.4. The second column of Tables 8.5 and 8.6 presents the number of successful recovery of the prime factor out of 100 combinations for each attack key. The last three (resp., four) columns indicate the time cost and lattice dimensions of the two-step EPKE attack on the MSB (resp., LSB) cases. In the third column of Table 8.5, we present the average factoring time to compute k' and the corresponding minimum and maximum time (in the brackets).

Table 8.5: Summary of the real SCA data EPKE experiments - MSB case

Attack Key	Success Nr.	Step 1a Factoring time	Step 2 Lattice Dim.	LLL time
K_{MSB1}	26	110s ([0.93s, 599.7s])	21	6s
K_{MSB2}	35	90s ([0.5s, 1379.7s])	21	6s
K_{MSB3}	15	176s ([1.0s, 1817.7s])	21	6s
K_{MSB4}	3	10s ([6.1s, 16.4s])	21	6s
K_{MSB5}	9	202s ([1.5s, 1211.8s])	21	6s
K_{MSB6}	4	251s ([59.7s, 477.9s])	21	6s
K_{MSB7}	4	65s ([2.1s, 235.7s])	21	6s
K_{MSB8}	4	212s ([1.6s, 775.2s])	21	6s
K_{MSB9}	4	969s ([764.4s, 1373.7s])	21	6s
K_{MSB10}	12	21s ([2.6s, 60.7s])	21	6s

The results further confirm the effectiveness of our EPKE attack in a realistic context, that is, disclosing a fraction of blinded CRT exponents via SCA followed by an EPKE attack to reveal the entire private key. In addition, the SCA experiments show that the proposed EPKE attack can tolerate slight SCA errors as demonstrated by our proposed verification strategy, *i.e.*, combining the recovered error-free partial MSBs or LSBs of d'_p from one trace and MSBs or LSBs of d'_q from another one. When considering real-world CRT implementations, SCA attacks will often result in errors, which is why it is essential for an attacker/evaluator to be able to

Table 8.6: Summary of the real SCA data EPKE experiments - LSB case

Attack Key	Success Nr.	Step 1c Lattice Dim.	LLL time	Step 2 Lattice Dim.	LLL time
K_{LSB1}	6	121	388s	21	2s
K_{LSB2}	2	121	404s	21	2s
K_{LSB3}	4	121	407s	21	3s
K_{LSB4}	3	121	421s	21	3s
K_{LSB5}	8	121	410s	21	3s
K_{LSB6}	5	121	407s	21	3s
K_{LSB7}	9	121	399s	21	3s
K_{LSB8}	12	121	405s	21	3s
K_{LSB9}	16	121	442s	21	3s
K_{LSB10}	9	121	433s	21	3s

apply (E)PKE attacks.

8.5 Conclusion and Future Work

Many existing PKE works focused on implementations of RSA and its CRT variant without exponent blinding countermeasures. The state-of-the-art PKE attack on CRT without exponent blinding [MNS22] can recover the whole CRT private key with only a third MSBs or LSBs of CRT exponents when the public exponent $e \approx N^{\frac{1}{12}}$. In this chapter, we showed that it can be extended to CRT implementations with additive exponent blinding, which is the most widely deployed exponent blinding countermeasure in real-world products. We proposed an extended PKE attack to recover the full CRT private key using partial disclosed MSBs or LSBs of additively blinded CRT exponents. It follows a two-step approach, first computing the key-dependent constant k' and then factoring the N using partial MSBs or LSBs of blinded CRT exponents d'_p and d'_q.

The mathematical proof and time-complexity analyses suggest that a third MSBs or LSBs of blinded CRT exponents d'_p and d'_q are enough to recover the entire CRT private key: in polynomial time based on the standard Coppersmith-type heuristic assumption in the LSB case when $e \approx N^{\frac{1}{12}-\gamma}$, as well as in sub-exponential (yet practically feasible) time or in probabilistic polynomial time under a heuristic assumption in the MSB case when $e \approx N^{\frac{1}{12}-\frac{2}{3}\gamma}$. Under different settings of typical key size and exponent blinding factor length, our extensive experiments verify the validity of the Coppersmith-type heuristic for the LSB case, as well as the affordability

of the required sub-exponential time and the heuristic assumption for the MSB case. More precisely, in practice, using an average PC, the required sub-exponential time is mostly in seconds and minutes, the worst-case occasionally observed is less than one and a half hours.

Moreover, as an application of the proposed EPKE attack in real life, utilizing real SCA partial key leakage of a real-world SPA/DPA-resistant Montgomery Ladder CRT implementation on a 45 nm secure microcontroller with an RSA co-processor, our SCA experimental results suggest that the EPKE attack can tolerate slight SCA errors of the recovered MSBs or LSBs of the blinded CRT exponents. Such errors are expected to occur in a realistic scenario, especially when only a single attack trace is available. It sheds some light on the error-tolerant potential of the EPKE attack, and points out a possible future direction of our work, *i.e.*, how to improve the EPKE to tolerate more generic SCA errors of the recovered MSBs or LSBs, *e.g.*, in the binary symmetric model where every recovered exponent bit will be "flipped" with a crossover probability (*i.e.*, the Bernoulli distribution).

Chapter 9

Conclusions and Perspective

In this thesis, targeting different widely used symmetric (focusing on AES) and asymmetric (focusing on ECC and CRT-RSA) cryptography algorithms in the security evaluations, we aimed to make side-channel security evaluations involving common SCA countermeasures more efficient while preserving a high assurance level. To reach this goal, we analyzed both information extraction and exploitation phases of side-channel attacks and considered conventional multivariate SCA attacks and emerging deep learning SCA attacks and partial key exposure attacks. In the following, we summarize the results and contributions and discuss some potential future research directions along with our works.

9.1 Multivariate side-channel security evaluations

In Part I, we focused on information extraction using multivariate attacks on ECC and AES algorithms with common SCA countermeasures, that is, linear regression-based single-trace attack on ECC scalar multiplication with typical SPA/SEMA countermeasure in a worst-case scenario, and investigating the applicability of Scatter [TGWC18] attack on AES with typical jitter, shuffling/masking countermeasures in security evaluations context.

In this regard, we first presented in Chapter 3 a generic evaluation framework for HDPA [BJPW13a] against scalar multiplication algorithms, which are the core of ECC cryptosystems. Therefore, they are also com-

monly targeted in security evaluations. It is essential for security evaluators to know to which extent they can identify and extract the potential side-channel leakages from a typical protected scalar multiplication implementation, in which SPA/SEMA and scalar blinding countermeasures are present. In this chapter, through an abstract view of representative SPA/SEMA-secure constant time scalar multiplication implementations as an example, we exemplified how to efficiently identify and then further extract SCA leakage information in view of a worst-case HDPA attack scenario. We instantiated it with state-of-the-art tools for preprocessing, POI detection, profiling and extraction. Afterwards, we applied the proposed framework to concrete implementations reflective of the variety of targets likely used in embedded cryptographic applications. Given the limited experimental reports available on this topic, we hope that the implementation and systematization efforts help clarify the practicality of such advanced attacks in reality, even against challenging targets running at high clock frequencies. And therefore, it is to take the side of their integration into the current certification practices. From a designer's perspective, our results also spotlight that scalar multiplications implementations with scalar randomization on commercial platforms are generally at risk given so many informative samples such implementations provide. The straightforward solutions to improve this situation include performance optimizations (since implementations with fewer cycles inevitably leak less to the horizontal adversary given a fixed sampling rate) and adding noise (time and amplitude). Yet, our analysis of an ARM Cortex-A8 running at 1 GHz (a presumably noisy target) suggested that this may not be enough. To this end, the systematic implementation of point randomization seems strictly necessary to achieve high-security assurance levels. To better assess its worst-case security, evaluation of such implementations (with a systematic evaluation framework described in this chapter) is an engaging scope for further research work.

Furthermore, in Chapter 4, we looked into how to more efficiently extract SCA leakage information with multivariate attacks on symmetric algorithms. For security evaluations, most targeted implementations are protected with jitter, shuffling and masking countermeasures. It is a well-known challenging problem to exploit the side-channel leakages in such cases. At COSADE 2018, Thiebeauld *et al.* proposed a novel histogram-based Scatter transform to make side-channel attacks on such symmetric algorithms implementations more efficient, which could be very beneficial for security evaluations. In this chapter, we established its specificity and

compared it with a natural competitor for non-profiled multivariate side-channel analysis. It turned out that a standard on-the-fly linear regression distinguisher resulted in more efficient attacks in many practically-relevant contexts targeting the instantiations of simulated or real AES-128 implementations, including all the ones initially put forward by the Scatter authors. Therefore, it remains an interesting open question to determine whether this Scatter transform can sometimes be a helpful ingredient in side-channel security evaluations. The exhibition of a simulated case where one can observe such a gain appears as a natural next step to answer this question, and that is a clear future work direction. Without such a showcase, evaluators had to leave with the conclusion that the Scatter transform currently does not improve over existing solutions in such scenarios.

9.2 Deep learning side-channel security evaluations

Deep learning SCA tests are mandated for security evaluations and are usually very time-consuming due to the heavy computations. Therefore, improving the efficiency of deep learning SCA attacks in the security evaluation context is critical. To this end, in Part II, our focus is information extraction using deep learning attacks on ECC and AES algorithms with common SCA countermeasures. More specifically, (i) defining the impact of alignments on deep learning SCA attacks on AES implementations, (ii) using deep learning techniques to simplify the single-trace attack on ECC scalar multiplication presented in Chapter 3, (iii) addressing how to improve the efficiency of deep learning-based profiled attacks on AES algorithm implementations while keeping similar attack performance.

We first looked into the impact of alignments on deep learning SCA attacks for security evaluations. Because the iterative alignment process described in Section 2.4.2 in security evaluations usually requires much effort from the evaluators. While deep learning SCA is not sensitive to misalignments [CDP17]. In Chapter 5, our results first confirmed that neural network models are powerful kits for black box leakage analysis. Second, the results indicated the remarkable generalization of the proposed ResNet model compared with the other state-of-the-art NN models in a side-channel context. Nevertheless, even with such powerful tools, preprocessing the leakage traces with alignment/re-synchronization methods

(and possibly filtering, and so on) can be highly beneficial to the attacks/evaluations. That is true from the data complexity point of view, and data complexity is generally accounted for as the most fundamental complexity measure in side-channel analysis. But our second DUT showed that it is also true from the time complexity point of view. Indeed, alignment plays the role of data compressing due to a much shorter interval targeted. It then significantly reduces the number of samples in the traces fed to the ResNet, which may decrease its execution time from weeks to days (or even hours). The latter conclusions naturally become increasingly relevant for devices protected with different countermeasures. As stated beforehand, our judgments are specific to the investigated datasets. In particular, deep ResNets only outperform other tested models in our case study (and are not claimed to be universally better, as popularized by the no-free-lunch theorem). It leaves an open question of how to efficiently find a suitable neural network model for each SCA test dataset. And it could be an appealing future research direction. We believe our conclusion regarding the interest and sometimes need for preprocessing to be more general (since consistently observed with different models and datasets).

Meanwhile, we also considered in the asymmetric algorithms cases how to extract SCA leakage information more efficiently by utilizing deep learning techniques. In this respect, motivated by simplifying our previous systematic approach to worst-case HDPA attacks on secure scalar multiplication implementations presented in Chapter 3, we used deep learning techniques to automate the critical steps of such attacks. Our empirical results of an ARM Cortex-A8 running at 1 GHz (a presumably noisy target) suggested that employing deep learning techniques in a very black-box manner can replace the sophisticated information extraction and information combination steps. Whereas, they also urged the necessity of dimensionality reduction/detection of POIs before conducting the attack (or at the very least, our results demonstrated that such a preparatory step significantly improves the efficiency of deep learning attacks), which is a generic problem to address for attacking asymmetric cryptography implementations using deep learning techniques considering a large number of sample points per trace. To this end, as exhibited in a single-trace attack on the RSA case [CCC+19a] employing deep learning techniques, simple SNR-based POI selection could be a cheaper and faster substitution. In addition, similar to what we have pointed out in Chapter 3, for the developers of scalar multiplication implementations, our results implied that scalar implementations with scalar blinding countermeasures are commonly endangered,

taking into account both the huge amount of informative samples such implementations offer and the simplicity deep learning techniques provide. Point randomization countermeasure is likely the most suitable and easy-to-deploy solution. Evaluating that kind of implementation is a fascinating track for further investigations. Another potential research direction would be targeting register manipulations of point addition and point doubling operations from the elliptic curve arithmetic level. Eventually, a straightforward while interesting future work would be the comparison between FCN and traditional CNN in this context.

We shifted the focus back to symmetric algorithms again in Chapter 7, but we paid attention to handling how to perform deep learning-based profiled attacks on symmetric algorithms more efficiently while retaining similar attack performance. During the security evaluations of symmetric algorithm implementations, the evaluators usually have to repeat profiling on each S-box to attack them individually. Since the certification bodies require both classical profiled attacks (such as template attacks) and deep learning attacks nowadays, the burden is getting even heavier for evaluators. Besides, the requested number of profiling traces for some certification bodies is also increasing due to many deployed countermeasures in the evaluated implementations. Therefore, whether there exists a way to more efficiently conduct such profiling is an essential and practically motivated research question. Nonetheless, the SCA community so far mostly put its efforts into enhancing the attack performance by targeting one S-box. In this chapter, we analyzed the complementary approach of trying to facilitate the profiling (data and time) complexity thanks to S-box pooling. We conducted intensive experiments on three different AES-128 implementations with 108 different configurations. For representative unmasked software sequential S-boxes implementation and the masked software sequential S-boxes implementation, the results exhibited that using this S-box pooling approach can reduce both the required number of profiling traces and profiling time by a factor of 16 (or 14). That corresponds to an optimal gain. Moreover, for the masked hardware parallel S-boxes implementation we analyzed, we still obtained a sub-optimal gain in terms of data and time complexity. That said, we could reduce the required number of profiling traces and required profiling time by a factor of at least 8 in this practically relevant setting. We used only the first round S-box computations for simplicity, this S-box pooling can be extended to all S-box computations of all AES rounds. It is demonstrated that in some practically-relevant contexts, S-box pooling can lead to significant gains.

However, it may not be applicable if the S-boxes leak differently. We acknowledged that these results indicated that the efficient exploitation of profiling measurements, for instance, thanks to S-box pooling, could be a helpful supplement to the evaluators' toolbox. On the one hand, it is easy to adapt and integrate into existing toolchains. On the other hand, there are contexts in which this simple optimization could lead to substantial gains (recalling that concretely, a factor of 16 is highly applicable for evaluation tasks that can take days). It would be interesting to further explore this S-box pooling approach on more different symmetric algorithms implementations with various SCA countermeasures.

9.3 Partial Key Exposure

In Part III, we focused on information exploitation after extracting partial information of secret key data, more specifically, partial key exposure (PKE) attack on the CRT-RSA algorithm. Security evaluators must know the bounds of how much partial information of secret key(s) is to break the target cryptographic implementations. The community has studied PKE attacks on asymmetric cryptography algorithms for decades focusing on RSA, CRT-RSA, ECDSA, and ECDH implementations. Many PKE attacks make use of Coppersmith's method or lattice basis reduction.

In the security evaluations context, most evaluated CRT-RSA implementations are protected with additive exponent blinding countermeasures (most likely together with other countermeasures as well). And that is our target in Chapter 8, *i.e.*, to determine the bounds of PKE attacks on CRT-RSA implementations with additive exponent blinding. Many RSA-related PKE publications focused on RSA and CRT implementations without exponent blinding countermeasures. In this regard, the state-of-the-art PKE attack on CRT without exponent blinding [MNS22] can recover the complete CRT private key merely using a third MSBs or LSBs of CRT exponents given a public exponent $e \approx N^{\frac{1}{12}}$. Our work explored extending it to CRT implementations with additive exponent blinding. That is the most widely deployed exponent blinding countermeasure in real-world products. More specifically, considering partial disclosed MSBs or LSBs of additively blinded CRT exponents, an extended PKE attack has been proposed to reveal the entire CRT private key following the two-step approach introduced in [MNS22]. That is, calculating the key-dependent constant k' followed by factoring the N utilising partial MSBs or LSBs of the

blinded CRT exponents d'_p and d'_q. With the mathematical proof and time-complexity analyses, it turned out that merely a third MSBs or LSBs of blinded CRT exponents d'_p and d'_q are sufficient to disclose the entire CRT private key. More precisely, in the LSB case provided $e \approx N^{\frac{1}{12}-\gamma}$ it can reveal the key in polynomial time based on the standard Coppersmith-type heuristic assumption, as well as in the MSB case provided $e \approx N^{\frac{1}{12}-\frac{2}{3}\gamma}$ in sub-exponential (nonetheless practically achievable) time or in probabilistic polynomial time under a heuristic assumption. Under various configurations of typical key size and exponent blinding factor length, we verified the validity of the Coppersmith-type heuristic for the LSB case through intensive experiments, and also the affordability of the demanded sub-exponential time as well as the heuristic assumption for the MSB case. That said, employing a moderately efficient PC, in practice, the needed sub-exponential time was primarily in seconds and minutes, and the worst case seldom observed was less than one and a half hours. Furthermore, utilizing the practical SCA partial key leakage of a real-world SPA/DPA-resistant Montgomery Ladder CRT implementation on a 45 nm secure microcontroller with an RSA co-processor, we exemplified an application of the proposed EPKE attack in real life. As our SCA empirical results implied, this EPKE attack is tolerant of slight SCA errors of the disclosed MSBs or LSBs of the blinded CRT exponents. In a realistic scenario, those types of errors likely happen provided merely a single attack trace. The SCA results give insight into the error-tolerant potential of the EPKE attack. Meanwhile, it spotlights a probable future research direction. That is, adapting the EPKE to handle more generic SCA errors of the disclosed MSBs or LSBs, for instance, in the binary symmetric model where each retrieved exponent bit might be "reversed" with a crossover probability following a Bernoulli distribution.

One general perspective of this thesis is how to extract and exploit side-channel leakage information efficiently regarding the post-quantum cryptography algorithms. Recently, NIST announced the standardisation of the post-quantum cryptography algorithm candidates of CRYSTALS-Dilithium (digital signature scheme) [DKL+18], CRYSTALS-Kyber (key establishment scheme) [BDK+18], Falcon(digital signature scheme) [FHK+20], and SPHINCS$^+$(digital signature scheme) [BHK+19]. The side-channel evaluations of those algorithms implementations will be another challenge on the SCA evaluators' desks.

Bibliography

[AAB+15] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org. URL: https://www.tensorflow.org/.

[AGTB19] Alejandro Cabrera Aldaya, Cesar Pereida García, Luis Manuel Alvarez Tapia, and Billy Bob Brumley. Cache-timing attacks on RSA key generation. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2019(4):213–242, 2019. doi:10.13154/tches.v2019.i4.213-242.

[Aon09] Yoshinori Aono. A new lattice construction for partial key exposure attack for RSA. In Stanislaw Jarecki and Gene Tsudik, editors, *Public Key Cryptography - PKC 2009, 12th International Conference on Practice and Theory in Public Key Cryptography, Irvine, CA, USA, March 18-20, 2009. Proceedings*, volume 5443 of *Lecture Notes in Computer Science*, pages 34–53. Springer, 2009. doi:10.1007/978-3-642-00468-1_3.

[APSQ06] Cédric Archambeau, Eric Peeters, François-Xavier Standaert, and Jean-Jacques Quisquater. Template attacks in principal subspaces. In Louis Goubin and Mitsuru Matsui, editors, *Cryptographic Hardware and Embedded Systems - CHES 2006, 8th International Workshop, Yokohama, Japan, October 10-13, 2006, Proceedings*, volume 4249 of *Lecture Notes in Computer Science*, pages 1–14. Springer, 2006. doi:10.1007/11894063_1.

[Bau12] Sven Bauer. Attacking exponent blinding in RSA without CRT. In Werner Schindler and Sorin A. Huss, editors, *Constructive Side-Channel Analysis and Secure Design - Third International Workshop, COSADE 2012, Darmstadt, Germany, May 3-4, 2012. Proceedings*, volume 7275 of *Lecture Notes in Computer Science*, pages 82–88. Springer, 2012. doi:10.1007/978-3-642-29912-4_7.

[BBGV16] Arthur Beckers, Josep Balasch, Benedikt Gierlichs, and Ingrid Verbauwhede. Design and implementation of a waveform-matching based triggering system. In François-Xavier Standaert and Elisabeth Oswald, editors, *Constructive Side-Channel Analysis and Secure Design - 7th International Workshop, COSADE 2016, Graz, Austria, April 14-15, 2016, Revised Selected Papers*, volume 9689 of *Lecture Notes in Computer Science*, pages 184–198. Springer, 2016. doi:10.1007/978-3-319-43283-0_11.

[bbu22] bbuhrow. YAFU, Automated integer factorization Version: 2.0.8. Available at https://github.com/bbuhrow/yafu, 2022. URL: https://github.com/bbuhrow/yafu.

[BCH+20] Shivam Bhasin, Anupam Chattopadhyay, Annelie Heuser, Dirmanto Jap, Stjepan Picek, and Ritu Ranjan Shrivastwa. Mind the portability: A warriors guide through realistic profiled side-channel analysis. In *27th Annual Network and Distributed System Security Symposium, NDSS 2020, San Diego, California, USA, February 23-26, 2020*. The Internet Society, 2020.

[BCO04] Eric Brier, Christophe Clavier, and Francis Olivier. Correlation power analysis with a leakage model. In Marc Joye and

Jean-Jacques Quisquater, editors, *Cryptographic Hardware and Embedded Systems - CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11-13, 2004. Proceedings*, volume 3156 of *Lecture Notes in Computer Science*, pages 16–29. Springer, 2004. doi:10.1007/978-3-540-28632-5_2.

[BCP+14] Lejla Batina, Lukasz Chmielewski, Louiza Papachristodoulou, Peter Schwabe, and Michael Tunstall. Online template attacks. In Willi Meier and Debdeep Mukhopadhyay, editors, *Progress in Cryptology - INDOCRYPT 2014 - 15th International Conference on Cryptology in India, New Delhi, India, December 14-17, 2014, Proceedings*, volume 8885 of *Lecture Notes in Computer Science*, pages 21–36. Springer, 2014. doi:10.1007/978-3-319-13039-2_2.

[BDF98] Dan Boneh, Glenn Durfee, and Yair Frankel. An attack on RSA given a small fraction of the private key bits. In Kazuo Ohta and Dingyi Pei, editors, *Advances in Cryptology - ASIACRYPT '98, International Conference on the Theory and Applications of Cryptology and Information Security, Beijing, China, October 18-22, 1998, Proceedings*, volume 1514 of *Lecture Notes in Computer Science*, pages 25–34. Springer, 1998. doi:10.1007/3-540-49649-1_3.

[BDK+18] Joppe W. Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS - kyber: A cca-secure module-lattice-based KEM. In *2018 IEEE European Symposium on Security and Privacy, EuroS&P 2018, London, United Kingdom, April 24-26, 2018*, pages 353–367. IEEE, 2018. doi:10.1109/EuroSP.2018.00032.

[BGP+11] Lejla Batina, Benedikt Gierlichs, Emmanuel Prouff, Matthieu Rivain, François-Xavier Standaert, and Nicolas Veyrat-Charvillon. Mutual information analysis: a comprehensive study. *J. Cryptol.*, 24(2):269–291, 2011. doi:10.1007/s00145-010-9084-8.

[BGRV15] Josep Balasch, Benedikt Gierlichs, Oscar Reparaz, and Ingrid Verbauwhede. Dpa, bitslicing and masking at 1 ghz.

In Tim Güneysu and Helena Handschuh, editors, *Cryptographic Hardware and Embedded Systems - CHES 2015 - 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings*, volume 9293 of *Lecture Notes in Computer Science*, pages 599–619. Springer, 2015. doi:10.1007/978-3-662-48324-4_30.

[BHK+19] Daniel J. Bernstein, Andreas Hülsing, Stefan Kölbl, Ruben Niederhagen, Joost Rijneveld, and Peter Schwabe. The sphincs+ signature framework. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019*, pages 2129–2146. ACM, 2019. doi:10.1145/3319535.3363229.

[BHM+19] Olivier Bronchain, Julien M. Hendrickx, Clément Massart, Alex Olshevsky, and François-Xavier Standaert. Leakage certification revisited: Bounding model errors in side-channel security evaluations. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part I*, volume 11692 of *Lecture Notes in Computer Science*, pages 713–737. Springer, 2019. doi:10.1007/978-3-030-26948-7_25.

[BJPW13a] Aurélie Bauer, Éliane Jaulmes, Emmanuel Prouff, and Justine Wild. Horizontal and vertical side-channel attacks against secure RSA implementations. In Ed Dawson, editor, *Topics in Cryptology - CT-RSA 2013 - The Cryptographers' Track at the RSA Conference 2013, San Francisco,CA, USA, February 25-March 1, 2013. Proceedings*, volume 7779 of *Lecture Notes in Computer Science*, pages 1–17. Springer, 2013. doi:10.1007/978-3-642-36095-4_1.

[BJPW13b] Aurélie Bauer, Éliane Jaulmes, Emmanuel Prouff, and Justine Wild. Horizontal collision correlation attack on elliptic curves. In Tanja Lange, Kristin E. Lauter, and Petr Lisonek, editors, *Selected Areas in Cryptography - SAC 2013*

- *20th International Conference, Burnaby, BC, Canada, August 14-16, 2013, Revised Selected Papers*, volume 8282 of *Lecture Notes in Computer Science*, pages 553–570. Springer, 2013. doi:10.1007/978-3-662-43414-7_28.

[BLvV15] Daniel J. Bernstein, Tanja Lange, and Christine van Vredendaal. Tighter, faster, simpler side-channel security evaluations beyond computing power. *IACR Cryptol. ePrint Arch.*, page 221, 2015. URL: http://eprint.iacr.org/2015/221.

[BM03] Johannes Blömer and Alexander May. New partial key exposure attacks on RSA. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, volume 2729 of *Lecture Notes in Computer Science*, pages 27–43. Springer, 2003. doi:10.1007/978-3-540-45146-4_2.

[Bot22] Botan. Botan, a Crypto and TLS for Modern C++ library, Version: 2.19.1. Available at https://github.com/randombit/botan/blob/master/src/lib/pubkey/rsa/rsa.cpp, 2022. URL: https://github.com/randombit/botan.

[BS21] Joppe Bos and Martijn Stam, editors. *Computational Cryptography: Algorithmic Aspects of Cryptology*. London Mathematical Society Lecture Note Series. Cambridge University Press, 2021. doi:10.1017/9781108854207.

[BSI13] BSI. BSI AIS46 RSA SCA Resistance Guideline, Minimum Requirements for Evaluating Side-Channel Attack Resistance of RSA, DSA and Diffie-Hellman Key Exchange Implementations Version: 1.0. Available at https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Zertifizierung/Interpretationen/AIS_46_BSI_guidelines_SCA_RSA_V1_0_e_pdf.pdf?__blob=publicationFile&v=1, 2013. URL: https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Zertifizierung/Interpretationen/AIS_46_BSI_guidelines_SCA_RSA_V1_0_e_pdf.pdf?__blob=publicationFile&v=1.

[BvdPSY14] Naomi Benger, Joop van de Pol, Nigel P. Smart, and Yuval Yarom. "ooh aah... just a little bit" : A small amount of side

channel can go a long way. In Lejla Batina and Matthew Robshaw, editors, *Cryptographic Hardware and Embedded Systems - CHES 2014 - 16th International Workshop, Busan, South Korea, September 23-26, 2014. Proceedings*, volume 8731 of *Lecture Notes in Computer Science*, pages 75–92. Springer, 2014. doi:10.1007/978-3-662-44709-3_5.

[C+15] François Chollet et al. Keras. https://github.com/fchollet/keras, 2015.

[CCC+19a] Mathieu Carbone, Vincent Conin, Marie-Angela Cornelie, François Dassance, Guillaume Dufresne, Cécile Dumas, Emmanuel Prouff, and Alexandre Venelli. Deep learning to evaluate secure RSA implementations. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2019(2):132–161, 2019. doi:10.13154/tches.v2019.i2.132-161.

[CCC+19b] Mathieu Carbone, Vincent Conin, Marie-Angela Cornelie, François Dassance, Guillaume Dufresne, Cécile Dumas, Emmanuel Prouff, and Alexandre Venelli. Deep learning to evaluate secure RSA implementations. *IACR Cryptol. ePrint Arch.*, page 54, 2019. URL: https://eprint.iacr.org/2019/054.

[CDP17] Eleonora Cagli, Cécile Dumas, and Emmanuel Prouff. Convolutional neural networks with data augmentation against jitter-based countermeasures - profiling attacks without preprocessing. In Wieland Fischer and Naofumi Homma, editors, *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*, volume 10529 of *Lecture Notes in Computer Science*, pages 45–68. Springer, 2017. doi:10.1007/978-3-319-66787-4_3.

[CFG+10] Christophe Clavier, Benoit Feix, Georges Gagnerot, Mylène Roussellet, and Vincent Verneuil. Horizontal correlation analysis on exponentiation. In Miguel Soriano, Sihan Qing, and Javier López, editors, *Information and Communications Security - 12th International Conference, ICICS 2010, Barcelona, Spain, December 15-17, 2010. Proceedings*, volume 6476 of *Lec-*

ture Notes in Computer Science, pages 46–61. Springer, 2010. doi:10.1007/978-3-642-17650-0_5.

[CFG+12] Christophe Clavier, Benoit Feix, Georges Gagnerot, Christophe Giraud, Mylène Roussellet, and Vincent Verneuil. ROSETTA for single trace analysis. In Steven D. Galbraith and Mridul Nandi, editors, *Progress in Cryptology - INDOCRYPT 2012, 13th International Conference on Cryptology in India, Kolkata, India, December 9-12, 2012. Proceedings*, volume 7668 of *Lecture Notes in Computer Science*, pages 140–155. Springer, 2012. doi:10.1007/978-3-642-34931-7_9.

[Che14] Chien-Ning Chen. Memory address side-channel analysis on exponentiation. In Jooyoung Lee and Jongsung Kim, editors, *Information Security and Cryptology - ICISC 2014 - 17th International Conference, Seoul, Korea, December 3-5, 2014, Revised Selected Papers*, volume 8949 of *Lecture Notes in Computer Science*, pages 421–432. Springer, 2014. doi:10.1007/978-3-319-15943-0_25.

[CJ01] Christophe Clavier and Marc Joye. Universal exponentiation algorithm. In Çetin Kaya Koç, David Naccache, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2001, Third International Workshop, Paris, France, May 14-16, 2001, Proceedings*, volume 2162 of *Lecture Notes in Computer Science*, pages 300–308. Springer, 2001. doi:10.1007/3-540-44709-1_25.

[CJRR99] Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards sound approaches to counteract power-analysis attacks. In Michael J. Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 398–412. Springer, 1999. doi:10.1007/3-540-48405-1_26.

[CK10] Jean-Sébastien Coron and Ilya Kizhvatov. Analysis and improvement of the random delay countermeasure of CHES 2009. In Stefan Mangard and François-Xavier Standaert, editors, *Cryptographic Hardware and Embedded Systems, CHES*

2010, 12th International Workshop, Santa Barbara, CA, USA, August 17-20, 2010. Proceedings, volume 6225 of *Lecture Notes in Computer Science*, pages 95–109. Springer, 2010. doi:10.1007/978-3-642-15031-9_7.

[CK13] Omar Choudary and Markus G. Kuhn. Efficient template attacks. In Aurélien Francillon and Pankaj Rohatgi, editors, *Smart Card Research and Advanced Applications - 12th International Conference, CARDIS 2013, Berlin, Germany, November 27-29, 2013. Revised Selected Papers*, volume 8419 of *Lecture Notes in Computer Science*, pages 253–270. Springer, 2013. doi:10.1007/978-3-319-08302-5_17.

[CK14] Omar Choudary and Markus G. Kuhn. Template attacks on different devices. In Emmanuel Prouff, editor, *Constructive Side-Channel Analysis and Secure Design - 5th International Workshop, COSADE 2014, Paris, France, April 13-15, 2014. Revised Selected Papers*, volume 8622 of *Lecture Notes in Computer Science*, pages 179–198. Springer, 2014. doi:10.1007/978-3-319-10175-0_13.

[CK18] Marios O. Choudary and Markus G. Kuhn. Efficient, portable template attacks. *IEEE Trans. Inf. Forensics Secur.*, 13(2):490–501, 2018. doi:10.1109/TIFS.2017.2757440.

[Cop96a] Don Coppersmith. Finding a small root of a bivariate integer equation; factoring with high bits known. In Ueli M. Maurer, editor, *Advances in Cryptology - EUROCRYPT '96, International Conference on the Theory and Application of Cryptographic Techniques, Saragossa, Spain, May 12-16, 1996, Proceeding*, volume 1070 of *Lecture Notes in Computer Science*, pages 178–189. Springer, 1996. doi:10.1007/3-540-68339-9_16.

[Cop96b] Don Coppersmith. Finding a small root of a univariate modular equation. In Ueli M. Maurer, editor, *Advances in Cryptology - EUROCRYPT '96, International Conference on the Theory and Application of Cryptographic Techniques, Saragossa, Spain, May 12-16, 1996, Proceeding*, volume 1070 of *Lecture Notes in Computer Science*, pages 155–165. Springer, 1996. doi:10.1007/3-540-68339-9_14.

[Cop97] Don Coppersmith. Small solutions to polynomial equations, and low exponent RSA vulnerabilities. *J. Cryptol.*, 10(4):233–260, 1997. doi:10.1007/s001459900030.

[Cor99] Jean-Sébastien Coron. Resistance against differential power analysis for elliptic curve cryptosystems. In Çetin Kaya Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems, First International Workshop, CHES'99, Worcester, MA, USA, August 12-13, 1999, Proceedings*, volume 1717 of *Lecture Notes in Computer Science*, pages 292–302. Springer, 1999. doi:10.1007/3-540-48059-5_25.

[CRR02] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. Template attacks. In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*, volume 2523 of *Lecture Notes in Computer Science*, pages 13–28. Springer, 2002. doi:10.1007/3-540-36400-5_3.

[CZH19] Han Cai, Ligeng Zhu, and Song Han. ProxylessNAS: Direct neural architecture search on target task and hardware. In *International Conference on Learning Representations*, 2019. URL: https://arxiv.org/pdf/1812.00332.pdf.

[DDFP20] Gabriel Destouet, Cécile Dumas, Anne Frassati, and Valérie Perrier. Wavelet scattering transform and ensemble methods for side-channel analysis. In Guido Marco Bertoni and Francesco Regazzoni, editors, *Constructive Side-Channel Analysis and Secure Design - 11th International Workshop, COSADE 2020, Lugano, Switzerland, April 1-3, 2020, Revised Selected Papers*, volume 12244 of *Lecture Notes in Computer Science*, pages 71–89. Springer, 2020. doi:10.1007/978-3-030-68773-1_4.

[DE04] Persi Diaconis and Paul Erdös. On the distribution of the greatest common divisor. In *A festschrift for Herman Rubin*, pages 56–61. Institute of Mathematical Statistics, 2004.

[DGH+16] Jean-Luc Danger, Sylvain Guilley, Philippe Hoogvorst, Cédric Murdica, and David Naccache. Improving the big

mac attack on elliptic curve cryptography. In Peter Y. A. Ryan, David Naccache, and Jean-Jacques Quisquater, editors, *The New Codebreakers - Essays Dedicated to David Kahn on the Occasion of His 85th Birthday*, volume 9100 of *Lecture Notes in Computer Science*, pages 374–386. Springer, 2016. doi:10.1007/978-3-662-49301-4_23.

[DKL+18] Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crystals-dilithium: A lattice-based digital signature scheme. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(1):238–268, 2018. doi:10.13154/tches.v2018.i1.238-268.

[DLO+19] Ibrahima Diop, Yanis Linge, Thomas Ordas, Pierre-Yvan Liardet, and Philippe Maurine. From theory to practice: horizontal attacks on protected implementations of modular exponentiations. *J. Cryptogr. Eng.*, 9(1):37–52, 2019. doi:10.1007/s13389-018-0181-1.

[DPN+16] Margaux Dugardin, Louiza Papachristodoulou, Zakaria Najm, Lejla Batina, Jean-Luc Danger, and Sylvain Guilley. Dismantling real-world ECC with horizontal and vertical template attacks. In François-Xavier Standaert and Elisabeth Oswald, editors, *Constructive Side-Channel Analysis and Secure Design - 7th International Workshop, COSADE 2016, Graz, Austria, April 14-15, 2016, Revised Selected Papers*, volume 9689 of *Lecture Notes in Computer Science*, pages 88–108. Springer, 2016. doi:10.1007/978-3-319-43283-0_6.

[DPRS11] Julien Doget, Emmanuel Prouff, Matthieu Rivain, and François-Xavier Standaert. Univariate side channel attacks and leakage modeling. *J. Cryptogr. Eng.*, 1(2):123–144, 2011. doi:10.1007/s13389-011-0010-2.

[DS16] François Durvaux and François-Xavier Standaert. From improved leakage detection to the detection of points of interests in leakage traces. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part I*, volume 9665 of *Lecture Notes*

in Computer Science, pages 240–262. Springer, 2016. doi:10.1007/978-3-662-49890-3_10.

[DSG21] Margaux Dugardin, Werner Schindler, and Sylvain Guilley. Stochastic methods defeat regular RSA exponentiation algorithms with combined blinding methods. *J. Math. Cryptol.*, 15(1):408–433, 2021. doi:10.1515/jmc-2020-0010.

[DSP17] François Durvaux, François-Xavier Standaert, and Santos Merino Del Pozo. Towards easy leakage certification: extended version. *J. Cryptogr. Eng.*, 7(2):129–147, 2017. doi:10.1007/s13389-017-0150-0.

[DSV14] François Durvaux, François-Xavier Standaert, and Nicolas Veyrat-Charvillon. How to certify the leakage of a chip? In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, volume 8441 of *Lecture Notes in Computer Science*, pages 459–476. Springer, 2014. doi:10.1007/978-3-642-55220-5_26.

[dt16] The FPLLL development team. fplll, a lattice reduction library. Available at https://github.com/fplll/fplll, 2016. URL: https://github.com/fplll/fplll.

[EG12] M. Abdelaziz Elaabid and Sylvain Guilley. Portability of templates. *J. Cryptogr. Eng.*, 2(1):63–74, 2012. doi:10.1007/s13389-012-0030-6.

[EJMdW05] Matthias Ernst, Ellen Jochemsz, Alexander May, and Benne de Weger. Partial key exposure attacks on RSA up to full size exponents. In Ronald Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, volume 3494 of *Lecture Notes in Computer Science*, pages 371–386. Springer, 2005. doi:10.1007/11426639_22.

[FHK+20] Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Prest, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang.

Falcon. technical report, national institute of standards and technology, 2020. 2020. URL: Availableathttps://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions.

[FKM+06] Pierre-Alain Fouque, Sébastien Kunz-Jacques, Gwenaëlle Martinet, Frédéric Muller, and Frédéric Valette. Power attack on small RSA public exponent. In Louis Goubin and Mitsuru Matsui, editors, *Cryptographic Hardware and Embedded Systems - CHES 2006, 8th International Workshop, Yokohama, Japan, October 10-13, 2006, Proceedings*, volume 4249 of *Lecture Notes in Computer Science*, pages 339–353. Springer, 2006. doi:10.1007/11894063_27.

[FRV14] Benoit Feix, Mylène Roussellet, and Alexandre Venelli. Side-channel analysis on blinded regular scalar multiplications. In Willi Meier and Debdeep Mukhopadhyay, editors, *Progress in Cryptology - INDOCRYPT 2014 - 15th International Conference on Cryptology in India, New Delhi, India, December 14-17, 2014, Proceedings*, volume 8885 of *Lecture Notes in Computer Science*, pages 3–20. Springer, 2014. doi:10.1007/978-3-319-13039-2_1.

[GB10] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and D. Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010, Chia Laguna Resort, Sardinia, Italy, May 13-15, 2010*, volume 9 of *JMLR Proceedings*, pages 249–256. JMLR.org, 2010. URL: http://www.jmlr.org/proceedings/papers/v9/glorot10a.html.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[GBTP08] Benedikt Gierlichs, Lejla Batina, Pim Tuyls, and Bart Preneel. Mutual information analysis. In Elisabeth Oswald and Pankaj Rohatgi, editors, *Cryptographic Hardware and Embedded Systems - CHES 2008, 10th International Workshop, Wash-*

ington, D.C., USA, August 10-13, 2008. Proceedings, volume 5154 of *Lecture Notes in Computer Science*, pages 426–442. Springer, 2008. doi:10.1007/978-3-540-85053-3_27.

[GGP+15] Cezary Glowacz, Vincent Grosso, Romain Poussier, Joachim Schüth, and François-Xavier Standaert. Simpler and more efficient rank estimation for side-channel security assessment. In Gregor Leander, editor, *Fast Software Encryption - 22nd International Workshop, FSE 2015, Istanbul, Turkey, March 8-11, 2015, Revised Selected Papers*, volume 9054 of *Lecture Notes in Computer Science*, pages 117–129. Springer, 2015. doi:10.1007/978-3-662-48116-5_6.

[GLP06] Benedikt Gierlichs, Kerstin Lemke-Rust, and Christof Paar. Templates vs. stochastic methods. In Louis Goubin and Mitsuru Matsui, editors, *Cryptographic Hardware and Embedded Systems - CHES 2006, 8th International Workshop, Yokohama, Japan, October 10-13, 2006, Proceedings*, volume 4249 of *Lecture Notes in Computer Science*, pages 15–29. Springer, 2006. doi:10.1007/11894063_2.

[GP99] Louis Goubin and Jacques Patarin. DES and differential power analysis (the "duplication" method). In Çetin Kaya Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems, First International Workshop, CHES'99, Worcester, MA, USA, August 12-13, 1999, Proceedings*, volume 1717 of *Lecture Notes in Computer Science*, pages 158–172. Springer, 1999. doi:10.1007/3-540-48059-5_15.

[GST14] Daniel Genkin, Adi Shamir, and Eran Tromer. RSA key extraction via low-bandwidth acoustic cryptanalysis. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 444–461. Springer, 2014. doi:10.1007/978-3-662-44371-2_25.

[Guy97] Isabelle Guyon. A scaling law for the validation-set training-set size ratio. In *AT & T Bell Laboratories*, 1997.

[HGG18] Benjamin Hettwer, Stefan Gehrer, and Tim Güneysu. Profiled power analysis attacks using convolutional neural networks with domain knowledge. In Carlos Cid and Michael J. Jacobson Jr., editors, *Selected Areas in Cryptography - SAC 2018 - 25th International Conference, Calgary, AB, Canada, August 15-17, 2018, Revised Selected Papers*, volume 11349 of *Lecture Notes in Computer Science*, pages 479–498. Springer, 2018. doi:10.1007/978-3-030-10970-7_22.

[HGM+11] Gabriel Hospodar, Benedikt Gierlichs, Elke De Mulder, Ingrid Verbauwhede, and Joos Vandewalle. Machine learning in side-channel analysis: a first study. *J. Cryptogr. Eng.*, 1(4):293–302, 2011. doi:10.1007/s13389-011-0023-x.

[HKT15] Neil Hanley, HeeSeok Kim, and Michael Tunstall. Exploiting collisions in addition chain-based exponentiation algorithms using a single trace. In Kaisa Nyberg, editor, *Topics in Cryptology - CT-RSA 2015, The Cryptographer's Track at the RSA Conference 2015, San Francisco, CA, USA, April 20-24, 2015. Proceedings*, volume 9048 of *Lecture Notes in Computer Science*, pages 431–448. Springer, 2015. doi:10.1007/978-3-319-16715-2_23.

[Hla09] Martin Hlavác. Known-plaintext-only attack on RSA-CRT with montgomery multiplication. In Christophe Clavier and Kris Gaj, editors, *Cryptographic Hardware and Embedded Systems - CHES 2009, 11th International Workshop, Lausanne, Switzerland, September 6-9, 2009, Proceedings*, volume 5747 of *Lecture Notes in Computer Science*, pages 128–140. Springer, 2009. doi:10.1007/978-3-642-04138-9_10.

[HMH+12] Johann Heyszl, Stefan Mangard, Benedikt Heinz, Frederic Stumpf, and Georg Sigl. Localized electromagnetic analysis of cryptographic implementations. In Orr Dunkelman, editor, *Topics in Cryptology - CT-RSA 2012 - The Cryptographers' Track at the RSA Conference 2012, San Francisco, CA, USA, February 27 - March 2, 2012. Proceedings*, volume 7178 of *Lecture Notes in Computer Science*, pages 231–244. Springer, 2012. doi:10.1007/978-3-642-27954-6_15.

[HOM06] Christoph Herbst, Elisabeth Oswald, and Stefan Mangard. An AES smart card implementation resistant to power analysis attacks. In Jianying Zhou, Moti Yung, and Feng Bao, editors, *Applied Cryptography and Network Security, 4th International Conference, ACNS 2006, Singapore, June 6-9, 2006, Proceedings*, volume 3989 of *Lecture Notes in Computer Science*, pages 239–252, 2006. doi:10.1007/11767480_16.

[How97] Nick Howgrave-Graham. Finding small roots of univariate modular equations revisited. In Michael Darnell, editor, *Cryptography and Coding, 6th IMA International Conference, Cirencester, UK, December 17-19, 1997, Proceedings*, volume 1355 of *Lecture Notes in Computer Science*, pages 131–142. Springer, 1997. doi:10.1007/BFb0024458.

[How01] Nick Howgrave-Graham. Approximate integer common divisors. In Joseph H. Silverman, editor, *Cryptography and Lattices, International Conference, CaLC 2001, Providence, RI, USA, March 29-30, 2001, Revised Papers*, volume 2146 of *Lecture Notes in Computer Science*, pages 51–66. Springer, 2001. doi:10.1007/3-540-44670-2_6.

[HRG14] Annelie Heuser, Olivier Rioul, and Sylvain Guilley. Good is not good enough - deriving optimal distinguishers from communication theory. In Lejla Batina and Matthew Robshaw, editors, *Cryptographic Hardware and Embedded Systems - CHES 2014 - 16th International Workshop, Busan, South Korea, September 23-26, 2014. Proceedings*, volume 8731 of *Lecture Notes in Computer Science*, pages 55–74. Springer, 2014. doi:10.1007/978-3-662-44709-3_4.

[HS09] Nadia Heninger and Hovav Shacham. Reconstructing RSA private keys from random key bits. In Shai Halevi, editor, *Advances in Cryptology - CRYPTO 2009, 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings*, volume 5677 of *Lecture Notes in Computer Science*, pages 1–17. Springer, 2009. doi:10.1007/978-3-642-03356-8_1.

[HSH+08] J. Alex Halderman, Seth D. Schoen, Nadia Heninger, William Clarkson, William Paul, Joseph A. Calandrino, Ariel J. Feld-

man, Jacob Appelbaum, and Edward W. Felten. Lest we remember: Cold boot attacks on encryption keys. In Paul C. van Oorschot, editor, *Proceedings of the 17th USENIX Security Symposium, July 28-August 1, 2008, San Jose, CA, USA*, pages 45–60. USENIX Association, 2008. URL: http://www.usenix.org/events/sec08/tech/full_papers/halderman/halderman.pdf.

[HWW21] Fanliang Hu, Huanyu Wang, and Junnian Wang. Cross-subkey deep-learning side-channel analysis. *IACR Cryptol. ePrint Arch.*, page 1328, 2021. URL: https://eprint.iacr.org/2021/1328.

[HZ12] Annelie Heuser and Michael Zohner. Intelligent machine homicide - breaking cryptographic devices using support vector machines. In Werner Schindler and Sorin A. Huss, editors, *Constructive Side-Channel Analysis and Secure Design - Third International Workshop, COSADE 2012, Darmstadt, Germany, May 3-4, 2012. Proceedings*, volume 7275 of *Lecture Notes in Computer Science*, pages 249–264. Springer, 2012. doi:10.1007/978-3-642-29912-4_18.

[HZRS15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL: http://arxiv.org/abs/1512.03385, arXiv:1512.03385.

[HZRS16a] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society, 2016. doi:10.1109/CVPR.2016.90.

[HZRS16b] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. *CoRR*, abs/1603.05027, 2016. URL: http://arxiv.org/abs/1603.05027, arXiv:1603.05027.

[IS15] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis R. Bach and David M. Blei, edi-

tors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 448–456. JMLR.org, 2015. URL: http://proceedings.mlr.press/v37/ioffe15.html.

[Jea16] Jérémy Jean. TikZ for Cryptographers. https://www.iacr.org/authors/tikz/, 2016.

[JL12] Marc Joye and Tancrède Lepoint. Partial key exposure on RSA with private exponents larger than N. In Mark Dermot Ryan, Ben Smyth, and Guilin Wang, editors, *Information Security Practice and Experience - 8th International Conference, ISPEC 2012, Hangzhou, China, April 9-12, 2012. Proceedings*, volume 7232 of *Lecture Notes in Computer Science*, pages 369–380. Springer, 2012. doi:10.1007/978-3-642-29101-2_25.

[JT01] Marc Joye and Christophe Tymen. Protections against differential analysis for elliptic curve cryptography. In Çetin Kaya Koç, David Naccache, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2001, Third International Workshop, Paris, France, May 14-16, 2001, Proceedings*, volume 2162 of *Lecture Notes in Computer Science*, pages 377–390. Springer, 2001. doi:10.1007/3-540-44709-1_31.

[JY02] Marc Joye and Sung-Ming Yen. The montgomery powering ladder. In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*, volume 2523 of *Lecture Notes in Computer Science*, pages 291–302. Springer, 2002. doi:10.1007/3-540-36400-5_22.

[KB07] Boris Köpf and David A. Basin. An information-theoretic model for adaptive side-channel attacks. In Peng Ning, Sabrina De Capitani di Vimercati, and Paul F. Syverson, editors, *Proceedings of the 2007 ACM Conference on Computer and Communications Security, CCS 2007, Alexandria, Virginia, USA, October 28-31, 2007*, pages 286–296. ACM, 2007. doi:10.1145/1315245.1315282.

[Kel60] Henry J Kelley. Gradient theory of optimal flight paths. *Ars Journal*, 30(10):947–954, 1960.

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael J. Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999. doi:10.1007/3-540-48405-1_25.

[KJJR11] Paul C. Kocher, Joshua Jaffe, Benjamin Jun, and Pankaj Rohatgi. Introduction to differential power analysis. *J. Cryptographic Engineering*, 1(1):5–27, 2011. URL: http://dx.doi.org/10.1007/s13389-011-0006-y, doi:10.1007/s13389-011-0006-y.

[Kob87] Neal Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48:203–209, 1987.

[Koc96] Paul C. Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In Neal Koblitz, editor, *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer, 1996. doi:10.1007/3-540-68697-5_9.

[LBM15] Liran Lerman, Gianluca Bontempi, and Olivier Markowitch. A machine learning approach against a masked AES - reaching the limit of side-channel attacks with a learning model. *J. Cryptogr. Eng.*, 5(2):123–139, 2015. doi:10.1007/s13389-014-0089-3.

[Lib21] Libgcrypt. Libgcrypt, the gnu crypto library, Version: 1.9.0. Available at https://github.com/gpg/libgcrypt/blob/master/cipher/rsa.c, 2021. URL: https://github.com/gpg/libgcrypt.

[LLL82] A.K. Lenstra, H.W.jun. Lenstra, and László Lovász. Factoring polynomials with rational coefficients. *Math. Ann.*, 261:515–534, 1982.

[LMPT15] Jake Longo, Elke De Mulder, Dan Page, and Michael Tunstall. Soc it to EM: electromagnetic side-channel attacks on a complex system-on-chip. In Tim Güneysu and Helena Handschuh, editors, *Cryptographic Hardware and Embedded Systems - CHES 2015 - 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings*, volume 9293 of *Lecture Notes in Computer Science*, pages 620–640. Springer, 2015. doi:10.1007/978-3-662-48324-4_31.

[LPB+15] Liran Lerman, Romain Poussier, Gianluca Bontempi, Olivier Markowitch, and François-Xavier Standaert. Template attacks vs. machine learning revisited (and the curse of dimensionality in side-channel analysis). In Stefan Mangard and Axel Y. Poschmann, editors, *Constructive Side-Channel Analysis and Secure Design - 6th International Workshop, COSADE 2015, Berlin, Germany, April 13-14, 2015. Revised Selected Papers*, volume 9064 of *Lecture Notes in Computer Science*, pages 20–33. Springer, 2015. doi:10.1007/978-3-319-21476-4_2.

[LSD15] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 3431–3440. IEEE Computer Society, 2015. doi:10.1109/CVPR.2015.7298965.

[LvVW14] Tanja Lange, Christine van Vredendaal, and Marnix Wakker. Kangaroos in side-channel attacks. In Marc Joye and Amir Moradi, editors, *Smart Card Research and Advanced Applications - 13th International Conference, CARDIS 2014, Paris, France, November 5-7, 2014. Revised Selected Papers*, volume 8968 of *Lecture Notes in Computer Science*, pages 104–121. Springer, 2014. doi:10.1007/978-3-319-16763-3_7.

[LZL14] Yao Lu, Rui Zhang, and Dongdai Lin. New partial key exposure attacks on CRT-RSA with large public exponents. In Ioana Boureanu, Philippe Owesarski, and Serge Vaudenay, editors, *Applied Cryptography and Network Security - 12th International Conference, ACNS 2014, Lausanne, Switzerland,*

June 10-13, 2014. Proceedings, volume 8479 of *Lecture Notes in Computer Science*, pages 151–162. Springer, 2014. doi:10.1007/978-3-319-07536-5_10.

[Mag20] Houssem Maghrebi. Deep learning based side-channel attack: a new profiling methodology based on multi-label classification. *IACR Cryptol. ePrint Arch.*, 2020:436, 2020. URL: https://eprint.iacr.org/2020/436.

[Man04] Stefan Mangard. Hardware countermeasures against DPA ? A statistical analysis of their effectiveness. In Tatsuaki Okamoto, editor, *Topics in Cryptology - CT-RSA 2004, The Cryptographers' Track at the RSA Conference 2004, San Francisco, CA, USA, February 23-27, 2004, Proceedings*, volume 2964 of *Lecture Notes in Computer Science*, pages 222–235. Springer, 2004. doi:10.1007/978-3-540-24660-2_18.

[Mbe21] MbedTLS. MbedTLS, a TLS and SSL library, Version: 3.1.0. Available at https://github.com/Mbed-TLS/mbedtls/blob/development/library/rsa.c, 2021. URL: https://github.com/Mbed-TLS/mbedtls.

[MDP20] Loïc Masure, Cécile Dumas, and Emmanuel Prouff. A comprehensive study of deep learning for side-channel analysis. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(1):348–375, 2020. doi:10.13154/tches.v2020.i1.348-375.

[MDS99] Thomas S. Messerges, Ezzy A. Dabbish, and Robert H. Sloan. Power analysis attacks of modular exponentiation in smartcards. In Çetin Kaya Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems, First International Workshop, CHES'99, Worcester, MA, USA, August 12-13, 1999, Proceedings*, volume 1717 of *Lecture Notes in Computer Science*, pages 144–157. Springer, 1999. doi:10.1007/3-540-48059-5_14.

[Mil85] Victor S. Miller. Use of elliptic curves in cryptography. In Hugh C. Williams, editor, *Advances in Cryptology - CRYPTO '85, Santa Barbara, California, USA, August 18-22, 1985, Proceedings*, volume 218 of *Lecture Notes in Computer Science*, pages 417–426. Springer, 1985. doi:10.1007/3-540-39799-X_31.

[MNS21] Alexander May, Julian Nowakowski, and Santanu Sarkar. Partial key exposure attack on short secret exponent CRT-RSA. In Mehdi Tibouchi and Huaxiong Wang, editors, *Advances in Cryptology - ASIACRYPT 2021 - 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6-10, 2021, Proceedings, Part I*, volume 13090 of *Lecture Notes in Computer Science*, pages 99–129. Springer, 2021. doi:10.1007/978-3-030-92062-3_4.

[MNS22] Alexander May, Julian Nowakowski, and Santanu Sarkar. Approximate divisor multiples - factoring with only a third of the secret crt-exponents. In Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology - EUROCRYPT 2022 - 41st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Trondheim, Norway, May 30 - June 3, 2022, Proceedings, Part III*, volume 13277 of *Lecture Notes in Computer Science*, pages 147–167. Springer, 2022. doi:10.1007/978-3-031-07082-2_6.

[MO08] Marcel Medwed and Elisabeth Oswald. Template attacks on ECDSA. In Kyo-Il Chung, Kiwook Sohn, and Moti Yung, editors, *Information Security Applications, 9th International Workshop, WISA 2008, Jeju Island, Korea, September 23-25, 2008, Revised Selected Papers*, volume 5379 of *Lecture Notes in Computer Science*, pages 14–27. Springer, 2008. doi:10.1007/978-3-642-00306-6_2.

[MOOS15] Daniel P. Martin, Jonathan F. O'Connell, Elisabeth Oswald, and Martijn Stam. Counting keys in parallel after a side channel attack. In Tetsu Iwata and Jung Hee Cheon, editors, *Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part II*, volume 9453 of *Lecture Notes in Computer Science*, pages 313–337. Springer, 2015. doi:10.1007/978-3-662-48800-3_13.

[MOP07] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power analysis attacks - revealing the secrets of smart cards*. Springer, 2007.

[MPP16] Houssem Maghrebi, Thibault Portigliatti, and Emmanuel Prouff. Breaking cryptographic implementations using deep learning techniques. In Claude Carlet, M. Anwar Hasan, and Vishal Saraswat, editors, *Security, Privacy, and Applied Cryptography Engineering - 6th International Conference, SPACE 2016, Hyderabad, India, December 14-18, 2016, Proceedings*, volume 10076 of *Lecture Notes in Computer Science*, pages 3–26. Springer, 2016. doi:10.1007/978-3-319-49445-6_1.

[MRSS18] Amir Moradi, Bastian Richter, Tobias Schneider, and François-Xavier Standaert. Leakage detection with the x2-test. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(1):209–237, 2018. doi:10.13154/tches.v2018.i1.209-237.

[NCOS16] Erick Nascimento, Lukasz Chmielewski, David F. Oswald, and Peter Schwabe. Attacking embedded ECC implementations through cmov side channels. *IACR Cryptol. ePrint Arch.*, page 923, 2016. URL: http://eprint.iacr.org/2016/923.

[NH10] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In Johannes Fürnkranz and Thorsten Joachims, editors, *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*, pages 807–814. Omnipress, 2010. URL: https://icml.cc/Conferences/2010/papers/432.pdf.

[NS03] Phong Q. Nguyen and Igor E. Shparlinski. The insecurity of the elliptic curve digital signature algorithm with partially known nonces. *Des. Codes Cryptography*, 30(2):201–217, 2003. URL: http://dx.doi.org/10.1023/A:1025436905711, doi:10.1023/A:1025436905711.

[OC14] Colin O'Flynn and Zhizhang (David) Chen. Chipwhisperer: An open-source platform for hardware embedded security research. In Emmanuel Prouff, editor, *Constructive Side-Channel Analysis and Secure Design - 5th International Workshop, COSADE 2014, Paris, France, April 13-15, 2014. Revised Selected Papers*, volume 8622 of *Lecture Notes in Com-*

puter Science, pages 243–260. Springer, 2014. doi:10.1007/978-3-319-10175-0_17.

[OP11] David F. Oswald and Christof Paar. Breaking mifare desfire MF3ICD40: power analysis and templates in the real world. In Bart Preneel and Tsuyoshi Takagi, editors, *Cryptographic Hardware and Embedded Systems - CHES 2011 - 13th International Workshop, Nara, Japan, September 28 - October 1, 2011. Proceedings*, volume 6917 of *Lecture Notes in Computer Science*, pages 207–222. Springer, 2011. doi:10.1007/978-3-642-23951-9_14.

[PC15] Guilherme Perin and Lukasz Chmielewski. A semi-parametric approach for side-channel attacks on protected RSA implementations. In Naofumi Homma and Marcel Medwed, editors, *Smart Card Research and Advanced Applications - 14th International Conference, CARDIS 2015, Bochum, Germany, November 4-6, 2015. Revised Selected Papers*, volume 9514 of *Lecture Notes in Computer Science*, pages 34–53. Springer, 2015. doi:10.1007/978-3-319-31271-2_3.

[PGM+19] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 8024–8035, 2019. URL: https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html.

[PGZ+18] Hieu Pham, Melody Y. Guan, Barret Zoph, Quoc V. Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. In Jennifer G. Dy and Andreas Krause, ed-

itors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 4092–4101. PMLR, 2018. URL: http://proceedings.mlr.press/v80/pham18a.html.

[PH18] Christophe Pfeifer and Patrick Haddad. Spread: a new layer for profiled deep-learning side-channel attacks. *IACR Cryptol. ePrint Arch.*, page 880, 2018. URL: https://eprint.iacr.org/2018/880.

[PHJ+19] Stjepan Picek, Annelie Heuser, Alan Jovic, Shivam Bhasin, and Francesco Regazzoni. The curse of class imbalance and conflicting metrics with machine learning for side-channel evaluations. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2019(1):209–237, 2019. doi:10.13154/tches.v2019.i1.209-237.

[PHJL17] Stjepan Picek, Annelie Heuser, Alan Jovic, and Axel Legay. Climbing down the hierarchy: Hierarchical classification for machine learning side-channel attacks. In Marc Joye and Abderrahmane Nitaj, editors, *Progress in Cryptology - AFRICACRYPT 2017 - 9th International Conference on Cryptology in Africa, Dakar, Senegal, May 24-26, 2017, Proceedings*, volume 10239 of *Lecture Notes in Computer Science*, pages 61–78, 2017. doi:10.1007/978-3-319-57339-7_4.

[PRB09] Emmanuel Prouff, Matthieu Rivain, and Régis Bevan. Statistical analysis of second order differential power analysis. *IEEE Trans. Computers*, 58(6):799–811, 2009. doi:10.1109/TC.2009.15.

[PSB+18] Emmanuel Prouff, Rémi Strullu, Ryad Benadjila, Eleonora Cagli, and Cécile Dumas. Study of deep learning techniques for side-channel analysis and introduction to ASCAD database. *IACR Cryptol. ePrint Arch.*, page 53, 2018. URL: http://eprint.iacr.org/2018/053.

[PSG16] Romain Poussier, François-Xavier Standaert, and Vincent Grosso. Simple key enumeration (and rank estimation) using histograms: An integrated approach. In Benedikt

Gierlichs and Axel Y. Poschmann, editors, *Cryptographic Hardware and Embedded Systems - CHES 2016 - 18th International Conference, Santa Barbara, CA, USA, August 17-19, 2016, Proceedings*, volume 9813 of *Lecture Notes in Computer Science*, pages 61–81. Springer, 2016. doi:10.1007/978-3-662-53140-2_4.

[PSK+18] Stjepan Picek, Ioannis Petros Samiotis, Jaehun Kim, Annelie Heuser, Shivam Bhasin, and Axel Legay. On the performance of convolutional neural networks for side-channel analysis. In Anupam Chattopadhyay, Chester Rebeiro, and Yuval Yarom, editors, *Security, Privacy, and Applied Cryptography Engineering - 8th International Conference, SPACE 2018, Kanpur, India, December 15-19, 2018, Proceedings*, volume 11348 of *Lecture Notes in Computer Science*, pages 157–176. Springer, 2018. doi:10.1007/978-3-030-05072-6_10.

[PUB00] NIST FIPS PUB. 186-2: Digital signature standard (dss). *National Institute for Standards and Technology*, 2000.

[PVG+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[PZS17] Romain Poussier, Yuanyuan Zhou, and François-Xavier Standaert. A systematic approach to the side-channel analysis of ECC implementations with worst-case horizontal attacks. In Wieland Fischer and Naofumi Homma, editors, *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*, volume 10529 of *Lecture Notes in Computer Science*, pages 534–554. Springer, 2017. doi:10.1007/978-3-319-66787-4_26.

[QS01] Jean-Jacques Quisquater and David Samyde. Electromagnetic analysis (EMA): measures and counter-measures for smart cards. In Isabelle Attali and Thomas P. Jensen, editors,

Smart Card Programming and Security, International Conference on Research in Smart Cards, E-smart 2001, Cannes, France, September 19-21, 2001, Proceedings, volume 2140 of *Lecture Notes in Computer Science*, pages 200–210. Springer, 2001. doi:10.1007/3-540-45418-7_17.

[Riv08] Matthieu Rivain. On the exact success rate of side channel analysis in the gaussian model. In Roberto Maria Avanzi, Liam Keliher, and Francesco Sica, editors, *Selected Areas in Cryptography, 15th International Workshop, SAC 2008, Sackville, New Brunswick, Canada, August 14-15, Revised Selected Papers*, volume 5381 of *Lecture Notes in Computer Science*, pages 165–183. Springer, 2008. doi:10.1007/978-3-642-04159-4_11.

[RQL19] Pieter Robyns, Peter Quax, and Wim Lamotte. Improving CEMA using correlation optimization. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2019(1):1–24, 2019. doi:10.13154/tches.v2019.i1.1-24.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978. URL: http://doi.acm.org/10.1145/359340.359342, doi:10.1145/359340.359342.

[RSV+11] Mathieu Renauld, François-Xavier Standaert, Nicolas Veyrat-Charvillon, Dina Kamel, and Denis Flandre. A formal study of power variability issues and side-channel attacks for nanoscale devices. In Kenneth G. Paterson, editor, *Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings*, volume 6632 of *Lecture Notes in Computer Science*, pages 109–128. Springer, 2011. doi:10.1007/978-3-642-20465-4_8.

[Sch00] Werner Schindler. A timing attack against RSA with the chinese remainder theorem. In Çetin Kaya Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2000, Second International Workshop, Worcester, MA,*

USA, August 17-18, 2000, Proceedings, volume 1965 of *Lecture Notes in Computer Science*, pages 109–124. Springer, 2000. doi:10.1007/3-540-44499-8_8.

[SGV08] François-Xavier Standaert, Benedikt Gierlichs, and Ingrid Verbauwhede. Partition vs. comparison side-channel distinguishers: An empirical evaluation of statistical tests for univariate side-channel attacks against two unprotected CMOS devices. In Pil Joong Lee and Jung Hee Cheon, editors, *Information Security and Cryptology - ICISC 2008, 11th International Conference, Seoul, Korea, December 3-5, 2008, Revised Selected Papers*, volume 5461 of *Lecture Notes in Computer Science*, pages 253–267. Springer, 2008. doi:10.1007/978-3-642-00730-9_16.

[Sha49] Claude E. Shannon. Communication theory of secrecy systems. *Bell Syst. Tech. J.*, 28(4):656–715, 1949. doi:10.1002/j.1538-7305.1949.tb00928.x.

[SHK+14] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, 2014. URL: http://dl.acm.org/citation.cfm?id=2670313.

[SI11] Werner Schindler and Kouichi Itoh. Exponent blinding does not always lift (partial) spa resistance to higher-level security. In Javier López and Gene Tsudik, editors, *Applied Cryptography and Network Security - 9th International Conference, ACNS 2011, Nerja, Spain, June 7-10, 2011. Proceedings*, volume 6715 of *Lecture Notes in Computer Science*, pages 73–90, 2011. doi:10.1007/978-3-642-21554-4_5.

[SKS09] François-Xavier Standaert, François Koeune, and Werner Schindler. How to compare profiled side-channel attacks? In Michel Abdalla, David Pointcheval, Pierre-Alain Fouque, and Damien Vergnaud, editors, *Applied Cryptography and Network Security, 7th International Conference, ACNS 2009, Paris-Rocquencourt, France, June 2-5, 2009. Proceedings*, volume 5536 of *Lecture Notes in Computer Science*, pages 485–498, 2009. doi:10.1007/978-3-642-01957-9_30.

[SLP05] Werner Schindler, Kerstin Lemke, and Christof Paar. A stochastic model for differential side channel cryptanalysis. In Josyula R. Rao and Berk Sunar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2005, 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005, Proceedings*, volume 3659 of *Lecture Notes in Computer Science*, pages 30–46. Springer, 2005. doi:10.1007/11545262_3.

[SMY09] François-Xavier Standaert, Tal Malkin, and Moti Yung. A unified framework for the analysis of side-channel key recovery attacks. In Antoine Joux, editor, *Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings*, volume 5479 of *Lecture Notes in Computer Science*, pages 443–461. Springer, 2009. doi:10.1007/978-3-642-01001-9_26.

[SNK+12] Alexander Schlösser, Dmitry Nedospasov, Juliane Krämer, Susanna Orlic, and Jean-Pierre Seifert. Simple photonic emission analysis of AES - photonic side channel analysis for the rest of us. In Emmanuel Prouff and Patrick Schaumont, editors, *Cryptographic Hardware and Embedded Systems - CHES 2012 - 14th International Workshop, Leuven, Belgium, September 9-12, 2012. Proceedings*, volume 7428 of *Lecture Notes in Computer Science*, pages 41–57. Springer, 2012. doi:10.1007/978-3-642-33027-8_3.

[SSS+17] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of go without human knowledge. *Nature*, 550:354 EP –, Oct 2017. Article. URL: http://dx.doi.org/10.1038/nature24270.

[SV14] Santanu Sarkar and Ayineedi Venkateswarlu. Partial key exposure attack on CRT-RSA. In Willi Meier and Debdeep Mukhopadhyay, editors, *Progress in Cryptology - INDOCRYPT 2014 - 15th International Conference on Cryptology*

in India, New Delhi, India, December 14-17, 2014, Proceedings, volume 8885 of *Lecture Notes in Computer Science*, pages 255–264. Springer, 2014. doi:10.1007/978-3-319-13039-2_15.

[SVO+10] François-Xavier Standaert, Nicolas Veyrat-Charvillon, Elisabeth Oswald, Benedikt Gierlichs, Marcel Medwed, Markus Kasper, and Stefan Mangard. The world is not enough: Another look on second-order DPA. In Masayuki Abe, editor, *Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings*, volume 6477 of *Lecture Notes in Computer Science*, pages 112–129. Springer, 2010. doi:10.1007/978-3-642-17373-8_7.

[SW14] Werner Schindler and Andreas Wiemers. Power attacks in the presence of exponent blinding. *J. Cryptogr. Eng.*, 4(4):213–236, 2014. doi:10.1007/s13389-014-0081-y.

[SW17] Werner Schindler and Andreas Wiemers. Generic power attacks on RSA with CRT and exponent blinding: new results. *J. Cryptogr. Eng.*, 7(4):255–272, 2017. doi:10.1007/s13389-016-0146-1.

[SZ04] Damien Stehlé and Paul Zimmermann. A binary recursive gcd algorithm. In Duncan A. Buell, editor, *Algorithmic Number Theory, 6th International Symposium, ANTS-VI, Burlington, VT, USA, June 13-18, 2004, Proceedings*, volume 3076 of *Lecture Notes in Computer Science*, pages 411–425. Springer, 2004. doi:10.1007/978-3-540-24847-7_31.

[TGWC18] Hugues Thiebeauld, Georges Gagnerot, Antoine Wurcker, and Christophe Clavier. SCATTER: A new dimension in side-channel. In Junfeng Fan and Benedikt Gierlichs, editors, *Constructive Side-Channel Analysis and Secure Design - 9th International Workshop, COSADE 2018, Singapore, April 23-24, 2018, Proceedings*, volume 10815 of *Lecture Notes in Computer Science*, pages 135–152. Springer, 2018. doi:10.1007/978-3-319-89641-0_8.

[Tim19] Benjamin Timon. Non-profiled deep learning-based side-channel attacks with sensitivity analysis. *IACR Trans. Cryp-*

togr. Hardw. Embed. Syst., 2019(2):107–131, 2019. doi:10.13154/tches.v2019.i2.107-131.

[TK14] Atsushi Takayasu and Noboru Kunihiro. Partial key exposure attacks on RSA: achieving the boneh-durfee bound. In Antoine Joux and Amr M. Youssef, editors, *Selected Areas in Cryptography - SAC 2014 - 21st International Conference, Montreal, QC, Canada, August 14-15, 2014, Revised Selected Papers*, volume 8781 of *Lecture Notes in Computer Science*, pages 345–362. Springer, 2014. doi:10.1007/978-3-319-13051-4_21.

[TK15] Atsushi Takayasu and Noboru Kunihiro. Partial key exposure attacks on CRT-RSA: better cryptanalysis to full size encryption exponents. In Tal Malkin, Vladimir Kolesnikov, Allison Bishop Lewko, and Michalis Polychronakis, editors, *Applied Cryptography and Network Security - 13th International Conference, ACNS 2015, New York, NY, USA, June 2-5, 2015, Revised Selected Papers*, volume 9092 of *Lecture Notes in Computer Science*, pages 518–537. Springer, 2015. doi:10.1007/978-3-319-28166-7_25.

[TK19] Atsushi Takayasu and Noboru Kunihiro. Partial key exposure attacks on RSA: achieving the boneh-durfee bound. *Theor. Comput. Sci.*, 761:51–77, 2019. doi:10.1016/j.tcs.2018.08.021.

[TLP19] Atsushi Takayasu, Yao Lu, and Liqiang Peng. Small crt-exponent RSA revisited. *J. Cryptol.*, 32(4):1337–1382, 2019. doi:10.1007/s00145-018-9282-3.

[TVW19] Hugues Thiebeauld, Aurélien Vasselle, and Antoine Wurcker. Second-order scatter attack. *IACR Cryptol. ePrint Arch.*, page 345, 2019. URL: https://eprint.iacr.org/2019/345.

[vdODZ+16] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *CoRR*, abs/1609.03499, 2016. URL: http://arxiv.org/abs/1609.03499, arXiv:1609.03499.

[VGRS12] Nicolas Veyrat-Charvillon, Benoît Gérard, Mathieu Renauld, and François-Xavier Standaert. An optimal key enumeration algorithm and its application to side-channel attacks. In Lars R. Knudsen and Huapeng Wu, editors, *Selected Areas in Cryptography, 19th International Conference, SAC 2012, Windsor, ON, Canada, August 15-16, 2012, Revised Selected Papers*, volume 7707 of *Lecture Notes in Computer Science*, pages 390–406. Springer, 2012. doi:10.1007/978-3-642-35999-6_25.

[VGS13] Nicolas Veyrat-Charvillon, Benoît Gérard, and François-Xavier Standaert. Security evaluations beyond computing power. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 126–141. Springer, 2013. URL: http://dx.doi.org/10.1007/978-3-642-38348-9_8, doi:10.1007/978-3-642-38348-9_8.

[VMKS12] Nicolas Veyrat-Charvillon, Marcel Medwed, Stéphanie Kerckhof, and François-Xavier Standaert. Shuffling against side-channel attacks: A comprehensive study with cautionary note. In Xiaoyun Wang and Kazue Sako, editors, *Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings*, volume 7658 of *Lecture Notes in Computer Science*, pages 740–757. Springer, 2012. doi:10.1007/978-3-642-34961-4_44.

[Wal01] Colin D. Walter. Sliding windows succumbs to big mac attack. In Çetin Kaya Koç, David Naccache, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2001, Third International Workshop, Paris, France, May 14-16, 2001, Proceedings*, volume 2162 of *Lecture Notes in Computer Science*, pages 286–299. Springer, 2001. doi:10.1007/3-540-44709-1_24.

[WCN+17] Tian Wang, Xiaoxin Cui, Yewen Ni, Dunshan Yu, Xiaole Cui, and Gang Qu. A practical cold boot attack on RSA private keys. In *2017 Asian Hardware Oriented Security and Trust Symposium, AsianHOST 2017, Beijing, China, October 19-20, 2017*, pages 55–60. IEEE Computer Society, 2017. doi:10.1109/AsianHOST.2017.8353995.

[WdHG+20] Lennert Wouters, Jan Van den Herrewegen, Flavio D. Garcia, David F. Oswald, Benedikt Gierlichs, and Bart Preneel. Dismantling dst80-based immobiliser systems. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(2):99–127, 2020. doi:10.13154/tches.v2020.i2.99-127.

[WO15] Carolyn Whitnall and Elisabeth Oswald. Robust profiling for dpa-style attacks. In Tim Güneysu and Helena Handschuh, editors, *Cryptographic Hardware and Embedded Systems - CHES 2015 - 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings*, volume 9293 of *Lecture Notes in Computer Science*, pages 3–21. Springer, 2015. doi:10.1007/978-3-662-48324-4_1.

[WOS14] Carolyn Whitnall, Elisabeth Oswald, and François-Xavier Standaert. The myth of generic dpa...and the magic of learning. In Josh Benaloh, editor, *Topics in Cryptology - CT-RSA 2014 - The Cryptographer's Track at the RSA Conference 2014, San Francisco, CA, USA, February 25-28, 2014. Proceedings*, volume 8366 of *Lecture Notes in Computer Science*, pages 183–205. Springer, 2014. doi:10.1007/978-3-319-04852-9_10.

[WSC+16] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144, 2016. URL: http://arxiv.org/abs/1609.08144, arXiv:1609.08144.

[XDH+16] Wayne Xiong, Jasha Droppo, Xuedong Huang, Frank Seide, Mike Seltzer, Andreas Stolcke, Dong Yu, and Geoffrey Zweig. The microsoft 2016 conversational speech recognition system. *CoRR*, abs/1609.03528, 2016. URL: http://arxiv.org/abs/1609.03528, arXiv:1609.03528.

[YGH16] Yuval Yarom, Daniel Genkin, and Nadia Heninger. Cachebleed: A timing attack on openssl constant time RSA. In Benedikt Gierlichs and Axel Y. Poschmann, editors, *Cryptographic Hardware and Embedded Systems - CHES 2016 - 18th International Conference, Santa Barbara, CA, USA, August 17-19, 2016, Proceedings*, volume 9813 of *Lecture Notes in Computer Science*, pages 346–367. Springer, 2016. doi:10.1007/978-3-662-53140-2_17.

[YJ00] Sung-Ming Yen and Marc Joye. Checking before output may not be enough against fault-based cryptanalysis. *IEEE Trans. Computers*, 49(9):967–970, 2000. doi:10.1109/12.869328.

[ZBHV21] Gabriel Zaid, Lilian Bossuet, Amaury Habrard, and Alexandre Venelli. Efficiency through diversity in ensemble models applied to side-channel attacks - A case study on public-key algorithms -. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(3):60–96, 2021. doi:10.46586/tches.v2021.i3.60-96.

[ZDS20] Yuanyuan Zhou, Sébastien Duval, and François-Xavier Standaert. Scatter: a missing case? In Guido Marco Bertoni and Francesco Regazzoni, editors, *Constructive Side-Channel Analysis and Secure Design - 11th International Workshop, COSADE 2020, Lugano, Switzerland, April 1-3, 2020, Revised Selected Papers*, volume 12244 of *Lecture Notes in Computer Science*, pages 90–103. Springer, 2020. doi:10.1007/978-3-030-68773-1_5.

[Zei12] Matthew D. Zeiler. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701, 2012. URL: http://arxiv.org/abs/1212.5701, arXiv:1212.5701.

[ZS19] Yuanyuan Zhou and François-Xavier Standaert. Simplified single-trace side-channel attacks on elliptic curve scalar multiplication using fully convolutional networks. In Gilles Callebaut, Kevin Verniers, and Bert Cox, editors, *WIC Symposium on Information Theory in the Benelux - SITB 2019 - 40th WIC Symposium on Information Theory in the Benelux, Ghent, Belgium, May 28-29, 2019, Proceedings*, pages 139–146, 2019.

[ZS20] Yuanyuan Zhou and François-Xavier Standaert. Deep learning mitigates but does not annihilate the need of aligned traces and a generalized resnet model for side-channel attacks. *J. Cryptogr. Eng.*, 10(1):85–95, 2020. doi:10.1007/s13389-019-00209-3.

[ZS22] Yuanyuan Zhou and François-Xavier Standaert. S-box pooling: Towards more efficient side-channel security evaluations. In Jianying Zhou, Sridhar Adepu, Cristina Alcaraz, Lejla Batina, Emiliano Casalicchio, Sudipta Chattopadhyay, Chenglu Jin, Jingqiang Lin, Eleonora Losiouk, Suryadipta Majumdar, Weizhi Meng, Stjepan Picek, Jun Shao, Chunhua Su, Cong Wang, Yury Zhauniarovich, and Saman A. Zonouz, editors, *Applied Cryptography and Network Security Workshops - ACNS 2022 Satellite Workshops, AIBlock, AIHWS, AIoTS, CIMSS, Cloud S&P, SCI, SecMT, SiMLA, Rome, Italy, June 20-23, 2022, Proceedings*, volume 13285 of *Lecture Notes in Computer Science*, pages 146–164. Springer, 2022. doi:10.1007/978-3-031-16815-4_9.

[ZvdPYS22] Yuanyuan Zhou, Joop van de Pol, Yu Yu, and François-Xavier Standaert. A third is all you need: Extended partial key exposure attack on CRT-RSA with additive exponent blinding. *IACR Cryptol. ePrint Arch.*, page 1163, 2022. URL: https://eprint.iacr.org/2022/1163.