COLLEGES SCIENCES
BRETAGNE POUR L'INGENIEUR
LOIRE ET LE NUMERIQUE

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

THESE DE DOCTORAT DE

L'ÉCOLE NATIONALE SUPERIEURE
MINES-TELECOM ATLANTIQUE BRETAGNE PAYS DE LA LOIRE –
IMT ATLANTIQUE

ECOLE DOCTORALE N° 648
Sciences pour l'Ingénieur et le Numérique
Spécialité : *mathématiques et leurs interactions*

Par

Modou SARRY

Side channel analysis against AEAD

Thèse présentée et soutenue à IMT ATLANTIQUE, Rennes, le 30/09/2024
Unité de recherche : IRISA, UMR CNRS 607
Thèse N° : 2024IMTA0420

Rapporteurs avant soutenance :

Louis GOUBIN	Professeur, Université de Versaille
Nadia EL MRABET	Professeur, Université de Saint Etienne

Composition du Jury :

Présidente :	Marine MINIER	Professeur, Université de Lorraine
Examinateurs :	Louis GOUBIN	Professeur, Université de Versaille
	Nadia EL MRABET	Professeur, Université de Saint-Etienne
	Vincent GROSSO	Chercheur, CNRS
	Damien MARION	Maitre de conférences, Université de Rennes 1
Dir. de thèse :	Hélène LE BOUDER	Maitre de conférences HDR, IMT ATLANTIQUE
Encadrant :	Laurent TOUTAIN	Professeur, IMT ATLANTIQUE
Encadrant :	Gaël THOMAS	Docteur en cryptographie, DGA-MI

Acknowledgement

Thanking the people who have contributed to my doctoral adventure is essential to me. It's important to express my gratitude and to let them know that they have, each in their own way, played a key role in the completion of my doctorate.

First of all, I would like to thank Professor Louis Goubin and Nadia El Mrabet for agreeing to report on my thesis.

I'd also like to thank my examiners, Vincent Grosso, Marine Minier and Damien Marion.

A huge thank you to my incredible supervisors, Hélène Le Bouder, Gaël Thomas, and Laurent Toutain. Thanks to your multiple technical and human facets, as well as your support and constructive feedback, you have been real boosters for the realization of this thesis. Without you, this thesis would not have the same flavor. You are the Dream Team of supervisors!

I'd also like to thank Ronan Latherme for his availability and invaluable explanations on the subject of Side channel analysis at the LHS.

Samba Dieng, former principal of Guéoul 2 school, where I did my primary education, I thank you from the bottom of my heart. And a special thanks to Mr. Niang, my reference teacher, for his encouragement and dedication to mathematics.

To Mr. Seck, my mathematics teacher at Guéoul College: may he rest in peace. You taught me to love numbers and formulas.

Many thanks to Mr. Gueye (Maths), Mr. Coulibaly, Mr. Dieng, and Mr. Sall from Guéoul high school. Your lessons and your passion for teaching have guided me to this point.

To all my TS2 classmates, starting with Serigne Mboup, Magaye Samb, Fatou Binetou Fall, Aminata Ndiaye, Fadiama Gaye, Abdou Kébé, Ousmane Ka, Mbaye Ndiaye and so many others: you have been a constant source of inspiration.

Many thanks to the members of KAKOUYI, starting with President Pape Ndiaye, Bara Gueye, Masseye Fall, my namesake Modou Ndiaye, Dame Gueye and so many others. Your encouragement and support have been invaluable.

To the Ndiayéne family of Guéoul, especially Massylla Ndiaye and Ablaye Ndiaye: thank you for welcoming me and supporting me throughout my years of learning.
To the members of the LMI2015 class, and in particular to Papa Moussa Gueye: thank you for your team spirit and your commitment in founding with me the Club Racine Mathématiques at the University Iba Der THIAM.

I'd like to thank Professors Mr. Diagne, Mr. Diop, Mr. Ba, and Mr. Mbaye at the University Iba Der THIAM. Your passion for teaching and your constant support were a great help.

To the Fall family in Rennes, starting with Fatou Binetou Dieng, Mame Mor Fall, Mamita Ndiaye, and Astou Diop: thank you for your hospitality and warm support.

I would also like to thank the Paye family, especially Awa Paye.

I'd also like to thank all the members of Touba Rennes, who put their trust in me to manage the Human Resources department so willingly. Special thanks to Pape Sall for his kindness and integration.

I'm not forgetting my dearest friends: Docteur Mbaye Dieng, Gorgui Diallo, Papa Moussa Dramé, and Sidi Leye (Guéoul). You have been invaluable companions throughout this journey.

I'd also like to thank the members of the Cryptis team, notably Amadou Oury Diallo, Habib Barry, Abraham Gnanzou, and Awaleh Houssein Meraneh. Being with you has been a real pleasure.

A big thank you to all my colleagues at ADER, who trusted me to chair the association and helped me carry out my activities. You have made this experience a real success.

Last but not least, I'd like to thank my family, especially my mother and father, for their support and understanding in the realization of my projects. You are my heroes. Many thanks also to my lovely brothers and sisters: Ndeye Fatou SARRY, Saye SARRY, Fatou Sylla SARRY and the others.

And of course, a huge thank you to Serigne Saliou Mbacké for his constant support from afar since I was very young. Your guidance has helped me stay on the right path and pursue my dreams.

Contents

I Introduction **11**

1 Introduction **12**
1.1 Professional Context . . . 14
1.1.1 IMT Atlantique . . . 14
1.1.2 DGA-MI . . . 14
1.2 Context . . . 15
1.3 Motivation . . . 15
1.4 Contribution . . . 15
1.5 Outline of this thesis . . . 16

II Context and Background **17**

2 Background and physical cryptanalysis **18**
2.1 Introduction . . . 20
2.2 Symmetric cryptography . . . 20
2.2.1 Symmetric encryption . . . 20
2.2.2 Message authentication code (MAC) . . . 21
2.2.3 Authenticated encryption . . . 21
2.2.4 Authenticated encryption with associated data (AEAD) . . . 22
2.2.5 Stream ciphers . . . 24
2.2.6 Primitives and Modes . . . 24
2.2.6.1 Primitives . . . 24
2.2.6.1.1 Permutations . . . 24
2.2.6.1.2 Block ciphers . . . 25
2.2.6.1.3 Tweakable block cipher (TBC) . . . 26
2.2.6.2 Modes . . . 26
2.2.6.2.1 Mode of operation . . . 26
2.2.6.2.2 Mode of operation for AEAD . . . 29
2.3 Physical cryptanalysis . . . 35
2.3.1 Side channel analysis (SCA) . . . 35
2.3.2 Simple power analysis (SPA) . . . 35
2.3.3 Profiled Attacks . . . 35
2.3.4 Non-Profiled Attacks . . . 36
2.3.4.1 Leakage model . . . 36
2.3.4.2 Differential power analysis (DPA) . . . 37
2.3.4.3 Correlation power analysis (CPA) . . . 37
2.3.4.4 Differential analysis aided power attack (DAPA) . . . 37
2.3.4.5 Soft analytical side channel attack (SASCA) . . . 37
2.3.4.6 Blind side channel analysis (BSCA) . . . 38

3 An In-depth Exploration of side channel analysis (SCA) Against AEAD **39**
3.1 Introduction . . . 41
3.2 Ascon . . . 42
3.2.1 Overview of the design . . . 42
3.2.1.1 Permutation . . . 43
3.2.1.1.1 Addition of constants . . . 43
3.2.1.1.2 Substitution layer . . . 43

3.2.1.1.3 Linear diffusion layer . . . 43
3.2.2 SCA on Ascon . . . 44
3.3 Elephant . . . 47
3.3.1 Overview of the design . . . 47
3.3.2 SCA on Elephant . . . 50
3.4 GIFT-COFB . . . 51
3.4.1 Overview of the design . . . 51
3.4.2 SCA on GIFT-COFB . . . 51
3.5 Grain-128AEADv2 . . . 53
3.5.1 Overview of the design . . . 53
3.5.2 SCA on Grain-128AEADv2 . . . 53
3.6 ISAP . . . 54
3.6.1 Overview of the design . . . 54
3.6.2 SCA on ISAP . . . 54
3.7 PHOTON-Beetle . . . 55
3.7.1 Overview of the design . . . 55
3.7.2 SCA on PHOTON-Beetle . . . 55
3.8 Romulus . . . 56
3.8.1 Overview of the design . . . 56
3.8.2 SCA on Romulus-N . . . 56
3.9 Sparkle . . . 57
3.9.1 Overview of the design . . . 57
3.9.1.1 Zoom on Sparkle . . . 57
3.9.1.2 Alzette . . . 58
3.9.2 SCA on Sparkle . . . 59
3.10 TinyJAMBU . . . 60
3.10.1 Overview of the design . . . 60
3.10.2 SCA on TinyJAMBU . . . 60
3.11 Xoodyak . . . 61
3.11.1 Overview of the design . . . 61
3.11.2 SCA on Xoodyak . . . 61
3.12 SCA on linear feedback shift registers (LFSRs) . . . 62

III Attack against AEAD algorithms. 64

4 Attack with a theoretical model 65
4.1 Introduction . . . 67
4.2 Elephant . . . 67
4.3 Theoretical attack . . . 67
4.3.1 Goal . . . 67
4.3.2 Leakage in the LFSR . . . 67
4.3.3 Link between the different masks . . . 68
4.4 Attack strategy . . . 69
4.4.1 Algorithm of the attack . . . 69
4.4.2 Optimisation of the attack . . . 70
4.4.2.1 Goal of the optimization . . . 70
4.4.2.2 Principle . . . 71
4.4.2.3 Gathering information . . . 71
4.4.2.4 Crafting a merge estimator . . . 72
4.4.2.5 Estimating the best position . . . 75
4.5 Results and discussion . . . 75
4.5.1 Elephant attack . . . 75
4.5.2 Impact of the generation of masks . . . 76
4.5.3 Studies on different LFSRs . . . 76
4.6 Conclusion . . . 78

5 Attack with a noisy theoretical model 80
5.1 Introduction . . . 82

5.2 Elephant and Sparkle . . . 82
5.3 Our approach of belief propagation (BP) . . . 82
5.3.1 Belief propagation (BP) . . . 82
5.3.1.1 Tanner graph . . . 82
5.3.1.2 BP algorithm . . . 83
5.4 Attack on Elephant . . . 83
5.4.1 Attack path . . . 83
5.4.2 Tanner graph . . . 84
5.4.3 Results . . . 84
5.5 Attack on Sparkle . . . 86
5.5.1 Attack path . . . 86
5.5.2 Tanner graph . . . 86
5.5.3 Results . . . 86
5.6 Conclusion . . . 88

6 Attack with real leakages **90**
6.1 Introduction . . . 92
6.2 Description of Ascon . . . 92
6.3 Attack path . . . 92
6.4 Experiments . . . 94
6.4.1 Mutual information on S-box of Ascon . . . 96
6.4.1.1 Information leakage linked to horizontal values at the sbox output . . . 97
6.4.1.2 Information leakage linked to vertical values at the sbox output . . . 98
6.4.1.3 Information leakage associated with horizontal Hamming weight (HW) of the sbox output . . . 99
6.4.1.4 Information leakage related to the vertical Hamming weight (HW) of the sbox output . . . 100
6.4.1.5 Discussion of results . . . 102
6.5 Conclusion . . . 103

IV Conclusion and Future works **104**

7 Conclusion and Future works **105**
7.1 Conclusion . . . 107
7.2 Future works . . . 109
7.3 Introduction . . . 113
7.3.1 Contexte . . . 113
7.3.2 Motivation . . . 113
7.3.3 Contribution . . . 113
7.4 Les attaques par observation contre les chiffrements intègres légers . . . 114
7.4.1 Une exploration des attaques par observation sur les 10 chiffrements intègres légers . . . 114
7.4.2 Attaque avec un modèle théorique . . . 114
7.4.3 Attaque avec un modèle théorique en ajoutant du bruit . . . 114
7.4.3.1 Attaque sur Elephant . . . 114
7.4.3.2 Attaque sur Sparkle . . . 115
7.4.4 Attaque pratique avec des traces réelles . . . 115
7.5 Conclusion . . . 115
7.6 Perspectives . . . 117
7.7 Enseignements . . . 118

List of Figures

1. Introduction **14**
1.1 The Rennes campus of IMT ATLANTIQUE. 14
1.2 DGA-MI. Reproduced with permission from [1]. ©Wikipedia 14

2. Background and physical cryptanalysis **22**
2.1 Overview of AEAD. 22
2.2 The structure of a stream cipher.. 24
2.3 the structure of a block cipher. 25
2.4 counter (CTR) mode. Reproduced with permission from [1]. ©Wikipedia 27
2.5 cipher block chaining (CBC) mode. Reproduced with permission from [1]. ©Wikipedia 28
2.6 cipher feedback (CFB) mode. Reproduced with permission from [1]. ©Wikipedia 28
2.7 output feedback (OFB) mode. Reproduced with permission from [1]. ©Wikipedia 29
2.8 the sponge mode. Reproduced with permission from [1]. ©Wikipedia 30
2.9 the duplex mode. Reproduced with permission from [2]. ©SAC 2011 31
2.10 MonkeyDuplex mode. Reproduced with permission from [3]. ©DIAC 2012 32
2.11 the beetle mode of operation. Reproduced with permission from [4]. ©IACR TCHES 2018 . . . 32
2.12 encryption of combined feedback (COFB) for 3-block associated data and plaintext. Reproduced with permission from [5]. ©TCHES 2017 34

3. An In-depth Exploration of SCA Against AEAD **42**
3.1 Ascon. This figure comes from [6] according to the description of Ascon [6]. 42
3.2 The AEAD variants of Ascon 42
3.3 State. This figure comes from [6] according to the description of Ascon [6]. 43
3.4 Round constant addition p_C. This figure comes from [6] according to the description of Ascon [6]. 43
3.5 Substition layer p_S with 5-bit S-box.. This figure comes from [6] according to the description of Ascon [6]. 43
3.6 Substition layer p_S with 5-bit S-box. This figure comes from [6] according to the description of Ascon [6]. 44
3.7 Linear layer with 64-bit diffusion function. This figure comes from [6] according to the description of Ascon [6]. 44
3.8 Elephant associated data authentication (top), plaintext encryption (middle), and ciphertext authentication (bottom). This figure comes from [7] according to the description of Elephant [8]. 48
3.9 160-bit LFSR φ_{Dumbo}. This figure comes from [7] according to the description of Elephant [8]. . 49
3.10 176-bit LFSR φ_{Jumbo}. This figure comes from [7] according to the description of Elephant [8]. . 49
3.11 200-bit LFSR $\varphi_{\mathsf{Delirium}}$. This figure comes from [7] according to the description of Elephant [8]. . 49
3.12 The AEAD variants of the Elephant 49
3.13 The AEAD variant of the GIFT-COFB 51
3.14 The AEAD variant of the Grain-128AEADv2 53
3.15 The AEAD variants of the ISAP 54
3.16 The AEAD variant of the PHOTON-Beetle 55
3.17 The AEAD variant of the Romulus 56
3.18 Encryption using Schwaemm256-128, where Sparkle384^R is the Sparkle384 permutation with R rounds, ρ is a linear application, $\mathcal{W}$ the linear whitening layer, and the constant Const_M indicating whether the plaintext M was padded or not. 57
3.19 The first round of the Sparkle384 permutation during the initialisation of Schwaemm256-128. 57
3.20 The Alzette S-box $\mathcal{A}_i$ used in Sparkle. 58

3.21 The AEAD variant of the SPARKLE. 58
3.22 The AEAD variant of the TINYJAMBU 60
3.23 The AEAD variant of the XOODYAK 61

4. Attack with a theoretical model **70**
4.1 Example of the tree representation of the LFSR initial state for the Hamming weights given on the left. Only the first three layers of the subtree rooted at $x'_0 = 03$ are shown. 70
4.2 Gathering information for the Dumbo LFSR. 71
4.3 Necessary data for the naive estimator (reduction triangle) and the neural network based merge estimator. 72
4.4 Distribution of err_{naive} for $m = 4$. 73
4.5 Distribution of $err_{network}$ for $m = 4$. 73
4.6 Architecture of the neural network for the merge estimator E. 74
4.7 Evolution of training and validation losses of the neural network. 75
4.8 Example output of the sliding window E estimation algorithm. 76
4.9 Estimator E versus actual computation time for finished runs (blue) and unfinished runs (red). Quartiles are represented by dashed lines. 77

5. Attack with a noisy theoretical model **82**
5.1 Example of a Tanner factor graph part. Circles are variable nodes and squares are factor nodes. 82
5.2 Tanner factor graph on linear feedback shift register (LFSR) Dumbo of ELEPHANT. 85
5.3 Measurement leakage in the first round of Alzette and Figure 3.20. 86
5.4 Tanner graph of the first round of Alzette. Only a single bit with index $b \in [\![0, 31]\!]$ of each 32-bit register is shown. Relative indices are given mod 32. The carries in the modular operation are denoted $c_1(b)$ and $c_1(b-1)$. Bits belonging to the same 32-bit register are enclosed in dashed lines. 87

6.1 Initialization phase of ASCON. 92
6.2 oscilloscope Keysight DSOS404A. 95
6.3 probe on STM32F4. 96
6.4 Information leakage linked to horizontal values at the S-box output. 97
6.5 Information leakage linked to vertical values at the S-box output. 98
6.6 Information leakage associated with horizontal Hamming weight (HW) of the S-box output. . . . 99
6.7 Information leakage related to the vertical Hamming weight (HW) of the S-box output. 100
6.8 All hypotheses 101

Part I

Introduction

Chapter 1

Introduction



Contents

1.1 Professional Context **14**
1.1.1 IMT Atlantique 14
1.1.2 DGA-MI 14
1.2 Context **15**
1.3 Motivation **15**
1.4 Contribution **15**
1.5 Outline of this thesis **16**

1.1 Professional Context

This thesis was conducted at the IMT Atlantique laboratory located on the Rennes campus. I started in September 2021 and I am supervised by Hélène Le Bouder (HDR at IMT Atlantique), Gaël Thomas (Doctor in cryptography at DGA-MI) and Laurent Toutain (Professor at IMT Atlantique). My doctoral school is called école doctorale (ED) Sciences pour l'ingénieur et le numérique (SPIN).

1.1.1 IMT Atlantique

IMT Atlantique, École Mines-Télécom Atlantique Bretagne-Pays de la Loire, was established on January 1, 2017. Accredited as one of the 204 french engineering schools authorised to award an engineering degree, the school operates across three campuses: Brest, Nantes, and Rennes. I was welcomed at the Rennes campus, illustrated in Fig. 1.2. I'm a member of the "Objets Communicants pour l'Internet du Futur" (OCIF) team, which is designing the Internet of Things by developing new algorithms, protocols and architectures for interconnecting objects to the Internet.



Figure 1.1: The Rennes campus of IMT ATLANTIQUE.

1.1.2 DGA-MI

DGA-MI is a technical expertise center of the DGA, whose mission is to carry out studies, expertise and tests in the fields of electronic warfare for weapons systems, information systems, telecommunications, information security and electronic components.



Figure 1.2: DGA-MI. Reproduced with permission from [1]. ©Wikipedia

1.2 Context

The rapid development of small computing devices such as radio-frequency identification (RFID) tags, industrial controllers, sensor nodes, and smart cards has become increasingly common with advancements in technology. This shift from desktop computers to compact devices is driven by various factors, including the miniaturization of electronic components, the growing demand for automation in various industrial sectors, and the need to collect real-time data in diverse environments.
However, this transition to smaller devices raises new concerns regarding security and privacy. Unlike desktop computers, these small devices have limited resources in terms of computing power, memory, and storage. Additionally, they may be deployed in unsecured environments and be susceptible to physical and software attacks. Many traditional cryptographic standards were designed and optimized for desktop and server environments, where resources are abundant. These standards aim to strike a balance between security, performance, and resource requirements. However, implementing them on resource-constrained devices presents major challenges.

This necessity for new algorithms has prompted national institute of standards and technology (NIST) to launch a process aimed at soliciting, evaluating, and standardizing lightweight cryptographic algorithms suitable for use in constrained environments.

After hosting two public workshops (in 2015 and 2016), NIST published the submission requirements and evaluation criteria in 2018 and received 57 submissions in response to the call. In April 2019, NIST announced 56 first-round candidates. In August 2019, NIST announced 32 second-round candidates and published NIST internal report to explain the evaluation criteria and selection of the second-round candidates. Selection criteria include software performance, hardware performance, security, and additional features such as resistance to side channel analysis (SCA).

In August 2020, NIST invited the submitters of the second-round candidates to provide short updates on their algorithms. During the second round of the process, NIST hosted the third and fourth lightweight cryptography workshops to discuss various aspects of the second-round candidates and obtain valuable feedback for the selection of the finalists.

In March 2021, NIST announces 10 finalists, namely Ascon [6], Elephant [8], GIFT-COFB [9], Grain-128AEADv2 [10],
ISAP [11], PHOTON-Beetle [12], Romulus [13], Sparkle [14], TinyJAMBU [15], Xoodyak [16].

This thesis begins just after the announcement of the then finalists, who are still competing against each other. Among the selection criteria of the NIST is the resistance of algorithms to side channel analysis (SCA).

1.3 Motivation

The aim of this thesis is to study the the impact of physical attacks on the security of AEAD. In particular, the study focuses on algorithms proposed in the context of the NIST competition.

The main objective is to identify potential vulnerabilities in the ten finalist algorithms in the NIST competition that can be exploited in side channel analysis (SCA), in order to anticipate future attacks.

1.4 Contribution

This manuscrit includes an in-depth study of side channel analysis (SCA) on the 10 finalist algorithms. In addition, it offers a detailed analysis of a blind side channel analysis (BSCA) on NIST candidate Elephant. Furthermore, this research has led to the development of generic tools for belief propagation, providing new perspectives for evaluating the security of cryptographic systems. It also explores the use of belief propagation (BP) to improve the power of blind side channel analysis (BSCA) on NIST candidate Elephant and another candidate, Sparkle. In addition, it presents the beginnings of a practical attack on Ascon in progress, aimed at identifying potential vulnerabilities and assessing the robustness of the system.

1.5 Outline of this thesis

This thesis studied the impact of physical attacks on the security of AEAD. In particular, the study focuses on algorithms proposed in the context of the NIST competition.

This thesis is composed of 7 chapters divided into 4 parts, and is organized as follows.

1. In part I, introduction.
   - ➤ In chap 1, evolution and a brief overview of crypography and physical cryptanalysis are introduced.
2. In part II, context and background.
   - ➤ In chap 2, the context and background in which AEAD and SCA are generally defined, while defining the fundamental concepts used in this thesis.
   - ➤ In chap 3, explains an in-depth exploration of SCA on the 10 AEAD algorithms finalists in the NIST competition.
3. In part III, attack against authenticated encryption with associated data (AEAD) algorithms.
   - ➤ In chap 4, describes an attack with a theoretical model, we target one of the finalists, Elephant, where the leakage model is only Hamming weight (HW).
   - ➤ In chap 5, presents an attack with a theoretical noisy model, we target two of the finalists, Elephant and Sparkle, where the leakage model is just a hamming weight with Gaussian noise. In addition, we used BP to make the attack more successful.
   - ➤ In chap 6, describes an attack path, using real traces of our Ascon target, we try to carry out a new attack with real leaks and get away from existential attacks on Ascon.
4. In part IV, conclusion and future works.
   - ➤ In chap 7, concludes the dissertation and provides some research.

Part II

Context and Background

Chapter 2

Background and physical cryptanalysis

"*Évite la compagnie de celui qui a un mauvais comportement et de mauvaises paroles; tu seras, alors exempt de tout reproche car on adopte souvent le comportement de celui qu'on fréquente. Si l'on veut avoir de bons caractères on a qu'à fréquenter ceux qui en ont et éviter ceux qui en sont dépourvus. L'homme a naturellement des dispositions qui le poussent à agir dans le bon ou le mauvais sens. S'il fréquente des gens de bien, leurs bonnes paroles, leurs bonnes pensées et leurs bons actes éveilleront en lui ses bonnes aptitudes et inhiberont celles qui sont mauvaises*"
SILKUL JAWAAHIRI (Le Collier en Perles Précieuse) de Serigne TOUBA
Proposée par Sohibou NDIAYE.



Contents

2.1 Introduction **20**
2.2 Symmetric cryptography **20**
2.2.1 Symmetric encryption 20
2.2.2 Message authentication code (MAC) 21
2.2.3 Authenticated encryption 21
2.2.4 Authenticated encryption with associated data (AEAD) 22
2.2.5 Stream ciphers 24
2.2.6 Primitives and Modes 24
2.2.6.1 Primitives 24
2.2.6.2 Modes 26
2.3 Physical cryptanalysis **35**
2.3.1 Side channel analysis (SCA) 35
2.3.2 Simple power analysis (SPA) 35
2.3.3 Profiled Attacks 35
2.3.4 Non-Profiled Attacks 36
2.3.4.1 Leakage model 36
2.3.4.2 Differential power analysis (DPA) 37
2.3.4.3 Correlation power analysis (CPA) 37
2.3.4.4 Differential analysis aided power attack (DAPA) 37
2.3.4.5 Soft analytical side channel attack (SASCA) 37
2.3.4.6 Blind side channel analysis (BSCA) 38

2.1 Introduction

To provide readers with a solid knowledge base to help contextualize this thesis, in this chapter we first introduce the basic concepts of cryptography. This initial exploration solidifies the foundation by revisiting the fundamental principles of cryptography and helping to understand the construction of authenticated encryption with associated data (AEAD) algorithms.

Secondly, we describe the essential concepts needed to understand the state of observational attacks, with particular emphasis on the leakage models and blind side channel analysis (BSCA) we use in our contributions. These elements are crucial to understanding the advanced analyses we will present later in chapters 4, 5, and 6.

2.2 Symmetric cryptography

Cryptography is a discipline of mathematics and computer science that historically studied methods to preserve the confidentiality of the messages exchanged in a communication channel in the presence of an adversary. Until the second half of the twentieth century, the primary application of cryptography was in military communications, some famous examples including Caesar's Cipher or the Enigma Machine. The introduction and generalization of the computer in the civil society led to the generalization of the usage of cryptography and the diversification of its applications.

In generic terms, cryptography nowadays studies several building blocks that provide security properties in adversarial contexts. The building blocks are cryptographic algorithms and their implementation on actual hardware, as well as cryptographic protocols (algorithms that involve multiple devices communicating over a channel). These building blocks are often public and standardized, either by a formal standardization body, or by widespread use.

The security properties are multiple.

1. Confidentiality: ensure that information is accessible only to authorized persons;
2. Integrity: ensure that information cannot be altered;
3. Authenticity: validate the origin of an entity ;
4. Non-repudiation: to enable a person to take part in a contract with the impossibility of reneging on it at a later date.

There exist many kinds of cryptographic algorithms and protocols, designed for different usages. In this work, we only talk about private key cryptography or symmetric cryptography, in which all participants in the cryptographic protocol use a single shared secret key that they must not share with the adversary. The key or set of keys is a shared secret between the parties that is used to maintain a secure communication link. The symmetric primitives can be either permutation, block ciphers, and tweakable block cipher (TBC). The main difficulty with symmetric cryptography is the way used to obtain the shared secret and thus establish the secure link between the diferent users.

This is different from public key cryptography, or asymmetric cryptography, in which a secret key is associated to one or multiple public keys that can be shared with other people.

2.2.1 Symmetric encryption

Encryption describes the process of transforming a plaintext message (we use the terms plaintext and message interchangeably) M into a ciphertext C, so that it is easy for an authorized user given C to compute M but hard for an adversary. In symmetric cryptography, authorized users share a secrete key K on which they agree in some way before the exchange of data.

In general terms, a symmetric encryption schemes consists of two efficient and deterministic procedures parameterized by a key K: an encryption procedure `enc` and a decryption procedure `dec` . The encryption

procedure takes a message M and some optional arguments and turns them into a ciphertext C, while the decryption procedure takes the ciphertext C and the optional arguments and returns the original message M. We say that the scheme is correct if decrypting a ciphertext with the same key and optional arguments used to encrypt it gives the initial plaintext.

2.2.2 Message authentication code (MAC)

Authentication is a process that guarantees that a message is authentic and has not been altered by a third party. In symmetric cryptography, authentication is achieved through the use of Message authentication code (MAC).

Let a key space $\mathcal{K} = \{0,1\}^k$, a tag space $\mathcal{C} = \{0,1\}^*$, and a message space $\mathcal{P} = \{0,1\}^*$. A message authentication code (MAC) is a deterministic procedure. $\texttt{mac} : \mathcal{P} \times \mathcal{K} \times \mathcal{T} \to \mathcal{T}$.

It should also be noted that the message authentication code is also called a tag. It generally has a fixed size $t \in \mathbb{N}$, in others terms $\mathcal{T} = \{0,1\}^t$.
A message authentication code (MAC) is calculated as follows.

$$(K \in \mathcal{K}, T \in \mathcal{T}, M \in \mathcal{P}) : T = \texttt{mac}(M))$$

To authenticate a message M, the sender calculates its message authentication code T and sends it with the message. The recipient can calculate the tag of the received message and compare it with the tag he has received to ensure that the message has not been modified.

2.2.3 Authenticated encryption

Message authentication code (MAC) can be combined with symmetric cipher to simultaneously ensure the confidentiality and authenticity of a message. To this end, we can use a message authentication code (MAC) with a K_{mac} key, in parallel with symmetric cipher using a K key, to encrypt and authenticate a message M in the following way.

1. MAC-Then-Encrypt: authenticate the plaintext message and encrypt both the tag and message.
   - ➤ Encryption.
     - Compute the tag $T = \texttt{mac}_{K_{mac}}(M)$.
     - Encrypt $M||T$ with $C = \texttt{enc}(M||T)$.
   - ➤ Decryption.
     - Decrypt the ciphertext to obtain $M'||T' = \texttt{dec}(C)$.
     - Verify that $T' = \texttt{mac}_{K_{mac}}(M')$, if so return M', otherwise return $\perp$.
2. Encrypt-And-Mac: authenticate the plaintext message and encrypt the message.
   - ➤ Encryption.
     - Compute the tag $T = \texttt{mac}_{K_{mac}}(M)$.
     - Compute the ciphertext $C = \texttt{enc}(M)$.
     - Return $C||T$.
   - ➤ Decryption.
     - Extract T'.
     - Extract C'.
     - Decrypt the ciphertext $M' = \texttt{dec}(C')$.
     - Verify that $T' = \texttt{mac}_{K_{mac}}(M')$, if so return M', otherwise return $\perp$.
3. Encrypt-Then-MAC: authenticate the encrypted message.
   - ➤ Encryption.
     - Compute the ciphertext $C = \texttt{enc}(M)$.
     - Compute the tag $T = \texttt{mac}_{K_{mac}}(C)$.

- Return $C||T$.

➤ Decryption:

- Extract T'.
- Extract C'.
- Verify that $T' = \texttt{mac}_{K_{mac}}(C')$, if so decrypt and return $M' = \texttt{dec}(C')$, otherwise do not decrypt C' and return $\perp$.

It's important to note that, of these solutions, only the Encrypt-then-MAC method is generically secure.

2.2.4 Authenticated encryption with associated data (AEAD)

The content of the message is not the only thing that can be altered by an attacker: associated data, such as internet packet headers or unique object ids can also be targeted.
To illustrate the security problem it may cause, we can imagine the case of a security protocol in which a client sends encrypted commands to servers over the network, using a secret key K to encrypt and authenticate the commands. If the address of the targeted server is not authenticated, an active attacker can capture a command sent to a particular server and execute that same command to all the other servers using the same key simply by changing the target address of the packet.

Authenticated encryption with associated data (AEAD) solves the problem by enabling some data (the associated data) to be authenticated but not encrypted. This associated data must be provided both in encryption and decryption, since it is used in the determination of the tag. If the associated data changes between encryption and decryption, the tag becomes invalid and no plaintext is returned.

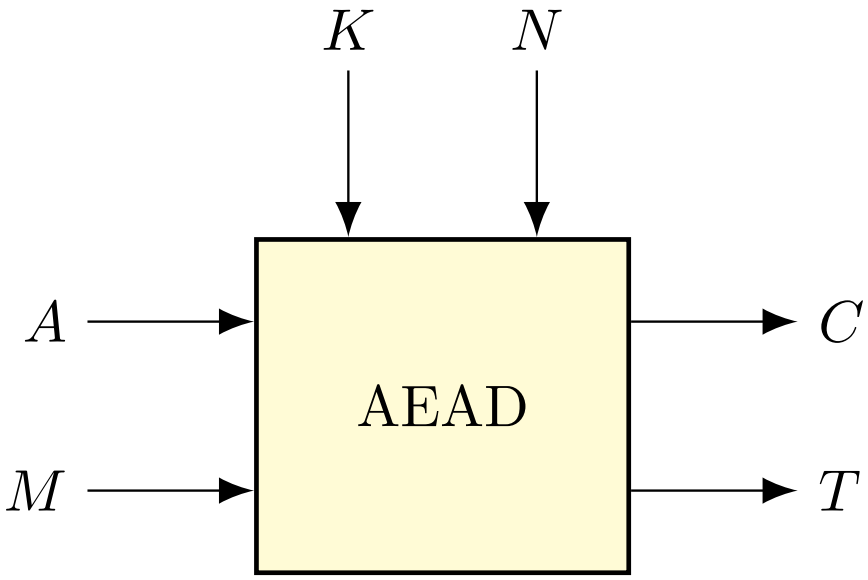


Figure 2.1: Overview of AEAD.

Authenticated encryption with associated data (AEAD), illustrated in Figure 2.1, should ensure confidentiality and integrity. It takes as input different parameters: a plaintext denoted M, the associated data denoted A, a secret key K, and an initialisation vector N also called a nonce. The nonce is public but must be different for each new plaintext. The algorithms ensure confidentiality of the plaintext M and integrity of both the plaintext M and the associated data A. It returns a ciphertext C and a tag T.

In the table 2.1, we present the different processes and events of the NIST competition for lightweight cryptography (LWC).

Table 2.1: Timeline of the NIST lightweight cryptography (LWC) algorithms.

Timeline of the NIST lightweight cryptography (LWC) algorithms	
July 2015	First lightweight cryptography (LWC) workshop at NIST
October 2016	Second lightweight cryptography (LWC) workshop at NIST
March 2017	NIST IR 8114 Report on LWC [17]
April 2017	Profles for LWC Standardization Process [18]
August 2018	Federal Register Notice [19] Submission requirements and evaluation criteria for the LWC standardization process [20]
April 2019	Announcement of the first-round candidates
August 2019	Announcement of the second-round candidates
October 2019	NIST IR 8268, Status report on the first Round of the NIST LWC standardization Process [21]
November 2019	Third LWC workshop at NIST
September 2020	Submission deadline for optional status updates
October 2020	Fourth LWC workshop
March 2021	Announcement of the finalists
July 2021	NIST IR 8369, Status report on the second round of the NIST LWC standardization process [22]
May 2022	Fifth LWC workshop
September 2022	Submission deadline for optional status updates
February 2023	Selection announcement: Ascon is the winner of the competition.
June 2023	Sixth LWC Workshop

In the table 2.2, we present the then AEAD algorithm finalists in the NIST competition. We also specify the three AEAD algorithms targeted in red during this thesis. These 10 finalists are described in chapter 3.

Table 2.2: 10 finalists for the AEAD algorithms.

10 finalists for the AEAD algorithms
Ascon [6]
Elephant [8]
GIFT-COFB [9]
Grain-128AEADv2 [10]
ISAP [11]
PHOTON-Beetle [12]
Romulus [13]
Sparkle [14]
TinyJAMBU [15]
Xoodyak [16]

2.2.5 Stream ciphers

Stream ciphers encrypt elements of plaintext using a dynamic transformation that evolves over time. The size of these elements can vary depending on design considerations, application requirements, and the platform used.
A stream cipher typically employs a pseudo-random generator. This generator take a secret key and initialization vector (IV)/nonce as inputs and produces a pseudo-random sequence of elements called the keystream. The keystream elements are XORed with the plaintext to produce the ciphertext.

Fig. 2.2 shows the structure of a stream cipher.

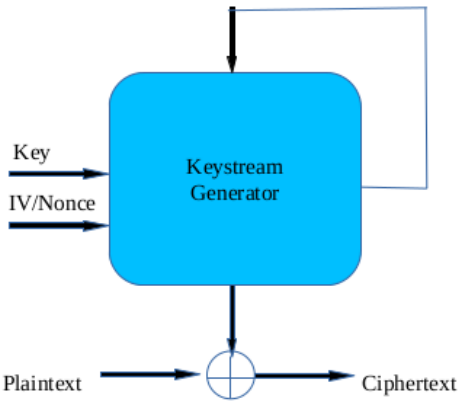


Figure 2.2: The structure of a stream cipher..

2.2.6 Primitives and Modes

Primitives are designed to handle data blocks of a fixed size. However, in real applications, the data to be encrypted or authenticated is often of variable size. Thus, modes are techniques that allow primitives to be used to process data of variable size.

In short, by combining cryptographic primitives with the appropriate modes, we can design robust cryptographic algorithms capable of efficiently securing data of variable size while ensuring both confidentiality and integrity.

2.2.6.1 Primitives

2.2.6.1.1. Permutations

A permutation is a function $\mathsf{P} : \mathcal{B} \to \mathcal{B}$. that transforms the bits of a data block of fixed length n, with a space $\mathcal{B} = \{0,1\}^n$.

The inverse permutation P^{-1} is the function that cancels out the effect of P. If $\mathsf{P}(x) = y$, then $\mathsf{P}^{-1}(y) = x$.

In this section, I refer to cryptographic permutations, which are mathematical objects used that have no special properties. Indeed, a cryptographic permutation must be indifferentiable from a random oracle to be considered secure [23].

2.2.6.1.2. Block ciphers

Block ciphers are families of permutations that are parameterized by a key. This K key is used to transform plaintext into ciphertext and vice versa.
Let a key space $\mathcal{K} = \{0,1\}^k$, and a block space $\mathcal{B} = \{0,1\}^n$.
A block cipher consists of $\mathcal{E}_K$ and $\mathcal{E}_K^{-1}$ defined as follows.

- $\mathcal{E}_K : \mathcal{K} \times \mathcal{B} \to \mathcal{B}$.

$$\forall K \in \mathcal{K}, M \in \mathcal{B} : \mathcal{E}_K(M) = C, \ C \text{ is ciphertext.}$$

- $\mathcal{E}_K^{-1} : \mathcal{K} \times \mathcal{B} \to \mathcal{B}$

$$\forall K \in \mathcal{K}, C \in \mathcal{B} : \mathcal{E}_K^{-1}(C) = M.$$

where $k \in \mathbb{N}$ is the size of the key and $n \in \mathbb{N}$ is the size of the blocks.

In Fig. 2.3, we shows the structure of a block cipher.

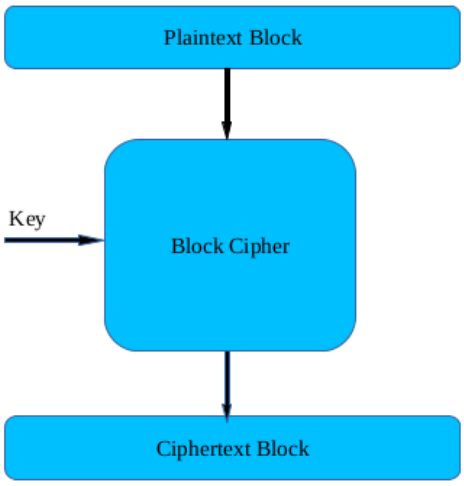


Figure 2.3: the structure of a block cipher.

2.2.6.1.3. Tweakable block cipher (TBC)

A tweakable block cipher (TBC) [24] is an extension of block ciphers. In addition to the secret key and plaintext, it uses an additional value called a "tweak". The tweak acts as a control parameter, allowing the behavior of the cipher to be modified without the need for a new key.
In other words, this T control parameter modifies the cipher for each instance, while using the same K key.

Let a key space $\mathcal{K} = \{0,1\}^k$, a block space $\mathcal{B} = \{0,1\}^n$, and a tweak space $\mathcal{T} = \{0,1\}^t$.
A tweakable block cipher (TBC) consists of $\mathcal{E}_{K,T}$ and $\mathcal{E}^{-1}_{K,T}$ defined as follows.

- $\mathcal{E}_{K,T} : \mathcal{K} \times \mathcal{T} \times \mathcal{B} \to \mathcal{B}$.

$$\forall K \in \mathcal{K}, T \in \mathcal{T}, M \in \mathcal{B} : \mathcal{E}_{K,T}(M) = C,\ C \text{ is ciphertext.}$$

- $\mathcal{E}^{-1}_{K,T} : \mathcal{K} \times \mathcal{T} \times \mathcal{B} \to \mathcal{B}$

$$\forall K \in \mathcal{K}, T \in \mathcal{T}, C \in \mathcal{B} : \mathcal{E}^{-1}_{K,T}(C) = M.$$

where $k \in \mathbb{N}$ is the size of the key, $t \in \mathbb{N}$ is the size of tweak, and $n \in \mathbb{N}$ is the size of the blocks.

2.2.6.2 Modes

2.2.6.2.1. Mode of operation

A mode of operation is a cryptographic algorithm designed to encrypt messages of variable length using a block cipher. A block cipher works on data blocks of a fixed size (e.g. 128 bits). However, the messages to be encrypted are generally not of fixed size, and may be shorter or longer than the block size. The operating modes can therefore handle these variable-length messages.

To achieve this, mode of operation use an IV in addition to the encryption key. The IV is a unique value used to ensure that the same data encrypted with the same key produces different results each time it is run. This is similar to the use of a nonce in stream ciphers. However, if the IV is reused with the same key, the security of the algorithm may be compromised, as it could allow an attacker to deduce information about the encrypted messages.

Since mode of operations are based on fixed-width primitives, they often require the message length to be a multiple of the block size. However, real messages don't always have this property. To solve this problem, additional data is added at the end of the message to adjust its length so that it is a multiple of the block size. This operation is known as padding. The type of padding used depends on the mode of operation.

The simplest method is to apply the block cipher several times. Using an $\mathcal{E}_K$ block cipher routine that processes blocks of length n, we encrypt an M (padded) message M as follows:

- Divide message: The M message is first divided into blocks of length n. If the length of M is not a multiple of n, padding is added to complete the last block.
- Encryption: Each M_i block of length n is then independently encrypted using the `enc` encryption routine and the encryption key. Encryption produces a series of encrypted blocks C_i:

$$\forall i \in 1, ..., l, C_i = \mathcal{E}_K(M_i), l = \frac{||M||}{n} \tag{2.1}$$

- Concatenation: The C_i encrypted blocks are then concatenated to form the final cipher C

$$C = C_1||C_2||C_3||\cdots||C_l. \tag{2.2}$$

This mode of operation, known as the electronic code book (ECB) mode, has major disadvantages in terms of security. Since each block is independently encrypted, identical blocks in the M plaintext will produce identical encrypted blocks in C. This can reveal patterns in the plaintext, leaving the cipher vulnerable to attack. Because of these limitations, other operating modes are often preferred to improve security.

In order to create a mode of operation, one commun method is to use block ciphers in combination with an initialization vector (IV) or nonce n to generate a key stream S. This stream is then used as a one-time pad to encrypt the plaintext.

$$\forall i \in 1, ..., l, C_i = M_i \oplus S_i, l = \frac{||M||}{n}. \tag{2.3}$$

In what follows, we show a few examples of these operating modes.

- Counter (CTR) mode: this mode, introduced by Whitheld Diffie and Martin Hellman in 1979 [25], transforms a block cipher into a stream cipher. It generates the next block in the key stream by encrypting successive values of a "counter". The counter can be any function producing a sequence that can be guaranteed not to be repeated for a long time, although an incremental counter is the simplest and most popular choice.

$$S_i = \mathcal{E}_K(N||i). \tag{2.4}$$

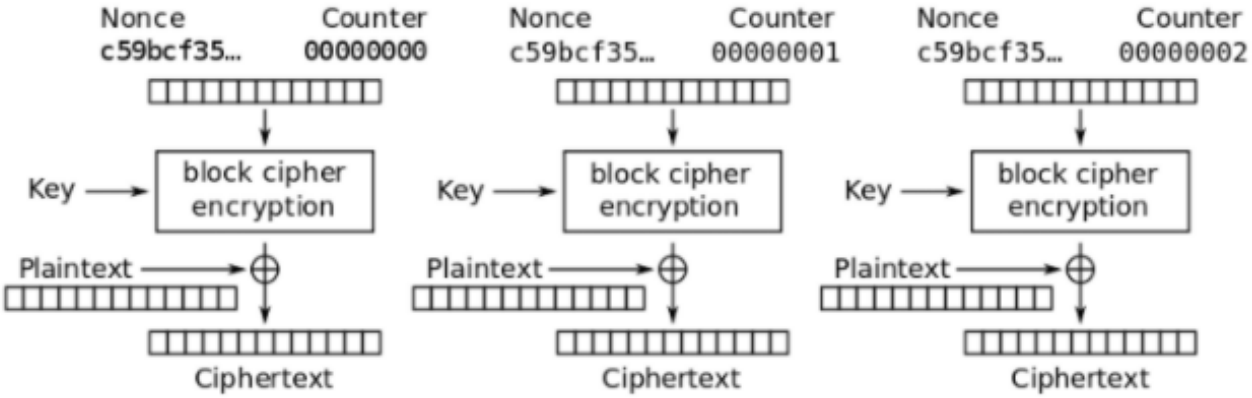


Figure 2.4: counter (CTR) mode. Reproduced with permission from [1]. ©Wikipedia

- Cipher block chaining (CBC) mode [26]: another common mode of operation not using this common method. This mode, invented and patented in 1976 by Ehrsam, Meyer, Smith and Tuchman, involves subjecting each block of plaintext to an XOR operation with the preceding block of ciphertext before encrypting it. In this way, each ciphertext block depends on all plaintext blocks processed up to that point. To guarantee the uniqueness of each message, an initialization vector must be used with the first block.

$$\forall i > 1, C_1 = \mathcal{E}_K(M_1 \oplus IV), C_i = \mathcal{E}_K(M_i \oplus C_{i-1}). \tag{2.5}$$

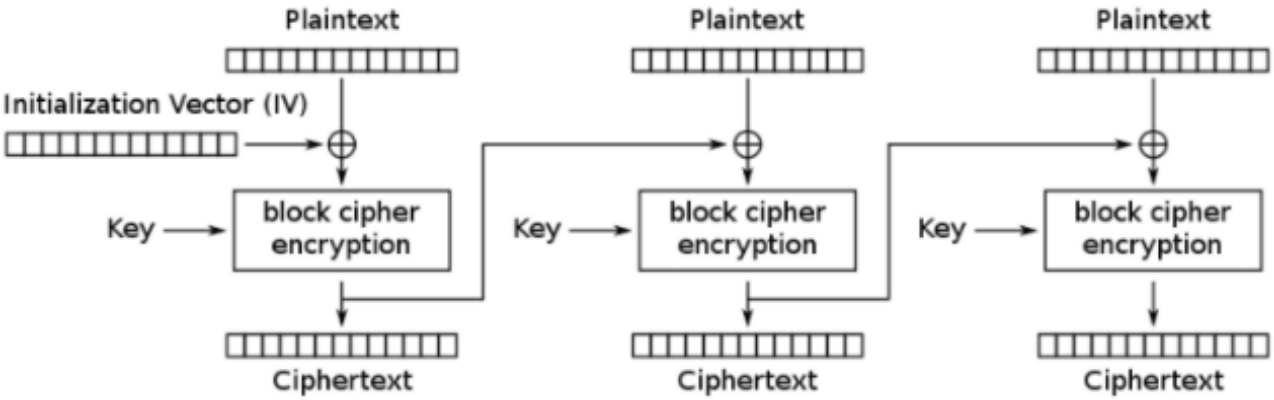


Figure 2.5: cipher block chaining (CBC) mode. Reproduced with permission from [1]. ©Wikipedia

- Cipher feedback (CFB) mode [27]: proposed in 1979, is a mode of operation that works like a stream cipher. It is quite similar similar to cipher block chaining (CBC), except that the plaintext is XORed with the cipher output. operation.

$$\forall i > 1, S_1 = \mathcal{E}_K(N), S_i = \mathcal{E}_K(C_{i-1}). \tag{2.6}$$

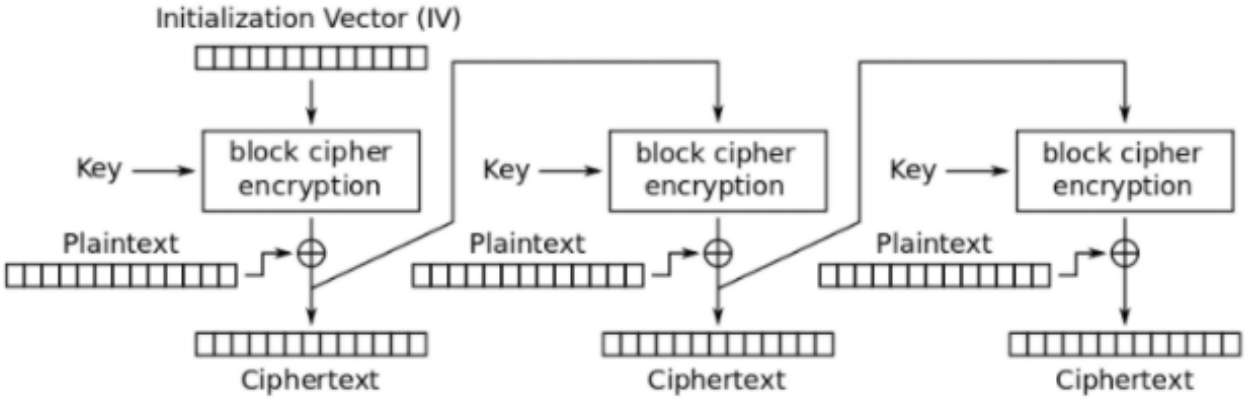


Figure 2.6: cipher feedback (CFB) mode. Reproduced with permission from [1]. ©Wikipedia

- Output feedback (OFB) mode [28]: proposed in 1981, this mode is also a synchronous stream cipher operating mode. It processes data in the same way as the cipher feedback (CFB) mode, but instead of transmitting the ciphertext calculated in the next step, the transmitted data is the output of the cipher calculation.

$$\forall i > 1, S_1 = \mathcal{E}_K(N), S_i = \mathcal{E}_K(S_{i-1}). \tag{2.7}$$

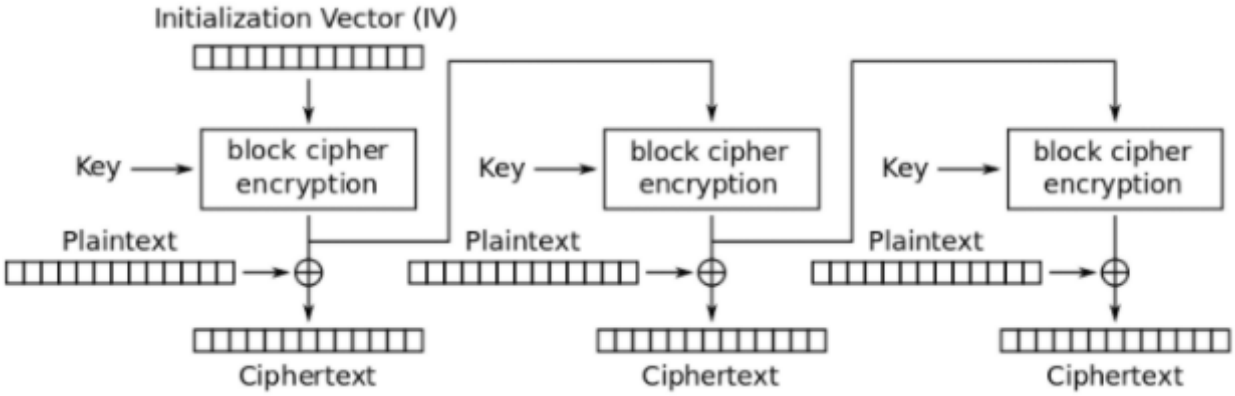


Figure 2.7: output feedback (OFB) mode. Reproduced with permission from [1]. ©Wikipedia

2.2.6.2.2. Mode of operation for AEAD

Sponge mode

Sponge mode [29] was introduced in 2007 by Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche.
This mode is a cryptographic structure used in hashing and MAC (Duplex mode [2] is required for encryption) algorithms. It is particularly well known for its use in the SHA-3 [30] hash algorithm and other modern cryptographic primitives. Sponge mode offers robust flexibility and security by using permutation on an internal state.
Sponge mode consists of three main phases: initialization, absorption (absorb) and extraction (squeeze). It uses a repeated cryptographic permutation on an internal state divided into two parts: r rate and c capacity.
We describe the main steps involved in building the sponge mode.

1. Initialization.
   - The internal state is initialized to zero or to a predetermined value.
2. Absorption (Absorb).
   - Input data is divided into blocks of size equal to the rate.
   - Each block is XORed with the rate part of the internal state.
   - Permutation is applied to the internal state after each block of data.
3. Extraction (Squeeze).
   - After absorbing all input data, output bits are produced by extracting bits from the rate part of the internal state.
   - Permutation is applied after each extraction until the desired output length is reached.

Fig. 2.8 presents the sponge mode.

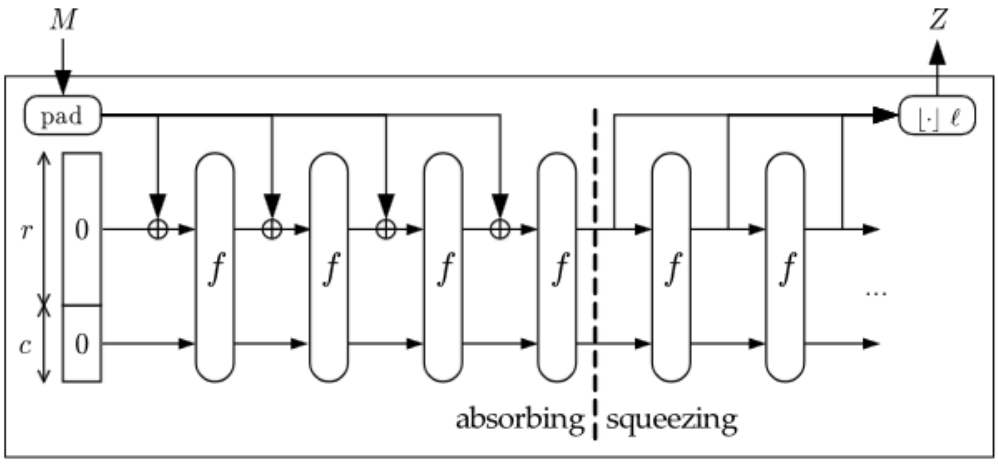


Figure 2.8: the sponge mode. Reproduced with permission from [1]. ©Wikipedia

The sponge-based design has several advantages compared to other available construction methods like some block cipher or Merkle-Damgard and other dedicated designs.

- The sponge mode is well-studied and has been analyzed and proven secure for different applications in a large amount of publications. Moreover, the sponge mode is used in the SHA-3 [30] winner Keccak [31].
- Flexible to adapt for other functionality (hash, MAC, and encryption).
- Elegant and simple design, clear state size, and no key schedule.

Duplex mode

Duplex mode is an extension of sponge mode, introduced in 2011 by the same authors [2]: Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. This mode enables continuous interaction with the data. In this way, the Duplex mode is an innovative cryptographic structure used to build symmetrical encryption algorithms. It is mainly used in the authenticated encryption with associated data (AEAD) encryption algorithm.
We describe the main steps involved in building Duplex mode from Sponge mode.

1. Initialization.
   - The initialization phase prepares the internal state of the duplex mode by configuring it with initial values. The internal state is made up of two parts: r the rate and c the capacity. The total size of the internal state is $b = r + c$.
     The internal state is initialized, usually to zero or to a value derived from a secret key.
2. Absorption.
   - The absorption phase consists of integrating the input data blocks into the internal state. Each block is XORed with the rate part of the internal state and the permutation is applied. This phase processes the data in successive blocks and combines them with the internal state.
3. Squeeze.
   - The extraction phase produces the output data by extracting bits from the internal state after each application of the permutation. This phase is used to generate ciphertext.
4. Finalization.
   - The finalization phase generates a tag that guarantees the integrity and authenticity of the processed data. This tag is based on the internal state after absorption and extraction.

Fig. 3.18 presents the duplex mode.

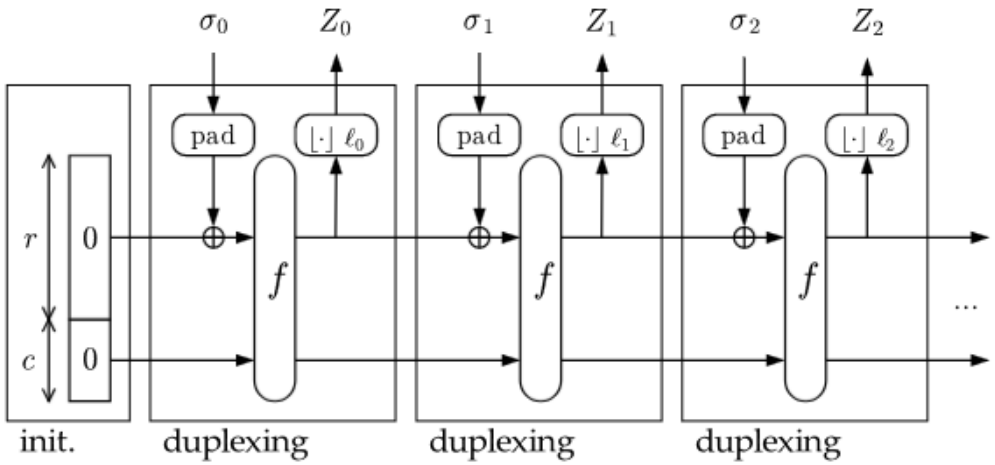


Figure 2.9: the duplex mode. Reproduced with permission from [2]. ©SAC 2011

MonkeyDuplex mode

MonkeyDuplex mode [3] is a variant of Duplex mode, introduced by Keyak creators Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche in 2014.
This is a specific variant of Duplex mode, designed to offer efficient and secure cryptographic operations in authenticated encryption and data integrity contexts, as it introduces certain specificities and optimizations to improve the latter. We describe the main steps involved in building the MonkeyDuplex mode.

1. Initialization.
   - ➤ The internal state is initialized, usually to a value dependent on the key and an initialization vector (IV).
2. Absorption.
   - ➤ Input data (plaintext, associated data, etc.) is absorbed in blocks. Each block is XORed with the rate portion of the internal state, and the permutation is applied to the complete state.
3. Squeeze.
   - ➤ After each absorption, part of the report (usually the rate part) can be issued as output. This can be used to produce blocks of ciphertext or authentication tags.
4. Finalization.
   - ➤ Once all the data has been absorbed, a finalization step is performed. A final block can be absorbed, and an authentication tag is calculated to guarantee data integrity.

Fig. 2.10 presents the MonkeyDuplex mode.

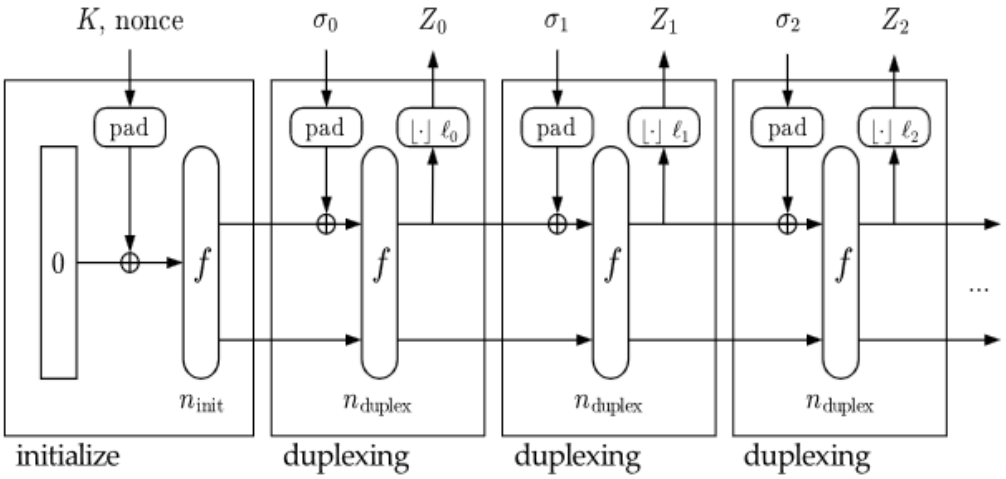


Figure 2.10: MonkeyDuplex mode. Reproduced with permission from [3]. ©DIAC 2012

Beetle mode

Introduced in 2018 by Itai Dinur, Orr Dunkelman, and Shai Halevi, Beetle mode [4] used by authenticated encryption with associated data (AEAD) algorithms. It belongs to the family of modes based on the Duplex structure, like the MonkeyDuplex mode. Beetle uses lightweight permutation and offers security guarantees for data confidentiality and integrity.

1. Initialization.
   - The internal state is initialized with a secret key and an initialization vector (IV).
2. Absorption.
   - The input data is divided into blocks of size equal to the rate.
   - Each block use fonction ρ with the rate part of the internal state.
   - A permutation is applied to the internal state after each block of data.
3. Squeezing.
   - Encrypted data is produced by extracting bits from the rate part of the internal state.
   - A permutation is applied after each extraction.
4. Authentication.
   - An authentication tag is generated using the capacity part of the internal state after absorption of all input data.
   - This tag guarantees the integrity and authenticity of the encrypted data.

Fig. 2.11 presents the beetle mode.

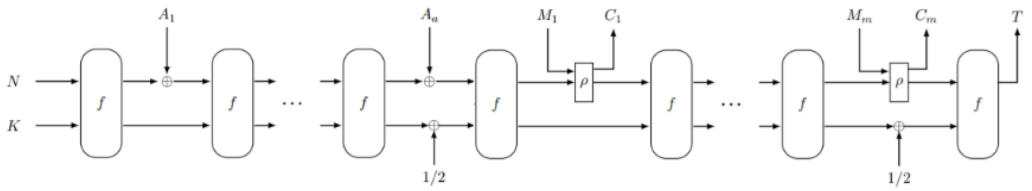


Figure 2.11: the beetle mode of operation. Reproduced with permission from [4]. ©IACR TCHES 2018

Combined feedback (COFB)

Combined feedback (COFB) [5] is block cipher based AEAD mode. COFB primarily focuses on the hardware implementation size. The COFB mode achieves several interesting features. It achieves a high value for rate which is 1 (i.e, needs only one block cipher call for one input block). The mode is inverse-free, i.e., it does not need a block cipher inverse during decryption. In addition to these features, this mode has a quite small state size, namely $1.5n + k$ bits, in case the underlying block cipher has an n-bit block and k-bit keys.

An initial version of COFB was presented in [5] and this latest version of COFB is a minor modification over the original COFB mode [5].
The modifications are as follows.

- Change the nonce to be 128 bit.
- Change the feedback to make it more hardware efficient.
- Deal with empty data. They change the mask update function for the purpose.
- Change the padding for the associated data. To be precise, if the associated data is empty, then padding the associated data will yield the constant block 10^{n-1} (n: block cipher state size).

We describe the main steps involved in building the COFB mode.

1. Initialization.
   - The internal state is initialized by combining the initialization vector (IV) and the secret key.
2. Absorption.
   - Data is processed in blocks. Each data block is combined with a feedback block generated from previous operations.
   - A block permutation or encryption function is used to transform the internal state and the data block.
3. Feedback generation.
   - Feedback is updated by combining the output of the cipher with the input data, ensuring that each block of data influences the next operations.
4. Encryption/Decryption.
   - The plaintext is encrypted by combining each block of plaintext with the output of encryption (or permutation) and feedback.
   - For decryption, the process is reversed, using the same feedback sequence to find the plaintext from the ciphertext.
5. Authentication.
   - An authentication tag is calculated from the final state of the feedback and the processed data, guaranteeing data integrity and authenticity.

The overall COFB process is illustrated on Figure 2.12.

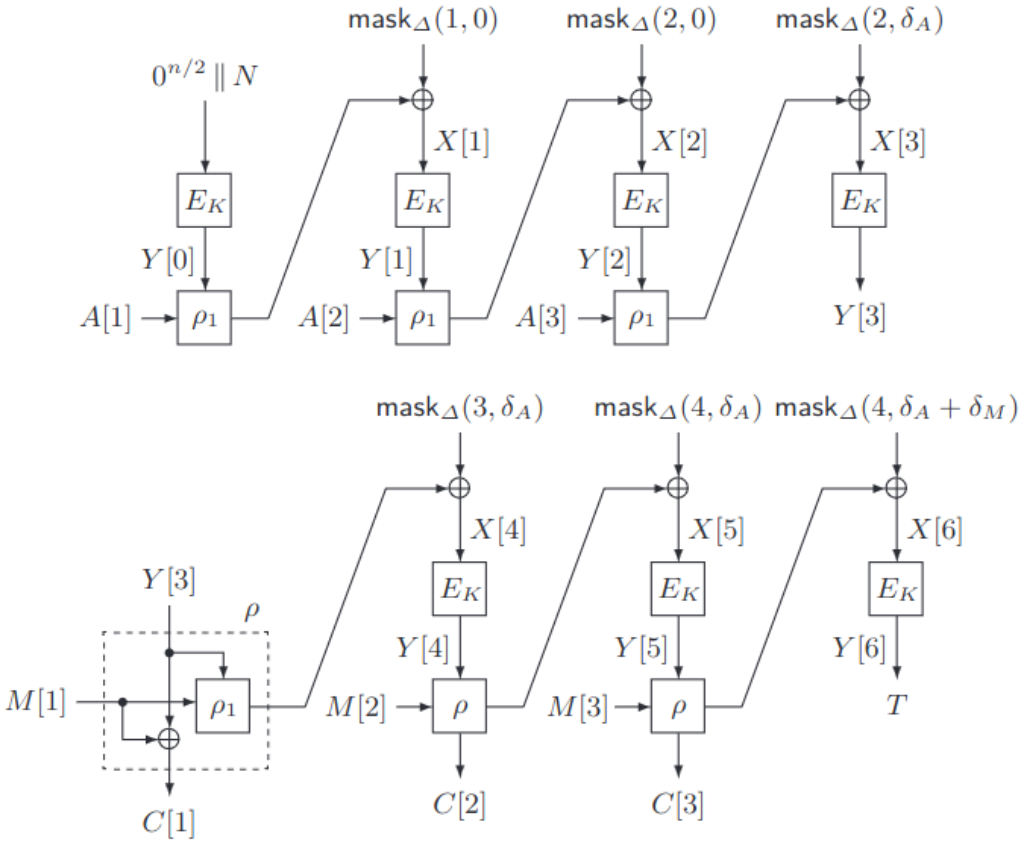


Figure 2.12: encryption of COFB for 3-block associated data and plaintext. Reproduced with permission from [5]. ©TCHES 2017

For COFB and Beetle, the advantage of using the ρ function over CBC and MonkeyDuplex lies in improved security.

Offset codebook (OCB)

Offset codebook (OCB) [32] is an authenticated encryption mode that combines encryption and authentication operations to offer both data confidentiality and integrity. In 2001, it was developed by Phillip Rogaway, Mihir Bellare, John Black and Ted Krovetz.
In addition, it is designed to be highly efficient, enabling high performance in terms of encryption and authentication speed. We describe the main steps involved in offset codebook (OCB) mode.

1. Initialization.
   - A unique initialization vector (IV) is chosen for each message. The IV is used to generate an initial offset mask.
2. Data block cipher.
   - The message is divided into fixed-size blocks.
   - Each block is combined with a unique offset mask, derived from the IV and systematically updated for each block.
   - The combined block is then encrypted using the block cipher algorithm.
   - The ciphertext for each block is produced after this operation.
3. Authentication of associated data.
   - Associated data is authenticated but not necessarily encrypted.
     They are processed in a similar way to produce an intermediate authentication value.

4. Authentication tag generation.
   - An authentication tag is calculated by combining the results of encrypted data blocks and associated data.
   - The tag is used to verify the integrity and authenticity of the message during decryption.

ΘCB

ΘCB [33] is a variant of offset codebook (OCB) that uses a tweakable block cipher (TBC) instead of a standard block cipher. In 2002, it was developed by Tadayoshi Kohno, John Viega, and Doug Whiting.
The main difference lies in ΘCB's use of tweakable block cipher (TBC)s, whereas OCB uses block ciphers.
ΘCB brings notable improvements by using tweaks. This additional flexibility improves security.

2.3 Physical cryptanalysis

2.3.1 Side channel analysis (SCA)

Even if an algorithm has been proven to be mathematically secure, its implementation can open the gate to physical attacks. SCA are a subcategory of physical attacks. They exploit the fact that some physical states of a device depend on intermediate values during the computation. This is the so-called leakage of information of the circuit. It could be used to retrieve sensitive data, such as secret keys, or to reverse engineer an algorithm. An SCA is often led with a divide-and-conquer approach. It is a domain in its own right, as shown different books [34, 35]. Namely, the secret is divided into small pieces that are analysed independently.

Different kinds of leakage sources can be exploited as execution time [36], power consumption [37] or electromagnetic (EM) radiations [38]. In this section, a power consumption or EM leakage channel are considered. At each instant, the measurement of the intensity of the electric current reflects the activity of the circuit. The power consumption of a device is a combination of the power consumption of each of its logic gates.

In the literature, simple power analysis (SPA) [39] is called simple because it determine directly, from an observation of the power consumption, during a normal execution of an algorithm, information on the calculation performed or the manipulated data. Other attacks like correlation power analysis (CPA) [40] use a mathematical model for the leakage. A confrontation between measurement and model is performed. More precisely, a statistic tool called distinguisher gives score to the different targets. Template attacks are statistical categorizations [41] that require no leakage model a priori.

2.3.2 Simple power analysis (SPA)

Simple power analysis (SPA) [42] is the term given to the direct interpretation of power consumption observed during cryptographic operations, and it can reveal a wealth of information about the target.

A example with a RSA [43] signing operation. This operation uses a message M that is raised to a private exponent d producing the signature $M^d \pmod N$, with N is the product of these two prime numbers, i.e. $N = p \times q$, N is part of the public key. If the exponentiation uses a naive square-and-multiply algorithm, the sequence of performed square and multiply operations reveals the secret exponent d . This sequence of operations may be distinguishable in the power trace, and thus be susceptible to a simple power analysis (SPA). In the classical consumption model, each square operation will appear as a 0 and each multiply as a 1, hence leaking the secret exponent.

2.3.3 Profiled Attacks

In 2002, Suresh Chari et al. introduced "Template Attacks" [44]. They argue that these attacks represent the most powerful form of possible SCA, based on multivariate probability density functions. Unlike other approaches that classify data into different categories, they introduced a profiling technique. This method directly compares collected traces with a pre-established model, thus avoiding the need to construct a leakage model. Template attack (TA)s involve comparing traces collected from a target device with a model of how that device operates to obtain the secret key. To conduct a template attack (TA), the attacker must have a complete replica of the target device and perform intensive preprocessing to create the model. Although this

process requires many power traces, once the model is established, only a few traces from the target are needed to recover the keys. There are four phases in a model attack.

1. Using a replica of the target device, perform encryption operations with different combinations of plaintexts and keys, then collect a large number of power traces. Make sure to gather enough traces to be able to distinguish the value of each sub key.
2. Develop a working model of the device. This model will identify specific "points of interest" in the traces and establish a multivariate distribution of power traces for this set of points.
3. Collect a limited number of power traces from the targeted device.
4. Use the model on the collected traces. Analyze each sub key and calculate the most probable values based on their fit to the model. Continue until the key is fully recovered.

Machine learning (ML) algorithms are algorithms that learn to solve a problem without being explicitly programmed to do so. In this context, “to learn” can be perceived as “to build an empirical model using training data”, whereas “to solve” can be perceived as “to evaluate real data using the model”. A profiled side-channel attack as described in this section can be reduced to a classifying task, which is thoroughly studied in the context of supervised machine learning.
Profiled machine learning-based attacks [45] are typically performed in a similar fashion as the template attack, and can often be classified as one. They also share many of its advantages and disadvantages. The crucial difference is in the choice of the empirical model; instead of Gaussian, a machine learning-based classifier is used. A support vector machine (SVM) [46] is a machine learning algorithm commonly used for classification, based on creating an optimal hyperplane between different classes. It was shown to be more efficient than a Gaussian template attack in some aspects.

2.3.4 Non-Profiled Attacks

2.3.4.1 Leakage model

The power consumption or electromagnetic leakages are very correlated to the Hamming weight (HW) of the data. It is a classical model used in the domain of SCA.

One important advantage of HW is that it rapidly reduces guesses. For example, let x be a byte, so x can take 256 values in $[\![0, 255]\!]$. With the HW of x, the attacker reduces the list of possible values, as shown in Table 2.3.

Table 2.3: Number of possible values for a byte x according its HW. This table comes from [47]

HW(x)	0	1	2	3	4	5	6	7	8
$\#x$	1	8	28	56	70	56	28	8	1

Another model for the leakage is a HW with an additive Gaussian noise. For a given discrete random variable byte x, and its HW(x) $\in [\![0, 8]\!]$; $\tilde{\text{HW}}(x)$ denotes the continuous random variable representing the simulated Hamming weight so called **noisy**; defined in $\mathbb{R}$ as:

$$\tilde{\text{HW}}(x) = \text{HW}(x) + \sigma_{x,t} \quad ; \tag{2.8}$$

with $\sigma_{x,t}$ an event of the Gaussian random variable $\mathcal{N}\left(0, \sigma^2\right)$ at a time t. The probability density function $\mathbf{F}$ associated to $\mathcal{N}\left(0, \sigma^2\right)$ is given by:

$$\mathbf{F}_\sigma(x) = \frac{1}{\sigma \cdot \sqrt{2\pi}} \cdot \exp\left(-\frac{1}{2} \cdot \left(\frac{x}{\sigma}\right)^2\right) \quad . \tag{2.9}$$

In chapter 4 and 5, we use HW and $\tilde{\text{HW}}$ 2.8 respectively as leakage model.

2.3.4.2 Differential power analysis (DPA)

In 1999, Paul Kocher, Joshua Jaffe, and Benjamin Jun published a paper titled "Differential Power Analysis" [37] considered the cornerstone of research on SCA at the device level. This pioneering work extends the testing of the mathematical structure of an algorithm to that of a device implementing the cryptographic algorithm. The authors posited that security vulnerabilities often involve unexpected interactions between components designed by different individuals. One such unexpected interaction concerns energy consumption during encryption, a subject that was scarcely addressed in academic literature before their 1999 publication. In practice, the challenge of directly observing information on secret keys poses a difficulty in distinguishing them from traces. Kocher et al. [37] developed the idea of a model-based SCA, introducing a selection function [48] to group traces into two data sets or classes. A statistic is then used to compare these classes and determine if they differ significantly. In differential power analysis (DPA), the first step involves reducing all traces from each class to a master trace using the first moment or the average, and then comparing these master traces to detect significant differences.

2.3.4.3 Correlation power analysis (CPA)

In 2004, their work "Correlation Power Analysis with a Leakage Model " [40], Brier *et al.* furthered the work on DPA and TA. They explored the possibility of examining multiple positions in a power trace to find correlations, rather than being limited to a single point in time. Their model-based approach uses a multivariate method to form and distinguish classes, instead of comparing a single descriptor for a single point in time, such as the difference of means. Additionally, when examining multiple bits at once, they emphasize that power consumption depends solely on the number of logically "1" bits, not on the number of bits they represent.

2.3.4.4 Differential analysis aided power attack (DAPA)

In 2017, Dobraunig et al. [49] demonstrated that DPA can be used on shift registers to extract relation between neighboring bits, significantly reducing the guess space of the internal state. Specifically, by knowing whether two neighboring bits have the same or different values, guessing the value of one bit allows determining the value of the other. In other words, this reduces the entropy of a bit for each known bit relationship. Inspired by their work, Sim et al. [50] proposed an extension in which they generalize this analysis methodology and combine it with differential analysis, which they call differential analysis aided power attack (DAPA). DAPA allows discovering more bit relationships in a shift register, while also taking into account the linear or nonlinear feedback function.
The DAPA attack method breaks down into three steps:

1. identification of differential patterns,
2. power measurements,
3. recovery of the internal state.

2.3.4.5 Soft analytical side channel attack (SASCA)

DPA requires a large amount of data (often millions of traces). In contrast, analytical side channel attack (ASCA) [51] requires less data but is very sensitive to noise. Soft analytical side channel attack (SASCA) [52] balances these characteristics, tolerating more noise while having low data complexity, similar to a low-density parity-check (LDPC) code [53]. SASCA are powerful methods to perform SCA. SASCA algorithms are mostly based on BP theory.

Thus, SASCA begins with a template attack on the targeted variable, then obtains the posterior probabilities based on the leaks. It can be divided into three steps.

1. Template attack on the targeted variable.
2. Calculation of posterior probabilities based on the leaks with BP.
3. Recovery of the secret information using these probabilities.

2.3.4.6 Blind side channel analysis (BSCA)

The blind side channel analysis (BSCA) family is new improvement in SCA. Linge *et al.* has presented the concept in [54]. In parallel, Le Bouder *et al.* published an attack in [55]. Then, these works have been improved by Clavier *et al.* in [56], moreover this contribution introduces for the first time, the term of blind side channel. Now it is a new family of SCA [7, 57–59].

The main idea is to only perform the attack on the leakage measurements *i.e*, without data such as plaintexts or ciphertexts.

In BSCA, the Hamming weight (HW) leakage model have often been used and a strong assumption is made: the attacker is supposed to retrieve a HW in chap 4 and noisy HW in chap 5 from the leakage.
In chap 4, the considered adversary model is that the HW of all manipulated intermediate variables can be recovered by the attacker.
In chap 5, for Elephant, the considered adversary model is that the noisy HW of all manipulated intermediate variables can be recovered by the attacker. And this same model is used for Sparkle. Several techniques, such as signal filtering, trace averaging or templates [44], can be used in order to fulfill this prerequisite.

Chapter 3

An In-depth Exploration of SCA Against AEAD

"*En plein coeur de toute difficulté se cache une possibilité.*"
Albert Einstein.
Proposée par Papa Moussa GUEYE.



Contents

3.1 Introduction **41**
3.2 Ascon **42**
3.2.1 Overview of the design 42
3.2.1.1 Permutation 43
3.2.2 SCA on Ascon 44
3.3 Elephant **47**
3.3.1 Overview of the design 47
3.3.2 SCA on Elephant 50
3.4 GIFT-COFB **51**
3.4.1 Overview of the design 51
3.4.2 SCA on GIFT-COFB 51
3.5 Grain-128AEADv2 . . . . . **53**
3.5.1 Overview of the design 53
3.5.2 SCA on Grain-128AEADv2 53
3.6 ISAP **54**
3.6.1 Overview of the design 54
3.6.2 SCA on ISAP 54
3.7 PHOTON-Beetle **55**
3.7.1 Overview of the design 55
3.7.2 SCA on PHOTON-Beetle 55
3.8 Romulus **56**
3.8.1 Overview of the design 56
3.8.2 SCA on Romulus-N 56
3.9 Sparkle **57**
3.9.1 Overview of the design 57
3.9.1.1 Zoom on Sparkle 57
3.9.1.2 Alzette 58
3.9.2 SCA on Sparkle 59
3.10 TinyJAMBU **60**
3.10.1 Overview of the design 60
3.10.2 SCA on TinyJAMBU 60
3.11 Xoodyak **61**
3.11.1 Overview of the design 61
3.11.2 SCA on Xoodyak 61
3.12 SCA on LFSRs **62**

3.1 Introduction

In this chapter, we address several crucial aspects. First, we present a brief description of each of the 10 authenticated encryption with associated data (AEAD) algorithms in the NIST competition, examining their primitives and the modes used in their construction. This initial study aims to provide a structured overview of these algorithms, facilitating their understanding and comparison. Only the 3 target algorithms, namely Elephant(chapter 4 and 5), Sparkle(chapter 5) and Ascon(chapter 6), have been detailed for the purposes of their respective chapters.

In addition, we dive into an in-depth exploration of SCA on each of these 10 AEAD algorithms. This detailed analysis identifies potential vulnerabilities and challenges associated with securing these cryptographic systems against side channel analysis (SCA).

Finally, we review the state of the art of SCA on linear feedback shift register (LFSR), which represent one of the main targets of our contributions. We review existing methodologies, the results obtained, and situate our work in relation to these advances to justify our contributions in the next chapters 4 and 5 of this thesis.

For easier reading, we define the following symbols.

1. A practical attack.
   -  Represents that the attacker retrieves the entire secret key.
   -  Represents that the attacker finds a part of the secret key.
   -  Represents that the attacker is unable to find the key.
2. A theoretical or simulated attack.
   -  Represents that the attacker retrieves the entire secret key.
   -  Represents that the attacker finds a part of the secret key.
   -  Represents that the attacker is unable to find the key.

#T represents the number of traces. Sometimes this is not specified in articles with practical attacks, which is why it is not filled in the tables.

3.2 Ascon

3.2.1 Overview of the design

Ascon family [6] is a permutation-based AEAD and a hashing scheme to provide secure encryption and authentication in resource-constrained environments. Ascon-AEAD uses the monkeyDuplex mode [3], with additional keys added at process initialization and finalization. The Ascon family, which includes Ascon-128 and Ascon-128a, was chosen as the main choice for lightweight authenticated encryption in the competition for authenticated encryption (CAESAR) competition. In 2023, he also won the NIST competition on lightweight cryptography.
The algorithm take four inputs: plaintext P, associated data A, nonce n , and a key K. It outputs the authenticated ciphertext C and an authentication tag T. The algorithm comprises four operation phases: initialization, associated data process, plaintext process (ciphertext process in decryption), and finalization.

In Ascon-128, the input of the initialization phase is a 320-bit initial state $(IV||K||N)$ consisting of the 64-bit constant IV , the 128-bit key K, and the 128-bit fresh nonce N) in the form of five 64-bit words x_0 to x_4. This five-word state S updates through the algorithm phases and is used as the secret state for encryption (decryption) and tag generation. The initialization phase includes twelve (same) rounds that update the initial state. The permutation is explained and detailed in the section 3.2.1.1.

In this phase, when an associated data block (A_i) is received, its first $r = 64$ bits is XORed to the first $r = 64$ bits of the state S, then the whole state S is permuted using the 6-round variant of the permutation. The associated data processing phase updates the state S using the associated data blocks. Then, the updated state attains the plaintext process phase. The plaintext process phase XORs the 64-bit plaintext block P_i with the first $r = 64$ bits of the state S to produce ciphertext block C_i. Then, the whole state S is transformed by the permutations to update the state for the next plaintext block.
The finalization phase XORes the key with the state S and transforms the results to provide the 128-bit authentication tag T.

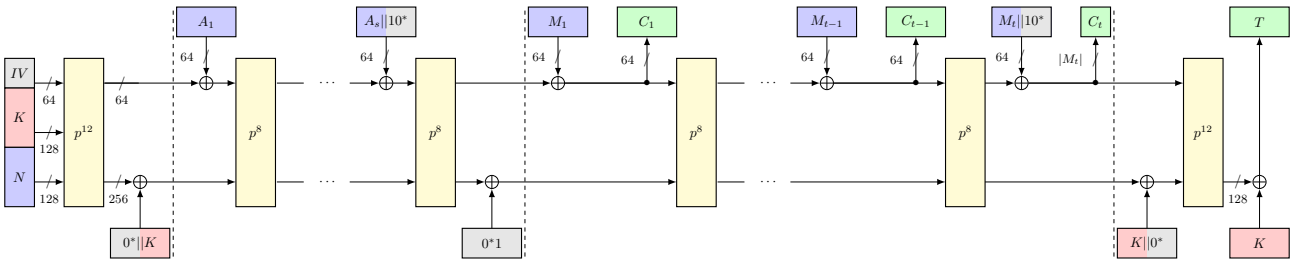


Figure 3.1: Ascon. This figure comes from [6] according to the description of Ascon [6].

Variants. The AEAD variants of Ascon are provided below in Fig. 3.2. The number of rounds for each AEAD variant is represented as a 3-tuple, which corresponds to the number of rounds during initialization, Associated data (AD) and message processing, and finalization, respectively. Ascon team also defined Ascon-80pq to provide stronger resistance against quantum key recovery attacks.

	size in bits				
AEAD variants	Key	Nonce	Tag	Block	#Round
Ascon-128	128	128	128	64	12/6/12
Ascon-128a	128	128	128	128	12/8/12
Ascon-80pq	160	128	128	64	12/6/12

Figure 3.2: The AEAD variants of Ascon

3.2.1.1 Permutation

The main components of the AEAD and hashing schemes of Ascon are the two 320-bit permutations p^a and p^b. The permutations iteratively apply an SPN-based round transformation p that in turn consists of three steps p_C , p_S , p_L :

$$p = p_L \circ p_S \circ p_C.$$

p^a and p^b differ only in the number of rounds. The number of rounds a and the number of rounds b are tunable security parameters.

For the description and application of the round transformations, the 320-bit state S is split into five 64-bit registers words x_i, $S = x_0||x_1||x_2||x_3||x_4$.

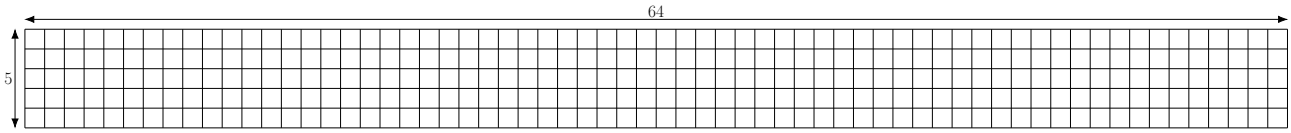


Figure 3.3: State. This figure comes from [6] according to the description of Ascon [6].

3.2.1.1.1. Addition of constants

The constant addition step p_C adds a round constant c_r to register word x_2 of the state S in round i, is depicted on Fig. 3.4.

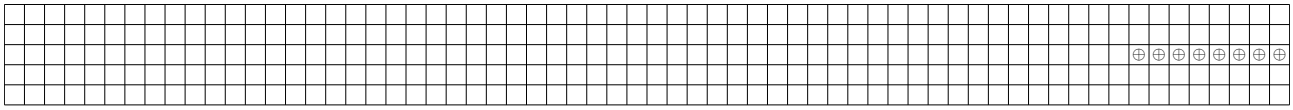


Figure 3.4: Round constant addition p_C. This figure comes from [6] according to the description of Ascon [6].

3.2.1.1.2. Substitution layer

The substitution layer p_S updates the state S with 64 parallel applications of the 5-bit S-box, is depicted on Fig. 3.6 and Fig. 3.5.

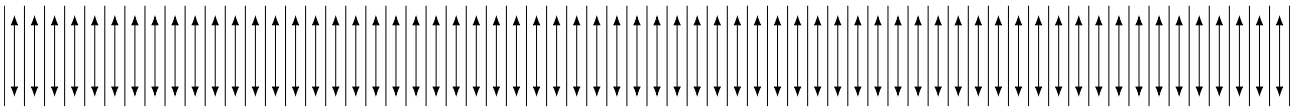


Figure 3.5: Substition layer p_S with 5-bit S-box.. This figure comes from [6] according to the description of Ascon [6].

3.2.1.1.3. Linear diffusion layer

The linear diffusion layer p_L provides diffusion within each 64-bit register word x_i, is depicted on Fig. 3.7.

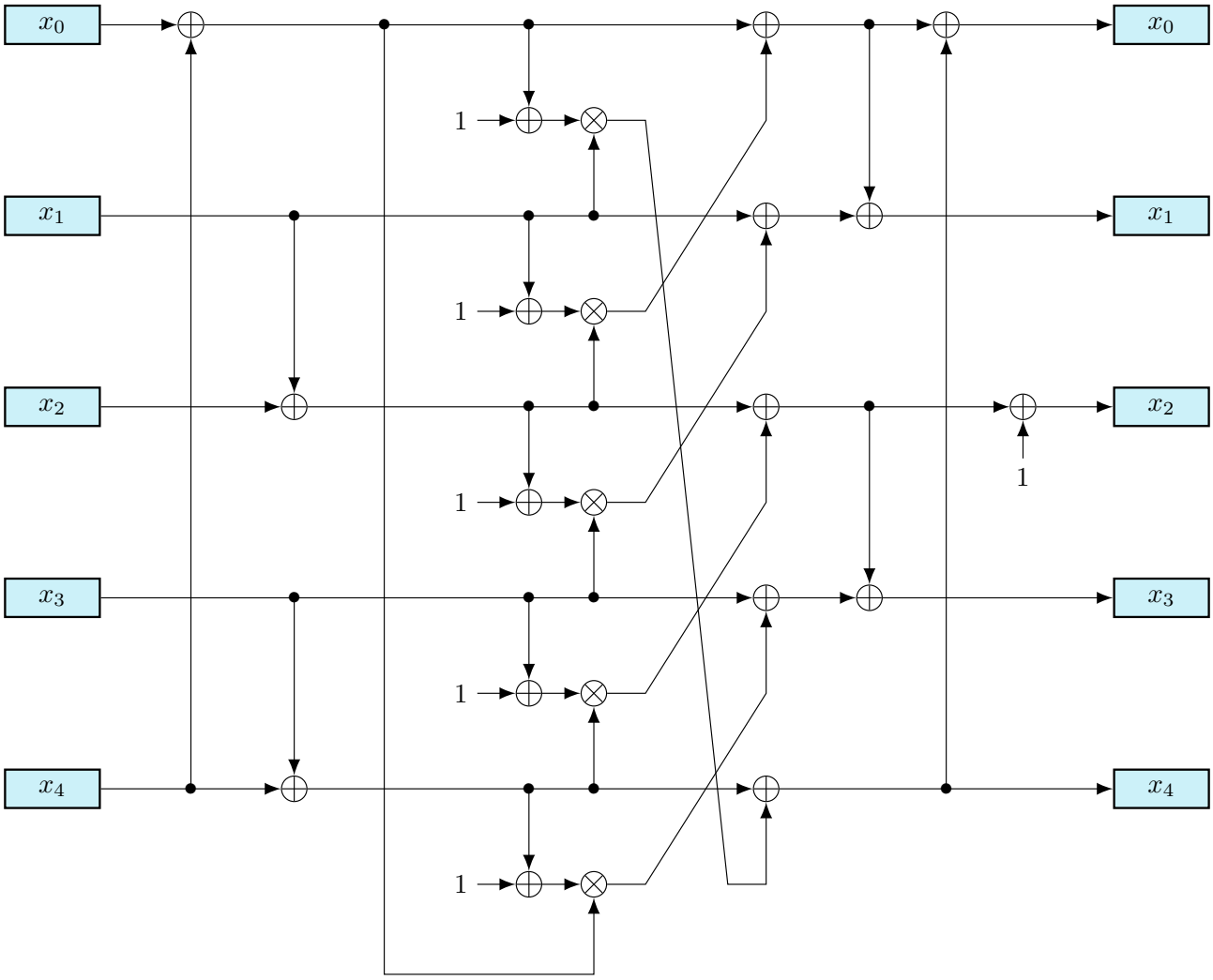


Figure 3.6: Substition layer p_S with 5-bit S-box. This figure comes from [6] according to the description of Ascon [6].

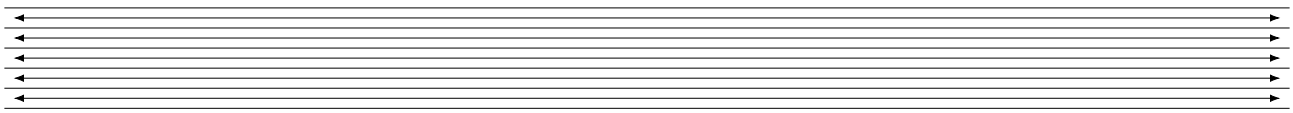


Figure 3.7: Linear layer with 64-bit diffusion function. This figure comes from [6] according to the description of Ascon [6].

3.2.2 SCA on Ascon

In 2017, the authors [60] discuss side-channel resistance, noting success in uncovering keys from an unprotected implementation with 500 power traces on average, focusing on the S-Box output of Ascon-x-low-area design. They employed 128 power analysis attacks with a SAT solver to find the secret key with 1000 power traces in Ascon-fast design. However, attacks on the protected implementation (Ascon TI) were unsuccessful, despite capturing over 1 million power traces, suggesting its robust resistance to side-channel attacks.

In 2020, another attack on Ascon [61], known as side channel analysis with reinforcement learning (SCARL), is based on deep learning. The authors utilized a lightweight implementation on an Artix-7 FPGA. Despite attempting DPA and CPA attacks with 40,000 power traces, these attacks were unsuccessful. However, the side channel analysis with reinforcement learning (SCARL) required only 24,000 power traces to uncover the secret key. This highlights the efficacy of deep learning-based approaches in SCA, especially in comparison to traditional DPA and correlation power analysis (CPA).

In 2022, Sinian et al. [62] conducted a soft analytical side channel attack (SASCA) on Ascon's encryption, focusing on its permutation function. They developed a unique factor graph to model intermediate computations and used a BP algorithm to exploit leakages. Through simulations, they recovered the encryption key using minimal traces, demonstrating low time and memory complexity.

In 2023, the study by Rezaeezade et al. [63] employs an ensemble of deep neural networks to attack two implementations of Ascon. They successfully uncover the secret key for the Ascon-protected version with less than 3,000 traces, a leading achievement. Furthermore, they identify the correct key with fewer than 100 traces for the unprotected implementation, comparable to state-of-the-art results.

In 2023 [64], Weissbart et al.'s study reveals vulnerabilities in the reference implementation of Ascon, where attackers can retrieve the full key with 8,000 traces using CPA and around 1,000 traces with deep learning analysis. While second-order CPA is ineffective, deep learning power analysis (DLPA) can recover partial keys with 800 traces on the protected implementation. However, the multi-task key recovery model lacks the generalization needed to correctly recover all partial keys for the full key attack.

Table 3.1: side channel analysis (SCA) on Ascon.

SCA on Ascon															
CPA			DPA			DLPA			SASCA			SCARL			
Unprotected															
Result	#T	Chip	Result	#T	Chip	Result	#T	Chip	Result	#T	Chip	Result	#T	Chip	
● [64]	8K	STM32F4	● [61]	40K	Artix-7	● [64]	1K	STM32F4	● [62]	few traces	Microcontoller	● [61]	24K	Artix-7	
● [61]	40K	Artix-7				● [63]	100	STM32F4							
Protected															
● [64]	8K	STM32F4				◐ [64]	800	STM32F4							
						● [63]	3K	STM32F4							

3.3 Elephant

3.3.1 Overview of the design

Algorithm 1 Elephant encryption algorithm `enc`

Input: $(K, N, A, M) \in \{0,1\}^{128} \times \{0,1\}^{96} \times \{0,1\}^* \times \{0,1\}^*$
Output: $(C, T) \in \{0,1\}^{|M|} \times \{0,1\}^t$

$M_1, \cdots, M_{\ell_M} \leftarrow \mathsf{Split}(M)$
for $t \leftarrow 1$ to ℓ_M **do**
  $C_t \leftarrow M_t \oplus \mathsf{P}(N||0^{n-96} \oplus \mathsf{mask}_K^{t-1,1}) \oplus \mathsf{mask}_K^{t-1,1}$
end for
$C \leftarrow \mathsf{Trunc}_{|M|}(C_1||\cdots||C_{\ell_M})$
$T \leftarrow 0^n$
$A_1, \cdots, A_{\ell_A} \leftarrow \mathsf{Split}(N||A||1)$
$C_1, \cdots, C_{\ell_C} \leftarrow \mathsf{Split}(C||1)$
$T \leftarrow A_1$
for $t \leftarrow 2$ to ℓ_A **do**
  $T \leftarrow T \oplus \mathsf{P}(A_t \oplus \mathsf{mask}_K^{t-1,0}) \oplus \mathsf{mask}_K^{t-1,0}$
end for
for $t \leftarrow 1$ to ℓ_C **do**
  $T \leftarrow T \oplus \mathsf{P}(C_t \oplus \mathsf{mask}_K^{t-1,2}) \oplus \mathsf{mask}_K^{t-1,2}$
end for
$T \leftarrow \mathsf{P}(T \oplus \mathsf{mask}_K^{0,0}) \oplus \mathsf{mask}_K^{0,0}$
return $(C, \mathsf{Trunc}_\tau(T))$

Elephant [8, 65] is a finalist to the NIST lightweight cryptography competition. It is a nonce-based authenticated encryption with associated data (AEAD). Its construction is based on an Encrypt-then-MAC that combines CTR-mode encryption with a variant of the protected counter sum [66, 67]. Elephant uses a cryptographic permutation masked with LFSRs in an Even-Mansour-like fashion [68] in place of a blockcipher.

Let P be an n-bit cryptographic permutation, and φ an n-bit LFSR. Let the function $\mathsf{mask} : \{0,1\}^{128} \times \mathbb{N} \times \{0,1,2\} \to \{0,1\}^n$ be defined as follows:

$$\mathsf{mask}_K^{t,b} = (\varphi \oplus \mathsf{id})^b \circ \varphi^t \circ \mathsf{P}(K||0^{n-128}) \tag{3.4}$$

Let $\mathsf{Split}(X)$ be the function that splits the input X into n-bit blocks, where the last block is zero-padded. Let $\mathsf{Trunc}_\tau(X)$ be the τ left-most bits of X.

Encryption `enc` under Elephant gets as input a 128-bit key K, a 96-bit nonce N, associated data $A \in \{0,1\}^*$, and a plaintext $M \in \{0,1\}^*$. It outputs a ciphertext C as large as M, and a t-bit tag T. The description `enc` is given in **Algorithm** 1 and is depicted on Fig. 3.8.

Decryption `dec` gets as input a 128-bit key K, a 96-bit nonce N, associated data $A \in \{0,1\}^*$, a ciphertext $C \in \{0,1\}^*$, and τ-bit tag T. It outputs a plaintext M as large as C if the tag T is correct, or the symbol $\perp$ otherwise. The description of `dec` is analoguous to the one of `enc`.

Elephant comes in three flavours which differ on the n-bit cryptographic permutation P and the LFSR φ used, as well as the tag size t.

Dumbo uses the 160-bit permutation Spongent-π[160] [69], the LFSR φ_{Dumbo} given by equation (3.1) and illustrated on Fig. 3.9, and has tag size $\tau = 64$ bits.

Jumbo uses the 176-bit permutation Spongent-π[176] [69], the LFSR φ_{Jumbo} given by equation (3.2) and illustrated on Fig. 3.10, and has tag size $\tau = 64$ bits.

Delirium uses the 200-bit permutation Keccak-f[200] [30, 31], the LFSR $\varphi_{\mathsf{Delirium}}$ given by equation (3.3) and illustrated on Fig. 3.11, and has tag size $\tau = 128$ bits.

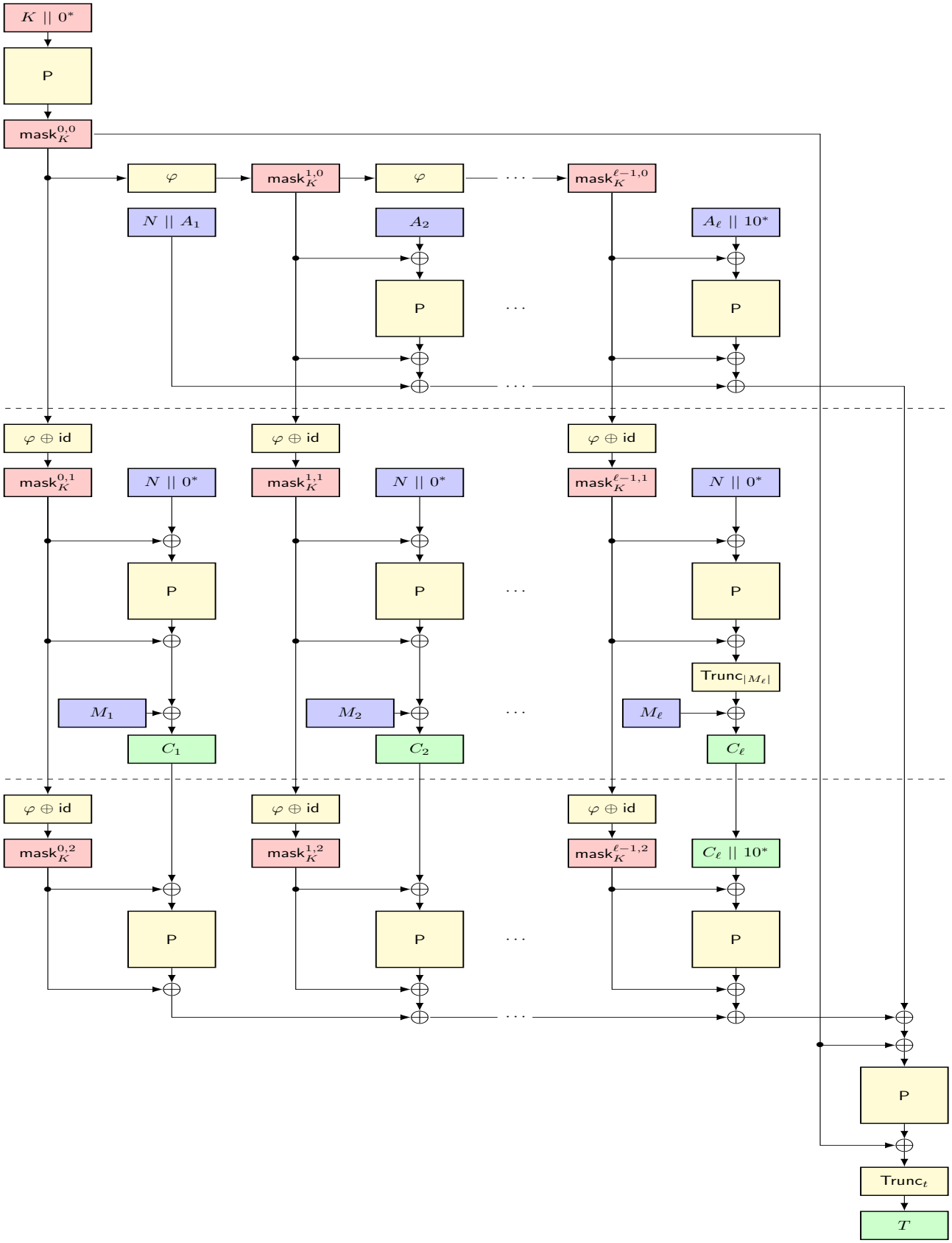


Figure 3.8: Elephant associated data authentication (top), plaintext encryption (middle), and ciphertext authentication (bottom). This figure comes from [7] according to the description of Elephant [8].

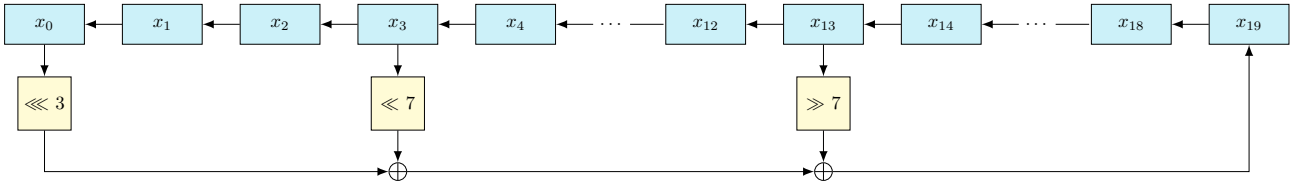


Figure 3.9: 160-bit LFSR φ_{Dumbo}. This figure comes from [7] according to the description of Elephant [8].

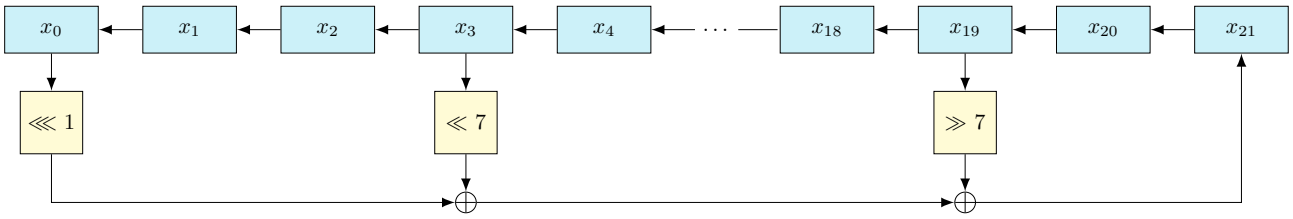


Figure 3.10: 176-bit LFSR φ_{Jumbo}. This figure comes from [7] according to the description of Elephant [8].

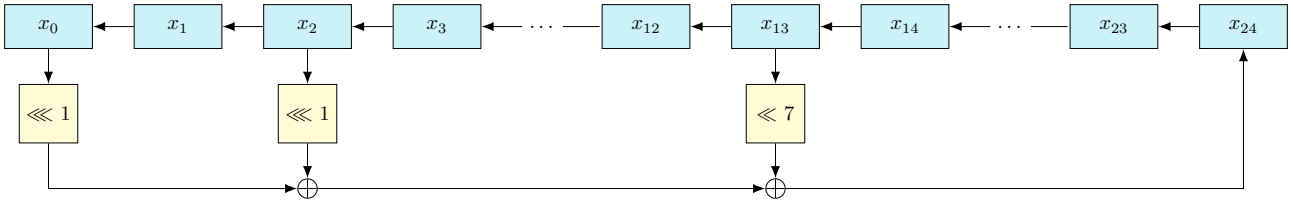


Figure 3.11: 200-bit LFSR $\varphi_{\mathsf{Delirium}}$. This figure comes from [7] according to the description of Elephant [8].

$$\varphi_{\mathsf{Dumbo}} : (x_0, \cdots, x_{19}) \mapsto (x_1, \cdots, x_{19}, x_0 \lll 3 \oplus x_3 \ll 7 \oplus x_{13} \gg 7) \tag{3.1}$$

$$\varphi_{\mathsf{Jumbo}} : (x_0, \cdots, x_{21}) \mapsto (x_1, \cdots, x_{21}, x_0 \lll 1 \oplus x_3 \ll 7 \oplus x_{19} \gg 7) \tag{3.2}$$

$$\varphi_{\mathsf{Delirium}} : (x_0, \cdots, x_{24}) \mapsto (x_1, \cdots, x_{24}, x_0 \lll 1 \oplus x_2 \lll 1 \oplus x_{13} \ll 7) \tag{3.3}$$

The first two variants are well suited to hardware implementation, while the third is specifically designed for software implementation. ELEPHANT is the only finalist based on a parallel mode.

Variants. The AEAD variants of the ELEPHANT family are listed below.

	size in bits					
AEAD variants	Key	Nonce	Tag	Block	Permutation	#Round
Dumbo	128	96	64	160	160-bit Spongent	80
Jumbo	128	96	64	176	176-bit Spongent	90
Delirium	128	96	128	200	Keccak	18

Figure 3.12: The AEAD variants of the ELEPHANT

3.3.2 SCA on Elephant

This study examines SCA on Elephant, focusing on five publications.
In 2021, the first publication [70] does not address a specific attack but rather focuses on test vector leakage assessment (TLVA) using Welch's t-test on both unprotected and protected implementations. Results indicate that the unprotected implementation exceeded the threshold of 4.5 with only 2K power traces, whereas the protected implementation did not exceed the threshold even with 100K power traces.

In 2022, the second publication [71] relies on correlation power analysis (CPA) and is designed for the Dumbo and Jumbo variants. The author utilized an ARM Cortex-M4 microcontroller and achieved full key discovery with approximately 30 power traces. A reference C implementation without any protection was utilized.

In 2022, with the third publication [7], Meraneh et al. present a theoretical blind side channel attack targeting the LFSR of the Elephant algorithm. The authors exploit the usage of intermediate variables that are statistically dependent to the secret (the secret LFSR initial state) and show that this structure could threaten the security of a cryptosystem's regarding SCA. Also, the study of the influence of the choice of the LFSR is presented.

In 2022, an extended version of a previous attack [7] was proposed by Julien et al [72]. The authors present a major improvement: an optimization to find the best time, relative to the beginning of encryption to start the attack.
This attack is a contribution of my thesis, it is described in the chapter 4.

In 2023, in the last publication, Modou et al. [73] proposed a major improvement of a first blind side channel analysis (BSCA) on Elephant. The leakage is considered as an Hamming weight (HW) with a Gaussian noise. A belief propagation (BP) algorithm is used to link the different leaks. This attack is a contribution of my thesis, it is described in the chapter 5.

Table 3.2: side channel analysis (SCA) on Elephant

SCA on Elephant					
Unprotected					
CPA			BSCA		
Result	#T	Chip	Result	#T	Chip
[70]	30	Corter M4	[7] [72] [73]		
Protected					

3.4 GIFT-COFB

3.4.1 Overview of the design

GIFT-COFB [9] is an authenticated encryption with associated data (AEAD) based on block ciphers. It uses GIFT-128 as the underlying block cipher. In addition, the mode used by GIFT-COFB is a variation of the combined feedback (COFB) mode. This COFB mode requires only one call to the block cipher for each input block.

Variants. The single variant of the GIFT-COFB family is listed below. The underlying block cipher GIFT 128-bit [74] consists of 40 rounds.

	size in bits			
AEAD variants	Key	Nonce	Tag	#Round
GIFT-COFB	128	128	128	40

Figure 3.13: The AEAD variant of the GIFT-COFB

3.4.2 SCA on GIFT-COFB

In 2021, Reinbrecht et al. [75] introduced a novel template attack (TA) on GIFT-COFB, called GRINCH attack [75]. They analyzed vulnerabilities in GIFT-COFB and successfully exploited them in their attack. Results demonstrate the effectiveness of the attack, with the ability to recover the full key in less than 400 encryptions.

In 2022, Aljuffri et al. [76] evaluated GIFT 128-bit [74] software implementations against power-based side channel analysis (SCA). They found vulnerabilities in GIFT 128-bit's S-Box against profiled and non-profiled attacks in unprotected or protected implementations using existing balancing or masking techniques. However, their proposed countermeasure, combining balancing and masking, offered full protection with minimal overhead.

In 2022, Benjamin et al. [77] conducted CPA, deep learning power analysis using MLP archictecture (DLPA-MLP), and deep learning power analysis using CNN archictecture (DLPA-CNN) on 10 datasets consisting of 345 power traces of GIFT-COFB, each containing 1250 temporal samples. The authors demonstrate that both CPA, deep learning power analysis using MLP archictecture (DLPA-MLP), and DLPA-CNN successfully recovered all round keys, while DLPA-MLP only managed to recover the round key in 6 out of 10 datasets.

Table 3.3: side channel analysis (SCA) on GIFT-COFB

SCA on GIFT-COFB											
Unprotected											
CPA			DLPA-CNN			DLPA-MLP			TA		
Result	#T	Chip	Result	#T	Chip	Result	#T	Chip	Result	#T	Chip
●	[77]	345	ATXMEGA128D4	●	[77]	345	ATXMEGA128D4	●	[75]		Kintex-7
Protected											

3.5 Grain-128AEADv2

3.5.1 Overview of the design

Grain-128AEADv2 [78] is an AEAD scheme based on stream cipher, specially optimized for hardware implementations. Its main component consists of two bit-oriented feedback shift registers. In 2008, the initial version of the algorithm, called Grain v1, was selected as a finalist in the hardware profile of the eSTREAM portfolio. The design of Grain-128AEADv2 is similar to that of Grain-128a, but has been updated to enable a larger tag size and to support AEAD. Grain-128AEADv2 consists of two main building blocks. The first is a pre-output generator, built using a linear feedback shift register (LFSR), a non-linear feedback shift register (NFSR) and a pre-output function. The last is an authentication generator, consisting of a shift register and an accumulator.

Variants. The AEAD variant of the Grain-128AEADv2 family is listed below.

AEAD variants	Key size (in bits)	Nonce size (in bits)	Tag size (in bits)	#Round
Grain-128AEADv2	128	96	64	512

Figure 3.14: The AEAD variant of the Grain-128AEADv2

3.5.2 SCA on Grain-128AEADv2

Bhasin et al. [79] present a chosen nonce DAPA attack which allows the entire key to be recovered on the 1-bit implementation of Grain-128AEADv2. This attack targets the key and nonce initialization phase of the scheme. The authors also show a way of partially recovering the key from a 32-bit implementation of Grain-128AEADv2 using DAPA techniques. This attack requires 65 instantiations of the cipher with different nonces to recover 64 bits of key in the noise-free scenario of the chosen nonce.

Table 3.4: SCA on Grain-128AEADv2

SCA on Grain-128AEADv2					
Unprotected					
DAPA(1-bit)			DAPA(32-bit)		
Result	#T	Chip	Result	#T	Chip
[79]			[79]		
Protected					

3.6 ISAP

3.6.1 Overview of the design

ISAP [11] is a permutation-based AEAD scheme, specifically designed to provide algorithm-level security against a wide range of implementation attacks, such as differential fault attacks, statistical fault attacks, inefficient statistical fault attacks and differential power consumption analysis.
ISAP's mode of operation is a nonce-based encryption and authentication construct. In this mode, encryption is achieved by performing an XOR between a message and a keystream, while authentication/verification is based on a hash-then-authenticate paradigm.
The authors recommend four instances of ISAP: ISAP-A-128, ISAP-A-128A, ISAP-K-128 and ISAP-K-128A. The first two instances use a 320-bit Ascon permutation, while the last two use a 400-bit Keccak permutation. To limit information leakage, a sponge-based reassignment function is used. This function is responsible for using fresh keys to process new data, helping to reinforce ISAP's security.

Variants. The variants of the ISAP family are listed below.

AEAD variants	Key(in bits)	Nonce(in bits)	Tag(in bits)	Permutation	Rate	#Round
ISAP-A-128a	128	128	128	320-bit Ascon	1/64	12/1/6/12
ISAP-K-128a	128	128	128	400-bit Keccak	1/144	16/1/8/8
ISAP-A-128	128	128	128	320-bit Ascon	1/64	12/12/12/12
ISAP-K-128a	128	128	128	400-bit Keccak	1/144	20/12/12/12

Figure 3.15: The AEAD variants of the ISAP

The Rate column provides two rate values, one for the nonce processing in the rekeying function IsapRk and the other one for the remaining phases. The #Rounds column provides the number of rounds for the authentication phase, nonce processing in the rekeying function, for the encryption and decryption phases, and for generating session keys in the rekeying function, respectively.

3.6.2 SCA on ISAP

There is only one publication [80] concerning the assessment of ISAP lateral channels using techniques such as $\mathcal{X}^2$-test, t-test [81] and CPA. Yuhang Ji shows in this report that ISAP is resistant to DPA based side-channel analysis. He used software implementations from the ISAP team, hardware implementations from institute of applied information processing and communications (IAIK) and hardware implementations from Ruhr University Bochum. Then, a CPA attack was used, but it failed to recover the private key from the given implementations.

Table 3.5: side channel analysis (SCA) on ISAP

SCA on ISAP					
Unprotected					
DPA			CPA		
Result	#T	Chip	Result	#T	Chip
[80]		STM32F303RCT6	[80]		STM32F303RCT6
Protected					
[80]		STM32F303RCT6	[80]		STM32F303RCT6

3.7 PHOTON-Beetle

3.7.1 Overview of the design

PHOTON-Beetle [12] is a permutation-based AEAD and hashing scheme. The basic permutation used in the PHOTON-Beetle family is the 256-bit PHOTON_{256} permutation, which consists of 12 rounds. PHOTON-Beetle-AEAD is based on a sponge-like AEAD mode Beetle with a combined feedback (COFB)(inspired by the COFB mode).

Variants. The variants of the PHOTON-Beetle family are listed below. The Rate column provides the absorbing and the squeezing rates, respectively.

AEAD variants	Key(in bits)	Nonce(in bits)	Tag(in bits)	Rate (in bits)	#Round
PHOTON-Beetle[128]	128	128	128	128/128	12
PHOTON-Beetle[32]	128	128	128	32/128	12

Figure 3.16: The AEAD variant of the PHOTON-Beetle

3.7.2 SCA on PHOTON-Beetle

A known template attack (TA) [82] has been documented on PHOTON-Beetle, focusing on the MixColumnSerial operation of the first round. Approximately 40.000 traces were captured to generate templates. The authors employed HW as a leakage model, dividing traces into 5 sets due to their attack on 4-bit elements. Interestingly, authors noted that even plugging the device into a different USB port of the same computer used for template creation resulted in incorrect predictions.

With 150 power traces captured, the successful key recovery rate was approximately 50%, achieved through an exhaustive search of the four most likely values returned from the template attack. Without this search, the success rate ranged between 12% and 23%.

Table 3.6: side channel analysis (SCA) on PHOTON-Beetle

SCA on PHOTON-Beetle		
Unprotected		
Template attack (TA)		
Result	#T	Chip
[82]	40K	ARM Cortex-M4
Protected		

3.8 Romulus

3.8.1 Overview of the design

Romulus [13] is an AEAD and hashing scheme based on the tweakable block cipher (TBC) Skinny [83]. Romulus-N uses a rate-1 TBC-based combined feedback mode, and the mode of Romulus-M follows a MAC-then-Encrypt approach. the authors submitted four variants:: Romulus-N (nonce-based AE), Romulus-M (nonce misuse-resistant), Romulus-T (leakage resilient) and Romulus-H (hash function). Each of these variants uses TBC Skinny-128-384+, a variant of Skinny-128-384 with 40 rounds instead of 56.

Variants. The variants of the Romulus family are listed below except Romulus-H.

AEAD variants	Key	Nonce	Tag	#Round
Romulus-N	128	128	128	40
Romulus-M	128	128	128	40
Romulus-T	128	128	128	40

Figure 3.17: The AEAD variant of the Romulus

3.8.2 SCA on Romulus-N

Two publications discuss side-channel attacks on Romulus-N. The first one [82] outlines a CPA attack. The author targeted the SubCells of the second round to uncover the 8 most significant bytes of the key, as it's the round immediately following the addition of the secret key. To obtain the 8 least significant bytes of the key, an attack on the SubCells in the third round was necessary, as it's the first time this part of the secret key is utilized. The attack succeeded between 69% and 85% with a number of traces ranging from 1800 to 2400.

The second publication [84] is a report, focuses on the side-channel leakage assessment of first-order masked Romulus-N. The authors conducted Welch's t-test, $\mathcal{X}^2$-test, and deep learning leakage assessment. Welch's t-test [85] and $\mathcal{X}^2$-test yielded statistically significant results only when checking the last bit of the first byte of the input nonce in hardware implementation. However, no statistically significant results were found in software implementation. Consequently, the authors concentrated on software implementation with deep learning leakage assessment. They achieved over 98% accuracy in checking the last bit of the first byte of the input nonce and intermediate value. Additionally, the authors attempted CPA and template attacks, but these were unsuccessful.

Table 3.7: side channel analysis (SCA) on Romulus

SCA on Romulus					
CPA			Template attack (TA)		
Unprotected					
Result	#T	Chip	Result	#T	Chip
[82]	1800 to 2000	STM32F303RCT6			
Protected					
Result	#T	Chip	Result	#T	Chip
[84]		STM32F303RCT6	[84]		STM32F303RCT6

3.9 SPARKLE

3.9.1 Overview of the design

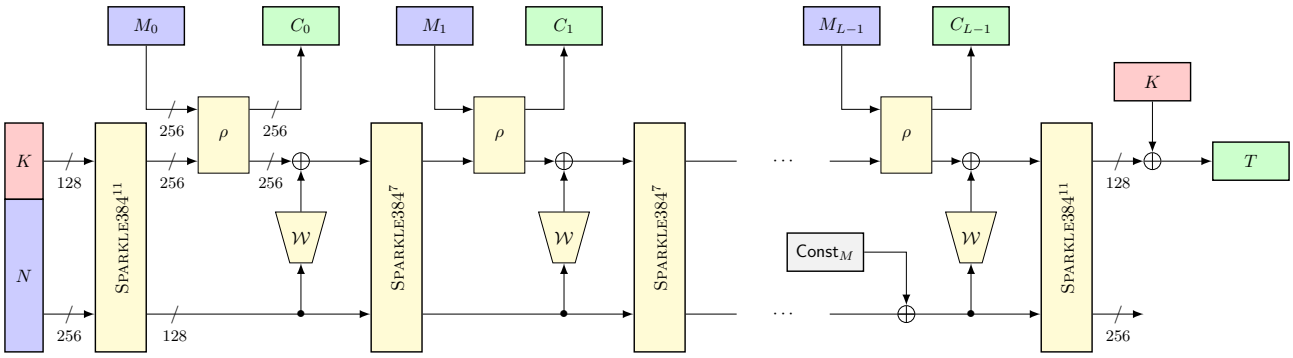


Figure 3.18: Encryption using SCHWAEMM256-128, where SPARKLE384^R is the SPARKLE384 permutation with R rounds, ρ is a linear application, $\mathcal{W}$ the linear whitening layer, and the constant Const$_M$ indicating whether the plaintext M was padded or not.

SCHWAEMM is the AEAD of the SPARKLE [14] submission to the NIST lightweight cryptography competition. It is a permutation-based AEAD that builds upon the Beetle [4] variant of the Duplex [2] construction by adding an extra linear whitening layer $\mathcal{W}$, from the inner state into the outer state. The underlying cryptographic permutation is named SPARKLE which gives the whole submission its name.

Several instances of both algorithms with various performance and security trade-offs are proposed.

The main instance is SCHWAEMM256-128. It is illustrated in Figure 3.18.

SCHWAEMM256-128 takes a 256-bit nonce N, a 128-bit key K, and produces a 128-bit authentication tag T. The corresponding SPARKLE permutation is the 384-bit-wide variant, denoted SPARKLE384.

3.9.1.1 Zoom on SPARKLE

The SPARKLE permuations are based on a substitution permutation network (SPN) structure. The linear layer is a Feistel-type transformation operating on 64-bit branches. It is based on a linear function $\mathcal{M}$ whose description is not necessary for the understanding of this thesis. The non-linear layer is a 64-bit S-box named Alzette $\mathcal{A}_i$, applied in parallel to every branch of the Feistel, where i denotes the index of the branch. More details on Alzette are given in the paragraph at the end of this section.

The number of rounds depends on the particular instance, and whether it is used in the initialisation, encryption or tag generation phase of SCHWAEMM. In the case of SCHWAEMM256-128, the number of rounds is $R = 11$ for the initialisation and tag generation, and only $R = 7$ for encryption. The first round of SPARKLE384 of the initialisation phase of SCHWAEMM256-128 is depicted on Figure 3.19. One important thing to note is that the state is initialised with the concatenation of the nonce N and the secret key K.

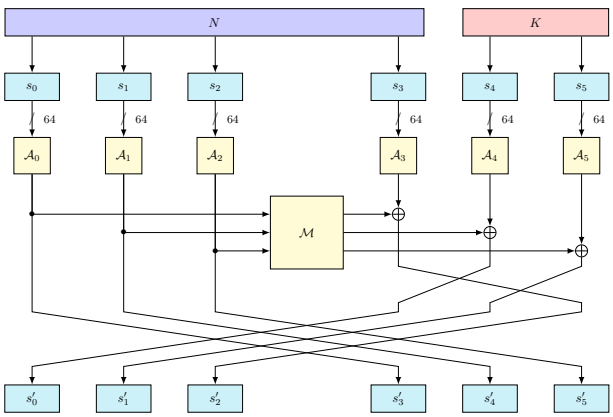


Figure 3.19: The first round of the SPARKLE384 permutation during the initialisation of SCHWAEMM256-128.

3.9.1.2 Alzette

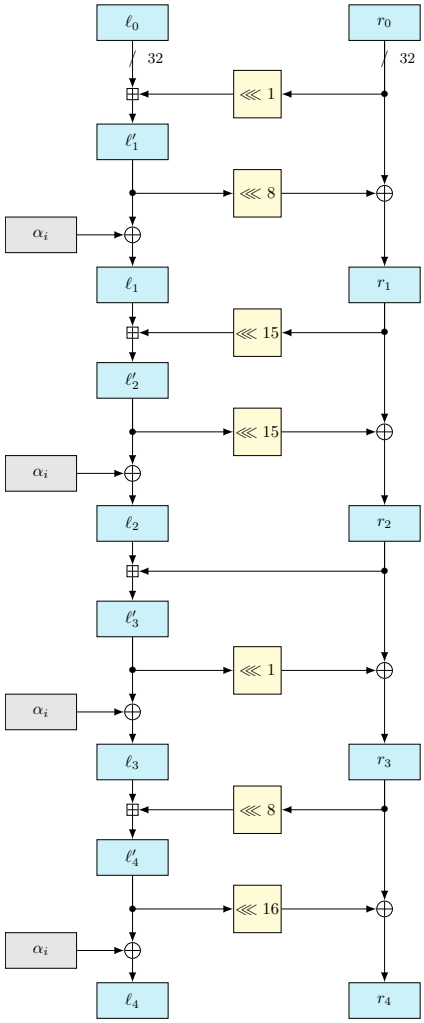


Figure 3.20: The Alzette S-box $\mathcal{A}_i$ used in SPARKLE.

The Alzette S-box $\mathcal{A}_i$ is a 64-bit permutation parametrised by a 32-bit known value α_i that depends only on the index i of branch inside the SPARKLE permutation. It is built as a 4-round addition-rotation-xor (ARX) Feistel cipher with all round keys equal to α_i. As the name ARX suggests, it uses a combination of 32-bit modular additions, rotations, and xors. The whole process is depicted on Figure 3.20.

Variants. The variants of the SPARKLE family are listed below.
Each variant uses the b-bit SPARKLE permutation with a bit rate of r bits and a capacity of c bits, where $b = r + c$. In the column #Steps for AEAD variants, x/y indicates that the SPARKLE permutation with y steps is used: during initialization, between associated data (AD) processing and message processing, and during finalization, while the SPARKLE permutation with x steps is used during: associated data (AD) processing and message processing.

AEAD variants	Key	Nonce	Tag	Rate	Capacity	#Step
SCHWAEMM256-128	128	256	128	256	128	7/11
SCHWAEMM128-128	128	128	128	128	128	7/10
SCHWAEMM196-192	192	196	196	196	196	7/11
SCHWAEMM256-256	256	256	256	256	256	8/12

Figure 3.21: The AEAD variant of the SPARKLE.

3.9.2 SCA on Sparkle

In 2020, the first publication [86] investigates side-channel attacks on Sparkle by implementing a protected version of Schwaemm. The authors conduct power trace measurements and T-test calculations, without attempting attacks on the mentioned algorithms, whether protected or unprotected. Their findings reveal that unprotected versions exceed a threshold of 4.5, whereas the protected version does not. However, they were unable to recover keys through CPA or deep learning power analysis (DLPA) with version protected.

In 2022, the second publication [87] delves into correlation power analysis (CPA) and distribution leakage assessment (DLPA) applied to Schwaemm-128. Authors implement Schwaemm256-128 on an unspecified device, collecting 2000 traces for each of 320 different private keys.

In 2023, the last publication, Modou et al. [73] proposed a new blind side channel analysis (BSCA) on Sparkle. The leakage is considered as an Hamming weight (HW) with a gaussian noise. A belief propagation (BP) algorithm is used to link the different leaks. This attack is a contribution of my thesis, it is described in chapter 5.

In table 3.8, we present the side channel analysis (SCA) on Sparkle.

Table 3.8: side channel analysis (SCA) on Sparkle.

SCA on Sparkle								
CPA			DLPA			BSCA		
Unprotected								
Result	#T	Chip	Result	#T	Chip	Result	#T	Chip
						[73]		
Protected								
Result	#T	Chip	Result	#T	Chip	Result	#T	Chip
[87]	2000	Artix-7	[87]	2000	Artix-7			

3.10 TinyJAMBU

3.10.1 Overview of the design

JAMBU is a lightweight authenticated encryption mode submitted to the CAESAR competition [88]. Jambu mode is a mode of operation for block ciphers designed to offer a combination of encryption and data authentication. It is based on an underlying block cipher, such as AES, and is designed to be performance-efficient.

TinyJAMBU is a small variant of the JAMBU mode. TinyJAMBU is based on a keyed permutation (with no key schedule) that is based on a 128-bit nonlinear feedback shift register. The nonlinearity in each round is provided by a single NAND operation. The state size of TinyJAMBU is only two thirds of that of JAMBU, the message block size of TinyJAMBU is half of that of JAMBU mode. When nonce is reused, TinyJAMBU provides better authentication security than JAMBU mode. The authentication security of TinyJAMBU mode is better than the Duplex mode [2] when nonce is reused (for the same permutation size and message block size).

Variants. TinyJAMBU family has three AEAD variants. The number of rounds is represented by a 2-tuple, where the first number is the number of rounds used in key setup, plaintext processing, and the generation of first 32-bit of tag, and the second number is the number of rounds used in nonce and associated data (AD) processing and the generation of last 32-bit of tag.

AEAD variants	Key	Nonce	Tag	State	#Round
TinyJAMBU-128	128	96	64	128	1024/640
TinyJAMBU-192	192	96	64	128	1152/640
TinyJAMBU-256	256	96	64	128	1280/640

Figure 3.22: The AEAD variant of the TinyJAMBU

3.10.2 SCA on TinyJAMBU

Bhasin et al. [79] apply SPA during "key configuration" to recover the key elements bit by bit. Then they demonstrate full key recovery on the 1-bit implementation of TinyJAMBU using differential analysis aided power attack (DAPA) and SPA techniques. This attack requires 33 instantiations of the cipher with different nonces, and a 6-bit guess under the chosen nonce, a noiseless scenario. However, it shows that it is more difficult to apply DAPA techniques to recover the key of TinyJAMBU's 32-bit implementation. While noting that it's quite simple to retrieve 32 bits of the internal state of the key register during the nonce setup phase, using DAPA rectangular differential analysis.

In table 3.9, we present the side channel analysis (SCA) on TinyJAMBU.

Table 3.9: side channel analysis (SCA) on TinyJAMBU

SCA on TinyJAMBU								
DAPA(1-bit implementation)			DAPA(32-bit implementation)			SPA		
Unprotected								
Result	#T	Chip	Result	#T	Chip	Result	#T	Chip
[79]			[79]			[79]		
Protected								

3.11 Xoodyak

3.11.1 Overview of the design

Xoodyak [16] is a permutation-based AEAD and hashing scheme. Xoodyak is built from a fxed 384-bit permutation (called Xoodoo) operated in cyclist mode. The design approach of Xoodoo is closely related to that of the Keccak permutation. The 384-bit state is arranged in a three-dimensional array of $3 \times 4 \times 32$ bits, nonlinearity is provided by simple operations on 3-bit columns, linear mixing is provided by mixing between sheets and moving the bits within the sheets around, and a constant addition ensures that there is some difference between rounds. Cyclist mode takes a fixed permutation and provides the functionality of both hashing (sponge mode) and AEAD (duplex mode) along with some new functionality, including tuple hashing, XOFs, and the generation of rolling subkeys.

Variants. The variants of the Xoodyak family are listed below.

AEAD variants	Key(in bits)	Nonce(in bits)	Tag(in bits)	Rate(in bits)	Capacity(in bits)	#Round
Xoodyak	128	256	128	192	192	12

Figure 3.23: The AEAD variant of the Xoodyak

3.11.2 SCA on Xoodyak

The only SCA is CPA inspired by DPA attack called Keyak, which was based on Keccak-p. Measurement was made on Piñata development board with STM32F4. However, the publication is quite brief on Xoodyak CPA attack description.
A simulated CPA on Xoodyak was demonstrated by Batina et al. using 30K traces [89].

In table 3.10, we present the side channel analysis (SCA) on Xoodyak.

Table 3.10: side channel analysis (SCA) on Xoodyak

SCA on Xoodyak		
CPA		
Unprotected		
Result	#T	Chip
[89]	30K	STM32F407IGT6
Protected		

In the following table 3.11, we present an overview of the 10 finalist algorithms. More specifically, the 3 variants targeted during the attack phase in part III are highlighted in red.

3.12 SCA on LFSRs

Linear feedback shift registers (LFSRs) with primitive polynomials are used in many symmetric cryptographic primitives because of their well-defined structure and remarkable properties such as long period, ideal autocorrelation and statistical properties.

The information leakage and the vulnerability of stream ciphers based on Galois LFSRs are studied in [90] and those based on Fibonacci LFSRs are analysed in [91]. In [90], the information leakage of XOR gates is exploited to perform a simple side-channel attack. However, if the leakage from the XOR gates is too low compared to other operations in the cipher, the attack fails. In [92], the attack recovers the initial state of a Galois LFSR by determining the output of the LFSR from the difference in power dissipation values in consecutive clock cycles.

In chap 4, a theoretical blind side channel attack targeting the LFSR of the Elephant algorithm is presented. Whereas several attacks on LFSRs have been described in the literature, the specific structure of the Elephant cryptosystem allows to elaborate a new approach that is depicted in the rest of this chapter 4 and 5.

Table 3.11: Overview of the finalist.

				Size in bits		
Finalists	Variant	Building Block	Mode	Key	Nonce	Tag
Ascon [6]	Ascon-128	Ascon Permutation	MonkeyDuplex [2]	128	128	128
	Ascon-128a			128	128	128
	Ascon-80pq			160	128	128
Elephant [8]	Dumbo	Spongent-π[160]	Encrypt-then-MAC 2.2.3	128	96	64
	Jumbo	Spongent-π[176]		128	96	64
	Delirium	Keccak-f[200]		128	96	128
GIFT-COFB [9]	GIFT-COFB	GIFT-128	COFB [5]	128	128	128
Grain-128AEADv2 [10]	Grain-128AEADv2	Feedback shift register	Encrypt-Then-MAC 2.2.3	128	96	64
ISAP [11]	ISAP-A-128a	Ascon Permutation	Encrypt-then-MAC 2.2.3	128	128	128
	ISAP-K-128a	Keccak-f[400]		128	128	128
	ISAP-A-128	Ascon Permutation		128	128	128
	ISAP-K-128	Keccak-f[400]		128	128	128
PHOTON-Beetle [12]	PHOTON-Beetle[128]	PHOTON_{256} Permutation	Beetle with COFB [5]	128	128	128
	PHOTON-Beetle[32]			128	128	128
Romulus [13]	Romulus-N	Skinny-128-384+ with TBC	COFB [5]	128	128	128
	Romulus-M		MAC-then-Encrypt 2.2.3	128	128	128
	Romulus-T		Encrypt-then-MAC 2.2.3	128	128	128
Sparkle [14]	Schwaemm256-128	Sparkle_{384}	Beetle with COFB [5]	128	256	128
	Schwaemm128-128	Sparkle_{256}		128	256	128
	Schwaemm192-192	Sparkle_{384}		192	192	192
	Schwaemm256-256	Sparkle_{512}		256	256	256
TinyJAMBU [15]	TinyJAMBU-128	Keyed Permutation [15]	Sponge [69]	128	96	64
	TinyJAMBU-192			192	96	64
	TinyJAMBU-256			256	96	64
Xoodyak [16]	Xoodyakv1	Xoodoo Permutation [16]	Sponge-variant Cyclist	128	128	128

Part III

Attack against AEAD algorithms.

Chapter 4

Attack with a theoretical model

"*Réaliser une thèse de doctorat, c'est comme courir un marathon en montagne: il y'a des hauts et des bas, mais seule la persévérance permet d'atteindre le jour de la soutenance et d'obtenir son diplôme.*"
Docteur Mbaye DIENG.



Contents

4.1 Introduction **67**
4.2 Elephant **67**
4.3 Theoretical attack **67**
4.3.1 Goal 67
4.3.2 Leakage in the LFSR 67
4.3.3 Link between the different masks 68
4.4 Attack strategy **69**
4.4.1 Algorithm of the attack 69
4.4.2 Optimisation of the attack 70
4.4.2.1 Goal of the optimization 70
4.4.2.2 Principle 71
4.4.2.3 Gathering information 71
4.4.2.4 Crafting a merge estimator 72
4.4.2.5 Estimating the best position 75
4.5 Results and discussion **75**
4.5.1 Elephant attack 75
4.5.2 Impact of the generation of masks 76
4.5.3 Studies on different LFSRs 76
4.6 Conclusion **78**

4.1 Introduction

In this chapter, we focus on ELEPHANT, one of the finalists in the NIST competition. We present a significant improvement on the first blind side channel analysis (BSCA) againstELEPHANT. Specifically, our study focuses on the linear feedback shift register (LFSR) counter used internally.LFSRs are classical functions widely used in symmetric cryptography. In the case of ELEPHANT, retrieving the initial state of the LFSR is equivalent to recovering the encryption key.

Our approach exploits the use of intermediate variables that are statistically dependent on the secret, in this case the initial state of the LFSR. We demonstrate that this structure can compromise the security of a cryptosystem against side channel analysis (SCA) attacks. This chapter also includes a study of the influence of the choice of LFSR on the overall security of the algorithm.

We propose a major improvement to the first attack [7] by integrating an optimization to determine the best time, relative to the start of encryption, to launch the attack. This optimization aims to maximize the chances of success by choosing the moment when information leaks are most exploitable.

For this contribution, I collaborated with two PhD students, Julien Maillard under the supervision of Christophe Clavier and Awaleh Houssein Meraneh under the supervision of Hélène Le Bouder.

4.2 Elephant

The description of ELEPHANT is already defined in the section 3.3.1.

4.3 Theoretical attack

4.3.1 Goal

LFSRs are used in different lightweight cryptography candidates, and its initial state often depends on both a key and a nonce. As the nonce needs to be changed for each encryption request, attacks on such schemes are limited to the decryption algorithm. In the case of Elephant, the LFSR only depends on the secret key. Consequently, our attack can be applied in an encryption scenario.

The goal of the presented attack is to retrieve the LFSR secret initial state. One has to remark three important points:

- Retrieving the initial state of the LFSR, which is equal to $\mathsf{mask}_K^{0,0}$, is equivalent to retrieving the secret key. Indeed, the initial state is the result of the known permutation P applied to the key.
- As the retroaction polynomial is publicly known, it is possible to shift the LFSR backwards: an attacker who recover enough consecutive bytes of the secret stream is able to reconstruct the initial state.
- The smaller the LFSR is, the more the attack is able to succeed. As a consequence, the Dumbo instance (see Fig. 3.9) is the most vulnerable one: the following of this paper is focused on Dumbo.

4.3.2 Leakage in the LFSR

In this attack, it is assumed that the Hamming weight (HW) of every byte of the LFSR can be obtained by an attacker.

Since the LFSR generates a single new byte at each iteration, let $(x_j, \cdots, x_{j+19}) = \mathsf{mask}_K^{j,0}$ be the content of the Dumbo LFSR and x_{j+20} the byte generated at iteration j. Precisely, the attacker has the following relation (L1), according to the equation (3.1).

L1 $x_{j+20} = (x_j \lll 3) \oplus (x_{j+3} \ll 7) \oplus (x_{j+13} \gg 7)$.

The first idea is to use the knowledge of the following Hamming weights: $HW(x_{j+20})$, $HW(x_{j+13})$ and

$$HW(x_j) = HW(x_j \lll 3). \tag{4.1}$$

So with the two equations (L1) and (4.1) the attacker has:

$$HW\left(x_{j+20}\right) = \begin{cases} HW\left(x_j\right) \\ HW\left(x_j\right) + 1 \\ HW\left(x_j\right) - 1 \\ HW\left(x_j\right) + 2 \\ HW\left(x_j\right) - 2 \end{cases} \tag{4.2}$$

Looking more precisely at equation (L1), it can be seen that the difference $HW(x_{j+20}) - HW(x_j)$ only depends on four bits. Let $x_j[i]$ denote the i-th least significant bit of byte x_j, these four bits are $\{x_{j+3}[0]; x_{j+13}[7]; x_j[4]; x_j[5]\}$. Table 4.1 gives the value of observed difference $HW(x_{j+20}) - HW(x_j)$ depending on the values of these four bits. In the worst case, there are only 6 possibilities left, out of 16.

$HW\left(x_{j+20}\right) - HW(x_j)$		$(x_{j+3}[0], x_{j+13}[7]) =$			
		$(0,0)$	$(0,1)$	$(1,0)$	$(1,1)$
$(x_j[4], x_j[5]) =$	$(0,0)$	0	$+1$	$+1$	$+2$
	$(1,0)$	0	$+1$	-1	0
	$(0,1)$	0	-1	$+1$	0
	$(1,1)$	0	-1	-1	-2

Table 4.1: Values of $HW\left(x_{j+20}\right) - HW\left(x_j\right)$ according to $\{x_{j+3}[0]; x_{j+13}[7]; x_j[4]; x_j[5]\}$ [7] .

4.3.3 Link between the different masks

The value $\mathsf{mask}_K^{j,1}$ can be expressed in terms of $\mathsf{mask}_K^{*,0}$ as in (4.3).

$$\begin{aligned} \mathsf{mask}_K^{j,1} &= (\varphi \oplus \mathsf{id}) \left(\mathsf{mask}_K^{j,0}\right) \\ &= \varphi\left(\mathsf{mask}_K^{j,0}\right) \oplus \mathsf{mask}_K^{j,0} \\ &= \mathsf{mask}_K^{j+1,0} \oplus \mathsf{mask}_K^{j,0}. \end{aligned} \tag{4.3}$$

Likewise, for $\mathsf{mask}_K^{j,2}$, equation (4.4) holds.

$$\begin{aligned} \mathsf{mask}_K^{j,2} &= (\varphi \oplus \mathsf{id})^2 \left(\mathsf{mask}_K^{j,0}\right) \\ &= (\varphi^2 \oplus \mathsf{id}) \left(\mathsf{mask}_K^{j,0}\right) \\ &= \varphi^2\left(\mathsf{mask}_K^{j,0}\right) \oplus \mathsf{mask}_K^{j,0} \\ &= \mathsf{mask}_K^{j+2,0} \oplus \mathsf{mask}_K^{j,0} \end{aligned} \tag{4.4}$$

As in the case of $\mathsf{mask}_K^{j,0}$, let y_j denote either the byte j of $\mathsf{mask}_K^{0,1}$ when $0 \leq j \leq 19$, or the new byte obtained after j iterations of the LFSR initialized with $\mathsf{mask}_K^{0,1}$. Likewise, let z_j denote either the byte j of $\mathsf{mask}_K^{0,2}$ when $0 \leq j \leq 19$, or the new byte obtained after j iterations of the LFSR initialized with $\mathsf{mask}_K^{0,2}$.

Equation (4.3) then translates to equation (4.5).

$$y_j = x_j \oplus x_{j+1} \tag{4.5}$$

Likewise, the equation 4.4 translates to (4.6).

$$z_j = x_j \oplus x_{j+2}. \tag{4.6}$$

The evolution of the LFSR is analogous to (L1):

$$y_{j+20} = (y_j \lll 3) \oplus (y_{j+3} \ll 7) \oplus (y_{j+13} \gg 7). \tag{4.7}$$

$$z_{j+20} = (z_j \lll 3) \oplus (z_{j+3} \ll 7) \oplus (z_{j+13} \gg 7). \tag{4.8}$$

The attacker can thus exploit two attack vectors: on the one hand, equations (L1), (4.7), and (4.8) coming from iterating the LFSR, and on the other hand, equations (4.5) and (4.6) coming from the different masks used for domain separation.

4.4 Attack strategy

For a byte x_j of the Dumbo LFSR with $j \geq 0$, let x'_j denote a guess of its value by the attacker. Given m successive bytes $(x_j, \cdots, x_{j+m-1})$, let $\mathbf{X}_j^m$ denote the set of guesses $(x'_j, \cdots, x'_{j+m-1})$ satisfying the constraints of the Hamming weights of masks x, y and z depicted in Equations 4.5 and 4.6.

4.4.1 Algorithm of the attack

Algorithm 2 $\mathsf{isvalid}(x'_0, \cdots, x'_j)$

Input: Byte-wise partial candidate $(x'_0, \cdots, x'_j)$ of length $1 \leq j+1 \leq 20$
Assumes $\mathsf{isvalid}(x'_0, \cdots, x'_{j-1})$ is true.
Output: **true** if candidate $(x'_0, \cdots, x'_j)$ is compatible with the observations, **false** otherwise

```
# Hamming weights of the xors
if HW(x'_j) ≠ HW(x_j) then
    return false
end if
if HW(x'_j ⊕ x'_{j-1}) ≠ HW(y_{j-1}) then
    return false
end if
if HW(x'_j ⊕ x'_{j-2}) ≠ HW(z_{j-2}) then
    return false
end if
# Hamming weights of the feedbacks
if |HW(x'_j ⋘ 3) − HW(x_{j+20})| > 2 then
    return false
end if
if |HW(x'_{j-3} ⋘ 3 ⊕ x'_j ≪ 7) − HW(x_{j+17})| > 1 then
    return false
end if
if HW(x'_{j-13} ⋘ 3 ⊕ x'_{j-10} ≪ 7 ⊕ x'_j ≫ 7) ≠ HW(x_{j+7}) then
    return false
end if
return true
```

The whole search space corresponding to the initial state of the LFSR is represented as a rooted tree. The nodes at depth j correspond to all the possible values for the bytes x_0 to x_j of the LFSR. The tested candidates are denoted by $(x'_0, \cdots, x'_{19})$. The nodes in the graph of the search space are labelled as follows:

- the nodes at depth j correspond to all the possible values of $(x'_0, \cdots, x'_j)$;
- the children of node $(x'_0, \cdots, x'_j)$, are the nodes labelled: $(x'_0, \cdots, x'_j, x'_{j+1})$ for all values of x'_{j+1}.

In practice, to reduce the number of nodes, only the nodes having the correct Hamming weights are considered. In other words, it suffices to consider nodes with $HW(x'_j) = HW(x_j)$. An example of such a tree is given on Fig. 4.1.

A backtracking algorithm is used. The tree is traversed in a depth-first manner. For each step, the attacker tests whether the current candidate $(x'_0, \cdots, x'_j)$ satisfies the different conditions given by the observed Hamming weights. This test is given by **Algorithm** 2.

If the test succeeds, the algorithm goes down to the next layer to test the values of the byte x'_{j+1}. If it reaches the bottom of the tree, then a good candidate has been found, and can be saved. The algorithm then iterates upon the next untested node.

If, at some point, the Hamming weights conditions do not hold for the current (partial) candidate, then no node in the sub-tree rooted at that node can lead to a good candidate. Thus, it can be pruned from the whole tree, saving the cost of browsing it. Finally, the algorithm ends when the whole tree has been explored. A pseudocode of the attack is given by **Algorithm** 3.

Algorithm 3 Attack

Input: Observed Hamming weights $HW(x_0), \cdots, HW(x_{19})$, $HW(y_0), \cdots, HW(y_{18})$, and $HW(z_0), \cdots, HW(z_{17})$. For the sake of clarity, they are seen as global variables.
Output: S set of keys compatible with the observed Hamming weights

1: $(x'_0, \cdots, x'_{19}) \leftarrow (0, \cdots, 0)$
2: $\ell \leftarrow 0$
3: $S \leftarrow \{\}$
4: **while true do**
5: **if** $j < 19$ **and** isvalid$(x'_0, \cdots, x'_j)$ **then**
6: $j \leftarrow j + 1$
7: $x'_j \leftarrow 0$
8: **else**
9: **if** $j = 19$ **and** isvalid$(x'_0, \cdots, x'_j)$ **then**
10: $S \leftarrow S \cup \{(x'_0, \cdots, x'_j)\}$
11: **end if**
12: **while** $j \geq 0$ **and** $x'_j = \texttt{FF}$ **do**
13: $j \leftarrow j - 1$
14: **end while**
15: **if** $j \geq 0$ **then**
16: $x'_j \leftarrow x'_j + 1$
17: **else**
18: **break**
19: **end if**
20: **end if**
21: **end while**
22: **return** S

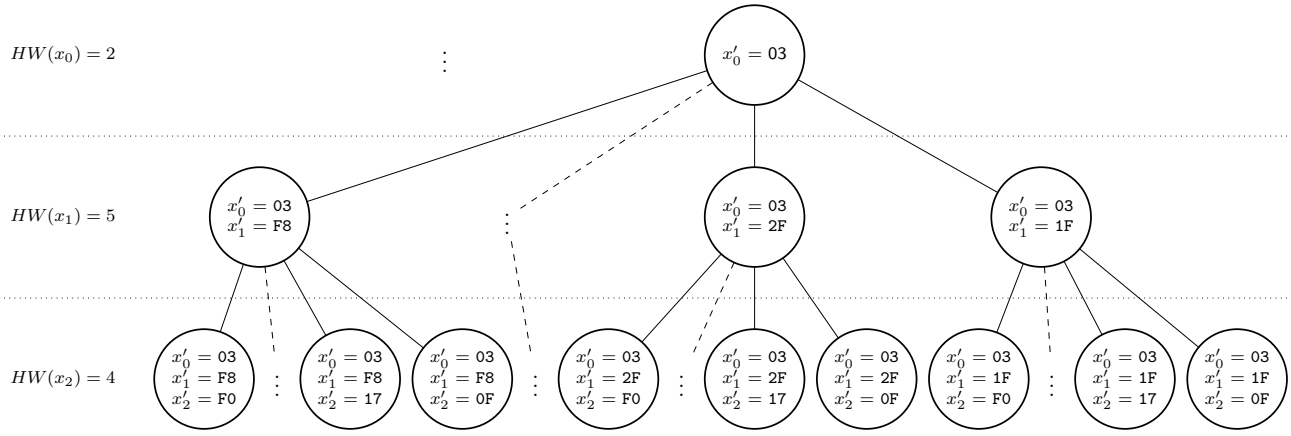


Figure 4.1: Example of the tree representation of the LFSR initial state for the Hamming weights given on the left. Only the first three layers of the subtree rooted at $x'_0 = 03$ are shown.

4.4.2 Optimisation of the attack

4.4.2.1 Goal of the optimization

Recall from **Algorithm** 1 that for each new block of data to encrypt, a different mask is used. Therefore the attacker actually can choose which values of index t at lines 2, 10, and 13 of **Algorithm** 1 they would rather attack.

The question raised at this point is: at which time t does the attacker minimize the complexity of the attack? The attacker wants to establish a metric $E(\cdot)$ such that $E(t)$ allows estimating the complexity of an attack at time t. The attacker constructs E such that the computation of $E(t)$:

1. is fast enough to allow exploring a large number of values of t and
2. is accurate enough so that the attacker can accurately select the value of t with minimal attack complexity.

4.4.2.2 Principle

The general idea for finding the best attack position is to obtain an estimation of the attack complexity at a given time t. For this sake, the LFSR state $(x_t, \cdots, x_{t+19})$ at time t is divided into tuples that can be treated with three different operations. Note that this approach does not exploit the retroaction polynomial of the LFSR. This allows to use it for all the flavours of the Elephant cryptosystem.

1. **Enumerate** This function takes as an input m indices $(i, \cdots, i+m-1)$ and returns all the candidates $\mathbf{X}_i^m$ for $(x'_i, \cdots, x'_{i+m-1})$ satisfying the constraints of masks x, y and z depicted in equations (4.5) and (4.6), based on the knowledge of corresponding $3m-3$ Hamming weights. This operation explores all the possible combinations for each x'_i and only retains the candidates matching the conditions imposed by masks x, y and z. Note that the *enumerate* function quickly becomes computationally intense as m grows. In our experiments, we use this function for $m \in \{1, 2, 3\}$.

2. **Merge** When m grows, the attacker proceeds with a divide and conquer strategy. Indeed, for $k < m$, the attacker first computes candidates $\mathbf{X}_i^k$ and $\mathbf{X}_{i+k}^{m-k}$ thanks to the *enumerate* function. Then candidates $\mathbf{X}_i^k$ and X_{i+k}^{m-k} are merged into $\mathbf{X}_i^m$ so that each remaining candidate in $\mathbf{X}_i^m$ satisfies the constraints on masks y and z. This approach allows benefiting from the reduced candidate sets $\mathbf{X}_i^k$ and $\mathbf{X}_{i+k}^{m-k}$, allowing to reduce the size of the search space for $\mathbf{X}_i^m$. For a growing m, storing the $\mathbf{X}_i^m$ set in memory can become impractical, hence the *merge* function comes with the *merge_ count* variant, that only returns the number $\mid \mathbf{X}_i^m \mid$ of candidates. We emphasize that calling *merge_ count* on X_i^m disables calling it for another merge.

3. **Merge estimator** The number of candidates $\mid \mathbf{X}_i^m \mid$ can be assessed by applying a reduction factor r on the product of $\mid \mathbf{X}_i^k \mid$ and $\mid \mathbf{X}_{i+k}^{m-k} \mid$. Namely, it is defined as:

$$r = \frac{\mid \mathbf{X}_i^k \mid \times \mid \mathbf{X}_{i+k}^{m-k} \mid}{\mid \mathbf{X}_i^m \mid} \tag{4.9}$$

Hence, the merging estimation procedure takes as input two indices tuples $(i, \cdots, i+k-1)$ and $(i+k, \cdots, i+m)$, for $k < m$, and their corresponding number of candidates $\mid \mathbf{X}_i^k \mid$ and $\mid \mathbf{X}_{i+k}^{m-k} \mid$. The aim of the estimator is to approximate r based on statistical tools without exploring the entire Cartesian product of $\mathbf{X}_i^k$ and $\mathbf{X}_{i+k}^{m-k}$. The construction of the estimator is discussed in the following.

4.4.2.3 Gathering information

The estimator E for the attack complexity can then be defined by applying the previous three functions. The procedure used for the Dumbo LFSR is depicted in Fig. 4.2.

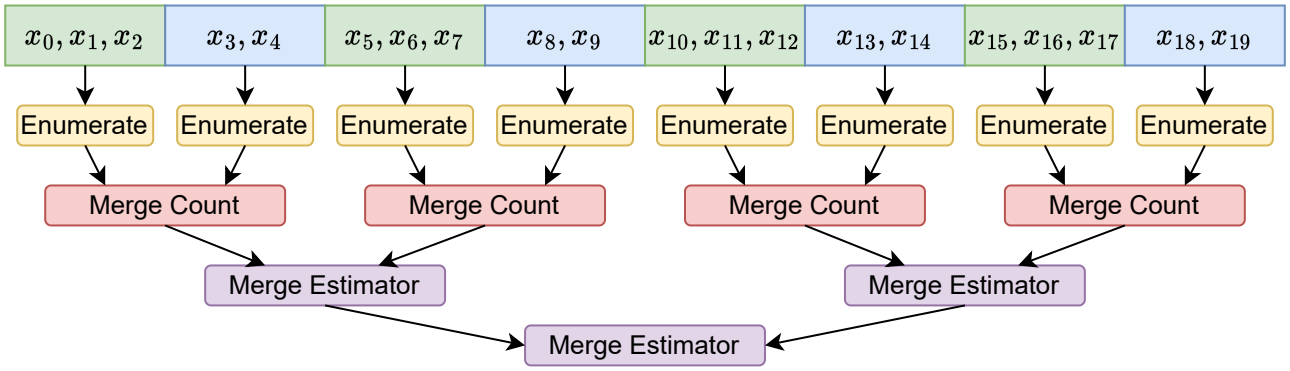


Figure 4.2: Gathering information for the Dumbo LFSR.

The 20 LFSR bytes are split into 4 quintuplets. Each quintuplet is itself split into a triplet and a pair of bytes. The attacker gathers the candidates for each triplet and pair with the *enumerate* function, and then call the *merge_ count* function on the union of the triplets and pairs to count the exact number of candidates for the quintuplets. Eventually, the attacker applies the *merge estimator* between the first and second, and the third and fourth quintuplets before calling the estimator a last time on the full state.

At this point, it can be seen that the accuracy of the complexity returned by E highly depends on the quality of the *merge_ estimator*.

4.4.2.4 Crafting a merge estimator

Naive estimator

Equations (4.5) and (4.6) allow to derive a *reduction triangle* for the merge of two tuples (see Fig. 4.3).

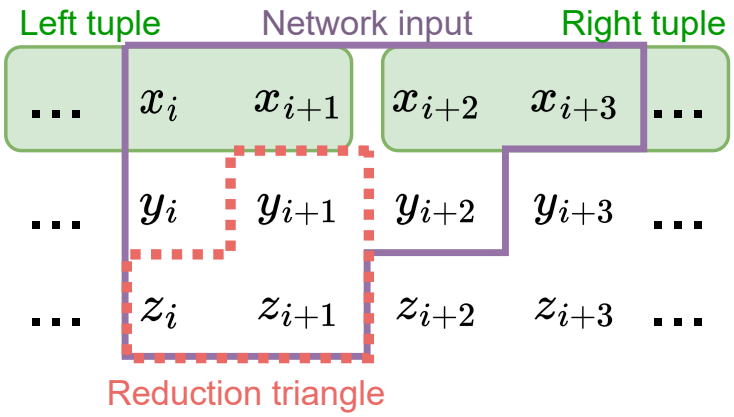


Figure 4.3: Necessary data for the naive estimator (reduction triangle) and the neural network based merge estimator.

This reduction triangle can be exploited to obtain a broad estimation on the information gained by a merge, hence, a estimation of the reduction factor of the Cartesian product of $\mathbf{X}_i^k$ and $\mathbf{X}_{i+k}^{m-k}$. Let $\mid \mathbf{X}_{triangle} \mid$ define the number of candidates for the reduction triangle. This corresponds to the product of the number of bytes values that match each three gathered Hamming weights. Then, r_{naive} is defined by:

$$r_{naive} = \frac{\mid \mathbf{X}_{triangle} \mid}{2^{24}} \tag{4.10}$$

The estimation error of r_{naive} is defined as follows:

$$err_{naive} = r_{naive} - r \tag{4.11}$$

When $err_{naive} > 0$, reduction factor has been overestimated. This situation raises an issue for the attacker, as the complexity of the attack is falsely underestimated. On the contrary, $err_{naive} < 0$ indicates that the reduction factor has been underestimated. This means that the attacker would probably not consider this step to perform the attack, even if, in reality, the attack complexity would have been much inferior.

A distribution of the err_{naive} values is depicted in Fig 4.4.

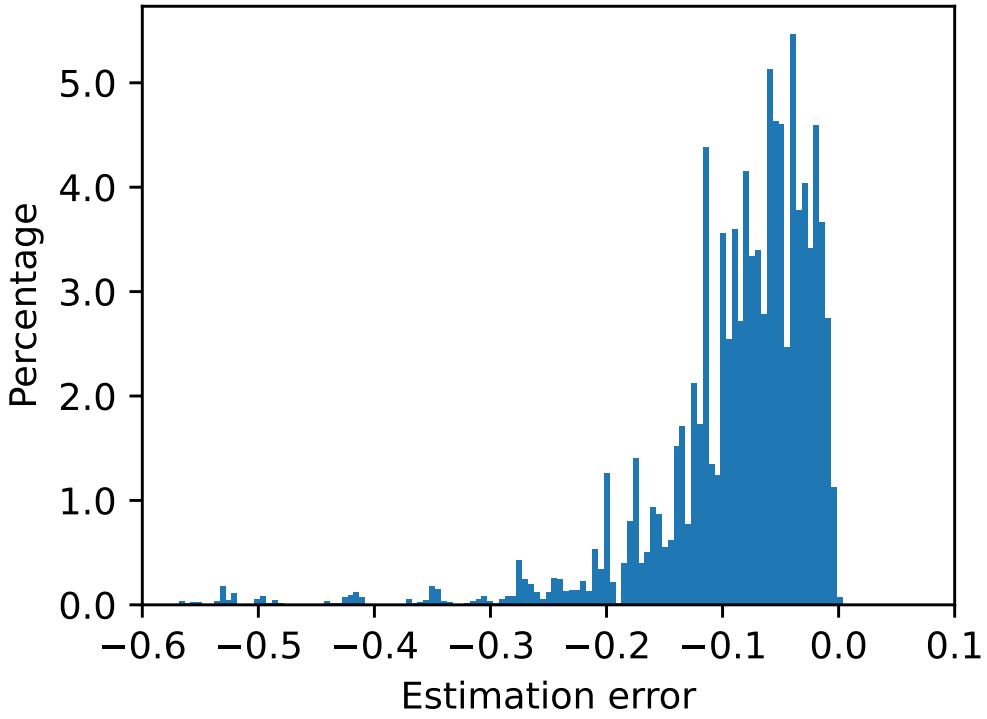


Figure 4.4: Distribution of err_{naive} for $m = 4$.

A distribution of $err_{network}$ values is depicted in Fig 4.5.

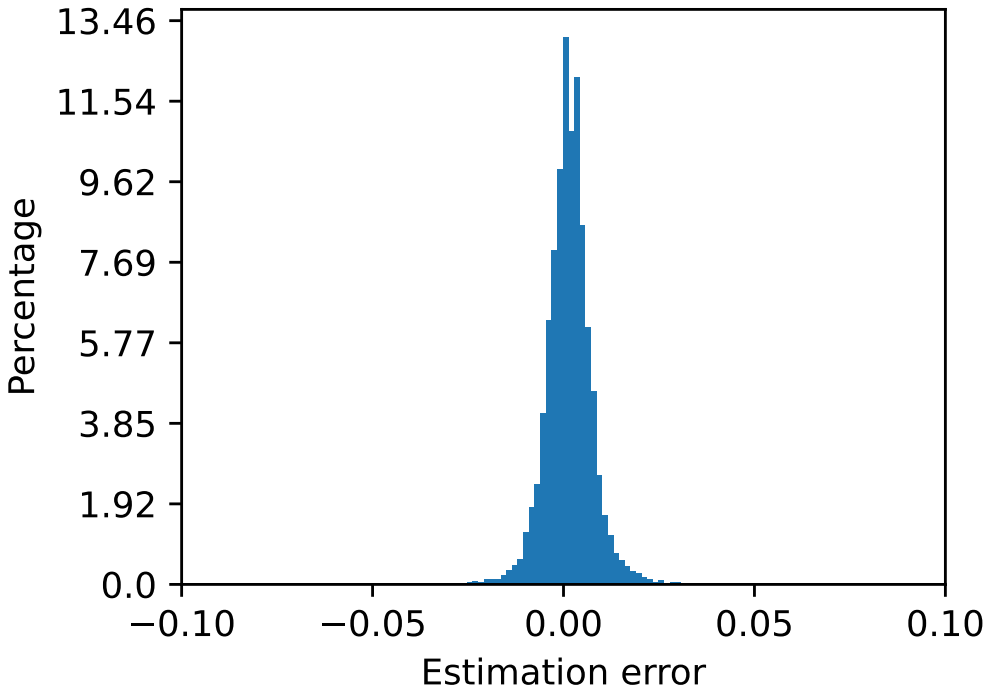


Figure 4.5: Distribution of $err_{network}$ for $m = 4$.

The attacker stresses that r_{naive}, despite being fast to compute, has the flaw of not considering all the inter-mask dependencies depicted in equations (4.5) and (4.6).

Neural network based estimator

Inter-masks dependencies are difficult to handle as-is for crafting a merge estimator. Luckily, the complex relations that bound these bytes can be exploited thanks to the power of a neural network. Hence, the Hamming weights of the bytes depicted in Fig. 4.3 can be fed to a neural network in order to predict r.

To do so, we create a dataset by performing 1.3M merges of candidates for two pairs of bytes. Merging pairs allows to quickly compute r, and thus allows building a significant dataset in a reasonable amount of time. We keep the real r values as labels that will be provided to the network in order for it to provide an estimator $r_{network}$.

A neural network composed of 6 fully connected hidden layers is crafted, whose sizes are depicted in Fig 4.6. The choice of the neural network architecture has been performed by progressively tuning the parameters in order to reduce the variance and standard deviation of the estimation error for $r_{network}$. Each layer uses the *Relu* activation function, and the model is compiled with the *adam* optimizer and the *Mean-Square-Error* loss function.

The model is trained upon 65 epochs upon a 1.287M sample datasets with a 0.01 validation ratio. Training and validation losses are depicted in Fig. 4.7.

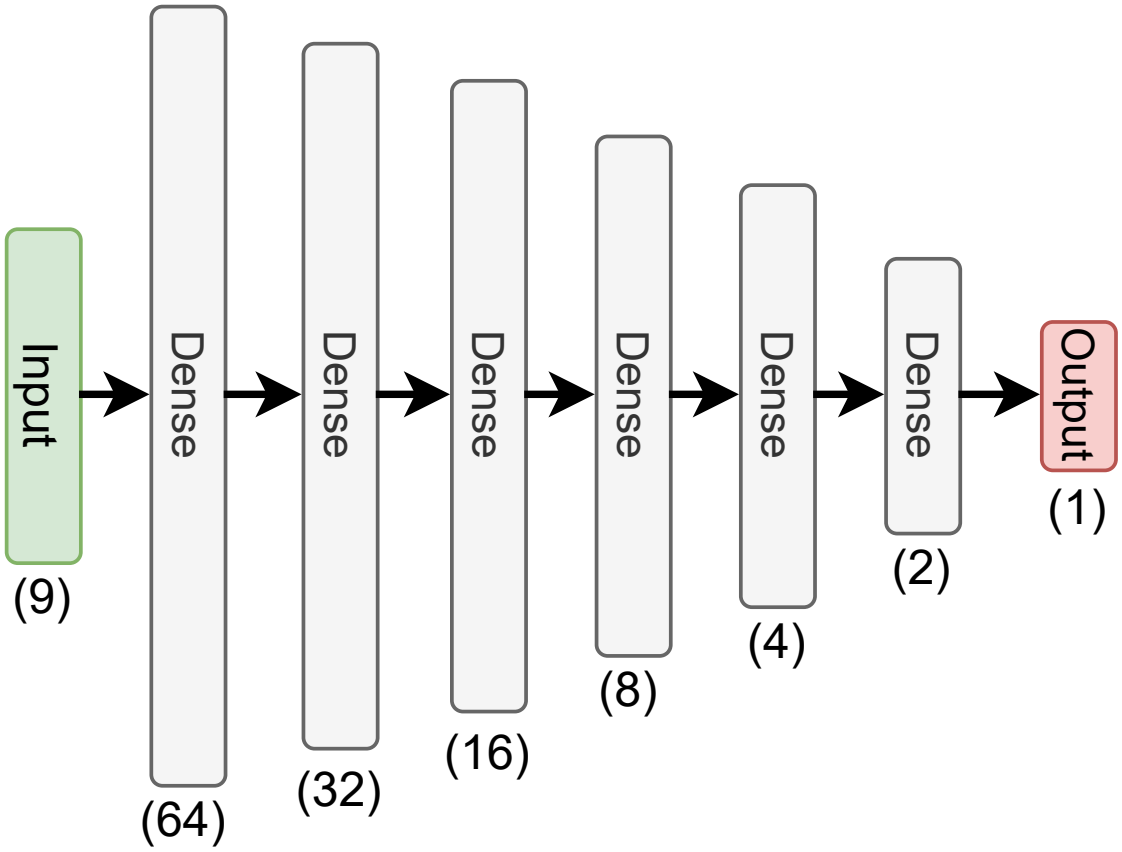


Figure 4.6: Architecture of the neural network for the merge estimator E.

The total training time of the model is approximately 15 minutes on a `Intel Core i7-8565U CPU`. As for r_{naive}, the estimation error of $r_{network}$ is measured as follows:

$$err_{network} = r_{network} - r \tag{4.12}$$

The distribution of $err_{network}$ on a testing dataset of 13K samples (*i.e.*, a dataset that has not been used to train the network) is displayed in Fig. 4.5.

Discussion

Fig. 4.4 shows that our naive estimator highly underestimates the reduction factor of a merge between two tuples. Some instances even show more than a 50% delta between r_{naive} and the real reduction factor r. Fig. 4.5 shows that the neural network provides an estimation of r which is more accurate than the naive estimator, as the mean is closer to 0. Moreover, the error variance and standard deviation of the distribution

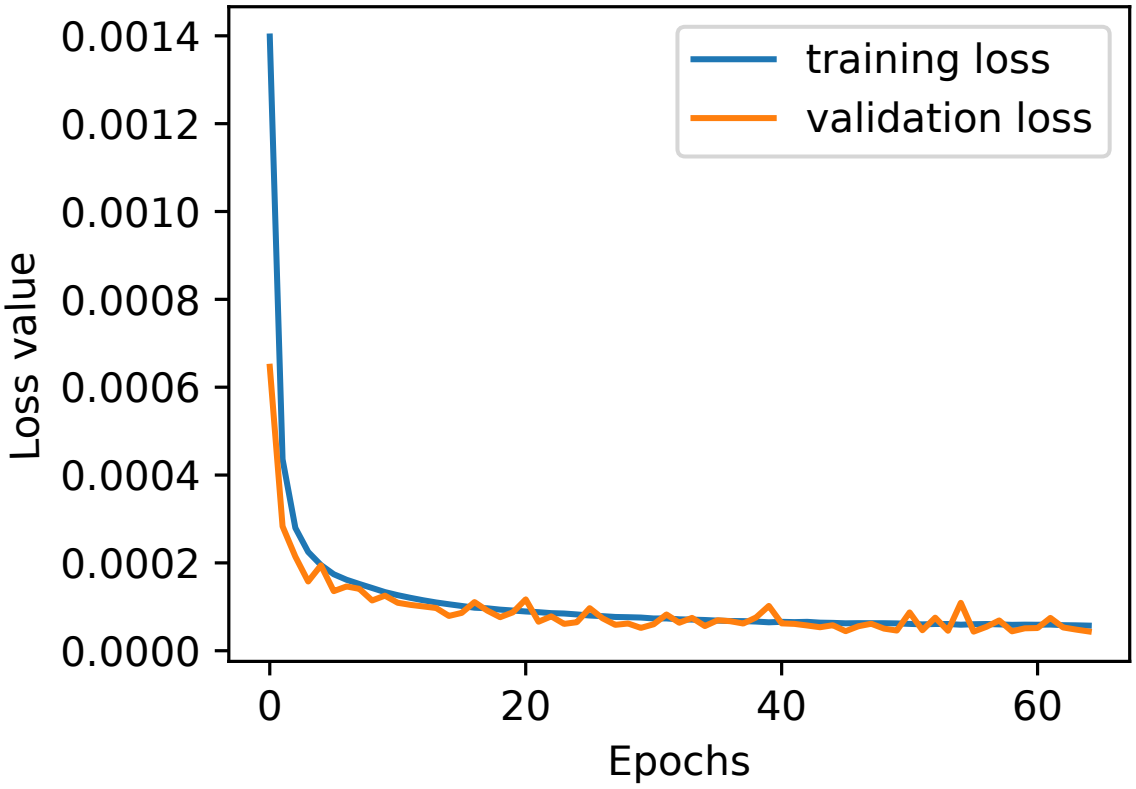


Figure 4.7: Evolution of training and validation losses of the neural network.

have been reduced compared to the naive estimator. More importantly, the $err_{network}$ distribution does not show outliers (*i.e*, predictions that are far from the mean) unlike err_{naive}. In our experiment, we hence use $r_{network}$ as the reduction factor estimation for the metric E (see Fig. 4.2).

4.4.2.5 Estimating the best position

The best position for the attack is assessed by applying estimator E in a sliding window fashion upon the iterations of the LFSR. In terms of performance, the computation time is approximately one second per iteration on a `Intel Core i7-8565U CPU`: this enables estimating the attack complexity on several dozens of thousands of steps in less than a day.

Depending on the attacker's computational power, several strategies can be considered. First, the attacker can set up a threshold upon attack complexity. When targeting an encryption stream, the attacker can apply the metric E on the sliding window until it returns an attack complexity that is below this threshold. Another approach is to fix a limit on the sliding window algorithm. Then, when all the attack complexity estimations have been returned, the attacker chooses the lowest one.

Fig. 4.8 illustrates a run of the sliding window algorithm on a Dumbo LFSR with initial state bytes sampled from the uniform distribution. The metric $E(t)$ is computed for $t \in \{0, \cdots, 20000\}$. In this example, minimal estimated attack complexity is approximately $2^{39.30}$ for $t = 10909$. Note that each $E(t)$ can be computed independently: the algorithm can easily be transformed into a parallel version.

4.5 Results and discussion

4.5.1 Elephant attack

We have simulated the attack on randomly generated Dumbo keys. We selected the lowest 25% of keys with respect to expected complexity given by the estimator E derived in section 4.4.2. This gave us $N_{runs} = 1275$ keys to test. For each, the number N_{nodes} of nodes effectively traversed in the tree has been counted. This number roughly corresponds to the time complexity of the attack. Among these nodes, we have specifically

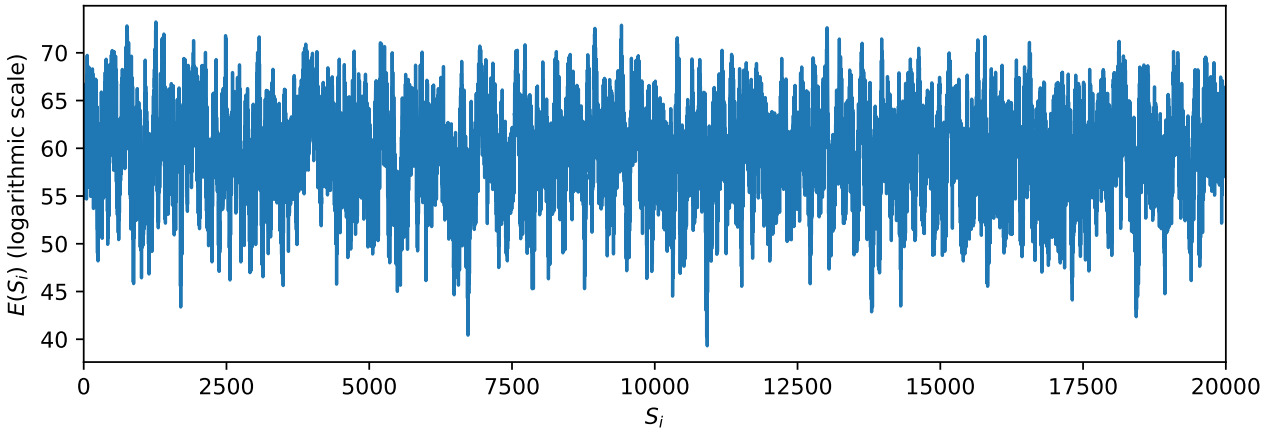


Figure 4.8: Example output of the sliding window E estimation algorithm.

counted the number N_{keys} of nodes on the last layer; *i.e.* nodes that correspond to plausible guesses that remain to be brute forced to finish the attack.

Only just above three quarters (77.73%) of the runs have ended after two days. This is an improvement (24.16% more) compared to [7] where only 53.57% had finished in the same amount of time. On average, for the runs that finished after two days, the number of nodes traversed is $N_{nodes} = 2^{41.92}$, and the number of remaining keys is $N_{keys} = 2^{36.52}$.

Figure 4.9 shows the link between the estimator E and the actual computation time for the differents keys tested. Quartiles for estimator and time distributions are given in Table 4.2. It should be remembered that data given here only represent the lowest 25% of the estimator distrbution and that time is capped at 48h, and therefore do not represent the full distributions.

	quartile Q_1	median Q_2	quartile Q_3
estimator E (log_2)	51.62	53.64	55.07
time (hours)	1.61	8.88	37.96

Table 4.2: Quartiles for estimator and time distributions.

4.5.2 Impact of the generation of masks

The threat brought by this attack upon the Elephant cryptosystem implies a discussion about mitigations. Apart from using generic countermeasures, like *e.g.* Boolean masking, there seem to be two possibilities for improvement. Indeed, the attacker gains information from two sources:

- from equations (4.3) and (4.4) used to derive the masks for domain separation;
- from the LFSR state update equation (L1).

Thus, either the mask derivation or the LFSR can be changed, or both. This section studies the former case.

We ran two experiments, similar to section 4.5.1 except that the attacker does not gain information from every Hamming weight. In the first experiment, they only know the values of the $HW(x_j)$, and the $HW(y_j)$. In other words, compared to the experiment in the section 4.5.1, they lost the knowledge of the $HW(z_j)$. Likewise, in the second experiment, they only know the values of the $HW(x_j)$. In both cases, none of the $N_{runs} = 120$ runs done has terminated after a week.

From these experiments, it seems that the combined knowledge of the $HW(x_j)$, $HW(y_j)$, and $HW(z_j)$ contributed heavily on the success of the attack. It would then seem a good idea to tweak the cryptographic mode of operation by finding another way of generating masks for domain separation.

4.5.3 Studies on different LFSRs

This section is dedicated to the study of the influence of the choice of the LFSR. We stress that the results shown here are obtained without the optimizations the optimization presented in 4.4.2: in practice, an attacker

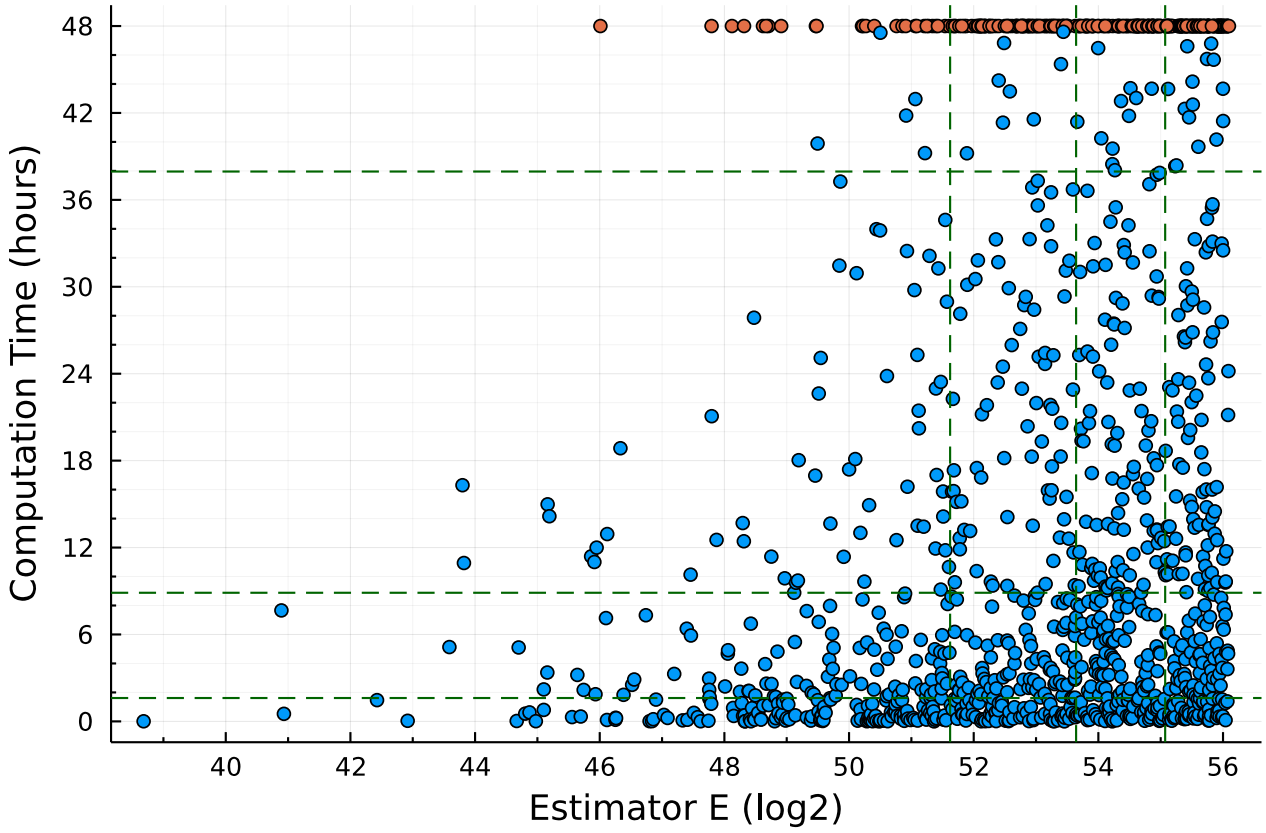


Figure 4.9: Estimator E versus actual computation time for finished runs (blue) and unfinished runs (red). Quartiles are represented by dashed lines.

can hope lowering the attack complexity by exploiting the optimization.

To keep the spirit of the original Elephant algorithm, only Fibonacci-like LFSRs, at the byte level, are considered. More specifically, LFSRs considered are: LFSRs where a single new byte is computed from a combination of three bytes using byte-wise shifts and rotations. As usual, the associated feedback polynomial must be primitive to ensure only maximum-length sequences can be generated. Among all possible candidates, different behaviours can be triggered.

In this paper, the *type* of a LFSR is defined as the sequence of the number of bits unknown to the attacker at each depth in the tree where a new feedback occurs.

Looking at equation (L1), it can be seen that:

$$|HW(x_{j+20}) - HW(x_j \lll 3)| \leq 2$$

since there are only 2 bits that are modified by:

$$x_{j+3} \lll 7 \oplus x_{j+13} \gg 7.$$

Thus, if other feedback functions are used, with more bits involved, it can be expected to have an impact on the attack.

Later in the attack, when at depth 3 in the tree, the same idea can be applied to check whether:

$$|HW(x_{j+17}) - HW(x_{j-3} \lll 3 \oplus x_j \ll 7)| \leq 1$$

So since now only the single bit $x_{j+10} \gg 7$ is unknown. Consequently, the type of the Dumbo LFSR is $[2, 1]$.

LFSRs with different types can be a first criterion when testing our attack.

A second criterion is the spacing between the feedback bytes. Indeed, the tighter they are, the faster the attacker can use equation (L1) at its full potential.

In the case of Dumbo, the feedback bytes are at indices 0, 3, and 13. We call 13 the *depth*, this is simply the highest index of the feedback.

We chose LFSRs based on these two criteria. Types is defined from $[2,1]$ to $[8,8]$. For types $[2,1]$, and $[5,*]$, we looked at all the possible LFSRs in order to study the influence of their depth.

The state update function of the different LFSR tested are given by equations (L2) to (L21). Their type and depth are given at the second, respectively third, column of Table 4.3.

L2 $x_{j+20} \leftarrow x_j \lll 3 \oplus x_{j+1} \ll 7 \oplus x_{j+11} \gg 7$

L3 $x_{j+20} \leftarrow x_j \lll 3 \oplus x_{j+14} \gg 3 \oplus x_{j+17} \gg 7$

L4 $x_{j+20} \leftarrow x_j \lll 1 \oplus x_{j+3} \gg 3 \oplus x_{j+13} \gg 7$

L5 $x_{j+20} \leftarrow x_j \lll 1 \oplus x_{j+9} \gg 3 \oplus x_{j+15} \gg 7$

L6 $x_{j+20} \leftarrow x_j \lll 3 \oplus x_{j+9} \ll 4 \oplus x_{j+19} \gg 7$

L7 $x_{j+20} \leftarrow x_j \lll 3 \oplus x_{j+1} \ll 5 \oplus x_{j+3} \gg 6$

L8 $x_{j+20} \leftarrow x_j \lll 1 \oplus x_{j+4} \gg 3 \oplus x_{j+19} \gg 5$

L9 $x_{j+20} \leftarrow x_j \lll 1 \oplus x_{j+7} \gg 3 \oplus x_{j+18} \gg 5$

L10 $x_{j+20} \leftarrow x_j \lll 1 \oplus x_{j+3} \gg 3 \oplus x_{j+9} \gg 5$

L11 $x_{j+20} \leftarrow x_j \lll 3 \oplus x_{j+1} \gg 7 \oplus x_{j+17} \ll 4$

L12 $x_{j+20} \leftarrow x_j \lll 3 \oplus x_{j+5} \gg 7 \oplus x_{j+19} \gg 3$

L13 $x_{j+20} \leftarrow x_j \lll 1 \oplus x_{j+5} \ll 7 \oplus x_{j+16} \ll 3$

L14 $x_{j+20} \leftarrow x_j \lll 1 \oplus x_{j+1} \gg 7 \oplus x_{j+9} \gg 3$

L15 $x_{j+20} \leftarrow x_j \lll 1 \oplus x_{j+13} \ll 5 \oplus x_{j+19} \ll 3$

L16 $x_{j+20} \leftarrow x_j \lll 3 \oplus x_{j+14} \gg 7 \oplus x_{j+17} \gg 3$

L17 $x_{j+20} \leftarrow x_j \lll 3 \oplus x_{j+4} \ll 1 \oplus x_{j+5} \gg 6$

L18 $x_{j+20} \leftarrow x_j \lll 1 \oplus x_{j+3} \gg 1 \oplus x_{j+9} \ll 1$

L19 $x_{j+20} \leftarrow x_j \lll 1 \oplus x_{j+4} \gg 1 \oplus x_{j+5} \ll 1$

L20 $x_{j+20} \leftarrow x_j \lll 3 \oplus x_{j+1} \gg 1 \oplus x_{j+8} \lll 7$

L21 $x_{j+20} \leftarrow x_j \lll 3 \oplus x_{j+3} \ll 5 \oplus x_{j+4} \lll 5$

We ran the same experience as in section 4.5.1 for every considered LFSR with $N_{tests} = 120$. For each LFSR, we noted the proportion of runs finished after two days of computations, the average number of nodes effectively traversed in the tree, and the average number of remaining keys. Results are summarized in Table 4.3.

From these experiments, it seems that the depth has a much more relevant impact than the type. Yet, this seems to be quite tailored to our particular attack. Changing the generation of the different masks is generally more impactful, since it can cut down in three the amount of information given to the attacker.

4.6 Conclusion

In nutshell, theoretical and simulated practical blind side-channel attack targeting the LFSR of the Elephant algorithm have been presented. Elephant is a pertinent target. First, Elephant is a finalist for the NIST competition for lightweight cryptography candidates for authenticated encryption. Moreover, Elephant is an interesting target because the internal LFSR only depends on the secret key. In other words, in the use case of Elephant, retrieving the initial state of the LFSR is equivalent to recovering the encryption key.

Different tweaking options have been considered. Going from the most impactful to the least, they are changing the mask derivation for domain separation, and modifying the LFSR, looking at the importance of depth and type.

LFSR	type	depth	finished	N_{nodes}	N_{keys}
(L1)	$[2,1]$	13	53.57%	$2^{41.82}$	$2^{36.59}$
(L2)	$[2,1]$	11	82.5%	$2^{41.23}$	$2^{36.39}$
(L3)	$[5,1]$	17	0.83%	$2^{42.89}$	$2^{34.68}$
(L4)	$[5,1]$	13	94.17%	$2^{39.68}$	$2^{33.68}$
(L5)	$[5,1]$	15	28.33%	$2^{42.13}$	$2^{35.25}$
(L6)	$[5,1]$	19	11.67%	$2^{42.38}$	$2^{36.77}$
(L7)	$[5,2]$	3	100.0%	$2^{30.93}$	$2^{24.93}$
(L8)	$[5,3]$	19	0.83%	$2^{43.99}$	$2^{37.59}$
(L9)	$[5,3]$	18	0.0%	–	–
(L10)	$[5,3]$	9	95.83%	$2^{40.32}$	$2^{34.0}$
(L11)	$[5,4]$	17	0.83%	$2^{43.58}$	$2^{35.6}$
(L12)	$[5,5]$	19	0.0%	–	–
(L13)	$[5,5]$	16	0.0%	–	–
(L14)	$[5,5]$	9	82.5%	$2^{41.43}$	$2^{34.95}$
(L15)	$[5,5]$	19	0.0%	–	–
(L16)	$[5,5]$	17	0.0%	–	–
(L17)	$[8,2]$	5	100.0%	$2^{35.53}$	$2^{29.17}$
(L18)	$[8,7]$	9	78.75%	$2^{41.56}$	$2^{34.79}$
(L19)	$[8,7]$	5	100.0%	$2^{35.41}$	$2^{29.42}$
(L20)	$[8,8]$	8	79.17%	$2^{41.59}$	$2^{35.78}$
(L21)	$[8,8]$	4	100.0%	$2^{34.76}$	$2^{29.29}$

Table 4.3: Type, depth, proportion of runs finished after two days of computations, the average number of nodes traversed, and the number of remaining keys for Dumbo (L1), and LFSRs (L2) to (L21) [7].

This attack is based on the fact that an attacker can retrieve the Hamming weights of the different bytes in the LFSR. The Elephant design, where there exist relations between the different masks of the LFSR, is an added vulnerability to our attack. In [7], in half the cases, the key is retrieved in less than two days.
In this chapter, an important improvement is made. The result [72] is that in three quarters of the cases, the key is retrieved in less than two days. To have this new result the main idea has been that the attacker can wait and decide when to begin the attack. In other words the attacker has to find the best attack position in the LFSR computation progress.

Chapter 5

Attack with a noisy theoretical model

"*La réussite est au bout de l'effort.*"
Fatou Binetou FALL



Contents

5.1 Introduction **82**
5.2 Elephant and Sparkle **82**
5.3 Our approach of BP **82**
5.3.1 Belief propagation (BP) 82
5.3.1.1 Tanner graph 82
5.3.1.2 BP algorithm 83
5.4 Attack on Elephant **83**
5.4.1 Attack path 83
5.4.2 Tanner graph 84
5.4.3 Results 84
5.5 Attack on Sparkle **86**
5.5.1 Attack path 86
5.5.2 Tanner graph 86
5.5.3 Results 86
5.6 Conclusion **88**

5.1 Introduction

In this chapter, we introduce two new attacks targeting authenticated encryption with associated data (AEAD) algoritms: Sparkle and Elephant. These attacks exploit blind side channel analysis (BSCA) to compromise algorithm security. Information leakage, in this context, is modeled as a Hamming weight (HW) with gaussian noise.

For each attack, we apply a belief propagation (BP) algorithm to correlate the different information leaks. This technique makes it possible to reconstruct cryptographic keys using the data obtained despite the added noise. Using the BP algorithm, we demonstrate how leaks can be effectively exploited to reveal sensitive information.

Furthermore, a objective of this chapter is to propose BSCA as a new methodological tool for evaluating the robustness of the subfunctions of a symmetric cryptographic primitive. This approach makes it possible to test the resistance of AEAD algorithms against SCA, thus providing an additional measure of their security.

5.2 Elephant and Sparkle

The description of Elephant and Sparkle are already defined in section 3.3.1 and 3.9.1 respectively.

5.3 Our approach of BP

5.3.1 Belief propagation (BP)

In our attack approach, relations between the different values for which a noisy HW is simulated, and exploited. For that belief propagation (BP) is used. BP was first used by Gallager [53, 93] for decoding low-density parity-check (LDPC) codes [94]. It was then rediscovered by Tanner [95] and formalized by Pearl [96]. The first time that BP was used in SCA on symmetric encryption, in the attack of Veyrat-Charvillon *et al.* [97], then it is studied in [47, 98].

5.3.1.1 Tanner graph

A BP algorithm relies on a bipartite graph called a factor graph (or Tanner graph). To each node in the factor graph is associated some information. The nodes of a factor graph as are of two kinds:

- variable nodes V representing the variables handled by the algorithm under attack;
- factor nodes, representing the equations E between these variables.

An edge links a variable node V with a factor node E, when the equation represented by the factor node E involves the variable node V. An example of a factor graph is illustrated in Figure 5.1.

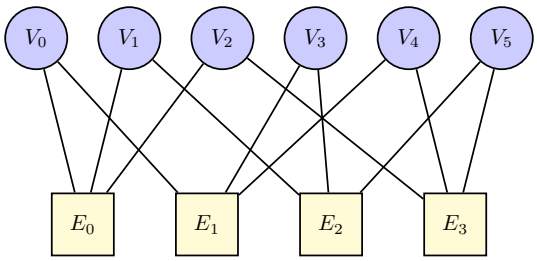


Figure 5.1: Example of a Tanner factor graph part. Circles are variable nodes and squares are factor nodes.

N denotes the set of neighbours for a node. Thus the set $\mathsf{N}(E)$ is made of the variable nodes V involved in equation E, and the set $\mathsf{N}(V)$ is made of factor nodes E that depend on V.

5.3.1.2 BP algorithm

The belief propagation (BP) algorithm inputs are

- the Tanner graph,
- prior probabilities $\mathbb{P}_A(V = v)$ on the different variable nodes V.

The BP algorithm returns a better belief, from the initial values $\mathbb{P}_A(V = v)$. More precisely, BP computes probabilities a posteriori $\mathbb{P}_P(V = v)$ on the different variable nodes according prior probabilities.

The values $\mathbb{P}_P(V = v)$ are computed according to the input prior probability $\mathbb{P}_A(V = v)$ and to the probabilities $\mathbb{P}(V = v|E)$ conditional on factor nodes $E \in \mathsf{N}(V)$ to be satisfied using the following equation:

$$\mathbb{P}_P(V = v) \propto \mathbb{P}_A(V = v) \times \prod_{\mathcal{E} \in \mathsf{N}(V)} \mathbb{P}(V = v|E) \tag{5.1}$$

In practice, nodes in the factor graph exchange information messages with their neighbours. More precisely, since the graph is bipartite, two types of messages are exchanged:

- factor to variable messages between a factor node E and a variable node V, denoted $\mu_{E\to V}$:

$$\mu_{E\to V}(v) = \sum_{v_1,v_2,\cdots,v_{\#\mathsf{N}(E)-1}} E(v, v_1, \cdots, v_{\#\mathsf{N}(E)-1}) \times \prod_{V_i \in \mathsf{N}(E)\setminus\{V\}} \mu_{V_i\to E}(v_i) \tag{5.2}$$

  This equation comes from the law of total probability. Roughly speaking, the quantity $\mu_{E\to V}(v)$ represents the belief from the perspective of factor node E that variable node V take the value v. The information is gathered from the other variables nodes in $\mathsf{N}(E)$.

- variable to factor messages between a variable node V and a factor node E, denoted $\mu_{E\to V}$.

$$\mu_{V\to E}(v) \propto \mathbb{P}_A(V = v) \times \prod_{E' \in \mathsf{N}(V)\setminus\{E\}} \mu_{E'\to V}(v) \tag{5.3}$$

  This equation is essentially Bayes' rule. Here variable node V updates its belief using informations from the factor nodes in $\mathsf{N}(V)$. This equation is meant to be used in alternance with Equation (5.2). So the contribution from node E is deliberately excluded to avoid self-persuasion.

Both equations make independence assumptions to make computations feasible in practice at the expanse of the risk to converge to an incorrect value.

To complete the description of the BP algorithm, an initialization step is done before applying the above equations. The variable to factor messages $\mu_{V\to E}(v)$ are initialized with the prior probabilities $\mathbb{P}_A(V = v)$. In summary, after an initialization phase, BP works by alternatively applying equations (5.2) then (5.3) for every edge (V, E) in the Tanner graph. At the end of the execution, the returned value $\mathbb{P}_P(V = v)$ is computed using equation (5.1) where the probabilities $\mathbb{P}(V = v|E)$ are replaced by their approximations $\mu_{E\to V}(v)$. The number of iterations depends of the rapidity of convergence, which in turn depends on the Tanner graph.

5.4 Attack on Elephant

5.4.1 Attack path

The objective of the described attack is to recover the secret initial state $\mathsf{mask}_K^{0,0}$ of the LFSR. The attack path is based on a similar approach as presented in the theoretical work by Meraneh *et al.* [7, 72]. However, their attack focuses on obtaining HW information and then uses graph traversal with backtracking to search the key space. The main limitation of their work is that they are restricted to scenarios without noise. It is a theoretical and mathematical assumption that is very useful for testing ideas, but it never happens in the real world. In this paper, we improve this attack to consider noisy HW closer to real measurements and propose the use of BP as a solution to address this challenge.

Linear feedback shift registers (LFSRs) find applications in various lightweight cryptography candidates, where their initial state is typically determined by both a key and a nonce. Since the nonce must be altered for

each encryption request, attacks against such schemes are confined to the decryption algorithm. However, in the case of Elephant, the LFSR solely relies on the secret key. As a result, our attack can be employed in an encryption scenario as well.

Three crucial points should be noted as following.

- Retrieving the initial state of the LFSR, which is equal to $\mathsf{mask}_K^{0,0}$, is equivalent to retrieving the secret key. Indeed, the initial state is the result of the known permutation P applied to the key.
- As the retroaction polynomial is publicly known, it is possible to shift the LFSR backwards: an attacker who recovers enough consecutive bytes of the secret stream is able to reconstruct the initial state.
- The smaller the LFSR is, the more the attack is able to succeed. As a consequence, the Dumbo instance (see Figure 3.9) is the most vulnerable one: the following of this section is focused on Dumbo.

Since the LFSR generates a single new byte at each iteration, let the content of the Dumbo LFSR be denoted as follows:

$$(x_j, \cdots, x_{j+19}) = \mathsf{mask}_K^{j,0} \tag{5.4}$$

and let x_{j+20} be the byte generated at iteration j. Similarly, let $(y_j, \cdots, y_{j+19}) = \mathsf{mask}_K^{j,1}$ and $(z_j, \cdots, z_{j+19}) = \mathsf{mask}_K^{j,2}$. By definition of mask, the following hold for all $j \geq 0$:

$$y_j = x_j \oplus x_{j+1} \tag{5.5}$$
$$z_j = x_j \oplus x_{j+2}. \tag{5.6}$$

The attacker can thus exploit two attack vectors: on the one hand, equations (3.1) coming from iterating the LFSR, and on the other hand, equations (5.5) and (5.6) coming from the different masks used for domain separation.

In this use case, the attacker obtain a leakage, simulated by noisy HW on the different bytes of the Dumbo LFSRs.

Once has to remark that a classic CPA is impossible on the $\mathsf{mask}_K^{0,0}$, because the attacker knows zero data in this function.

5.4.2 Tanner graph

In the case of Elephant, the Tanner graph is defined as follows. Variable nodes are the bytes x_j, y_j and z_j of the LFSRs. There are four types of factor nodes:

- $E_{y_0}, E_{y_1}, \ldots, E_{y_{20}}$ represent the equation (5.5).
- $E_{z_0}, E_{z_1}, \ldots, E_{z_{19}}$ represent the equation (5.6).
- We have divided the feedback equation (3.1) into two sub-equations. This helps reduce the amount of computation in BP. Indeed, the number of terms in equation (5.2) is exponential in the number of neighbours in $\mathsf{N}(E)$. The factor nodes corresponding to equation (3.1) are given in Table 5.1.

The Elephant Tanner graph is illustrated in Figure 5.2.

Table 5.1: Intermediate and final feedback equation

intermediate feedback equation	final feedback equation
$E_{tx_{20}} : tx_{20} = (x_0 \lll 3) \oplus (x_3 \ll 7)$	$E_{ux_{20}} : x_{20} = tx_{20} \oplus (x_{13} \gg 7)$
$E_{tx_{21}} : tx_{21} = (x_1 \lll 3) \oplus (x_4 \ll 7)$	$E_{ux_{21}} : x_{21} = tx_{21} \oplus (x_{14} \gg 7)$
$E_{ty_{20}} : ty_{20} = (y_0 \lll 3) \oplus (y_3 \ll 7)$	$E_{uy_{20}} : y_{20} = ty_{20} \oplus (y_{13} \gg 7)$

5.4.3 Results

For 1000 randomly generated keys, and 11 noise levels $0.1 \leq \sigma \leq 1$, the average, standard deviation, median, and quartiles rank of the correct 20 key bytes were computed. Results are given in Table 5.2.

Results indicate that when the noise level is low, BP helps pushing up the ranks of the correct key bytes. An attacker can then leverage the posterior probabilities returned by BP using smart key enumeration techniques

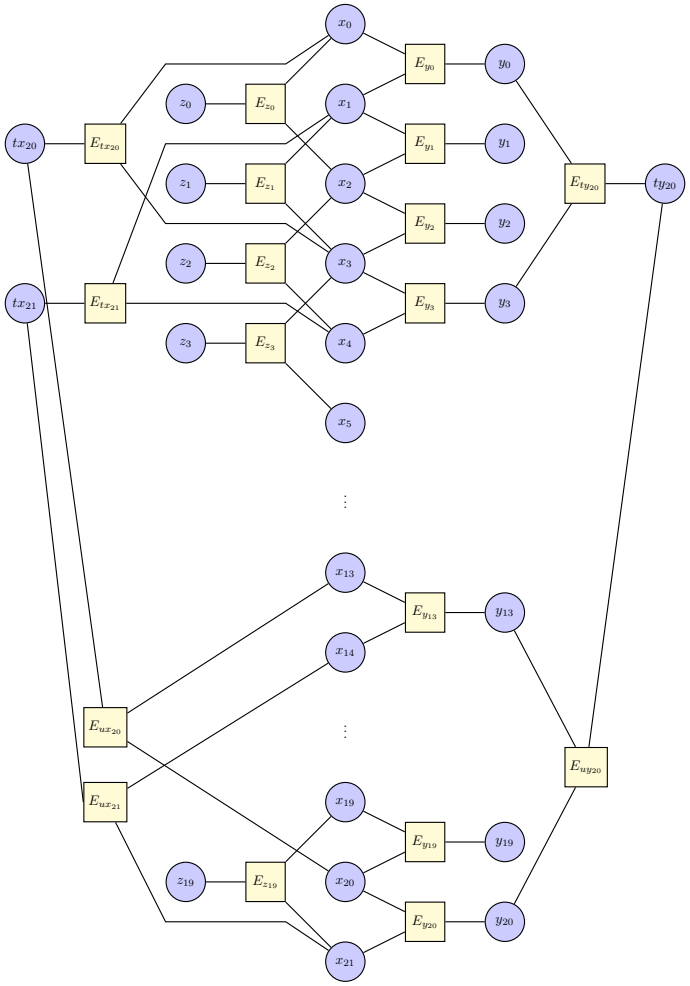


Figure 5.2: Tanner factor graph on LFSR Dumbo of ELEPHANT.

Table 5.2: Rank for all 20 key bytes on 1000 randomly generated keys and for different noise levels σ.

σ	Mean	Standard Deviation	Quartile Q1	Median	Quartile Q3	Max
0.1	3.54	5.97	0	3	3	27
0.15	3.54	5.97	0	3	3	27
0.2	3.67	6.11	0	3	3	31
0.25	4.95	9.31	0	3	3	97
0.3	6.00	10.76	0	3	3	97
0.35	7.46	13.10	0	3	8	97
0.4	9.59	15.72	0	3	8	97
0.5	15.97	23.03	3	8	31	153
0.6	23.65	31.41	3	8	31	157
0.7	33.73	40.11	3	27	36	213
1	70.68	61.42	31	36	92	246

to try and find the whole key. Note that since the LFSR computation only depend on the key, it is possible to reduce the noise level by averaging several traces.

Another important thing to note is that, finding the correct value of *any* $\mathsf{mask}_K^{j,i}$, not just the first one, counts as a successful attack. Depending on the actual value of $\mathsf{mask}_K^{j,i}$, some may be easier to attack than others, as was already noticed by Meraneh *et al.* [7] in the noise-free case.

5.5 Attack on SPARKLE

5.5.1 Attack path

During the initialisation phase of SCHWAEMM256-128, the key K is loaded into the last 128 bits of the SPARKLE384 state, see Figure 3.18. Let split $K = K_1||K_2$ into two 64-bits halves. Then, in the first round, K_1 is the input of Alzette $\mathcal{A}_4$, and K_2 of Alzette $\mathcal{A}_5$. Both these Alzettes are the target of our attack. The rest of this section only describes the attack for a single Alzette since they work exactly the same but for the constant α.

Alzette is a 4-round ARX Feistel cipher, illustrated in see Figure 3.20. In this use case, the attacker can obtain simulated leakage on the different internal values ℓ_i, ℓ'_i, and r_i of Alzette. Here again, we assume a leakage model as a noisy HW of the bytes, as described in equations (2.8) and (2.9). The leakage observed in the first round of Alzette is depicted in Figure 5.3.

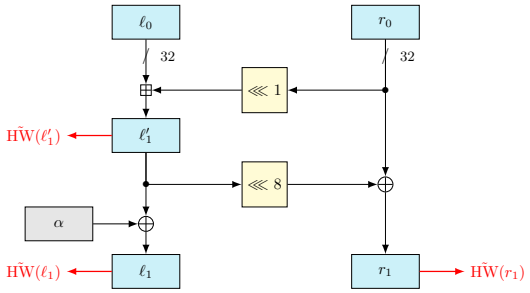


Figure 5.3: Measurement leakage in the first round of Alzette and Figure 3.20.

The different internal values related to each other thanks to the following Alzette equations:

$$\begin{array}{lll} \ell'_1 = \ell_0 \boxplus (r_0 \lll 1) & , \quad \ell_1 = \ell'_1 \oplus \alpha & , \quad r_1 = (\ell'_1 \lll 8) \oplus r_0 \\ \ell'_2 = \ell_1 \boxplus (r_1 \lll 15) & , \quad \ell_2 = \ell'_2 \oplus \alpha & , \quad r_2 = (\ell'_2 \lll 15) \oplus r_1 \\ \ell'_3 = \ell_2 \boxplus r_2 & , \quad \ell_3 = \ell'_3 \oplus \alpha & , \quad r_3 = (\ell'_3 \lll 1) \oplus r_2 \\ \ell'_4 = \ell_3 \boxplus (r_3 \lll 8) & , \quad \ell_4 = \ell'_4 \oplus \alpha & , \quad r_4 = (\ell'_4 \lll 16) \oplus r_3 \end{array} \tag{5.7}$$

To the best of our knowledge, this attack is the first BSCA against SPARKLE.

5.5.2 Tanner graph

In the case of SPARKLE, the variable node in the Tanner graph are defined at the bit level. They are the bits of the different ℓ_i, ℓ'_i, and r_i. for $0 \leq i \leq 4$. Since modular additions are involved, a 31-bit carry variable c_i is also added at each round.

As for the equations, they are of two kinds:

- the bitwise translation of equations (5.7) which describe Alzette,
- the HW equations (sum of bits gives the HW).

In this case, the HW are represented in the graph because they induce new relations between the different bits. The Figure 5.4 shows a part of the Tanner graph of the first round of Alzette.

5.5.3 Results

We ran our attack on both Alzettes in a parallel manner, the only difference being the α_4 constant used by $\mathcal{A}_4$ and α_5 used by $\mathcal{A}_5$.

For the different values of the noise σ and with a single simulated measurement, the number of correct decisions on the input bits, or the number of recovered bits was calculated. The experiment was repeated on 1000 uniformly drawn random keys.

Table 5.3 presents the number of recovered bits of the 64-bit K_1 input of Alzette $\mathcal{A}_4$. Table 5.4 does the same for K_2 and $\mathcal{A}_5$. Finally, Table 5.5 presents the number of bits recovered for the whole 128-bit master key K of SPARKLE.

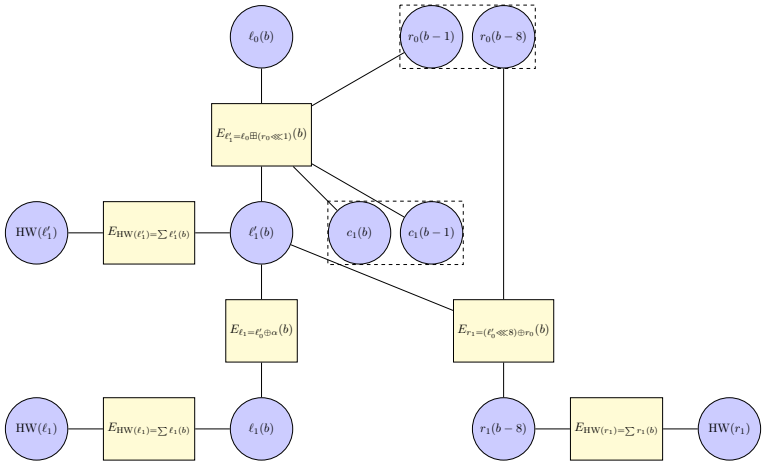


Figure 5.4: Tanner graph of the first round of Alzette. Only a single bit with index $b \in [\![0, 31]\!]$ of each 32-bit register is shown. Relative indices are given mod 32. The carries in the modular operation are denoted $c_1(b)$ and $c_1(b-1)$. Bits belonging to the same 32-bit register are enclosed in dashed lines.

Table 5.3: Number of bits recovered for the 64-bit key K_1 of Alzette $\mathcal{A}_4$.

σ	Mean	Standard Deviation	Min	Quartile Q1	Median	Quartile Q3	Max
0.1	57.08	3.02	36	56	57	59	63
0.15	57.16	2.66	38	56	57	59	63
0.2	57.20	2.57	40	56	57	59	64
0.25	57.15	2.76	38	56	57	59	64
0.3	57.07	2.74	40	56	57	59	64
0.35	56.77	2.94	36	55	57	58	64
0.4	56.63	2.81	39	55	57	58	64
0.45	56.12	3.01	35	55	56	58	64
0.5	55.81	2.81	39	54	56	57	64
0.6	54.83	2.94	36	53	55	56	64
0.7	54.19	2.68	36	53	54	56	62
1	52.86	3.47	35	52	54	55	59

Table 5.4: Number of bits recovered for the 64-bit key K_2 of Alzette $\mathcal{A}_5$.

σ	Mean	Standard Deviation	Min	Quartile Q1	Median	Quartile Q3	Max
0.1	56.98	2.78	36	56	57	59	63
0.15	57.05	2.40	39	56	57	59	63
0.2	57.05	2.43	39	56	57	59	63
0.25	56.96	2.55	39	55	57	59	63
0.3	56.82	2.53	40	55	57	58	63
0.35	56.59	2.64	39	55	57	58	64
0.4	56.29	2.67	40	55	56	58	64
0.45	55.92	2.75	37	54	56	58	64
0.5	55.53	2.66	38	54	55	57	64
0.6	54.64	2.70	36	53	55	56	63
0.7	53.86	2.79	36	53	54	55	61
1	52.78	3.29	33	52	54	54	59

First, note that as in the case of the attack on Elephant, it is possible to reduce the noise level by averaging several traces, since the variables handled only depend on the key.

Second, it is important to consider that the correct key is often not the first attempt, as there may be a few unknown bits for the attacker. Therefore, allowing the attacker to perform an exhaustive search can be relevant to uncover the remaining bits. That is why we grant the attacker a certain computational power.

Let n be the number of bits the attacker can flip on 64 bits. This means that they can try up to $\mathcal{P}$ keys

Table 5.5: Number of bits recovered for the 128-bit master key K.

σ	Mean	Standard Deviation	Min	Quartile Q1	Median	Quartile Q3	Max
0.1	114.06	4.07	89	112	114	117	123
0.15	114.22	3.59	92	112	114	116	123
0.2	114.26	3.51	97	112	114	117	124
0.25	114.11	3.76	92	112	114	116	124
0.3	113.89	3.67	97	112	114	116	125
0.35	113.36	3.94	90	111	113	116	124
0.4	112.92	3.82	95	111	113	115	124
0.45	112.05	4.09	89	110	112	115	123
0.5	111.35	3.85	92	109	111	114	123
0.6	109.48	3.97	89	108	110	112	121
0.7	108.06	3.77	87	106	108	110	120
1	105.65	4.65	85	105	107	108	115

where $\mathcal{P}$ is given by:

$$\mathcal{P} = \binom{64}{0} + \binom{64}{1} + \binom{64}{2} + \cdots + \binom{64}{n}.$$

Since they must do this for both half-keys, the total number of keys they can guess is $\mathcal{P}^2$. Table 5.6 indicates the number of keys found out of 1000 with an attacker possessing a computing power of $\mathcal{P}^2$ as a function of the number of bits n they can flip in a reasonnable amount of time.

It is essential to point out that this exhaustive search does not exploit the a posteriori probabilities returned by BP. Therefore, the use of smarter key enumeration techniques, exploiting these probabilities, will certainly lead to improved results without the need for a specific computing power to perform an exhaustive search. Future projects include the use of key enumeration.

Table 5.6: Number of keys, out of 1000, found for both Alzettes by an attacker with computational power $\mathcal{P}^2$, that can exhaustively flip up to n bits in each 64-bit subkey.

	$n=4, \mathcal{P}^2=2^{39}$		$n=5, \mathcal{P}^2=2^{46}$		$n=6, \mathcal{P}^2=2^{53}$		$n=7, \mathcal{P}^2=2^{59}$		$n=8, \mathcal{P}^2=2^{65}$	
σ	$\mathcal{A}_4$	$\mathcal{A}_5$	$\mathcal{A}_4$	$\mathcal{A}_5$	$\mathcal{A}_4$	$\mathcal{A}_5$	$\mathcal{A}_4$	$\mathcal{A}_5$	$\mathcal{A}_4$	$\mathcal{A}_5$
0.1	175	169	274	283	494	422	610	586	831	787
0.15	174	168	273	281	492	420	611	585	831	786
0.2	175	172	284	285	485	421	611	586	818	776
0.25	171	170	284	280	470	417	612	566	801	740
0.3	177	133	284	246	447	389	582	554	751	716
0.35	155	136	248	229	368	365	540	512	703	664
0.4	137	105	243	187	368	306	523	468	652	618
0.45	110	84	180	167	285	265	431	407	584	559
0.5	88	53	154	112	246	216	380	342	527	495
0.6	34	23	86	51	132	116	225	213	360	326
0.7	9	7	25	18	65	43	150	100	266	211
1	0	0	2	1	15	6	47	35	119	88

5.6 Conclusion

In this thesis, two attacks on two lightweight AEAD algorithms have been successfully conducted and for each attack we used a single simulated measurement.

Firstly, an improvement on the attack against Elephant [7] incorporating a noisy Hamming weight model is presented. This enhanced approach allows for intelligent backtracking of the key, increasing the chances of finding the correct value.

Next, the first BSCA on Sparkle has been described. By exploiting the attacker computational power, we have shown that the attack can indeed recover a significant portion of the secret keys when the noise level is moderate. Using backtracking on the key in an intelligent way that could improve results without doing an exhaustive search for the few remaining bits not predicted by our attack.

The power of the BP algorithm was also highlighted in our research, prompting us to develop a generic tool for it in the future. In future work, we can speed up the computation time of BP by using Walsh-Hadamard transform [99].

Our future research has to focus on transitioning these attacks into practical implementations and to develop a key backtracking algorithm that is intelligent with posterior probabilities. Additionally, we plan to explore the security of Ascon, the NIST competition winner, as a potential target for our future investigations.

Chapter 6

Attack with real leakages

"*Chaque échec est une réussite car on apprend de nos erreurs.*"
Fatou Sylla SARRY



Contents

6.1 Introduction **92**
6.2 Description of Ascon **92**
6.3 Attack path **92**
6.4 Experiments **94**
6.4.1 Mutual information on S-box of Ascon 96
6.4.1.1 Information leakage linked to horizontal values at the sbox output 97
6.4.1.2 Information leakage linked to vertical values at the sbox output 98
6.4.1.3 Information leakage associated with horizontal Hamming weight (HW) of the sbox output 99
6.4.1.4 Information leakage related to the vertical Hamming weight (HW) of the sbox output 100
6.4.1.5 Discussion of results 102
6.5 Conclusion **103**

6.1 Introduction

In this chapter, we target Ascon, the winner of the NIST competition in 2023, presenting the start of a new practical attack in progress.
We describe the identified attack path in detail. More specifically, our study focuses on the initialization phase of the Ascon algorithm, specifically targeting the permutation that contains the sbox, thus representing the vulnerable layer we will exploit. Sboxes are classical functions widely used in symmetric cryptography.

Our approach is to analyze the traces to determine what leaks best at the Ascon sbox level: the value or Hamming weight (HW), either vertically or horizontally. Once this analysis has been carried out, we will consider developing a new attack, distinct from the existing approaches mentioned in the chapter 3.

6.2 Description of Ascon

The descripion of Ascon is explained in section 3.2.1.

6.3 Attack path

The aim of this SCA against Ascon is to find the value of the key used during the initialization phase, is illustred on Fig. 6.1. The attack is possible at this stage because during the initialization phase we know the contents of the initialization vector (IV) and the nonce N, with the exception of the key K. During the first permutation round, the state is composed of a 64-bit IV, the 128-bit key K and a 128-bit nonce N.

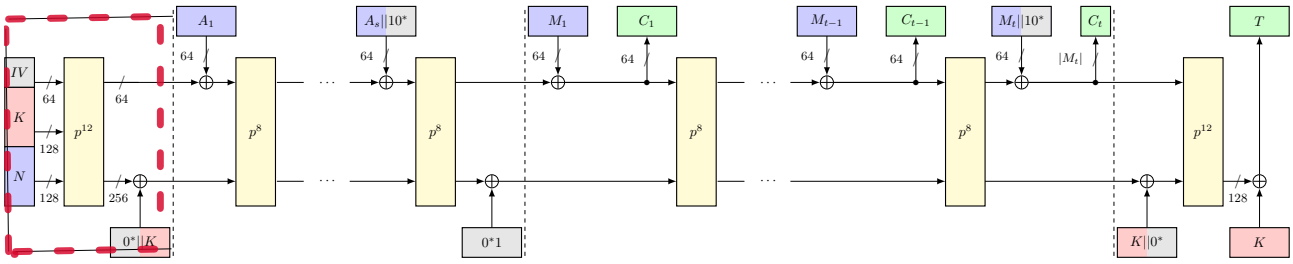


Figure 6.1: Initialization phase of Ascon.

The non-linear properties of Ascon's S-box with controllable nonce allow for a possible leakage that can be exploited with SCA. An intuitive leakage that can be applied is the S-box output. Ascon's S-box is a 5-bit S-box that is applied to the columns of the state. In order to better understand this information leakage, we write the 5 outputs of the S-box as an equation.

Let $(s_0, s_1, s_2, s_3, s_4)$ the 5 output bits of the Ascon S-box and $(v_0, k_0, k_1, n_0, n_1)$ the 5 input bits of the Ascon S-box and $(s_0, s_1, s_2, s_3, s_4)$ the 5 output bits of the Ascon S-box such that:

- v_0 represents the bit of the initialization vector.
- k_0 et k_1 represent the two bits of the key.
- n_0 et n_1 represent the two bits of the Nonce.

The 5 equations linking the 5 input and output bits are shown below (6.5), (6.4), (6.3), and (6.2), (6.1).

$$s_4 = n_0 + n_1 + k_0(1 + v_0 + n_1) \tag{6.1}$$

$$s_3 = (v_0 + 1)(n_1 + n_0) + v_0 + k_0 + k_1 \tag{6.2}$$

$$s_2 = n_1(n_0 + 1) + 1 + k_0 + k_1 \tag{6.3}$$

$$s_1 = n_1 + v_0 + (n_0 + 1)(k_0 + k_1) + k_0 k_1 \tag{6.4}$$

$$s_0 = n_0 + v_0 + k_1 + k_0(n_1 + k_1 + v_0 + 1) \tag{6.5}$$

It's clear that the equation (6.1) i.e. output s_4 S-box shows information leakage relative to key bit k_0. This is because all the other variables in the equation, apart from k_0, are known.
So our main target on the 5 outputs of the S-box is the s_4 output in the recovery of the k_0 key bit input.
Exploiting the fact that the two bits of the input nonce n_0 and n_1 are public, and v_0 is a constant, known bit. We transform equation s_4 as a function of the input values of n_0, n_1 and v_0 in order to find a direct link between s_4 and k_0.

If $n_0 + n_1 = 0$ then this implies the following equation (6.6).

$$s_4 = k_0(1 + v_0 + n_1) \tag{6.6}$$

If $n_0 + n_1 = 1$ then this implies the following equation (6.7).

$$s_4 = 1 + k_0(1 + v_0 + n_1) \tag{6.7}$$

When $(n_0, n_1) = (0, 0)$ or $(n_0, n_1) = (1, 1)$ then equation (6.6) implies the following equation(6.8).

$$s_4 = \begin{cases} k_0(1 + v_0), & if \quad (n_0, n_1) = (0, 0) \\ k_0(v_0), & if \quad (n_0, n_1) = (1, 1) \end{cases} \tag{6.8}$$

When $(n_0, n_1) = (0, 0)$ or $(n_0, n_1) = (1, 1)$ then equation (6.7) implies the following equation(6.9).

$$s_4 = \begin{cases} 1 + k_0(v_0), & if \quad (n_0, n_1) = (0, 1) \\ 1 + k_0(1 + v_0), & if \quad (n_0, n_1) = (1, 0) \end{cases} \tag{6.9}$$

Since v_0 is a known public constant, depending on the values of v_0 we can have finer relation between s_4 and k_0.
(6.8) implies Depending on whether $v_0 = 0$ or $v_0 = 1$, we simplify equation (6.8) into the following equation (6.10).

$$s_4 = \begin{cases} k_0, & if \quad (n_0, n_1, v_0) = (0, 0, 0) \\ k_0, & if \quad (n_0, n_1, v_0) = (1, 1, 1) \\ 0, & if \quad (n_0, n_1, v_0) = (0, 0, 1) \\ 0, & if \quad (n_0, n_1, v_0) = (1, 1, 0) \end{cases} \tag{6.10}$$

Depending on whether $v_0 = 0$ or $v_0 = 1$, we simplify equation (6.9) into the following equation (6.11).

$$s_4 = \begin{cases} 1 + k_0, & if \quad (n_0, n_1, v_0) = (0, 1, 1) \\ 1 + k_0, & if \quad (n_0, n_1, v_0) = (1, 0, 0) \\ 1, & if \quad (n_0, n_1, v_0) = (0, 1, 0) \\ 1, & if \quad (n_0, n_1, v_0) = (1, 0, 1) \end{cases} \tag{6.11}$$

Table 6.1: Link between output s_4 and k_0.

Link between s_4 and k_0	
Output s_4	(n_0, n_1, v_0)
k_0	$(0, 0, 0)$
0	$(0, 0, 1)$
1	$(0, 1, 0)$
$1 + k_0$	$(0, 1, 1)$
$1 + k_0$	$(1, 0, 0)$
1	$(1, 0, 1)$
0	$(1, 1, 0)$
k_0	$(1, 1, 1)$

In summary, The table 6.1 clearly shows that if $(n_0, n_1, v_0) = (0, 0, 0)$ or $(n_0, n_1, v_0) = (1, 1, 1)$ then the output s_4 is equal to the key bit k_0. And when the triplet $(n_0, n_1, v_0) = (0, 1, 1)$ or $(n_0, n_1, v_0) = (1, 0, 0)$ then the output s_4 is equal to the inverse bit of the key k_0.
From this information, we can first find half the key (64 bits) with the 64 outputs s_4 of the S-box in parallel. Secondly, having found the k_0 bits from s_4, we replace the k_0 value in the other equations, which in this case allows us to have a single unknown, the k_1 bit. Finally, it will allow us to find the other half of the remaining key.

6.4 Experiments

We have carried out a practical experiment to assess the robustness of the Ascon-128 v1.2 implementation against side channel analysis (SCA). This section details the experimental protocol used to collect and analyze power consumption traces during execution of the Ascon algorithm on a specific hardware platform. This experiment took place at IRISA's "Laboratoire de Haute Sécurité" (LHS), and was supervised by INRIA research engineer Ronan Lathersme.

The main objective is to identify potential vulnerabilities in the implementation of Ascon-128 v1.2 by observing information leakage through electromagnétique radiation measurements. We aim to determine whether internal cryptographic operations, in particular transformations involving S-boxes, reveal information that can be exploited by an attacker.
This approach is detailed along with the equipment used and the methodology adopted when capturing traces on Ascon's S-box during the initialization phase.

1. Equipment used.
   - Langer EM probe: used to capture electromagnetic emissions during algorithm execution, with a bandwidth of 30 MHz to 3 GHz.
   - Langer PA 303 preamplifier: amplifies the signal received by the probe, with a bandwidth of 3 GHz.
   - Oscilloscope Keysight DSOS404A: features a 4 GHz bandwidth and is capable of capturing up to 20 GSamples per second, enabling accurate and detailed trace collection.
   - STM32F4 target: a 32-bit microcontroller running at a default clock frequency of 7.37 MHz, on which the Ascon implementation is executed.
   - A control computer (Xeon E5-1603v3 2.80GHz, 4 cores, 40GB RAM) is used to orchestrate the measurements and perform the analysis with home-made tools.
2. Methodology.
   - Implementation preparation.
     We use the C implementation of Ascon-128 v1.2, optimized for ARMv7m microcontrollers, available on the Ascon team's GitHub repository. This implementation is chosen for its closeness to real-life scenarios.
   - Triggers configuration.
     Triggers are placed before and after Ascon's S-box to synchronize trace captures. This makes it possible to precisely target the permutation operations where leaks are most likely to occur.

- Communication with oscilliscope are coded in python language.
- Trace Collection.
  Unprotected implementation: We collect 300K traces, each containing 20003 samples. Traces include executions with fixed and random keys to vary test conditions.
  We also collected 300K traces, each containing 20003 samples, with fixed nonces and random keys.
  Protected implementation: For future tests, we plan to collect traces using masking techniques to assess the effectiveness of countermeasures against SCA.

3. Trace analysis.
   The collected traces are analyzed to identify points of interest. To do this, we have formulated four hypotheses that we can categorize according to values or HW, looking specifically at the vertical and horizontal axes .

   - On the horizontal axis:
     the 64 S-boxes are calculated in parallel, allowing us to analyze leakage separately along the horizontal axis. So, when we talk about values or HW on this axis, we're referring to the 64 output bits of the S-boxes. For example, the first byte is the concatenation of the 8 s_4 output bits retrieved from the first 8 calculated S-boxes, and the sum of these 8 bits gives the first HW. We have tested this first byte and its HWs along the horizontal axis.

     The hypothesis on values and hamming weights are presented in Fig. 6.4 and Fig. 6.6 respectively.
   - On the vertical axis:
     we look at the 5 S-box output bits, specifically calculating the value and hamming weights of its 5 S-box output bits. The hypothesis on values and hamming weights are presented in Fig. 6.5 and Fig. 6.7 respectively.

Fig. 6.2, we show the oscilloscope.

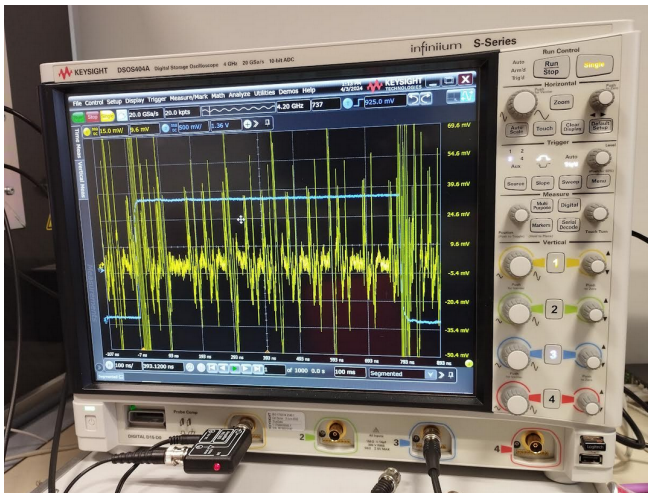


Figure 6.2: oscilloscope Keysight DSOS404A.

Fig. 6.3, we show the placement of the probe on our STM32F4 target.

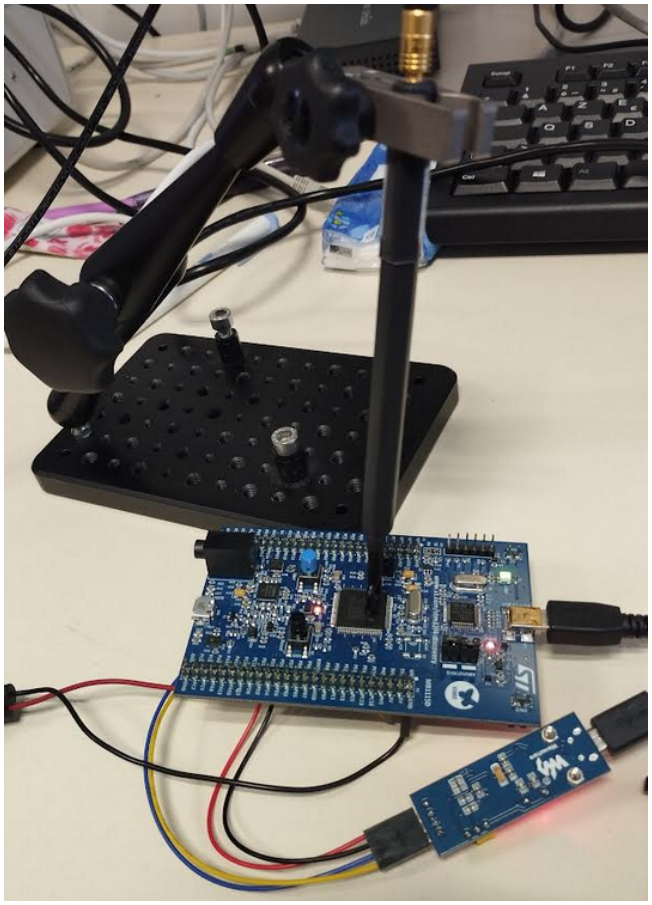


Figure 6.3: probe on STM32F4.

6.4.1 Mutual information on S-box of Ascon

Definition of Mutual Information

For two random vectors X and Y, the mutual information $I(X;Y)$ is defined as:

$$I(X;Y) = H(X) - H(X \mid Y) \tag{6.12}$$

where:

- $H(X)$ is the entropy of X, representing the intrinsic uncertainty of X.
- $H(X \mid Y)$ is the conditional entropy of X) given Y, representing the remaining uncertainty of X when Y is known.

Mutual information can also be expressed symmetrically as:

$$I(X;Y) = H(X) + H(Y) - H(X,Y) \tag{6.13}$$

where:

- $H(Y)$ is the entropy of Y.
- $H(X,Y)$ is the joint entropy of X and Y.

To best exploit the attack path defined on the Ascon S-box and identify the points of interest, we propose four distinct hypotheses using mutual information (MI) [100] as the main analysis tool. Each hypothesis aims to identify the main source of information leakage, enabling to better understand and exploit sbox vulnerabilities. The results of each hypotheses are presented in figures. In these various figures and results, it's important to note that the bits observed are one byte on one of Ascon's 64-bit words (as opposed to the 5 output bits of one of the S-boxes).

6.4.1.1 Information leakage linked to horizontal values at the sbox output

In this first hypothesis, we analyze the first byte, which the first byte is the concatenation of the 8 s_4 output bits retrieved from the first 8 calculated S-boxes. This allows to understand how specific value of horizontal bits can reveal sensitive information.
The results obtained are shown in the corresponding Fig. 6.4, illustrating the level of leakage detected for each horizontal value.

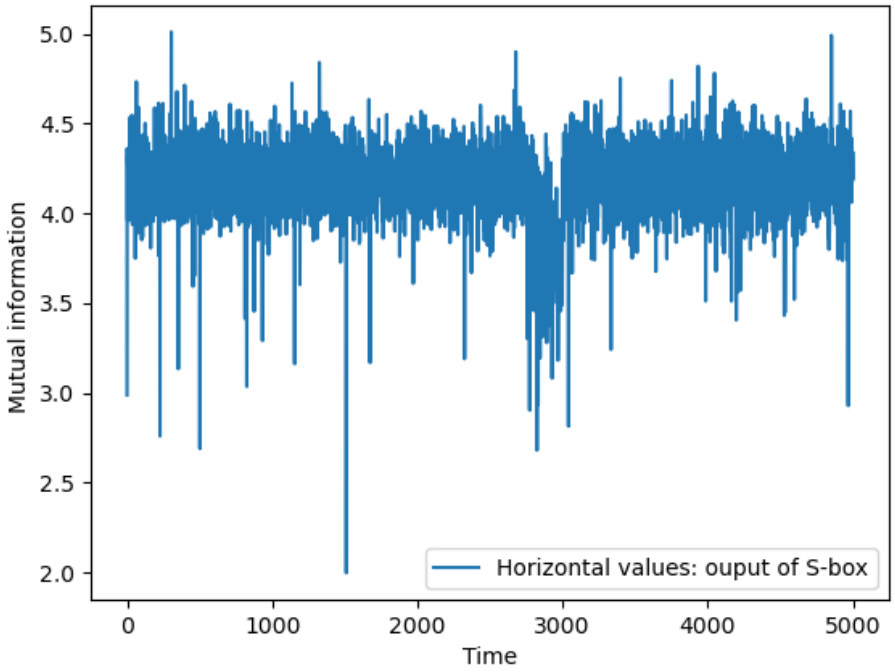


Figure 6.4: Information leakage linked to horizontal values at the S-box output.

6.4.1.2 Information leakage linked to vertical values at the sbox output

The second hypothesis focuses on the vertical values of the 5 S-box output bits. By identifying information leakage related to these values, we can determine whether vertical bits expose sensitive data in a more significant way.
The results of this analysis are presented on Fig. 6.5, showing the leakage levels associated with vertical values.

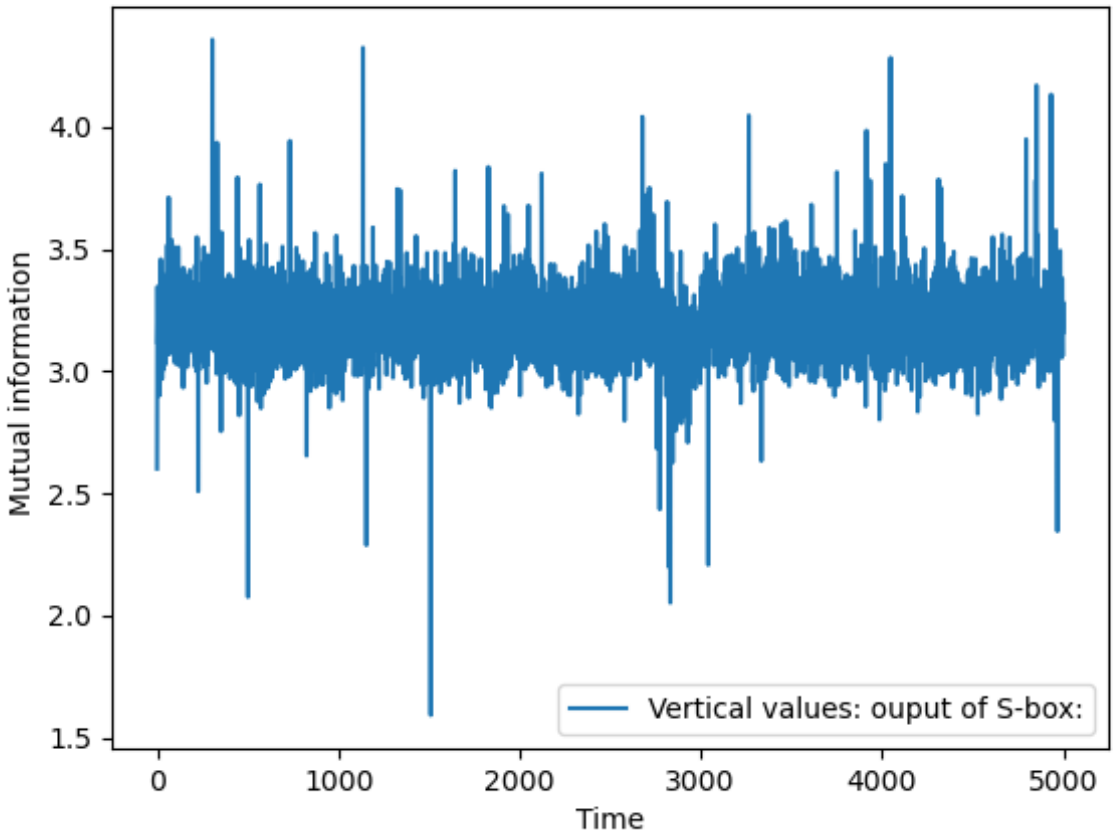


Figure 6.5: Information leakage linked to vertical values at the S-box output.

6.4.1.3 Information leakage associated with horizontal Hamming weight (HW) of the sbox output

In the third hypothesis, we examine the first HW of byte which . where the first byte is the concatenation of the 8 s_4 output bits retrieved from the first 8 calculated S-boxes, and the sum of these 8 bits gives the first HW.
The results of this hypothesis are illustrated on Fig. 6.6, highlighting the leaks detected as a function of horizontal Hamming weight (HW).

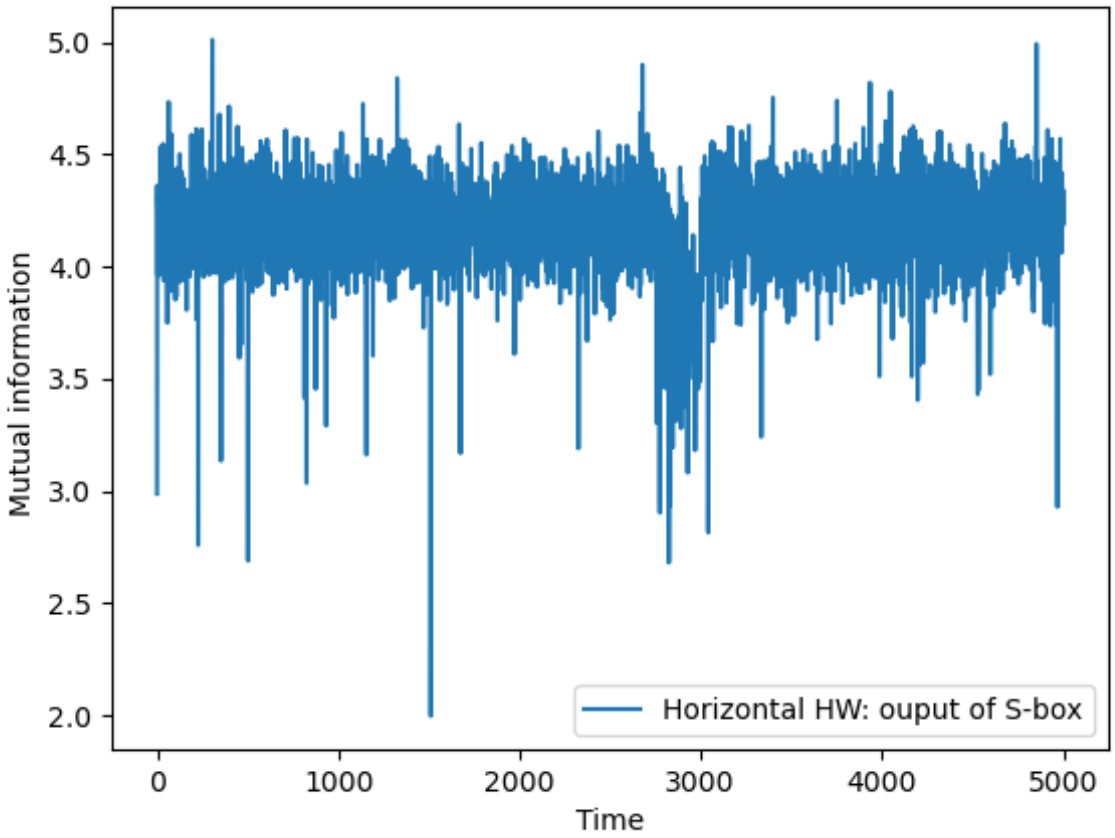


Figure 6.6: Information leakage associated with horizontal Hamming weight (HW) of the S-box output.

6.4.1.4 Information leakage related to the vertical Hamming weight (HW) of the sbox output

Finally, the fourth hypothesis explores information leakage related to the vertical Hamming weight (HW) of the 5 S-box output bits.
The results obtained are presented on Fig. 6.7, illustrating the leakage levels for vertical Hamming weight (HW).

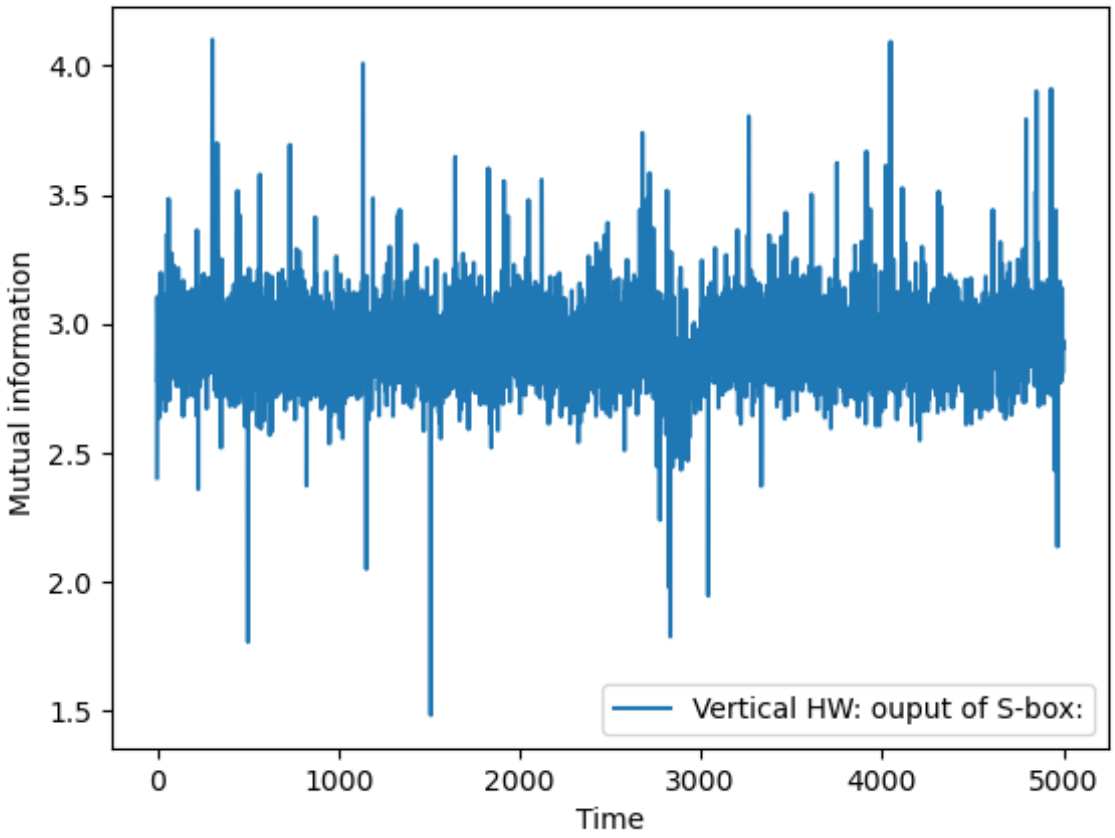


Figure 6.7: Information leakage related to the vertical Hamming weight (HW) of the S-box output.

To obtain an overview of information leakage, we have combined the results of all four hypotheses in a single comparative figure. This figure 6.8 allows to compare the different types of information leakage and determine which are the most significant. The results clearly show that information leakage is more related to the horizontal values and Hamming weight (HW) of the S-box output, rather than to the vertical values and Hamming weight (HW).

1. Horizontal values.
   Analysis of leakage related to horizontal values shows a significant mutual information between bit values on the horizontal lines of the sbox output and information leakage. This indicates that horizontal bits can be exploited to reveal sensitive information.

2. Vertical values.
   In contrast, leakage associated with vertical values is less pronounced. The bits in the vertical columns of the sbox output do not appear to leak information, making them less exploitable for for SCA.

3. Horizontal Hamming weight (HW).
   Horizontal Hamming weight (HW) also show significant leakage. This measure of 1-bit density is a key indicator of information leakage and can be used to refine attack strategies.

4. Vertical Hamming weight (HW).
   Finally, vertical Hamming weight (HW), while showing some leakage, are overall less significant than their horizontal counterparts. This reinforces the idea that the horizontal arrangement of bits is a more critical source of vulnerability.

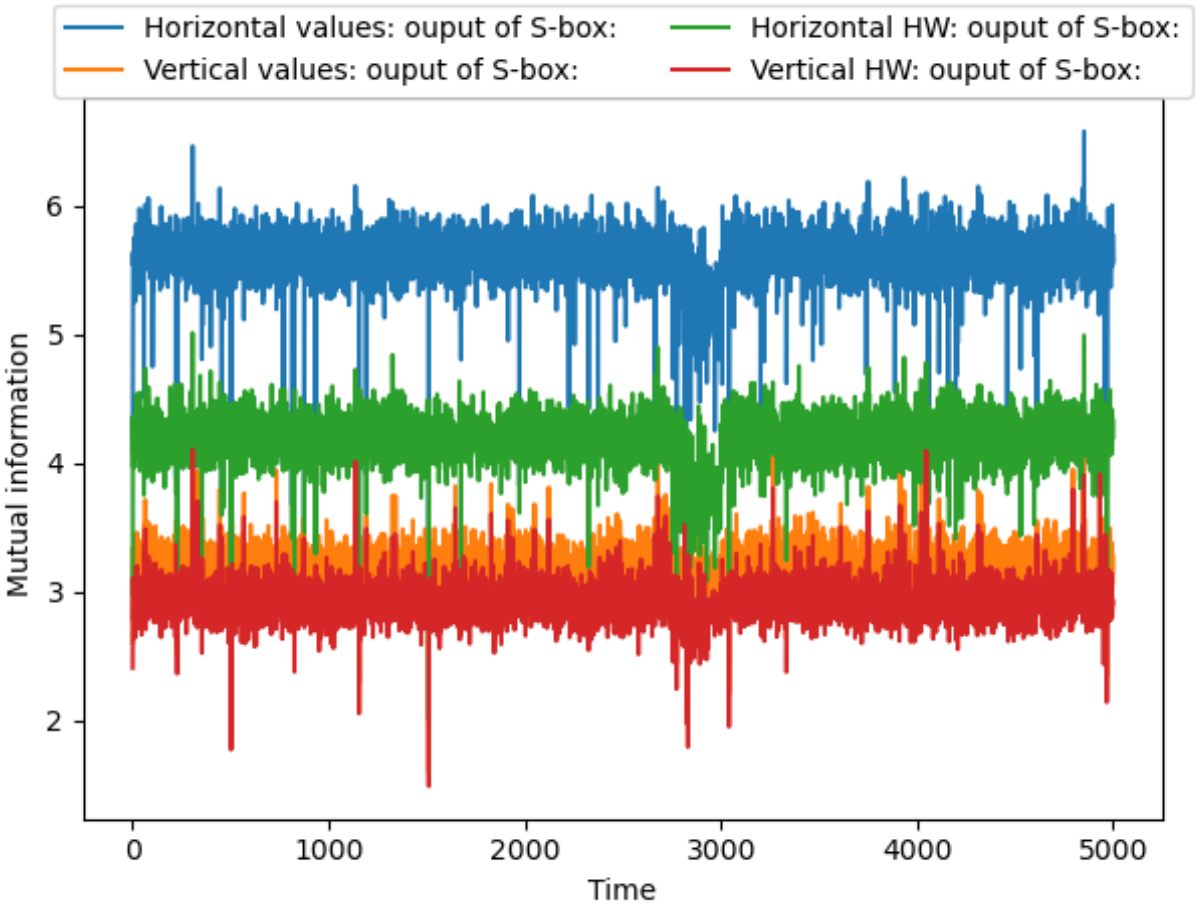


Figure 6.8: All hypotheses

6.4.1.5 Discussion of results

From our results, it's clear that mutual information does not correctly identify points of interest. Although we observe some anomalies with Hamming values or weights on the horizontal axis of our hypotheses, these observations are not sufficient to provide accurate identification. This limitation leads us to analyze potential problems and propose alternatives to overcome these challenges, which we describe in detail in this section.

Potential problems.
This section attempts to identify and lists in detail the various potential problems that may arise. I examine difficulties related to the calculation of mutual information, trace inadequacies, and measurement quality. In addressing each aspect, I aim to understand the sources of these problems and their impact on the correct identification of points of interest.

1. Calculating mutual information.
   - ➤ Inaccurate binning: if values are grouped incorrectly, this can introduce errors into the calculation of mutual information, making it difficult to identify points of interest.
2. Traces.
   - ➤ An insufficient number of traces means that statistical precision is not sufficient to distinguish signal from noise, thus compromising the reliability of the results.
   - ➤ Traces should be sufficiently clean and contain relevant information about the key. If traces are too noisy or of poor quality, points of interest may be masked or misinterpreted.

Potential resolutions.
This section proposes possible solutions to the problems identified above. The aim is to test these solutions during our next practical sessions, in order to improve the identification of points of interest.

1. Improving the way mutual information is calculated.
   - ➤ Use of advanced binning techniques: use techniques such as adaptive binning or k-means-based methods [101] [102] to improve accuracy.
   - ➤ More robust statistical models: employ more sophisticated models, such as Gaussian mixture models [103] [104], to better capture data characteristics.
2. Increase the number and quality of traces.
   - ➤ Acquire more traces: increase the number of traces to improve statistical accuracy.
   - ➤ Pre-processing traces: filter noise and normalize traces to improve data quality.
   - ➤ Use noise reduction techniques: apply techniques such as moving averaging, low-pass filters or Kalman filtering [105] [106] to reduce noise.
3. Dimensional reduction methods.
   - ➤ Principal Component Analysis (PCA) [107] [108]: reduce data dimensions to concentrate relevant information.
   - ➤ Linear Discriminant Analysis (LDA) [109] [110]: find the axes that maximize the separation between key classes.
4. T-test analysis.
   - ➤ T-test [81] for equality of means: compare sample means for two different assumptions (e.g. key assumptions) to identify leakage points. The t-test can reveal statistically significant differences between groups, which is useful for locating points of interest.
   - ➤ Welch's t-test [111] [85]: use this test when the variances of the two samples are not equal, for more robust results.
   - ➤ Difference T-test [112] [113]: used to detect information leakage by comparing traces associated with hypothetically different values of key bits.

6.5 Conclusion

In summary, I have targeted Ascon, the winner of the NIST competition in 2023, presenting the start of a new practical attack in progress. I describe the identified attack path in detail. More specifically, our study focuses on the initialization phase of the Ascon algorithm, specifically targeting the permutation that contains the S-box.

Our approach is to analyze the traces to determine what leaks best at the Ascon sbox: the value or HW, either vertically or horizontally. From our results, it's clear that mutual information does not correctly identify points of interest. Although we observe some differences with values or HW on the horizontal axis of our hypotheses, these observations are not sufficient to provide accurate identification. This limitation led to analyze potential problems and propose alternatives to overcome these challenges. Once this analysis has been carried out in relation to the alternatives, we will consider developing a new attack, distinct from existing approaches.

Part IV

Conclusion and Future works

Chapter 7

Conclusion and Future works

"*Mais dans le domaine de l'esprit, il y'a place pour toutes les oppositions: même ce qui ne triomphe jamais dans la réalité y conserve un dynamisme efficace, et c'est précisément les rêves qui ne se sont point accomplis qui s'y montrent les plus invincibles*"
Stefan Zweig
Proposée par Demba MAR.



Contents

7.1 Conclusion **107**
7.2 Future works **109**
7.3 Introduction **113**
7.3.1 Contexte 113
7.3.2 Motivation 113
7.3.3 Contribution 113
7.4 Les attaques par observation contre les chiffrements intègres légers **114**
7.4.1 Une exploration des attaques par observation sur les 10 chiffrements intègres légers . 114
7.4.2 Attaque avec un modèle théorique 114
7.4.3 Attaque avec un modèle théorique en ajoutant du bruit 114
7.4.3.1 Attaque sur Elephant 114
7.4.3.2 Attaque sur Sparkle 115
7.4.4 Attaque pratique avec des traces réelles 115
7.5 Conclusion **115**
7.6 Perspectives **117**
7.7 Enseignements **118**

7.1 Conclusion

This PhD was motivated by the identification of potential vulnerabilities in the ten finalist algorithms of the NIST competition that can be exploited in side channel analysis (SCA), in order to anticipate future attacks.

The state of the art research 2 enabled me to conduct an in-depth exploration 3 of the ten finalist algorithms in the NIST competition for the standardization of low-cost cryptographic algorithms. I have included brief descriptions of each algorithm, showing their basic primitives, modes of use and proposed variants. This analysis also includes a review of observational attacks specific to each algorithm, highlighting potential vulnerabilities and possible exploitation mechanisms.
By synthesizing this information, the state of the art provides a solid and practical basis for the original contributions presented in the following chapters of the thesis. It also provides a clear link between the fundamental concepts and the specific advances explored in this research.

In the chapter 4, I target Elephant, one of the finalists in the NIST competition, presenting a significant improvement on the first theoretical blind side channel analysis (BSCA) against Elephant. Our study focused on the linear feedback shift register (LFSR) counter used internally. In the case of Elephant, recovering the initial state of the LFSR amounts to recovering the encryption key.
I have described an optimization of this theoretical attack, which has considerably improved its effectiveness. Initially, the attack [7] succeeded in less than two days in only half the cases. Our first contribution thanks to this attack optimization, I achieved a success rate of three quarters of the time.

In the chapter 5, I have successfully carried out two attacks on AEAD algorithms using a single simulated measurement for each attack, demonstrating significant advances in the field of physical cryptanalysis.
In this second contribution to my thesis, I proposed the following two attacks.
Firstly, I presented an improved attack on Elephant with a noisy HW model. This approach enables the use of an intelligent bactraking key, considerably increasing the chances of discovering the correct byte value.
Secondly, I proposed the first blind side channel analysis (BSCA) on Sparkle. I demonstrated that the attack can recover a significant portion of secret keys (about 114 bits found out of 128 bits) when the noise level is moderate. The few remaining bits can be recovered by giving the attacker access to computing power to flip a few bits. Another way of finding a few remaining bits is to use intelligent key bactracking on the output of the BP algorithm.
Our research has also highlighted the potential of the BP algorithm with our generic approach.

In the chapter 6, I have undertaken an in-depth analysis targeting Ascon, the winner of the 2023 NIST competition, exploring the beginnings of a practical attack. In this third contribution, which is in progress at the time of writing, I focused our study on the initialization phase of the Ascon algorithm, with particular emphasis on the permutation containing the sbox, a crucial and often vulnerable component in symmetric encryption systems.
Our approach has been to analyze traces to determine which information leaks best at the Ascon sbox level: whether it's the value or the HW, also exploring the vertical and horizontal aspects of this information leakage. This analysis has laid the foundations for developing a new attack strategy in the future, different from the existing methods discussed in our exploration of SCA on AEAD previously 3.

My work has been published and presented.

➤ **List of publications.**

1. Julien Maillard, Awaleh Houssein Meraneh, Modou Sarry, Christophe Clavier, Hélène Le Bouder, Gaël Thomas: Blind Side Channel Analysis on the Elephant LFSR Extended Version. ICSBT/SECRYPT (Revised Selected Papers) 2022: 20-42
2. Modou Sarry, Hélène Le Bouder, Eïd Maaloouf, Gaël Thomas: Blind Side Channel Analysis Against AEAD with a Belief Propagation Approach. CARDIS 2023: 127-147
3. Modou Sarry: An In-depth Exploration of SCA Against AEAD. (in the process of being submitted)

➤ **List of presentations.**

1. Modou Sarry: Blind side channel analysis (BSCA) on the Elephant and Sparkle. OCIF seminar, Rennes.

2. Modou Sarry: Blind side channel analysis (BSCA) Against AEAD with a Belief Propagation Approach. CARDIS 2023, Amsterdam.
3. Modou Sarry: Side channel analysis (SCA) Against lightweight cryptography. Green seminar organized by IMT-ATLANTIQUE, Rennes.
4. Modou Sarry: Side channel analysis (SCA) Against AEAD. Doctoral students' day, France.
5. Modou Sarry: Side channel analysis (SCA) Against AEAD. Cryptography seminar, University of Rennes.

7.2 Future works

1. Potential of the belief propagation (BP) algorithm.
   - Research has revealed the potential of the BP algorithm to improve the effectiveness of SCA. Considering the development of a generic tool for BP could enable a broader and more flexible assessment of the robustness of cryptographic sub-functions.
   - Using the Walsh-Hadamard transform [99] to accelerate BP's computation time represents a significant improvement, making attacks faster and potentially more effective.
2. Blind side channel analysis (BSCA) on Elephant and Sparkle with real traces.
   - Moving from a mathematical leakage model to the use of real traces in BSCA on Elephant and Sparkle represents a significant step forward. This practical approach makes it possible to test and validate attacks under real-life conditions, offering more robust and applicable results.
   - This shift to practical attacks is essential for verifying the robustness of Elephant and Sparkle algorithms in the face of real and not just theoretical attacks..
3. Continuation of practical attack on Ascon.
   - Having already identified an attack path on Ascon and recovered traces targeting the sbox, the thesis proposes to continue this attack, hoping for results soon. The aim is to develop a new attack, distinct from the previously mentioned methods, exploiting the specific vulnerabilities of the sbox.
   - This approach aims to enrich the body of knowledge on SCA by proposing innovative and effective approaches to compromising Ascon's security, thereby enhancing understanding of current threats.
4. Other lines of research at the beginning of my thesis.
   - Combining invariant and fault attacks.
     Exploring the combination of these two types of attack could reveal new synergies and vulnerabilities, offering a more robust approach to compromising cryptographic algorithms.
   - Multiple linear regression on modular addition.
     This method could be used to identify subtle correlations in cryptographic operations such as XOR.
   - Attack on Grain-128AEADv2, in particular on non-linear feedback shift register (NLFSR) or LFSR:
     By specifically targeting the non-linear feedback shift register (NLFSR) or linear feedback shift register (LFSR) of the AEAD algorithm, namely Grain-128AEADv2. This research could uncover specific weaknesses in these structures, paving the way for more targeted and effective attacks.

Teaching

During my thesis years at IMT-Atlantique, I had the opportunity to serve as a teaching assistant, where I contributed significantly to several courses and academic activities with engineering students and masters specializing in cyber security. Here's a detailed overview of my role as a teaching assistant.

1. Algebra notions for cryptography.
   I've taught the basics of algebra as it relates to cryptography, covering topics such as groups, rings, and finite fields. These concepts are fundamental to understanding modern cryptographic algorithms. To help students master these abstract notions with practical exercises.

2. Probability with theoretical and practical sessions.

   I taught probability theory, explaining key concepts such as distributions, expectation, variance, and probability theorems. To reinforce learning, I organized practical sessions where students applied these concepts.

3. Introduction to cryptography.
   I taught an introduction to cryptography, covering its history, fundamental principles and applications. I also taught the main symmetric and asymmetric algorithms, and explained how they ensure the confidentiality, integrity and authenticity of data through tutorial sessions.

4. Physical attacks on cryptographic algorithms.

   - Side channel analysis (SCA).
     I taught the concepts and techniques of side-channel attacks, such as power analysis and electromagnetic analysis. For example, I supervised practical exercises where students executed attacks on AES to understand how information can leak through power measurements.
   - Fault injection attacks.
     I taught the principles of fault injection attacks, such as disrupting the normal operation of a cryptosystem to reveal its weaknesses. We conducted practical experiments on AES to illustrate how faults can be exploited to compromise security.

5. Support and jury member for engineering students' cryptography projects.
   I supervised and supported the engineering students in their projects on post-quantum cryptography and lightweight cryptography. As a member of the jury, I evaluated their project presentations, providing constructive criticism and suggestions for improving their work.

6. Facilitator and moderator of the MOOC on "sécurité matérielle" on www.fun-mooc.fr.
   I facilitated and moderated a massive open online course (MOOC) on hardware security. This course, offered on the www.fun-mooc.fr platform, covered topics such as basic concepts of hardware security, protection against physical attacks, and assessing the robustness of hardware devices. My role involved presenting the content, answering participants' questions, and leading discussions to enrich the online learning experience.

7. Welcoming the 10 interns and sharing research with ADER.

   As President of ADER, I was joined by fellow PhD students who are keen to share their passion for research. ADER brings together interns, PhD students, post-docs and research engineers. Our main aim is to foster exchanges, collaborations and mutual support between researchers at the start of their careers and those with more experience.
   I have prepared an interactive session with 10 trainees where we present our respective fields of research to discover the themes we are working on, ask questions, and exchange with them on their own projects and interests, and further arouse their curiosity.

Summary in french

"*Tout le monde désire la récompense, mais peu sont prêts à embrasser le chemin qui y mène*"
Docteur Modou SARRY.



Contents

7.1 Conclusion **107**
7.2 Future works **109**
7.3 Introduction **113**
7.3.1 Contexte 113
7.3.2 Motivation 113
7.3.3 Contribution 113
7.4 Les attaques par observation contre les chiffrements intègres légers **114**
7.4.1 Une exploration des attaques par observation sur les 10 chiffrements intègres légers . 114
7.4.2 Attaque avec un modèle théorique 114
7.4.3 Attaque avec un modèle théorique en ajoutant du bruit 114
7.4.3.1 Attaque sur Elephant 114
7.4.3.2 Attaque sur Sparkle 115
7.4.4 Attaque pratique avec des traces réelles 115
7.5 Conclusion **115**
7.6 Perspectives **117**
7.7 Enseignements **118**

7.3 Introduction

7.3.1 Contexte

Le développement rapide de petits dispositifs informatiques tels que les tags RFID, les automates industriels, les capteurs et les cartes à puce est devenu de plus en plus courant avec les progrès technologiques. Cette évolution des ordinateurs de bureau vers des appareils compacts est motivée par divers facteurs, notamment la miniaturisation des composants électroniques, la demande croissante d'automatisation dans divers secteurs industriels et la nécessité de collecter des données en temps réel dans divers environnements. Toutefois, cette transition vers des appareils plus petits soulève de nouvelles préoccupations en matière de sécurité et de protection de la vie privée. Contrairement aux ordinateurs, ces petits appareils disposent de ressources limitées en termes de puissance de calcul, de mémoire et de stockage. En outre, ils peuvent être déployés dans des environnements non sécurisés et être exposés à des attaques physiques. La plupart des normes cryptographiques standards ont été conçues et mises en oeuvre dans le but d'assurer la confidentialité, l'intégrité, l'authentification et la non-répudiation. De nombreuses normes cryptographiques traditionnelles ont été conçues et optimisées pour les environnements de bureau et de serveur, où les ressources sont abondantes. Ces normes visent à trouver un équilibre entre les exigences en matière de sécurité, de performances et de ressources. Toutefois, leur mise en oeuvre sur des appareils aux ressources limitées présente des défis majeurs.

Cette nécessité de disposer de nouveaux algorithmes a incité l'Institut National des Normes et de la Technologie (NIST pour National Institute of Standards and Technology) à lancer un processus visant à solliciter, évaluer et normaliser des algorithmes de chiffrements intègres légers adaptés à une utilisation dans des environnements contraints.

Le NIST a annoncé 56 candidats au premier tour. En août 2019, le NIST a annoncé 32 candidats au second tour et a publié le rapport interne du NIST pour expliquer les critères d'évaluation et la sélection des candidats du second tour. Les critères de sélection comprennent les performances logicielles, les performances matérielles, la sécurité et des fonctionnalités supplémentaires telles que la résistance aux attaques physiques.

En août 2020, le NIST a invité les candidats du deuxième tour à fournir de brèves mises à jour sur leurs algorithmes. Au cours du deuxième tour du processus, le NIST a organisé les troisième et quatrième ateliers sur la cryptographie légère afin de discuter de divers aspects des candidats du deuxième tour et d'obtenir des commentaires précieux pour la sélection des finalistes. En mars 2021, NIST annonce les 10 finalistes.

Le NIST annonce 10 finalistes, à savoir Ascon [6], Elephant [8], GIFT-COFB [9], Grain-128AEADv2 [10], ISAP [11], PHOTON-Beetle [12], Romulus [13], Sparkle [14], TinyJAMBU [15], Xoodyak [16].

Cette thèse débute juste après l'annonce des 10 finalistes, qui sont encore en compétition les uns contre les autres. Parmi les critères de sélection du NIST figure la résistance des algorithmes face aux attaques physiques.

7.3.2 Motivation

L'objectif de cette thèse est d'étudier l'impact des attaques physiques sur la sécurité des algorithmes de chiffrements intègres légers. En particulier, l'étude se concentre sur les algorithmes proposés dans le cadre du concours NIST.

L'objectif principal est d'identifier les vulnérabilités potentielles des dix algorithmes finalistes du concours NIST qui peuvent être exploitées dans les attaques par observation, afin d'anticiper les attaques futures.

7.3.3 Contribution

Ce manuscrit comprend une étude approfondie des attaques par observation sur les 10 algorithmes finalistes. Il propose une attaque par observation à texte inconnus sur le candidat NIST Elephant. En outre, cette recherche a conduit au développement d'outils pour la propagation des croyances, offrant de nouvelles perspectives pour l'évaluation de la sécurité des systèmes cryptographiques. Elle explore également l'utilisation de la propagation de croyance pour améliorer la puissance des attaques par observation à texte inconnus sur le candidat NIST Elephant et un autre candidat, Sparkle . Finalement, il cible le vainqueur de la compétition Ascon pour effectuer le début d'une attaque pratique en cours avec des traces réelles en exploitant le chemin d'attaque trouvé.

7.4 Les attaques par observation contre les chiffrements intègres légers

7.4.1 Une exploration des attaques par observation sur les 10 chiffrements intègres légers

Cette thèse m'a permis de réaliser une étude approfondie des dix algorithmes finalistes du concours NIST pour la normalisation des algorithmes cryptographiques à bas coût. Cette analyse comprend également une étude sur les attaques par obsercation à chaque algorithme, mettant en évidence les vulnérabilités potentielles et les mécanismes d'exploitation possibles.
Ainsi, l'état de l'art fournit non seulement un cadre théorique mais aussi une base pratique pour les contributions originales présentées dans les chapitres 4, 5, et 6 de la thèse. Il fournit également un lien entre les concepts fondamentaux et les avancées spécifiques explorées dans mes travaux.

7.4.2 Attaque avec un modèle théorique

Dans le chapitre 4, j'ai ciblé Elephant, l'un des finalistes de la compétition du NIST. Je présente une amélioration significative de la première attaque théorique par observation à textes inconnus contre cet algorithme [7]. Plus précisément, notre étude se concentre sur le compteur à base de registres à décalage à rétroaction linéaire (LFSR) utilisé en interne. Dans le cas d'Elephant, la récupération de l'état initial du LFSR équivaut à retrouver la clé de chiffrement.

Différentes options de modification ont été envisagées. En allant de la plus impactante à la moins impactante, elles modifient la dérivation du masque pour la séparation des domaines, et modifient le LFSR, en examinant l'importance de la profondeur et du type.

Notre approche exploite l'utilisation de variables intermédiaires qui sont statistiquement dépendantes du secret, en l'occurrence l'état initial du LFSR. Je démontre que cette structure peut compromettre la sécurité d'un cryptosystème face aux attaques par observation. Ce chapitre inclut également une étude de l'influence du choix du LFSR sur la sécurité globale de l'algorithme.

Je propose une amélioration de la première attaque en intégrant une optimisation permettant de déterminer le meilleur moment, par rapport au début du chiffrement, pour lancer l'attaque. Cette optimisation vise à maximiser les chances de succès en choisissant le moment où les fuites d'information sont les plus exploitables.

7.4.3 Attaque avec un modèle théorique en ajoutant du bruit

Dans le chapitre 5, je propose une nouvelle attaque ciblant deux algorithmes de chiffrements intègres légers: Sparkle et Elephant. J'exploite les attaques par observation à texte inconnu sur les deux algorithmes. La fuite d'information, dans ce contexte, est modélisée sous forme de poids de Hamming (HW) avec un bruit gaussien.
Pour chaque attaque, nous appliquons un algorithme de propagation de croyances (BP) afin de corréler les différentes fuites d'information. Cette technique permet de reconstituer les clés cryptographiques en utilisant les données obtenues malgré le bruit ajouté. En utilisant l'algorithme BP, nous démontrons comment les fuites peuvent être exploitées de manière efficace pour révéler des informations sensibles.
En outre, un objectif clé de ce chapitre est de proposer cet outil comme une nouvelle méthode pour évaluer la robustesse des sous-fonctions d'une primitive cryptographique symétrique. Cette approche permet de tester la résistance des algorithmes de algorithmes de chiffrements intègres légers face à des attaques par observation, fournissant ainsi une mesure supplémentaire de leur sécurité.

7.4.3.1 Attaque sur Elephant

J'ai présenté une attaque sur Elephant avec un modèle HW bruité dans le chapitre 5. Cette approche permet d'utiliser une clé bactraking intelligente, ce qui augmente considérablement les chances de découvrir la valeur correcte de l'octet.

7.4.3.2 Attaque sur Sparkle

J'ai proposé la première attaque par observation à texte inconu sur Sparkle dans le chapitre 5. J'ai démontré que l'attaque permet de récupérer une partie importante des clés secrètes (environ 114 bits trouvés sur 128 bits) lorsque le niveau de bruit est modéré. Les quelques bits restants peuvent être récupérés en donnant à l'attaquant l'accès à la puissance de calcul nécessaire . Une autre façon de trouver quelques bits restants est d'utiliser un bactracking intelligent des clés sur la sortie de l'algorithme de propagation croyance. J'ai également mis en évidence le potentiel de l'algorithme de propagation croyance.

7.4.4 Attaque pratique avec des traces réelles

Dans le chapitre 6, j'ai ciblé Ascon, le gagnant de la compétition du NIST en 2023, en présentant le début d'une nouvelle attaque pratique en cours. Je décris en détail le chemin d'attaque identifié. Plus particulièrement, notre étude se concentre sur la phase d'initialisation de l'algorithme Ascon, en ciblant spécifiquement la permutation qui contient la sbox, représentant ainsi la couche vulnérable que je souhaite exploiter.

Notre approche consiste à analyser les traces pour déterminer ce qui fuit le mieux au niveau de la sbox d'Ascon: la valeur ou le poids de Hamming, que ce soit au niveau vertical ou horizontal. Une fois cette analyse effectuée, j' envisage de développer une nouvelle attaque, distincte des approches existantes mentionnées dans le chapitre 6.

7.5 Conclusion

Ce doctorat a été motivé par l'identification de vulnérabilités potentielles dans les dix algorithmes finalistes du concours NIST qui peuvent être exploitées dans les attaques par observation, afin d'anticiper les attaques futures.

L'état de l'art dans la partie 2 m'a permis de mener une exploration des attaques par observation sur les dix algorithmes finalistes du concours NIST pour la normalisation des algorithmes de chiffrements intègres légers dans le chapitre 3.

Dans le chapitre 4, je cible Elephant, l'un des finalistes du concours NIST, en présentant une amélioration significative par rapport au premiere attaque théorique contre Elephant. Notre étude s'est concentrée sur le compteur LFSR utilisé en interne. Dans le cas de Elephant, récupérer l'état initial du LFSR revient à récupérer la clé de chiffrement. J'ai décrit une optimisation de cette attaque théorique, qui a considérablement amélioré son efficacité. Initialement, l'attaque [7] réussissait en moins de deux jours dans seulement la moitié des cas. Notre première contribution grâce à cette optimisation de l'attaque, j'ai atteint un taux de réussite de trois quarts du temps.

Dans le chapitre 5, j'ai réalisé avec succès deux attaques sur les algorithmes de chiffrements intègres légers en utilisant une seule mesure simulée pour chaque attaque. Dans cette deuxième contribution à ma thèse, j'ai proposé les deux attaques suivantes.
Premièrement, j'ai présenté une attaque sur Elephant avec un modèle HW bruité.
Deuxièmement, j'ai proposé la première attaque par observation à texte inconnu sur Sparkle. J'ai démontré que l'attaque peut permettre de récupérer une part importante de la clé secrète. Les quelques bits restants peuvent être récupérés en donnant à l'attaquant l'accès à la puissance de calcul nécessaire.

Dans le chapitre 6, j'ai entrepris une analyse approfondie ciblant Ascon, le vainqueur du concours NIST 2023, explorant les prémices d'une attaque pratique. Dans cette troisième contribution, qui est en cours au moment de la rédaction, j'ai concentré cette étude sur la phase d'initialisation de l'algorithme Ascon, en mettant particulièrement l'accent sur la permutation contenant la sbox, un composant crucial et souvent vulnérable dans les cryptographie symétrique. J'ai également analysé les traces recupérées à partir des mesures sur le sbox de Ascon. Notre approche consiste à analyser les traces afin de déterminer quelles sont les informations qui fuient le mieux au niveau de la sbox de Ascon: qu'il s'agisse de la valeur ou du poids de hamming, en explorant également les aspects verticaux et horizontaux de cette fuite d'informations. Cette analyse permet d'avoir les bases du développement d'une nouvelle stratégie d'attaque à l'avenir, différente des attaques existantes discutées dans notre exploration des attaques par observation sur les dix algorithmes finalistes 3.

Mes travaux ont été publiés et présentés.

➤ **La liste de mes publications.**

1. Julien Maillard, Awaleh Houssein Meraneh, Modou Sarry, Christophe Clavier, Hélène Le Bouder, Gaël Thomas: Blind Side Channel Analysis on the Elephant LFSR Extended Version. ICSBT/SECRYPT (Revised Selected Papers) 2022: 20-42
2. Modou Sarry, Hélène Le Bouder, Eïd Maaloouf, Gaël Thomas: Blind Side Channel Analysis Against AEAD with a Belief Propagation Approach. CARDIS 2023: 127-147
3. Modou Sarry: An In-depth Exploration of SCA Against AEAD. (in the process of being submitted)

➤ **La liste de mes presentations.**

1. Modou Sarry: Blind side channel analysis (BSCA) on the Elephant and Sparkle. Séminaire OCIF, Rennes.
2. Modou Sarry: Blind side channel analysis (BSCA) Against AEAD with a Belief Propagation Approach. CARDIS 2023, Amsterdam.
3. Modou Sarry: Side channel analysis (SCA) Against lightweight cryptography. Séminaire verte organisé par IMT-ATLANTIQUE, Rennes.
4. Modou Sarry: Side channel analysis (SCA) Against AEAD. Journée des doctorants, France.
5. Modou Sarry: Side channel analysis (SCA) Against AEAD. Séminaire en cryptographie, Université de Rennes.

7.6 Perspectives

1. Potentiel de l'algorithme de la propagation de croyance (BP).
   - ➤ La recherche a révélé le potentiel de l'algorithme BP pour améliorer l'efficacité des attaques par observation. Envisager le développement d'un outil générique pour BP pourrait permettre une évaluation plus large et plus souple de la robustesse des sous-fonctions cryptographiques.
   - ➤ L'utilisation de la transformée de Walsh-Hadamard [99] pour accélérer le temps de calcul de BP représente une amélioration significative, rendant les attaques plus rapides.
2. Attaque par observation à texte inconnu sur Elephant and Sparkle avec des traces réelles.
   - ➤ Le passage d'un modèle mathématique de fuite à l'utilisation de traces réelles dans les attaque par observation à texte inconnu sur Elephant et Sparkle, cette approche pratique permet de tester et de valider les attaques dans des conditions réelles, offrant ainsi des résultats plus robustes et plus applicables.
   - ➤ Ce passage à des attaques pratiques est essentiel pour vérifier la robustesse des algorithmes Elephant et Sparkle face à des attaques pratiques et pas seulement théoriques.
3. Poursuite de l'attaque pratique sur Ascon.
   - ➤ Ayant déjà identifié un chemin d'attaque sur Ascon et récupéré des traces ciblant la sbox, la thèse propose de poursuivre cette attaque, en espérant des résultats prochainement. L'objectif est de développer une nouvelle attaque, distincte des méthodes précédemment mentionnées 3, exploitant les vulnérabilités spécifiques de la sbox.
   - ➤ Cette approche vise à enrichir le corpus de connaissances sur SCA en proposant des approches innovantes et efficaces, améliorant ainsi la compréhension des menaces actuelles.
4. D'autres pistes de recherche au début de ma thèse.
   - ➤ Combinaison d'attaques invariantes et d'injection de fautes.
     L'étude de la combinaison de ces deux types d'attaques pourrait révéler de nouvelles vulnérabilités, offrant ainsi une nouvelle approche d'attaquer les algorithmes cryptographiques afin d'anticiper les attaques futures.
   - ➤ Régression linéaire multiple sur l'addition modulaire.
     Cette méthode pourrait être utilisée pour identifier des corrélations subtiles dans des opérations cryptographiques telles que XOR.
   - ➤ Attaque sur Grain-128AEADv2, en particulier sur NLFSR ou LFSR.
     En ciblant spécifiquement le NLFSR ou LFSR de l'algorithme de chiffrement intègre léger, à savoir Grain-128AEADv2. Cette recherche pourrait permettre de découvrir des faiblesses spécifiques dans ces structures, ouvrant ainsi la voie à de nouvelles attaques.

7.7 Enseignements

Pendant mes années de thèse à IMT-Atlantique, j'ai eu l'opportunité de servir en tant qu'assistant d'enseignement, où j'ai contribué de manière significative à plusieurs cours et activités académiques avec des élèves ingénieurs et des masters spécialisés en cybersécurité. Voici un aperçu détaillé de mon rôle en tant qu'assistant d'enseignement.

1. Notions d'algèbre pour la cryptographie.

   J'ai enseigné les bases de l'algèbre en rapport avec la cryptographie, en abordant des sujets tels que les groupes, les anneaux et les corps finis. Ces concepts sont fondamentaux pour comprendre les algorithmes cryptographiques modernes. Pour aider les étudiants à maîtriser ces notions abstraites à l'aide d'exercices pratiques.

2. Probabilité avec des sessions théoriques et pratiques.
   J'ai enseigné la théorie des probabilités, en expliquant des concepts clés tels que les distributions, l'espérance, la variance et les théorèmes de probabilité. Pour renforcer l'apprentissage, j'ai organisé des séances pratiques au cours desquelles les étudiants ont appliqué ces concepts.

3. Introduction à la cryptographie.
   J'ai enseigné une introduction à la cryptographie, couvrant son histoire, ses principes fondamentaux et ses applications. J'ai également enseigné les principaux algorithmes symétriques et asymétriques, et expliqué comment ils garantissent la confidentialité, l'intégrité et l'authenticité des données à travers des sessions de tutorat.

4. Attaques physiques contre les algorithmes cryptographiques.

   - Attaques par observation.
     J'ai enseigné les concepts et les techniques d'attaques par observation, tels que l'analyse de la puissance et l'analyse électromagnétique. Par exemple, j'ai supervisé des exercices pratiques au cours desquels les étudiants ont exécuté des attaques sur l'AES pour comprendre comment des informations peuvent fuir à travers les mesures de puissance.
   - Attaques par injection de fautes.
     J'ai enseigné les principes des attaques par injection de fautes, telles que la perturbation du fonctionnement normal d'un algorithme cryptographique à partir d'une injection de faute pour révéler des vulnérabilités.J'ai mené des expériences pratiques sur l'AES pour illustrer comment les failles peuvent être exploitées pour compromettre la sécurité et de retrouver la clé.

5. Soutien et membre du jury pour les projets de cryptographie des élèves ingénieurs.
   J'ai supervisé et soutenu les élèves ingénieurs dans leurs projets sur la cryptographie post-quantique et la cryptographie légère. En tant que membre du jury, j'ai évalué les présentations de leurs projets, en fournissant des critiques constructives et des suggestions pour améliorer leur travail.

6. Facilitateur et modérateur du MOOC sur la "sécurité matérielle" sur www.fun-mooc.fr.
   J'ai animé et modéré un cours en ligne ouvert et massif (MOOC) sur la sécurité matérielle. Ce cours, proposé sur la plateforme www.fun-mooc.fr, couvrait des sujets tels que les concepts de base de la sécurité matérielle, la protection contre les attaques physiques et l'évaluation de la robustesse des dispositifs matériels. Mon rôle consistait à présenter le contenu, à répondre aux questions des participants et à animer des discussions afin d'enrichir l'expérience d'apprentissage en ligne.

7. Accueillir les 10 stagiaires et partager les notions de la recherche avec l'ADER.
   En tant que présidente de l'ADER, j'ai été rejointe par des collègues doctorants désireux de partager leur passion pour la recherche. L'ADER regroupe des stagiaires, des doctorants, des post-doctorants et des ingénieurs de recherche. Notre objectif principal est de favoriser les échanges, les collaborations et l'entraide entre les chercheurs en début de carrière et ceux plus expérimentés.
   J'ai préparé une session interactive avec 10 stagiaires où nous présentons nos domaines de recherche respectifs pour découvrir les thèmes sur lesquels nous travaillons, poser des questions, et échanger avec eux sur leurs propres projets et intérêts, et éveiller leur curiosité.

Bibliography



[1] Wikipedia contributors. Wikimedia Commons. URL: https://commons.wikimedia.org/wiki.

[2] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Duplexing the Sponge: Single-Pass Authenticated Encryption and Other Applications. In *SAC*. Springer, 2011.

[3] Michaël Peeters Guido Bertoni, Joan Daemen and Gilles Van Assche. Permutation-based Encryption, Authentication and Authenticated Encryption. *DIAC*, pages 8, 21, 29, 2012.

[4] Avik Chakraborti, Nilanjan Datta, Mridul Nandi, and Kan Yasuda. Beetle Family of Lightweight and Secure Authenticated Encryption Ciphers. *IACR TCHES.*, 2018.

[5] Kazuhiko Minematsu Avik Chakraborti, Tetsu Iwata and Mridul Nandi. Blockcipher-based authenticated encryption: How small can we go? In *Cryptographic Hardware and Embedded Systems*, pages 277–298. CHES, 2017.

[6] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. Ascon. *Submission to the CAESAR competition*, 2014.

[7] Awaleh Houssein Meraneh, Christophe Clavier, Hélène Le Bouder, Julien Maillard, and Gaël Thomas. Blind side channel on the elephant lfsr. In *SECRYPT*. 2022.

[8] Tim Beyne, Yu Long Chen, Christoph Dobraunig, and Bart Mennink. Elephant v2. *NIST lightweight competition*, 2021.

[9] Subhadeep Banik, Avik Chakraborti, Akiko Inoue, Tetsu Iwata, Kazuhiko Minematsu, Mridul Nandi, Thomas Peyrin, Yu Sasaki, Siang Meng Sim, and Yosuke Todo. Gift-cofb. *Cryptology ePrint Archive*, 2020.

[10] Martin Hell, Thomas Johansson, Alexander Maximov, Willi Meier, and Hirotaka Yoshida. Grain-128aead, round 3 tweak and motivation. 2021.

[11] CE Dobraunig, Maria Eichlseder, Stefan Mangard, Florian Mendel, BJM Mennink, Robert Primas, and Thomas Unterluggauer. Isap v2. 0. 2020.

[12] Zhenzhen Bao, Avik Chakraborti, Nilanjan Datta, Jian Guo, Mridul Nandi, Thomas Peyrin, and Kan Yasuda. Photon-beetle authenticated encryption and hash family. *NIST Lightweight Compet. Round*, 1:115, 2019.

[13] Tetsu Iwata, Mustafa Khairallah, Kazuhiko Minematsu, and Thomas Peyrin. Duel of the titans: the romulus and remus families of lightweight aead algorithms. *IACR Transactions on Symmetric Cryptology*, 2020.

[14] Christof Beierle, Alex Biryukov, Luan Cardoso dos Santos, Johann Großschädl, Léo Perrin, Aleksei Udovenko, Vesselin Velichkov, Qingju Wang, and Alex Biryukov. Schwaemm and esch: lightweight authenticated encryption and hashing using the sparkle permutation family. *NIST round*, 2, 2019.

[15] Hongjun Wu and Tao Huang. Tinyjambu: A family of lightweight authenticated encryption algorithms (version 2). *Submission to the NIST Lightweight Cryptography Standardization Process*, 2021.

[16] Joan Daemen, Seth Hoffert, Michaël Peeters, G Van Assche, and R Van Keer. Xoodyak, a lightweight cryptographic scheme. 2020.

[17] K. A. McKay, I. L. E. Bassham, M. Sönmez Turan, and N. Mouha. Report on lightweight cryptography. Report, National Institute of Standards and Technology, 2017.

[18] L. Bassham, C. Çalık, K. McKay, N. Mouha, and M. Sönmez Turan. Profiles for the lightweight cryptography standardization process. Report, National Institute of Standards and Technology, 2017. Available at https://csrc.nist.gov/CSRC/media/Publications/white-paper/2017/04/26/profiles-for-lightweight-cryptography-standardization-process/draft/documents/profles-lwc-std-proc-draft.pdf.

[19] National Institute of Standards and Technology (NIST). Federal register notice. Report, National Institute of Standards and Technology, 2018. Available at https://www.govinfo.gov/content/pkg/FR-2018-08-27/pdf/2018-18433.pdf.

[20] X. Dong, L. Qin, S. Sun, and X. Wang. Key guessing strategies for linear key-schedule algorithms in rectangle attacks. In O. Dunkelman and S. 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 3–33. Springer, 2022.

[21] D. Chang, D. Hong, J. Kang, and M. Sönmez Turan. Resistance of ascon family against conditional cube attacks in nonce-misuse setting. *IEEE Access*, 11:4501–4516, 2023.

[22] M. Sönmez Turan, K. A. McKay, D. Chang, C. Çalık, I. L. E. Bassham, J. Kang, and J. Kelsey. Status report on the second round of the nist lightweight cryptography standardization process. Report, National Institute of Standards and Technology, 2021.

[23] Ueli Maurer. In Moni Naor, editor, *TCC 2004: Theory of Cryptography, First Theory of Cryptography Conference*, volume 2951 of *Lecture Notes in Computer Science*, Cambridge, MA, USA, February 2004. Springer. Add any additional information here.

[24] Moses Liskov, Ronald L. Rivest, and David Wagner. Tweakable block ciphers. In *Proceedings of the 22nd Annual International Cryptology Conference (CRYPTO)*, pages 31–46. Springer, 2002.

[25] Whitfield Diffie and Martin E. Hellman. Privacy and authentication: An introduction to cryptography. *Proceedings of the IEEE*, 67(3):397–427, 1979.

[26] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[27] Ronald L. Rivest. Data encryption using feedback. *Communications of the ACM*, 21(7):562–563, 1978.

[28] Ralph C. Merkle. Secure communications over insecure channels. *Communications of the ACM*, 24(4):294–299, 1981.

[29] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Cryptographic sponge functions. In *Proceedings of the 2007 NIST Workshop on Cryptographic Hash Functions*, 2007.

[30] NIST. SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions. *FIPS 202*, 2015.

[31] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles van Assche. The Keccak Reference. 2011.

[32] Phillip Rogaway, Mihir Bellare, and John Black. Ocb: A block-cipher mode of operation for efficient authenticated encryption. In *Proceedings of the 8th ACM Conference on Computer and Communications Security (CCS 2001)*, pages 196–205. ACM, 2001.

[33] Tadayoshi Kohno, John Viega, and Doug Whiting. The theta-cb mode for authenticated encryption. In *Proceedings of the 8th International Conference on Fast Software Encryption (FSE 2001)*, volume 2355 of *Lecture Notes in Computer Science*, pages 139–147. Springer, 2002.

[34] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power analysis attacks: Revealing the secrets of smart cards*, volume 31. Springer, 2008.

[35] Maamar Ouladj and Sylvain Guilley. *Side-Channel Analysis of Embedded Systems*. Springer, 2021.

[36] Helena Handschuh and Howard M Heys. A timing attack on rc5. In *International Workshop on Selected Areas in Cryptography*. Springer, 1998.

[37] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *CRYPTO*. Springer, 1999.

[38] François-Xavier Standaert. Introduction to side-channel attacks. In *Secure integrated circuits and systems*. Springer, 2010.

[39] Stefan Mangard. A simple power-analysis (SPA) attack on implementations of the AES key expansion. In *ICISC*. Springer, 2002.

[40] Eric Brier, Christophe Clavier, and Francis Olivier. Correlation power analysis with a leakage model. In *CHES*. Springer, 2004.

[41] Cédric Archambeau, Eric Peeters, François-Xavier Standaert, and Jean-Jacques Quisquater. Template attacks in principal subspaces. In *CHES*. Springer, 2006.

[42] Stefan Mangard. A simple power-analysis (SPA) attack on implementations of the AES key expansion. In *ICISC*. Springer, 2002.

[43] Ronald L. Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[44] Suresh Chari, Josyula R Rao, and Pankaj Rohatgi. Template attacks. In *Cryptographic Hardware and Embedded Systems-CHES*. Springer, 2002.

[45] Lilian Lerman, Benoît Ôrtaçtepe, and Emmanuel Prouff. Side channel attack: an approach based on machine learning. *Journal of Cryptographic Engineering*, 4(2):123–136, 2014.

[46] Gabriela Hospodar, Zdenek Kotásek, Lejla Batina, Benoît Gierlichs, Bart Preneel, and Ingrid Verbauwhede. Machine learning in side-channel analysis: A first study. *Journal of Cryptographic Engineering*, 1(4):293–302, 2011.

[47] Hélène Le Bouder, Ronan Lashermes, Yanis Linge, Gaël Thomas, and Jean-Yves Zie. A Multi-round Side Channel Attack on AES Using Belief Propagation. In *FPS*. Springer, 2016.

[48] E.; Standaert F.X. Whitnall, C.; Oswald. The myth of generic dpa...and the magic of learning. In *Cryptology—CT-RSA*, pages 183–205. Springer: Berlin, Germany, 2014.

[49] Thomas Korak Christoph Dobraunig, Maria Eichlseder and Florian Mendel. Side-channel analysis of keymill. In *Channel Analysis and Secure Design - 8th International Workshop, COSADE 2017, Paris, France*, volume 10348 of *Lecture Notes in Computer Science*, pages 138–152. Springer, 2017.

[50] Dirmanto Jap Siang Meng Sim and Shivam Bhasin. Dapa: differential analysis aided power attack on (non-) linear feedback shift registers. In *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, Lecture Notes in Computer Science, pages 169–191. TCHES, 2021.

[51] Christine Rivier and Jean-Jacques Quisquater. Improvement of side channel attacks by the algebraic attack against DSA. In *2001 International Conference on Information Security and Cryptology (ICISC)*, pages 212–222. Springer, 2001.

[52] Lejla Batina, Benedikt Gierlichs, and Pim Tuyls. Csi–a side-channel attack on the MICAz wireless sensor network node. In *2011 12th International Conference on Cryptographic Hardware and Embedded Systems (CHES)*, pages 283–299. Springer, 2011.

[53] Robert G. Gallager. Low-density parity-check codes. *IRE Trans. on Information Theory*, 1962.

[54] Yanis Linge, Cécile Dumas, and Sophie Lambert-Lacroix. Using the joint distributions of a cryptographic function in side channel analysis. In *COSADE*. Springer, 2014.

[55] Hélène Le Bouder, Ronan Lashermes, Yanis Linge, Gaël Thomas, and Jean-Yves Zie. A Multi-round Side Channel Attack on AES Using Belief Propagation. In *Foundations and Practice of Security*. Springer, 2016.

[56] Christophe Clavier and Léo Reynaud. Improved blind side-channel analysis by exploitation of joint distributions of leakages. In *CHES*. Springer, 2017.

[57] Christophe Clavier, Léo Reynaud, and Antoine Wurcker. Quadrivariate improved blind side-channel analysis on boolean masked aes. In *International Workshop on Constructive Side-Channel Analysis and Secure Design*, pages 153–167. Springer, 2018.

[58] Melissa Azouaoui, Kostas Papagiannopoulos, and Dominik Zürner. Blind side-channel sifa. In *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE.

[59] Ville Yli-Mäyry, Rei Ueno, Noriyuki Miura, Makoto Nagata, Shivam Bhasin, Yves Mathieu, Tarik Graba, Jean-Luc Danger, and Naofumi Homma. Diffusional side-channel leakage from unrolled lightweight block ciphers: A case study of power analysis on prince. *IEEE Transactions on Information Forensics and Security*, 2020.

[60] Hannes Gross, Erich Wenger, Christoph Dobraunig, and Christoph Ehrenhofer. Ascon hardware implementations and side-channel evaluation. *Microprocessors and Microsystems*, 2017. Consulté le 5 juillet 2024.

[61] Keyvan Ramezanpour, Paul Ampadu, and William Diehl. Scarl: Side channel analysis with reinforcement learning on the ascon authenticated cipher. `https://doi.org/10.48550/arxiv.2006.03995`, 2020.

[62] Sinian Luo, Weibin Wu, Yanbin Li, Ruyun Zhang, and Zhe Liu. An efficient soft analytical side-channel attack on ascon. Springer, 2022.

[63] Azade Rezaeezade, Abraham Basurto-Becerra, Léo Weissbart, and Guilherme Perin. One for all, all for ascon: Ensemble-based deep learning side-channel analysis. `https://eprint.iacr.org/2023/1922`, 2023.

[64] Léo Weissbart and Stjepan Picek. Lightweight but not easy: Side-channel analysis of the ascon authenticated cipher on a 32-bit microcontroller. `https://eprint.iacr.org/2023/1598.pdf`, 2023.

[65] Tim Beyne, Yu Long Chen, Christoph Dobraunig, and Bart Mennink. Dumbo, Jumbo, and Delirium: Parallel Authenticated Encryption for the Lightweight Circus. *IACR Transactions on Symmetric Cryptology*, 2020.

[66] Daniel J. Bernstein. How to Stretch Random Functions: The Security of Protected Counter Sums. *J. Cryptol.*, 1999.

[67] Atul Luykx, Bart Preneel, Elmar Tischhauser, and Kan Yasuda. A MAC Mode for Lightweight Block Ciphers. In Thomas Peyrin, editor, *Fast Software Encryption FSE*. Springer, 2016.

[68] Robert Granger, Philipp Jovanovic, Bart Mennink, and Samuel Neves. Improved Masking for Tweakable Blockciphers with Applications to Authenticated Encryption. In *EUROCRYPT*. Springer, 2016.

[69] Andrey Bogdanov, Miroslav Knezevic, Gregor Leander, Deniz Toz, Kerem Varici, and Ingrid Verbauwhede. Spongent: a Lightweight Hash Function. In *CHES*. Springer, 2011.

[70] Richard Haeussler. *Implementation, Benchmarking, and Protection of Lightweight Cryptography Candidates*. PhD thesis, George Mason University, 2021.

[71] Louis Vialar. Fast Side-Channel Key-Recovery Attack against Elephant Dumbo. 2022.

[72] Julien Maillard, Awaleh Houssein Meraneh, Modou Sarry, Christophe Clavier, Hélène Le Bouder, and Gaël Thomas. Blind side channel analysis on the Elephant LFSR Extended version. *SECRYPT BOOK*, 2023.

[73] Modou Sarry, Eïd Maaloouf, Hélène Le Bouder, and Gaël Thomas. Blind Side Channel Analysis Against AEAD with a Belief Propagation Approach. pages 127–147. 2023.

[74] Subhadeep Banik, Andrey Bogdanov, Takanori Isobe, Kyoohyung Jeong, Tetsu Kobayashi, Jorge Nakahara Jr., SangJae Sim, and Yosuke Wang. GIFT: A small present towards reaching the NIST lightweight cryptography competition goals. In *Cryptographic Hardware and Embedded Systems - CHES 2017*, volume 10529 of *Lecture Notes in Computer Science*, pages 321–345. Springer, 2017.

[75] Said Hamdioui Mottaqiallah Taouil Johanna Sepúlveda Cezar Reinbrecht, Abdullah Aljuffri. Grinch: A cache attack against gift lightweight cipher. In *2021 Design, Automation, Test in Europe Conference, Exhibition (DATE) 2021, Grenoble, France, February*, pages 1–5. IEEE, 2021.

[76] Hamdioui S Taouil M Sepulveda J Aljuffri A, Reinbrecht C. Balanced dual-mask protection scheme for gift cipher against power attacks. In *IEEE Computer Society, Los Alamitos, CA, USA*, pages 1–6. IEEE 40th VLSI Test Symposium (VTS), 2022.

[77] Herzoff J. Babinkostova L. & Serra E Benjamin, A. Deep learning based side channel attacks on lightweight cryptography (student abstract). In *Proceedings of the AAAI Conference on Artificial Intelligence*. AAAI, 2022.

[78] Martin Hell, Thomas Johansson, Alexander Maximov, Willi Meier, and Hirotaka Yoshida. Grain-128aead, round 3 tweak and motivation. 2021.

[79] Wei Cheng Ng Siang Meng Sim Shivam Bhasin, Dirmanto Jap. Survey on the Effectiveness of DAPA-Related Attacks against Shift Register Based AEAD Schemes . 2022.

[80] Yuhang Ji. Side-channel Evaluation of ISAP , 2022.

[81] William S. Gosset. The probable error of a mean. *Biometrika*, 6(1):1–25, 1908.

[82] Louis Vialar. *Side Channel Analysis of NIST Lightweight Cryptography Candidates* . PhD thesis, 2022.

[83] C. Beierle, G. Leander, A. Moradi, T. Peyrin, Y. Sasaki, and H. Wang. Skinny-aead and skinny-hash: Authenticated encryption and hashing using the skinny block cipher. In K. G. Paterson and D. Stebila, editors, *Advances in Cryptology - EUROCRYPT 2021 - 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, volume 12697 of *Lecture Notes in Computer Science*, pages 39–69. Springer, 2021.

[84] Yuhang Ji Xiangjun Lu Shipei Qu Tengfei Wang Chi Zhang Hongyi Zhang Xiaolin Zhang Dawu Gu, Pei Cao. Side-Channel Evaluation on Protected Implementations of Several NIST LWC Finalists , 2022.

[85] Gerard Goodwill, Benjamin Jun, Josh Jaffe, and Pankaj Rohatgi. A testing methodology for side-channel resistance validation. In *NIST Non-Invasive Attack Testing Workshop*, 2011.

[86] Sachin Sachin Flora Coleman, Behnaz Rezvani and William Diehl. Side channel resistance at a cost: A comparison of arx-based authenticated encryption. In *30th International Conference on Field-Programmable Logic and Applications (FPL), Gothenburg, Sweden, 2020*, pages 193–199, 2020.

[87] et al Cassi Chen. Cpa and dlpa on hardware implementations of schwaemm and gift. In *30th International Conference on Field-Programmable Logic and Applications (FPL), Gothenburg, Sweden, 2020*, 2022.

[88] CAESAR. *Competition for Authenticated Encryption: Security, Applicability, and Robustness.*

[89] Lukasz Chmielewski Ellen Gunnarsdottir Vahid Jahandideh Tom Stock Lejla Batina, Ileana Buhan and Léo Weissbart. Side-Channel Evaluation Report on Implementations of Several NIST LWC Finalists , 2022.

[90] Antoine Joux and Pascal Delaunay. Galois LFSR, Embedded Devices and Side Channel Weaknesses. In *INDOCRYPT*, 2006.

[91] Sanjay Burman, Debdeep Mukhopadhyay, and Kamakoti Veezhinathan. LFSR based stream ciphers are vulnerable to power attacks. In *INDOCRYPT*, volume 4859 of *Lecture Notes in Computer Science*, pages 384–392. Springer, 2007.

[92] Abhishek Chakraborty, Bodhisatwa Mazumdar, and Debdeep Mukhopadhyay. Fibonacci LFSR vs. galois LFSR: which is more vulnerable to power attacks? In *SPACE*, volume 8804 of *Lecture Notes in Computer Science*, pages 14–27. Springer, 2014.

[93] Robert G. Gallager. *Low Density Parity check codes.* PhD thesis, MIT, Cambridge, MA, 1963.

[94] Sae-Young Chung, G. David Forney Jr., Thomas J. Richardson, and Rüdiger L. Urbanke. On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit. *IEEE Communications Letters*, 2001.

[95] Robert M. Tanner. A recursive approach to low complexity codes. *IEEE Trans. on Information Theory*, 1981.

[96] Judea Pearl. Reverend Bayes on Inference Engines: A Distributed Hierarchical Approach. In *National Conference on Artificial Intelligence*. AAAI Press, 1982.

[97] Nicolas Veyrat-Charvillon, Benoît Gérard, and François-Xavier Standaert. Soft Analytical Side-Channel Attacks. In *Advances in Cryptology - ASIACRYPT 2014*.

[98] Vincent Grosso and François-Xavier Standaert. ASCA, SASCA and DPA with Enumeration: Which One Beats the Other and When? In *ASIACRYPT 2015*, pages 291–312. Springer.

[99] Ahmed Nasir and Rao Kamisetty Ramamohan. Walsh hadamard transform. pages 99–152. 1975.

[100] Benedikt Gierlichs, Lejla Batina, Pim Tuyls, and Bart Preneel. Mutual information analysis. In *Cryptographic Hardware and Embedded Systems-CHES*. Springer, 2008.

[101] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, 1:281–297, 1957.

[102] Annelie Heuser and Werner Schindler. Learning from uncertainty - theoretical results and practical applications of side channel analysis. In *International Conference on Cryptographic Hardware and Embedded Systems*, pages 429–446. Springer, 2012.

[103] D. A. Reynolds. Gaussian mixture models. *Encyclopedia of Biometrics*, 741(659):827–832, 1966.

[104] Huan Lou, Praveen Kumar Yadav, David Naccache, Debdeep Mukhopadhyay, and Rajat Subhra Chakraborty. On estimating the secret from the source distribution. In *International Conference on Cryptographic Hardware and Embedded Systems*, pages 629–649. Springer, 2018.

[105] R. E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45, 1960.

[106] Thanh-Ha Le, Pierre-Alain Moellic, Alexandre Berzati, Benoit Robisson, Emmanuel Prouff, and Adeline Thillard. A kalman filter-based side-channel attack on multiply-accumulate operations. *Journal of Cryptographic Engineering*, 6(3):189–208, 2016.

[107] Karl Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2(11):559–572, 1901.

[108] Léo Lerman, Benoît Gierlichs, and Ingrid Verbauwhede. Power analysis attack: An approach based on machine learning. In *Smart Card Research and Advanced Applications*, pages 125–140. Springer, 2011.

[109] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(2):179–188, 1936.

[110] Léo Lerman, Amir Moradi, and Claire Veyrat-Charvillon. Side-channel analysis of cryptographic software via early-terminating em measurements. In *International Conference on Smart Card Research and Advanced Applications*, pages 62–76. Springer, 2015.

[111] Bernard L. Welch. The generalization of student's problem when several different population variances are involved. *Biometrika*, 34(1-2):28–35, 1947.

[112] Henry Scheffé. The analysis of variance. *Biometrika*, 76(1):13–22, 1989.

[113] Tobias Schneider and Amir Moradi. Leakage assessment methodology - a clear roadmap for side-channel evaluations. *International Conference on Smart Card Research and Advanced Applications*, 9514:495–513, 2015.

COLLEGES BRETAGNE LOIRE SCIENCES POUR L'INGENIEUR ET LE NUMERIQUE

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Titre : Les attaques physiques sur les chiffrements intègres légers

Mots clés : NIST, attaques physiques, propagation de croyance, attaques par observation à textes inconnus, chiffrements intègres légers

Résumé : Les dispositifs interconnectés tels que les réseaux de capteurs, la santé, les systèmes de contrôle distribués, l'internet des objets et les systèmes cyber-physiques sont de plus en plus répandus. Cette transition vers de petits appareils engendre de nouvelles préoccupations en matière de sécurité et de vie privée, car de nombreux algorithmes cryptographiques classiques ne répondent pas toujours aux exigences spécifiques de ces dispositifs restreints. Face à ce besoin de nouvelles solutions, l'Institut National des Normes et de la Technologie (NIST pour National Institute of Standards and Technology) a lancé une compétition visant à solliciter, évaluer et normaliser les algorithmes cryptographiques légers adaptés à ces environnements contraints. Ma thèse se concentre sur la compréhension de l'impact des attaques physiques sur la sécurité des chiffrements intègres légers, en particulier ceux proposés dans le cadre de la compétition du NIST. Nous proposons ainsi une attaque par observations à textes inconnus sur Elephant. Ces travaux aboutissent également au développement d'outils pour la propagation de croyance, ouvrant ainsi de nouvelles perspectives pour l'évaluation de la sécurité des systèmes cryptographiques. Parallèlement, nous explorons l'utilisation de la propagation de croyance pour effectuer des attaques par observation à textes inconnus sur Elephant et Sparkle. Finalement, nous ciblons le vainqueur de la compétition Ascon pour effectuer le début d'une attaque pratique en cours avec des traces réelles en exploitant le chemin d'attaque trouvé. Ces recherches contribuent de manière significative à la compréhension et à l'amélioration de la sécurité des systèmes embarqués et des dispositifs interconnectés dans des domaines critiques.

Title : Side channel analysis against AEAD.

Keywords : : BSCA, NIST, BP, physical attacks, lightweight cryptography, AEAD

Abstract : Today, various emerging sectors such as sensor networks, healthcare, distributed control systems, the Internet of Things, and cyber-physical systems involve interconnected devices with significant constraints. These devices typically communicate wirelessly and collaborate to perform specific tasks. The transition from desktop computers to small devices raises new security and privacy concerns. Many conventional cryptographic standards have been optimized for desktop and server environments, sometimes overlooking the specific security, performance, and resource requirements of constrained devices. This necessity for new algorithms has prompted national institute of standards and technology (NIST) to launch a process aimed at soliciting, evaluating, and standardizing lightweight cryptographic algorithms suitable for use in constrained environments. This thesis studies the impact of physical attacks on the security. Specifically, the study focuses on algorithms proposed in the context of the NIST competition. It includes an in-depth study of physical attacks on the 10 finalist algorithms. Additionally, it offers a detailed analysis of a blind side channel analysis (BSCA) on NIST candidate Elephant. Furthermore, this research has led to the development of tools for belief propagation (BP), providing new perspectives for evaluating the security of cryptographic systems. It also explores the use of belief propagation (BP) to improve the power of blind side channel analysis (BSCA) on NIST candidate Elephant and another candidate, Sparkle. Finally, this thesis presents the beginnings of a practical attack on Ascon in progress, aimed at identifying potential vulnerabilities and assessing the robustness of the system. These results make a significant contribution to understanding and improving the security of embedded systems and interconnected devices in critical domains.