Méthodologie et outils pour la mise en pratique des attaques par collision et attaques horizontales sur l'exponentiation modulaire

Ibrahima Diop

N°d'ordre NNT : **2017LYSEM010**

THESE de DOCTORAT DE L'UNIVERSITE DE LYON
opérée au sein de
l'Ecole des Mines de Saint-Etienne

Ecole Doctorale N° 488
Sciences, Ingénierie, Santé

Spécialité de doctorat : Microélectronique
Discipline : Attaques par canaux auxiliaires

Soutenue publiquement le 11/04/2017, par :
Ibrahima DIOP

Méthodologie et outils pour la mise en pratique des attaques par collision et attaques horizontales sur l'exponentiation modulaire

---

Devant le jury composé de :

Guilley Sylvain, Professeur, Telecom-ParisTech	Président
Leveugle Régis, Professeur, Institut polytechnique de Grenoble	Rapporteur
Clavier Christophe, Professeur, Université de Limoges	Rapporteur
Guilley Sylvain, Professeur, Telecom-ParisTech	Examinateur
Teglia Yannick, Expert en Cryptologie, Gemalto	Examinateur
Maurine Philippe, Maitre de conférences, Université de Montpelier 2	Directeur de thèse
Liardet Pierre-Yvan, Expert en Cryptologie, STMicroelectronics	Encadrant
Linge Yanis, Ingénieur en Cryptologie, STMicroelectronics	Co-encadrant

Spécialités doctorales	Responsables :	Spécialités doctorales	Responsables
SCIENCES ET GENIE DES MATERIAUX	K. Wolski Directeur de recherche	MATHEMATIQUES APPLIQUEES	O. Roustant, Maître-assistant
MECANIQUE ET INGENIERIE	S. Drapier, professeur	INFORMATIQUE	O. Boissier, Professeur
GENIE DES PROCEDES	F. Gruy, Maître de recherche	SCIENCES DES IMAGES ET DES FORMES	JC. Pinoli, Professeur
SCIENCES DE LA TERRE	B. Guy, Directeur de recherche	GENIE INDUSTRIEL	X. Delorme, Maître assistant
SCIENCES ET GENIE DE L'ENVIRONNEMENT	D. Graillot, Directeur de recherche	MICROELECTRONIQUE	Ph. Lalevée, Professeur

EMSE : Enseignants-chercheurs et chercheurs autorisés à diriger des thèses de doctorat (titulaires d'un doctorat d'État ou d'une HDR)

ABSI	Nabil	CR	Génie industriel	CMP
AUGUSTO	Vincent	CR	Image, Vision, Signal	CIS
AVRIL	Stéphane	PR2	Mécanique et ingénierie	CIS
BADEL	Pierre	MA(MDC)	Mécanique et ingénierie	CIS
BALBO	Flavien	PR2	Informatique	FAYOL
BASSEREAU	Jean-François	PR	Sciences et génie des matériaux	SMS
BATTON-HUBERT	Mireille	PR2	Sciences et génie de l'environnement	FAYOL
BEIGBEDER	Michel	MA(MDC)	Informatique	FAYOL
BLAYAC	Sylvain	MA(MDC)	Microélectronique	CMP
BOISSIER	Olivier	PR1	Informatique	FAYOL
BONNEFOY	Olivier	MA(MDC)	Génie des Procédés	SPIN
BORBELY	Andras	MR(DR2)	Sciences et génie des matériaux	SMS
BOUCHER	Xavier	PR2	Génie Industriel	FAYOL
BRODHAG	Christian	DR	Sciences et génie de l'environnement	FAYOL
BRUCHON	Julien	MA(MDC)	Mécanique et ingénierie	SMS
CAMEIRAO	Ana	MA(MDC)	Génie des Procédés	SPIN
CHRISTIEN	Frédéric	PR	Science et génie des matériaux	SMS
DAUZERE-PERES	Stéphane	PR1	Génie Industriel	CMP
DEBAYLE	Johan	CR	Sciences des Images et des Formes	SPIN
DEGEORGE	Jean-Michel	MA(MDC)	Génie industriel	Fayol
DELAFOSSE	David	PR0	Sciences et génie des matériaux	SMS
DELORME	Xavier	MA(MDC)	Génie industriel	FAYOL
DESRAYAUD	Christophe	PR1	Mécanique et ingénierie	SMS
DJENIZIAN	Thierry	PR	Science et génie des matériaux	CMP
DOUCE	Sandrine	PR2	Sciences de gestion	FAYOL
DRAPIER	Sylvain	PR1	Mécanique et ingénierie	SMS
FAUCHEU	Jenny	MA(MDC)	Sciences et génie des matériaux	SMS
FAVERGEON	Loïc	CR	Génie des Procédés	SPIN
FEILLET	Dominique	PR1	Génie Industriel	CMP
FOREST	Valérie	MA(MDC)	Génie des Procédés	CIS
FOURNIER	Jacques	Ingénieur chercheur CEA	Microélectronique	CMP
FRACZKIEWICZ	Anna	DR	Sciences et génie des matériaux	SMS
GARCIA	Daniel	MR(DR2)	Sciences de la Terre	SPIN
GAVET	Yann	MA(MDC)	Sciences des Images et des Formes	SPIN
GERINGER	Jean	MA(MDC)	Sciences et génie des matériaux	CIS
GOEURIOT	Dominique	DR	Sciences et génie des matériaux	SMS
GONDRAN	Natacha	MA(MDC)	Sciences et génie de l'environnement	FAYOL
GRAILLOT	Didier	DR	Sciences et génie de l'environnement	SPIN
GROSSEAU	Philippe	DR	Génie des Procédés	SPIN
GRUY	Frédéric	PR1	Génie des Procédés	SPIN
GUY	Bernard	DR	Sciences de la Terre	SPIN
HAN	Woo-Suck	MR	Mécanique et ingénierie	SMS
HERRI	Jean Michel	PR1	Génie des Procédés	SPIN
KERMOUCHE	Guillaume	PR2	Mécanique et Ingénierie	SMS
KLOCKER	Helmut	DR	Sciences et génie des matériaux	SMS
LAFOREST	Valérie	MR(DR2)	Sciences et génie de l'environnement	FAYOL
LERICHE	Rodolphe	CR	Mécanique et ingénierie	FAYOL
MALLIARAS	Georges	PR1	Microélectronique	CMP
MOLIMARD	Jérôme	PR2	Mécanique et ingénierie	CIS
MOUTTE	Jacques	CR	Génie des Procédés	SPIN
NIKOLOVSKI	Jean-Pierre	Ingénieur de recherche	Mécanique et ingénierie	CMP
NORTIER	Patrice	PR1	Génie des Procédés	SPIN
O CONNOR	Rodney Philip	MA(MDC)	Microélectronique	CMP
OWENS	Rosin	MA(MDC)	Microélectronique	CMP
PERES	Véronique	MR	Génie des Procédés	SPIN
PICARD	Gauthier	MA(MDC)	Informatique	FAYOL
PIJOLAT	Christophe	PR0	Génie des Procédés	SPIN
PINOLI	Jean Charles	PR0	Sciences des Images et des Formes	SPIN
POURCHEZ	Jérémy	MR	Génie des Procédés	CIS
ROBISSON	Bruno	Ingénieur de recherche	Microélectronique	CMP
ROUSSY	Agnès	MA(MDC)	Microélectronique	CMP
ROUSTANT	Olivier	MA(MDC)	Mathématiques appliquées	FAYOL
SANAUR	Sébastien	MA(MDC)	Microélectronique	CMP
STOLARZ	Jacques	CR	Sciences et génie des matériaux	SMS
TRIA	Assia	Ingénieur de recherche	Microélectronique	CMP
VALDIVIESO	François	PR2	Sciences et génie des matériaux	SMS
VIRICELLE	Jean Paul	DR	Génie des Procédés	SPIN
WOLSKI	Krzystof	DR	Sciences et génie des matériaux	SMS
XIE	Xiaolan	PR1	Génie industriel	CIS
YUGMA	Gallian	CR	Génie industriel	CMP

Mise à jour : 12/12/2016

Remerciements

Je remercie en premier lieu mon directeur de thèse Philippe Maurine pour ses conseils et tout son temps passé à me guider dans la bonne direction au cours de ces trois années. Sa gentillesse, son ouverture d'esprit ne m'ont jamais été de trop durant ces trois années.

J'adresse un remerciement spécial à mes responsables scientifiques à STMicroelectronics Pierre-Yvan Liardet et Yanis Linge, qui m'ont été d'un grand secours durant cette thèse.
Cette thèse n'aurait jamais vu le jour sans Pierre-Yvan qui m'a fait confiance et qui a su m'accompagner depuis le premier jour de mon stage chez STMicroelectrnoincs. Merci à Yanis d'avoir su me redonner la confiance aux moments où j'en avais le plus besoin. Sa rigueur scientifique m'a beaucoup servi d'exemples durant cette thèse.

J'adresse également ma gratitude à Régis Leveugle et à Christophe Clavier qui m'ont fait l'honneur d'être les rapporteurs de cette thèse. Je suis également très reconnaissant de Yannick Teglia et de Sylvain Guilley d'avoir accepté d'être membre de mon jury.
J'ai la chance d'avoir dans mon jury de thèse des personnes dont j'ai pris du plaisir, depuis des années, à étudier leurs travaux respectifs.

J'ai passé d'excellents moments au sein de l'équipe CryptoLab de STMicroelectronics. Je tiens tout particulièrement à remercier Thomas Ordas. Sa gentillesse, son soutien et ses conseils m'ont énormément aidé durant cette thèse.
Merci également à mes collègues Bruno Nicolas et Daniele Fronte.
Je me dois aussi de remercier Vincent Deveaud, Laurent Di-Russo, Aurélien Poupeau, Alexandre Sarafianos, Christophe Laurencin, Eric Wong et Isabelle Reynaud.

Mes derniers remerciements, et sans doute les plus importants, vont évidemment à ma famille et mes amis. Je remercie tout particulièrement mon épouse Fama Fall pour tout le bonheur que nous partageons dans notre vie quotidienne.

À ma
grand-mère, à
ma fille.

Table des matières

Table des figures

Liste des tableaux

Liste des algorithmes

Liste des acronymes

AES *Advanced Encryption Standard*

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

BCDC *Bounded Collision Detection Criterion*

CMOS *Complementary Metal Oxyde Semiconductor*

CPA *Correlation Power Analysis*

CRT *Chinese Remainder Theorem*

DEMA *Differential Electromagnetic Analysis*

DES *Data Encryption Standard*

DCA *Differential Cluster Analysis*

DPA *Differential Power Analysis*

DSA *Digital Signature Algorithm*

DSO *Digital Sampling Oscilloscope*

DTW *Dynamic Time Warping*

DUT *Device Under Test*

EM *Électromagnétique*

FFT *Fast Fourier Transformation*

GE *Guessing Entropy*

HCA *Horizontal Correlation Analysis*

IC *Integrated Circuit*

L2R *Left-To-Right*

LIM *Long Integer Multiplication*

LNR *Leakage-to-Noise Ratio*

LRA *Leak Resistant Arithmetic*

MESD *Multiple Exponent Single Data*

NICV *Normalized Inter-Class Variance*

NIST *National Institute of Standard and Technology*

PCA *Principal Component Analysis*

PCDC *Perin Collision Detection Criteria*

POC *Phase-Only Correlation*

PoI *Point of Interest*

PSD *Power Spectral Density*

R2L *Right-To-Left*

ROSETTA *Recovery of Secret Exponent by Triangular Trace Analysis*

RSA *Rivest Shamir Adleman*

SEMA *Simple Electromagnetic Analysis*

SEMD *Single Exponent Multiple Data*

SMA *Square-and-Multiply-Always*

SPA *Simple Power Analysis*

SPN *Substitution and Permutation Network*

SNR *Signal-to-Noise Ratio*

SR *Success Rate*

TA *Timing Attack*

ZEMD *Zero Exponent Multiple Data*

Introduction générale

De nos jours, les cartes à puces sont des objets incontournables. Elles sont notamment utilisées pour stocker des données confidentielles (santé, bancaire, identité, etc). Le caractère confidentiel des données enregistrées sur les cartes à puces en font des cibles de choix pour les attaquants et pirates de tous bords.

La cryptographie propose des mécanismes pour assurer la confidentialité, l'authenticité et l'intégrité des données. Elle est divisée en deux grandes familles : la cryptographie symétrique et la cryptographie asymétrique.

La cryptographie moderne a été marquée par la proposition d'une méthode de chiffrement de données basée sur un schéma de Feistel : le *Data Encryption Standard* (DES) [11]. Cet algorithme a été remplacé quelques années plus tard par un nouveau standard plus robuste basé sur une structure algébrique, plus précisément sur un réseau de substitution-permutation : l'*Advanced Encryption Standard* (AES) [12]. Ces deux méthodes de chiffrement font partie des cryptosystèmes symétriques car deux parties désirant communiquer doivent partager un même secret, appelé clé secrète, pour chiffrer et déchiffrer des données.

Ce type de cryptosystème est très efficace en termes de performance. Cependant, chaque utilisateur doit disposer d'une clé secrète pour chaque personne avec qui il souhaite communiquer. Dans la pratique la gestion de ces clés peut être problématique lorsque plusieurs utilisateurs veulent communiquer les uns avec les autres.

C'est en 1976 que W. Diffie et M. Hellman [13] ont proposé une solution à cette problématique en introduisant le concept de cryptographie asymétrique ou cryptographie à clé publique. Cependant, il a fallu attendre 1978 et l'invention du *Rivest Shamir Adleman* (RSA) [14] pour voir émerger cette nouvelle famille d'algorithmes. Les algorithmes à clé publique utilisent deux clés différentes, l'une pour chiffrer, l'autre pour déchiffrer et sont basés sur des problèmes mathématiques réputés difficiles à résoudre comme la factorisation pour le RSA et le logarithme discret pour El-Gamal [15] par exemple.

Pour chiffrer un message que l'on souhaite envoyer à un correspondant, il suffit de connaître sa clé publique qui sert à chiffrer le message. Dès lors, le message est indéchiffrable même avec la connaissance de cette clé publique. Lorsque le correspondant reçoit un message chiffré, il utilise sa clé privée, connue de lui seul, afin de le déchiffrer.

La cryptographie à clé publique souffre d'un défaut majeur : il est difficile d'être sûr que l'on utilise bien la clé publique de notre correspondant et non celle d'une personne mal intentionnée. D'autre part, ces algorithmes sont plutôt lents et leur sécurité nécessite l'utilisation de clés de taille très importante. Ainsi, la recommandation de l'*Agence Nationale de la Sécurité des Systèmes d'Information*

(ANSSI) en 2014 [16] est d'utiliser des clés de taille supérieure ou égale à 2048 bits pour l'algorithme RSA pour une sécurité jusqu'en 2030.

L'application la plus courante des algorithmes asymétriques est ce que l'on appelle le protocole d'échange de clé. Il permet à deux interlocuteurs de définir une clé commune et de pouvoir utiliser un algorithme de chiffrement à clé privée pour échanger plus facilement des données.

Les cartes à puces permettent de garantir différentes propriétés cryptographiques telles que la confidentialité des données. Paradoxalement, l'utilisation d'algorithmes de chiffrement rend les composants embarqués sensibles à de nombreuses attaques, s'ils ne sont pas implémentés avec une grande prudence. Parmi ces attaques, une famille d'attaques appelée attaques par canaux auxiliaires (ou attaques par canaux cachés), exploite les grandeurs physiques (consommation de courant, émission électromagnétique, etc) observables lors de l'exécution d'un algorithme cryptographique par le circuit intégré afin d'en extraire de l'information sur le secret manipulé.

Ces attaques ont été introduites comme une menace pour les cartes à puces pour la première fois en 1996 par P. Kocher [17]. Dès lors, de nombreux travaux dans ce domaine ont fait leur apparition dans la littérature. On assiste alors à l'apparition des attaques verticales utilisant plusieurs traces d'exécution [18, 19, 20, 21, 22]. Ces attaques ont ainsi été beaucoup étudiées au début du siècle et n'ont cessé d'évoluer et de gagner en efficacité en passant de plusieurs milliers (voire millions) de traces à peu de traces avec les premiers exemples d'attaques par collision [1, 2, 8, 23].

Cette évolution a eu un impact très important sur l'industrie de la carte à puce qui doit garantir la sécurité des produits tout en maintenant leur efficacité en termes de temps d'exécution et de consommation tant mémoire qu'électrique. Il est donc important de vérifier la résistance d'une carte à puce face à ces attaques. Il est aussi important d'élaborer des solutions logicielles, matérielles, algorithmiques ou arithmétiques pour se prémunir de ces attaques.

Parallèlement, des efforts ont été consentis par la communauté industrielle, mais aussi la communauté académique en définissant des contremesures tant environnementales que mathématiques. Cependant, les circuits intégrés (*Integrated Circuit* (IC)s) restent théoriquement vulnérables à une sous-famille d'attaques par canaux cachés apparue récemment. Ces attaques, dites *attaques horizontales* [3, 4, 5, 24, 7], ne nécessitent qu'une seule mesure sur un canal auxiliaire d'une exécution unique de l'algorithme cible pour extraire l'information secrète et apparaissent donc particulièrement dangereuses. Vérifier l'efficacité des attaques horizontales dans la pratique est donc une priorité pour l'industrie mais aussi pour les laboratoires d'évaluation sécuritaire.

L' objectif de cette thèse est principalement d'étudier la mise en pratique des attaques horizontales et des attaques par collision sur l'exponentiation modulaire. L'objectif secondaire est la définition de possibles contremesures.

Durant ces trois années, j'ai pu découvrir le monde des attaques par canaux

cachés au sein de *STMicroelectronics*. Afin de mettre en pratique les attaques par collision et les attaques horizontales, j'ai commencé par développer un simulateur de fuite. Pour un jeu de données et un algorithme choisi il génère des données correspondant à un canal caché théorique selon un modèle de fuite choisi. En ajoutant à ces simulations "parfaites" du bruit, je me suis rapproché suffisamment de ce que représente un canal caché comme la consommation et j'ai pu développer et tester mes premières attaques. La seconde étape a consisté à appliquer ces mêmes attaques sur des composants modernes et sécurisés.

Dans cette thèse je vais m'appliquer à décrire concrètement les difficultés que j'ai rencontrées dans mes expérimentations pratiques (sur des composants modernes et sécurisés) des attaques horizontales ou par collision. Pour chaque problème posé j'étudierai à la fois les outils les plus pertinents trouvés dans la littérature afin de surmonter ces difficultés. Pour certains problèmes clés, souvent négligés par la littérature des canaux cachés, comme la resynchronisation, la réduction de bruit, les problèmes de seuil, je propose au travers de cette thèse des approches systématiques que je compare à l'état de l'art.

La structure de cette thèse est la suivante : le chapitre 1 décrit l'évolution des attaques par canaux cachés sur l'exponentiation modulaire et les contremesures classiques généralement déployées pour protéger les implémentations. Une description plus détaillée des attaques par collision et des attaques horizontales y est proposée. Le chapitre 2 présente le simulateur réalisé durant cette thèse, pour valider en amont les attaques avant leur application dans la pratique. Dans la deuxième partie de ce chapitre, une méthodologie pour effectuer les attaques par collisions en présence de composants réels est détaillée. Ainsi un critère de détection automatique de collisions est proposé. Le chapitre 3 présente une méthode d'estimation du rapport signal à bruit ou *Signal-to-Noise Ratio* (SNR) et donc de la qualité des mesures collectées pour mener des attaques par canaux cachés et des applications associées. Cette méthode se base sur le critère présenté dans le chapitre 2. Le chapitre 4 présente une étude détaillée de chaque étape d'une attaque horizontale dans la pratique en utilisant les méthodologies présentées dans les chapitre 2 et 3. Finalement, différentes techniques pour se prémunir des attaques horizontales sont proposées.

Chapitre 1

Cryptanalyse par Canaux Cachés

Dans ce chapitre, un rapide rappel du système RSA, des algorithmes d'exponentiation et de multiplication modulaires est présenté. Un survol de certaines attaques par canaux cachés classiques et des contremesures notamment de masquage, y est effectuée. L'objet de cette thèse étant l'étude pratique des attaques par collision et des attaques horizontales, les principaux exemples de ces dernières y sont détaillés.

1.1 Introduction

Les algorithmes proposés en cryptographie moderne sont construits à partir de problèmes difficiles comme la factorisation du produit de deux grands nombres premiers, ou encore le problème du logarithme discret sur un groupe de cardinalité importante, etc.

Traditionnellement la cryptanalyse classique se focalise sur le problème sous-jacent à l'algorithme ou le protocole en termes mathématiques et détermine les instances ou les paramètres pour lesquels le problème est hors de portée des moyens présents ou même futurs. Mais un algorithme nécessite un support matériel pour être exécuté car l'être humain n'est pas en capacité de faire les calculs voulus sans assistance informatique. Afin de rendre pratiques les applications de la cryptographie, comme la signature numérique par exemple, elles sont implémentées dans des circuits intégrés.

Ces composants influencent l'environnement dans lequel ils opèrent. Par exemple, les composants intégrés, comme les cartes à puce, consomment du courant fourni par l'environnement et émettent des radiations électromagnétiques lors de leur fonctionnement en fonction des algorithmes et des données qu'elles traitent. De la même manière, ces composants peuvent être sensibles aux changements de température ou aux champs magnétiques dans lesquels ils se trouvent.

On appelle ces sources d'informations sur le composant, son activité et les données qu'il traite des canaux auxiliaires, ou canaux cachés. En effet, ils ne sont pas à proprement parler le canal de communication prévu par les concepteurs. Leur observation peut néanmoins donner de nombreuses informations sur les données traitées, notamment du fait qu'un adversaire peut avoir accès à des images des résultats de calculs intermédiaires.

Les canaux cachés les plus utilisés sont le temps d'exécution, la consommation électrique ainsi que le rayonnement électromagnétique. Afin d'exploiter ces canaux cachés plusieurs attaques ont été proposées. Par exemple dans [17], P. Kocher exploite la connaissance des temps d'exécution d'une exponentiation avec différents messages pour retrouver un exposant secret. Dans [18], P. Kocher et al. commencent par montrer comment exploiter le courant consommé par le composant pendant la construction des sous-clés d'un DES pour extraire la clé de tour (cette attaque est appelée attaque en puissance simple ou *Simple Power Analysis* (SPA)). Dans cet article fondateur des attaques dites "différentielles" ou (*Differential Power Analysis* (DPA)) ont été proposées. En 2001, dans [25] K. Gandolfi et al. proposent l'exploitation des champs électromagnétiques comme canal caché. Ces attaques sont alors appelées attaques électromagnétiques simples (*Simple Electromagnetic Analysis* (SEMA)) ou différentielles (*Differential Electromagnetic Analysis* (DEMA)).

Ce chapitre est organisé comme suit. La section 1.2 propose une étude des algorithmes d'exponentiation modulaire, qui constituent les principales opérations du système RSA. La section 1.3 dresse un état de l'art des attaques par canaux cachés sur l'exponentiation modulaire. Une attention plus particulière est apportée

aux attaques par collision et aux attaques horizontales, qui constituent le cœur de cette thèse en section 1.4 et section 1.5 respectivement.

1.2 Pré-requis

L'exponentiation joue un rôle essentiel dans le système RSA comme dans de nombreux systèmes à clé publique (*Digital Signature Algorithm* (DSA), El Gamal, etc). Nous détaillons dans cette section certaines méthodes d'exponentiation modulaires car leur compréhension est importante dans le cadre des attaques horizontales et des attaques par collision. Nous nous intéressons également dans cette section à la brique de base que constitue la multiplication modulaire. Cette dernière est en effet un élément important de l'implémentation en rapport avec les fuites exploitées par les attaques.

1.2.1 Présentation du système RSA

Le système RSA a été inventé en 1978 par R. Rivest, A. Shamir et L. Adleman [14]. C'est la première instance connue de méthodes de cryptographie à clé publique. Le système RSA est largement étudié depuis des années, que ce soit d'un point de vue théorique [26, 27, 28, 29, 30] ou pratique [17, 31].

La sécurité théorique du système RSA repose sur la difficulté supposée du problème de la factorisation de grands nombres entiers. Afin de rendre difficile la factorisation du modulo RSA, on choisit des nombres de taille au moins égale à 2048 bits [16] voire même à 4096 bits.

La génération de clés RSA commence par la sélection de deux grands nombres premiers aléatoires p et q. Le modulo public N (ou modulo RSA) est alors défini comme étant le produit pq. La taille de N, au sens binaire, est noté n. On choisit ensuite l'exposant public e de telle sorte que le pgcd $(e, \phi(N)) = 1$. Ici $\phi(N)$ désigne l'indicatrice d'Euler du modulo N et est égale au nombre d'entiers compris entre 1 et N (inclus) et premiers avec N (autrement dit le nombre d'entiers inversibles modulo N). Elle est calculée par $\phi(N) = (p-1)(q-1)$.

Le couple (N, e) forme alors la partie publique de la clé. La partie privée d est quant à elle obtenue en calculant :

$$d = e^{-1} \mod \phi(N),$$

dans le groupe multiplicatif $(\mathbb{Z}/n\mathbb{Z}^*, \times)$. Ainsi, il est facile de calculer d lorsque p et q sont connus. En revanche, calculer d sans connaître p et q est un problème difficile.

Le chiffrement RSA d'un message m, $0 \leq m < N$, consiste simplement en une exponentiation modulaire avec l'exposant public e :

$$C = m^e \mod N.$$

L'opération de déchiffrement est réalisée de manière similaire mais, en utilisant l'exposant privé d :

$$\tilde{m} = C^d \bmod N.$$

Si aucune erreur n'intervient pendant le chiffrement, la transmission du chiffré ou le déchiffrement, alors $\tilde{m} = m$.

Comme pour l'opération de déchiffrement, la signature RSA, S, d'un message m consiste en une exponentiation modulaire avec l'exposant privé d :

$$S = m^d \bmod N.$$

La vérification est faite en utilisant la clé publique du signataire :

$$m \stackrel{?}{=} S^e \bmod N.$$

Dans la pratique, on préfère signer une empreinte du message m plutôt que de signer directement sa valeur. Pour cela, on utilise une fonction de hachage pour calculer l'empreinte du document à signer.

1.2.2 Exponentiation modulaire

L'exponentiation modulaire est une opération de base pour l'utilisation de nombreuses méthodes de cryptographie asymétrique. En effet, que ce soit pour réaliser une signature DSA ou un chiffrement RSA, il faut pouvoir calculer une puissance modulaire du message d'entrée. Si cette opération est réalisée naïvement par la formule suivante :

$$m^d \bmod N = (\underbrace{m \cdot m \cdots m}_{d\ fois}) \bmod N, \tag{1.1}$$

alors la complexité calculatoire croît exponentiellement avec la taille de l'exposant ce qui n'est pas acceptable pour une implémentation efficace de l'exponentiation modulaire. En tenant compte des tailles des données en relation directe avec la sécurité de ces protocoles cryptographiques, une méthode bien plus efficace consiste à réaliser l'exponentiation modulaire en parcourant les bits d'exposant :

$$m^d \bmod N = \prod_{i=0}^{n-1} m^{2^i d_i} \bmod N = \left(\left(\left(\left(\ldots\left(m^{d_{n-1}}\right)^2 \times m^{d_{n-2}}\right)^2 \ldots \times m^{d_i}\right)^2 \ldots \times m^{d_1}\right)^2 \times m^{d_0}\right) \bmod N. \tag{1.2}$$

avec $d = \sum_{i=0}^{n-1} 2^i \cdot d_i$.

Ces méthodes d'exponentiation modulaires, appelées méthodes carré-multiplication ou (*Square-and-multiply*), ont une complexité qui croît linéairement avec la taille de l'exposant, ce qui est donc plus efficace. Les sections suivantes présentent deux méthodes de carré-multiplication appelées usuellement méthodes classiques ou naïves d'exponentiation modulaire binaire.

1.2.2.1 Méthodes d'exponentiation modulaire binaire classiques

Une façon simple de réaliser l'exponentiation modulaire binaire est de parcourir les bits d'exposant du poids fort vers le poids faible comme le montre l'équation 1.2. De ce fait, cette méthode est couramment appelée méthode d'exponentiation modulaire *"Gauche-Droite"* ou *Left-To-Right* (L2R). Comme le montre l'algorithme 1, l'exponentiation est réalisée en répétant des carrés modulaires, suivis, selon la valeur du bit courant d'exposant d_i, par des multiplications modulaires.

algorithme 1 Exponentiation binaire *Gauche-Droite* ou L2R

Entrées: $m, N, d = (d_{n-1}, ..., d_1, d_0), d_i \in \{0, 1\}$ pour tout $i \in \{0, \cdots, n-1\}$
Sorties: $S = m^d \bmod N$
1: $R_0 = 1$
2: **pour** $i = n-1$ **à** 0 **faire**
3: $\quad R_0 = R_0^2 \bmod N$
4: $\quad$ **si** $d_i = 1$ **alors**
5: $\quad\quad R_0 = R_0 \cdot m \bmod N$
6: $\quad$ **fin si**
7: **fin pour**
8: **retourner** R_0

Une deuxième méthode consiste à parcourir les bits de l'exposant en sens inverse, c'est-à-dire du poids faible vers le poids fort, soit de droite à gauche d'où le nom exponentiation binaire *"Droite-Gauche"* (ou *Right-To-Left* (R2L)). Elle se différencie de la première méthode par l'enchaînement des carrés et des multiplications modulaires ainsi que par l'utilisation d'une variable supplémentaire R_1 qui accumule les puissances carrées du message calculées à chaque itération.

algorithme 2 Exponentiation binaire *Droite-Gauche* ou R2L

Entrées: $m, N, d = (d_{n-1}, ..., d_1, d_0), d_i \in \{0, 1\}$ pour tout $i \in \{0, \cdots, n-1\}$
Sorties: $S = m^d \bmod N$
1: $R_0 = 1$
2: $R_1 = m$
3: **pour** $i = 0$ **à** $n-1$ **faire**
4: $\quad$ **si** $d_i = 1$ **alors**
5: $\quad\quad R_0 = R_0 \cdot R_1 \bmod N$
6: $\quad$ **fin si**
7: $\quad R_1 = R_1^2 \bmod N$
8: **fin pour**
9: **retourner** R_0

L'algorithme 1 permet d'économiser une variable d'accumulation (i.e. la variable R_1 cf. Algorithme 2) par rapport à l'algorithme R2L.

Les algorithmes 1 et 2 permettent de calculer efficacement l'exponentiation

modulaire, mais souffrent du point de vue des attaques par canaux cachés, de la dépendance entre les opérations effectuées et le secret manipulé. En effet, dans ces algorithmes, l'exécution de la multiplication ou non à l'étape 5 est conditionnée par la valeur du bit d'exposant. Dans le cadre d'une implantation sur un composant, cette dépendance, comme détaillé dans la section 1.3, peut permettre à un attaquant potentiel de déduire la valeur de l'exposant en analysant le comportement physique de la puce.

Au cours des dernières années, d'autres types d'algorithmes d'exponentiation plus résistants ont été proposés dans la littérature. Ces algorithmes sont détaillés plus tard dans le document en fonction de leur contexte d'apparition.

1.2.3 Algorithmes de multiplication modulaire

Tous les algorithmes d'exponentiation présentés précédemment effectuent une séquence de multiplications modulaires. Une multiplication modulaire $x \cdot y \bmod N$ comporte généralement deux étapes. D'abord, la multiplication régulière $x \cdot y$ est effectuée, puis une réduction modulo N est réalisée. Dans la pratique ces deux étapes sont entrelacées afin d'augmenter les performances de l'algorithme tant du point de vue calculatoire que de la mémoire nécessaire.

La réduction modulo N est le reste de la division euclidienne du résultat du produit par N. La difficulté des problèmes sous-jacents impliquant de travailler sur des nombres de taille importante, des contraintes d'efficacité pèsent donc sur cette opération, en particulier dans le contexte des systèmes embarqués.

Plusieurs méthodes similaires ont été mises au point afin de limiter le coût relatif à la réduction modulaire. Parmi celles-ci, on trouve la réduction de Montgomery [32], celle de Barrett [33], ou encore de Quisquater [34]. Une étude détaillée de ces techniques peut être trouvée dans [35]. Dans cette thèse, parmi ces méthodes seule la méthode de Montgomery [32] est décrite. Avant d'aborder la multiplication de Montgomery, la multiplication classique de grands nombres ou *Long Integer Multiplication* (LIM) [36], considérée dans certaines attaques décrites dans cette thèse [4, 5], est d'abord détaillée.

1.2.3.1 Multiplication d'entiers longs (LIM)

La multiplication de longs entiers implique généralement une multiplication multi-précision mise en œuvre selon la méthode scolaire. Cette méthode est décrite dans l'algorithme 3.

Soient $x = (x_{n-1}, \cdots, x_1, x_0)_b$ et $y = (y_{n-1}, \cdots, y_1, y_0)_b$ les décompositions respectives de x et de y en base $b = 2^t$. L'algorithme 3 effectue plusieurs petites multiplications sur des entiers de t-bits $(x_i \cdot y_j)$ et donne un résultat w sur $2n$-bits. Pour effectuer le calcul $w \bmod N$, l'algorithme LIM peut être combiné avec la méthode de réduction de Montgomery pour la réduction modulo N.

algorithme 3 Multiplication de grands nombres entiers

Entrées: $x = (x_{n-1}, \cdots, x_1, x_0)_b$, $y = (y_{n-1}, \cdots, y_1, y_0)_b$
Sorties: $w = x \cdot y$

pour $i = 0$ **à** $2n - 1$ **faire**
  $w_i = 0$
fin pour
pour $i = 0$ **à** $n - 1$ **faire**
  $c \leftarrow 0$
  pour $j = 0$ **à** $n - 1$ **faire**
    $(uv)_b \leftarrow (w_{i+j} + x_i \cdot y_j) + c$
    $w_{i+j} \leftarrow v$
    $c \leftarrow u$
  fin pour
  $w_{i+n} \leftarrow c$
fin pour
retourner w

1.2.3.2 La méthode de multiplication de Montgomery (MMM)

En 1985, P. Montgomery a introduit une méthode efficace pour la multiplication modulaire dans [32]. L'algorithme de la multiplication modulaire de Montgomery transforme la réduction modulo N en une division par une puissance de la base $R = 2^n$. Ce qui revient dans la pratique à effectuer une division en une puissance de 2 qui peut être implémentée par des décalages à droite.

Soit N un un entier positif de n-bits, et soient R et T des entiers tels que $R > N$, pgcd $(R, N) = 1$ et $0 \leq T < NR$. Le calcul de TR^{-1} mod N sans diviser par N est appelé réduction de Montgomery de T modulo N par rapport à R. La réduction de Montgomery peut être calculée efficacement en utilisant l'algorithme 4.

Supposons x et y deux entiers vérifiant $0 \leq x, y < N$. Soient $\widetilde{x} = xR$ mod N et $\widetilde{y} = yR$ mod N, les représentations respectives dans le domaine Montgomery de x et y. La réduction de Montgomery de $\widetilde{x}\,\widetilde{y}$ est $\widetilde{x}\,\widetilde{y}R^{-1}$ mod $N = xyR$ mod N. Le résultat xyR mod N est aussi appelée multiplication de Montgomery.

L'algorithme 5 présente une version multi-précision de la multiplication de Montgomery.

Après la description des algorithmes d'exponentiation modulaire binaire classiques, la section suivante détaille l'évolution des attaques par canaux cachés en fonction de l'évolution des algorithmes d'exponentiation modulaires.

algorithme 4 Réduction de Montgomery

Entrées: $N, R = 2^n, T, N' = N^{-1} \bmod N$
Sorties: $T \cdot R^{-1} \bmod N$
1: $Q = T \cdot N' \bmod R$
2: $S = (T + Q \cdot N)/R \bmod R$
3: **si** $S > N$ **alors**
4: $\quad S = S - N$
5: **fin si**
6: **retourner** S

algorithme 5 Multiplication de Montgomery

Entrées: N, $x = (x_{n-1}, \cdots, x_1, x_0)_b$, $y = (y_{n-1}, \cdots, y_1, y_0)_b$, avec $0 \leq x, y < N$, $R = b^n$ avec $\text{pgcd}(N, b) = 1$ et $N' = N^{-1} \bmod b$
Sorties: $A = xyR^{-1} \bmod N$
1: $A = 0$ $(A = (a_{n-1}, \cdots, a_1, a_0)_b)$
2: **pour** $i = 0$ to $n - 1$ **faire**
3: $\quad q_i = (a_0 + x_i y_0) \bmod b$
4: $\quad A = (A + x_i y + q_i N)/b$
5: **fin pour**
6: **si** $A \geq N$ **alors**
7: $\quad A = A - N$
8: **fin si**
9: **retourner** A

1.3 Attaques par canaux cachés classiques

Cette section donne un aperçu des principales attaques par canaux auxiliaires ciblant les différentes variantes des algorithmes d'exponentiation modulaires.

1.3.1 Attaques Temporelles

Comme évoqué précédemment, le temps d'exécution d'une opération telle qu'une exponentiation modulaire est suffisamment important pour être mesurable au travers du temps de réponse d'une commande envoyée à un composant et qui effectue cette opération.

P. Kocher en 1996 dans [17], pour la première fois, montre qu'un attaquant peut exploiter ce temps et obtenir la clé secrète (l'exposant) utilisée par le composant, notamment sur les algorithmes d'exponentiations modulaires binaires classiques. Comme abordé dans la section 1.2.2.1, ces algorithmes ont des temps de calcul dépendants des données d'entrée et de la clé secrète. Par exemple, dans l'algorithme L2R (algorithme 1), la boucle d'itération requiert plus ou moins d'opérations en fonction de l'exposant. En effet, si le bit de l'exposant manipulé vaut 0, seule l'opération carré est effectuée, alors que si le bit vaut 1, une opération supplémentaire est effectuée. Ainsi, le temps d'exécution dépend du poids de Hamming de l'exposant. Par conséquent, un attaquant peut retrouver pas à pas tous les bits de l'exposant en vérifiant si l'étape 5 de l'algorithme 1 est effectuée ou non.

Quelques années plus tard, J.-F. Dhem et al. dans [37] observent que dans une implémentation de la multiplication de Montgomery (algorithme 5), une opération de soustraction par le modulo est susceptible d'être calculée. Le temps additionnel de calcul de cette possible soustraction peut donc être utilisé par un attaquant pour retrouver des bits de l'exposant secret utilisé dans une signature RSA par exemple.

Toutefois, avec l'évolution des contremesures (algorithmes réguliers (section 1.3.3), masquage (section 1.4.5 etc), la majorité des ICs sont aujourd'hui robustes aux attaques temporelles. Cependant, combinées avec une attaque DPA par exemple, les attaques temporelles restent une menace concrète pesant sur les systèmes sécurisés comme le montre W. Schindler dans [38].

1.3.2 Attaque simple par analyse du courant (ou par analyse du rayonnement électromagnétique)

Pour mesurer le temps d'exécution de divers calculs on peut se baser sur le profil de consommation de courant du composant ou celui des radiations électromagnétiques. Ce profil peut révéler des éléments en relation plus directe avec le secret, pas seulement le poids de Hamming de l'exposant, mais également de manière plus microscopique le traitement au sein de l'algorithme du bit d'exposant. On appelle attaque par simple analyse du courant (ou SPA), une attaque consistant à effectuer une analyse directe de ces profils lors d'une seule exécution d'un algorithme cryptographique.

P. Kocher et al. montrent dans [18] comment exploiter le courant consommé par le composant pendant la construction des sous-clés d'un DES pour extraire la clé secrète et démontrent que ce type d'attaque peut être utilisée contre des algorithmes d'exponentiation modulaire. Dans la section 2.3.1, nous allons montrer que ce type d'attaque peut facilement être appliqué aux algorithmes L2R et R2L.

Il a ensuite été montré que l'analyse du rayonnement électromagnétique émis par le composant peut également être exploité de façon similaire [25, 39]. Lorsque le rayonnement électromagnétique est exploité au lieu de la consommation de courant, l'attaque est alors désignée par SEMA.

Toutefois, la réussite d'une SPA (ou d'une SEMA) peut nécessiter la connaissance de l'algorithme cryptographique considéré et surtout de la manière dont il est implémenté.

Pour lutter contre les attaques SPA et les attaques temporelles de nouveaux algorithmes ont été proposés dans la littérature. Contrairement aux algorithmes irréguliers binaires (algorithme L2R ou R2L), ces derniers se comportent de la même manière quelle que soit la valeur du secret en cours de traitement. Ils sont donc qualifiés d'algorithmes réguliers.

1.3.3 Algorithmes réguliers

Une solution naturelle pour lutter contre les attaques de type SPA, SEMA ou temporelle, consiste à rendre l'exécution des opérations dans un algorithme d'exponentiation indépendante de la valeur de l'exposant. Pour cela, plusieurs solutions ont été proposées dans la littérature.

Une première méthode consiste à ajouter une fausse multiplication dans le cas où la valeur du bit de l'exposant est égale à 0. Cette méthode proposée par J.-S. Coron en 1999 [40] est connue sous le nom Carré-et-Multiplication-Systématique ou *Square-and-Multiply-Always* (SMA).

L'algorithme 6 détaille cette variante dans le cas du parcours de l'exposant de gauche à droite. Dans cet algorithme le test sur la valeur est remplacé par une affectation du produit $R_0 \cdot m \bmod n$ dans $R_{\neg d_i}$. Ainsi, quelle que soit la valeur du bit de l'exposant, les mêmes opérations sont effectuées (carré suivi d'une multiplication). Le résultat de la multiplication fictive est mis dans un registre R_1. La valeur contenue dans R_1 est réutilisée uniquement lorsque bit traité vaut 0 et R_1 n'influent donc pas sur le résultat final de l'exponentiation.

Cependant, une attaque par injection de fautes appelée *Safe-error* [41] sans conséquence sur les calculs, est possible. Étant donné que cet algorithme ajoute une multiplication inutile dans le cas où le bit de l'exposant vaut 0, un attaquant peut injecter une faute dans cette multiplication et obtenir à la fin de l'exponentiation le résultat correct. Il en déduit donc la valeur du bit de l'exposant. Cette attaque est difficile à contrer par les moyens habituels comme la vérification du calcul à la fin de l'exponentiation.

Une autre méthode permettant d'améliorer la sécurité des méthodes classiques d'exponentiation, a été proposée par M. Joye et al. en 2002 dans [42]. Elle est

algorithme 6 Exponentiation binaire carré-et-multiplication systématique Gauche-Droite

Entrées: $m, N, d = (d_{n-1}, \cdots, d_1, d_0),\ d_i \in \{0,1\}$ pour tout $i \in \{0, \cdots, n-1\}$
Sorties: $S = m^d \bmod N$

1: $R_0 = 1$
2: $R_1 = 1$
3: **pour** $i = n-1$ **à** 0 **faire**
4: $\quad R_0 = R_0 \cdot R_0 \bmod N$
5: $\quad R_{\neg d_i} = R_0 \cdot m \bmod N$
6: **fin pour**
7: **retourner** R_0

connue sous le nom d'échelle de Montgomery (*Montgomery Powering Ladder* en anglais). Cette variante introduit des calculs non-redondants dans le but de se protéger contre la SPA et cette implémentation est aussi protégée des attaques de type *Safe-error* [41] qui s'applique sur la variante SMA.

L'algorithme 7 présente l'échelle de Montgomery. Il consiste à effectuer les mêmes opérations (multiplication suivie d'un carré) quelle que soit la valeur du bit de l'exposant. Contrairement à la solution précédente, le contenu des deux variables R_0 et R_1 dans l'échelle de Montgomery influe sur le résultat.

La sécurité de l'échelle de Montgomery repose sur la supposition qu'un adversaire ne peut pas distinguer une écriture dans R_0 d'une écriture dans R_1.

algorithme 7 Échelle de Montgomery

Entrées: $m, N, d = (d_{n-1}, ..., d_1, d_0),\ d_i \in \{0,1\}$ pour tout $i \in \{0, \cdots, n-1\}$
Sorties: $S = m^d \bmod N$

1: $R_0 = 1$
2: $R_1 = m$
3: **pour** $i = n-1$ **à** 0 **faire**
4: $\quad R_{\neg d_i} = R_{d_i} \cdot R_{\neg d_i} \bmod N$
5: $\quad R_{d_i} = R_{d_i} \cdot R_{d_i} \bmod N$
6: **fin pour**
7: **retourner** R_0

Une autre direction a été proposée B. Chevallier-Mames et al. dans [43]. Elle consiste à transformer la boucle non régulière (dans le sens où la boucle comporte un test) constituée par les étapes 3 à 5 de l'algorithme 1, en une suite régulière de multiplications, sans utiliser de multiplications factices, pour un gain de temps dans l'exécution de l'algorithme. L'algorithme 8 ainsi obtenu est parfaitement régulier en ce sens qu'il ne comporte que des multiplications et que chaque itération de la boucle principale ne comprend qu'une seule multiplication.

La sécurité de l'algorithme 8 repose sur la supposition selon laquelle les carrés sont indistinguables des multiplications.

Cependant, F. Amiel et al. ont montré dans [44] qu'il est toujours possible de distinguer, à partir de traces moyennées de canaux cachés, une multiplication $x \cdot y$ avec $x = y$ d'une multiplication avec $x \neq y$.

algorithme 8 Exponentiation binaire Atomique Gauche-Droite

Entrées: $m, N, d = (d_{n-1}, ..., d_1, d_0),\ d_i \in \{0, 1\}$ pour tout $i \in \{0, \cdots, n-1\}$
Sorties: $S = m^d \bmod N$

1: $R_0 = 1$
2: $R_1 = m$
3: $i = n - 1$
4: $k = 0$
5: **tant que** $i >= 0$ **faire**
6: $\quad R_0 = R_0 \cdot R_k \bmod N$
7: $\quad k = k \oplus d_i$
8: $\quad i = i - 1 + k$
9: **fin tant que**
10: **retourner** R_0

Les algorithmes réguliers permettent de se protéger contre les attaques SPA ou SEMA. Cependant ils restent vulnérables à certaines attaques comme cela est expliqué dans la section 1.3.4.

1.3.4 Les attaques statistiques ou attaques verticales

Les attaques SPA ou SEMA s'intéressent à des relations entre données traitées et canal caché, observables directement sur la courbe du canal caché considéré avec peu de traitements. Généralement, les détails de l'implémentation du composant attaqué sont requis pour mener une attaque SPA réussie.

D'autres attaques plus performantes (dans le sens où une parfaite connaissance de l'implémentation n'est pas nécessaire) existent. Ces attaques nécessitent l'observation de fuites d'informations mesurées lors de plusieurs exécutions utilisant le même secret. Elles sont appelées attaques statistiques ou attaques verticales car les mêmes mesures y sont interprétées à l'aide d'outils statistiques et par échantillon temporel.

Les attaques statistiques se déroulent en trois étapes distinctes. Une première étape, qui requiert l'accès au produit testé (ou *Device Under Test* (DUT)) a pour objet la collecte des mesures lors du déroulement de l'algorithme cible utilisant le secret. Une seconde étape, pouvant être faite en parallèle de la précédente, consiste à construire pour chaque sous-clé un modèle de consommation (cf. section 1.3.4.1) représentant la fuite théorique de la variable intermédiaire visée dans l'algorithme. Cette variable dépend de la clé secrète. Enfin, une dernière étape consiste à utiliser un ou plusieurs outils statistiques afin de tester la validité de chacune des hypothèses de clé.

Dans ce qui suit, quelques modèles de fuite sont détaillés dans la section 1.3.4.1, avant d'aborder les tests statistiques dans la section 1.3.4.2.

1.3.4.1 Modèles de Fuite

La plupart des circuits intégrés sont basés sur la technologie *Complementary Metal Oxyde Semiconductor* (CMOS). Cette technologie possède des propriétés de consommation en courant qui sont exploitées par les attaques. En effet, celle-ci varie en grande partie en fonction de la valeur des données traitées par le composant à un instant donné.

Pour savoir quelle donnée est traitée par un circuit à partir de traces récupérées lors de son fonctionnement, plusieurs modélisations ont été proposées mais deux modèles restent très largement utilisés en pratique. Il s'agit des modèles de distance et de poids de Hamming.

Dans [19], T. Messerges et al. observent une relation linéaire entre le poids de Hamming d'un registre interne à un instant t et la consommation électrique $M(t)$ du composant à ce même instant. Cette observation conduit au modèle :

$$M(t) = a \cdot HW(\text{Valeur}) + b.$$

où a est une constante liée au composant utilisé, $HW(\text{Valeur}(t))$ est le poids de Hamming de l'état actuel Valeur(t) d'un registre interne et b est une constante intégrant un bruit gaussien provenant de sources diverses. Ce modèle, appelé modèle poids de Hamming, a été étendu en considérant la distance de Hamming entre deux états consécutifs du composant par E. Brier al. dans [21]. Ainsi, $M(t)$ est modélisé par :

$$M(t) = a \cdot HW\left(\text{Valeur}(t) \oplus \text{Valeur}(t-1)\right) + b,$$

où $HW(\text{Valeur}(t) \oplus \text{Valeur}(t-1))$ est la distance de Hamming entre Valeur(t) et son état précédent Valeur($t-1$).

D'autres types de modélisation existent, comme celle proposée par B. Gierlichs et al. dans [45] où les auteurs considèrent directement la valeur de la donnée.

1.3.4.2 Méthodes Statistiques

De nombreux travaux ont été réalisés afin d'évaluer l'efficacité de différents tests statistiques désignés par le terme distingueurs (car permettant de distinguer des populations) dans le contexte d'attaques par canaux cachés. On présente ici une liste non exhaustive des principales propositions.

1.3.4.2.1 Notations

On notera dans cette section :

- $g : \mathcal{X} \times \mathcal{K} \to \mathcal{Z}$ la fonction intermédiaire ciblée par l'attaque,

- $\mathcal{K} = \{0, \cdots, \mathbf{k} - 1\}$ l'ensemble des valeurs possibles de la clé et $k^\star$ la valeur de la clé secrète,
- $\mathcal{X} = \{0, \cdots, \mathbf{x} - 1\}$ l'ensemble des entrées possibles de g,
- $\mathcal{Z} = \{0, \cdots, \mathbf{z} - 1\}$ l'ensemble des résultats possibles pour g.
- $L(Z) = L(g(X, k)) = \Phi(X, k)$, la variable aléatoire représentant la fuite générée par le calcul de Z

La fonction intermédiaire g connue par l'attaquant, est déterministe et dépend de la clé $k \in \mathcal{K}$.

Le but d'une attaque par observation est d'estimer la clé secrète $k^\star \in \mathcal{K}$. On dispose pour cela de n réalisations physiques notées $l_i = \phi(x_i, k^\star)$, $i = 1, \cdots, n$ de $L(Z) = \Phi(X, k^\star)$. On considère que ces réalisations proviennent de variables aléatoires indépendantes. Ces réalisations sont obtenues à partir de mesures physiques bruitées. D'autre part, on modélise par la fonction $m(X, k)$, l'information pour une clé $k \in \mathcal{K}$. On notera $\mathcal{M}$ l'ensemble des valeurs possibles du modèle.

On dispose de n réalisations $m(x_i, k)$ pour différentes clés à tester dans un ensemble $\mathcal{K}$. À partir de ces données, l'attaquant va chercher à quantifier l'intensité de la *dépendance* à l'aide d'un distingueur entre les variables $L(Z)$ et $m(X, k)$. Les paragraphes suivants rappellent les distingueurs les plus couramment utilisés.

1.3.4.2.2 Différence de moyennes

L'attaque originelle proposée par P. Kocher et al. dans [18] est basée sur la différence de moyennes. P. Kocher et al. y proposent d'attaquer la valeur d'un bit de la sortie d'une boite-S du DES lors de la première ronde. Ainsi, les variables aléatoires $m(X, k)$, $k \in \mathcal{K}$, n'ont ici que deux valeurs possibles, 0 et 1. L'attaquant estime donc la différence des moyennes des traces telles que $m(X, k)$ vaut 0 et telles que $m(X, k)$ vaut 1. Pour cela, il forme deux partitions :

$$\mathcal{T}_0 = \{l_i : m(x_i, k) = 0\} \text{ et } \mathcal{T}_1 = \{l_i : m(x_i, k) = 1\}.$$

Ensuite, il calcule la différence des moyennes $\hat{\Delta}_k$ entre les deux partitions $\mathcal{T}_0$ et $\mathcal{T}_1$ comme suit :

$$\hat{\Delta}_k = \frac{\sum_{l \in \mathcal{T}_0} l}{|\mathcal{T}_0|} - \frac{\sum_{l \in \mathcal{T}_1} l}{|\mathcal{T}_0|} = \overline{\mathcal{T}}_0 - \overline{\mathcal{T}}_1$$

où $\overline{\mathcal{T}}_i$, et $|\mathcal{T}_i|$ correspondent respectivement à la moyenne d'une partition $\mathcal{T}_i$ et à son cardinal.

Lorsque $m(X, k)$ et $L(Z)$ sont indépendantes, les deux moyennes sont presque identiques. En revanche, lorsque un ou plusieurs points de Δ_k sont élevés (en valeur absolue), il est probable qu'il existe une relation de dépendance entre les deux

variables qui sont considérées. Ainsi, une estimation de la bonne clé sera donnée par l'hypothèse qui réalise le plus grand maximum en valeur absolue :

$$\hat{k^\star} = \underset{k \in \mathcal{K}}{argmax}(|\hat{\Delta}_k|).$$

Dans la pratique la différence des moyennes peut nécessiter une définition de seuil pour la phase de décision. Ce problème peut être résolu en utilisant le test t de Welch [46] comme nous allons le voir dans la sous-section qui suit.

1.3.4.2.3 Le Test t de Welch

En statistiques, le test t de Welch [46] est une adaptation du test t de Student [47]. Il peut être utilisé notamment pour tester statistiquement l'hypothèse H_0 d'égalité des moyennes de deux échantillons de variances inégales. Il s'agit en fait d'une solution approchée du problème de Behrens-Fisher [48]. Le test t de Welch s'appuie sur le calcul de la statistique t :

$$t = \frac{\overline{\mathcal{T}}_0 - \overline{\mathcal{T}}_1}{\sqrt{\frac{\sigma_1^2}{N_1} + \frac{\sigma_2^2}{N_2}}} \tag{1.3}$$

où σ_i^2 et N_i correspondent respectivement à la variance d'une partition $\mathcal{T}_i$ et à son cardinal $|\mathcal{T}_i|$.

Contrairement au test t de Student, le dénominateur n'est pas basé sur une estimation de l'ensemble des variances. Le calcul des degrés de liberté ν associés à cette estimation de la variance est approché par l'équation de Welch-Satterthwaite :

$$\nu = \frac{\left(\frac{\sigma_1^2}{N_1} + \frac{\sigma_2^2}{N_2}\right)^2}{\frac{\sigma_1^4}{N_1^2 \cdot \nu_1} + \frac{\sigma_2^4}{N_2^2 \cdot \nu_2}} = \frac{\left(\frac{\sigma_1^2}{N_1} + \frac{\sigma_2^2}{N_2}\right)^2}{\frac{\sigma_1^4}{N_1^2 \cdot (N_1 - 1)} + \frac{\sigma_2^4}{N_2^2 \cdot (N_2 - 1)}}. \tag{1.4}$$

Ainsi $\nu_i = N_i - 1$, les degrés de liberté sont associés à la $n^{\text{ième}}$ estimation de la variance. Une fois t et ν calculés, ces statistiques peuvent être utilisées avec une distribution de Student pour tester l'hypothèse nulle qui stipule que les moyennes de deux populations sont égales (un test bilatéral), ou l'hypothèse nulle stipulant que la moyenne d'une population est plus grande ou égale à une autre (un test unilatéral). Lorsque le test est réalisé, celui-ci donne une probabilité ou p-valeur qui permet de rejeter ou non l'hypothèse nulle (H_0).

Mettre cette p-valeur à 10^{-3} détermine la valeur du seuil de la statistique t à 4.5 pour valider ou rejeter H_0. En d'autres termes, toutes les positions avec une valeur de t supérieure à 4.5 sont considérées comme des positions où les deux familles ont des moyennes différentes (avec une probabilité 0.001 de se tromper) ; ce qui permet de résoudre le problème de définition de seuil arbitraire avec la méthode de la différence des moyennes.

Le test de Welch est généralement utilisé dans la littérature des attaques par canaux cachés pour vérifier si un composant fuit ou non. Le test est souvent appliqué à une famille composée de courbes obtenues avec des entrées fixes (et une clé fixe) et une autre famille avec des entrées aléatoires (et une clé fixe). E. Oswald et al. [49] puis F.-Y Standaert et al. [50] ont utilisé ensuite le test de Welch pour identifier les points d'intérêt (ou *Point of Interest* (PoI)s) dans une trace (les points contenant l'information utile liée à la manipulation de l'information secrète dans une courbe).

D'autres techniques proposées visent à offrir un plus vaste choix pour les valeurs du modèle de fuite. Parmi celles-ci, on trouve le coefficient de corrélation de Pearson.

1.3.4.2.4 Coefficient de corrélation de Pearson

Pour analyser la *dépendance* linéaire entre deux variables aléatoires, la mesure utilisée est souvent le coefficient de corrélation linéaire ou corrélation de Pearson entre $m(x., k)$ et $l.$. Ce coefficient est défini par :

$$\rho_k = \frac{\frac{1}{n}\sum_{i=1}^{n}(l_i - \bar{l})(m(x_i,k) - \bar{m_k})}{\sqrt{\frac{1}{n}\sum_{i=1}^{n}(l_i - \bar{l})^2 \frac{1}{n}\sum_{i=1}^{n}(m(x_i,k) - \bar{m_k})^2}}. \tag{1.5}$$

où $\bar{l}$ est la moyenne associée aux l_i et $\bar{m_k}$ celle associée aux $m(x_i, k)$. $\rho_k = 1$ ou $\rho_k = -1$ indique une relation fonctionnelle linéaire parfaite entre le modèle considéré et la fuite. En effet, ρ_k mesure la proximité des n points à la droite de régression dont le signe de la pente est donné par le signe de ρ_k. La bonne clé est donc estimée comme étant la clé pour laquelle on a, en valeur absolue, le *coefficient de Pearson* le plus élevé :

$$\hat{k^\star} = \underset{k \in \mathcal{K}}{argmax}(|\rho_k|).$$

L'analyse par corrélation de la consommation (ou *Correlation Power Analysis* (CPA)) [21] est actuellement l'une des attaques par observation les plus connues. Cette popularité est principalement due au fait que les modèles de fuites linéaires utilisés sont représentatifs de la fuite. Elle est également due à sa faible complexité calculatoire requis.

Comme montré dans [51], l'utilisation de la CPA avec un modèle mono-bit revient à la DPA de P. Kocher et al.. La CPA est également très résistante au bruit présent dans les réalisations de la variable $L(Z)$. En effet, le bruit étant indépendant des données traitées, aura tendance à être moyenné par le calcul du coefficient de Pearson. Ainsi, plus le nombre de réalisations augmente, moins la contribution du bruit est importante. Néanmoins, afin de se prémunir cette attaque, les fabricants de cartes à puce essayent de mettre en œuvre des contremesures visant à casser cette dépendance linéaire.

1.3.5 Exemples d'attaques verticales sur l'exponentiation modulaire

En 1999, T. Messerges et al. ont proposé dans [19] différentes méthodes d'attaques verticales sur l'exponentiation modulaire. Ces méthodes furent les premiers exemples d'attaques verticales par canaux cachés applicables sur des cryptosystèmes comme le RSA. La plupart des attaques verticales sur le système RSA sont des cas particuliers de ces attaques. Dans ce qui suit sont détaillées ces attaques qui permettent de comprendre la plupart des attaques considérées dans la suite du document.

Notons qu'ici, nous reprenons les notations habituelles pour une exponentiation modulaire. Ainsi m désigne un message d'entrée, d et k des exposants, N le modulo de taille n.

1.3.5.1 Attaque d'un exposant avec de multiples données [19]

L'attaque *Single Exponent Multiple Data* (SEMD) [19] suppose que l'attaquant est capable d'exécuter plusieurs fois l'opération d'exponentiation m_i^k et d'enregistrer les courbes de consommation $\mathcal{T}_i^k$ correspondantes lors d'une phase d'apprentissage. Ici, k désigne l'exposant choisi par l'attaquant et les m_i (avec $i = 1, ..., l$) sont des messages aléatoires.

Une fois l'apprentissage terminé, l'attaquant cherche à retrouver le secret d, caché dans un composant du même type. Pour cela, il enregistre les courbes de consommation $\mathcal{T}_i^d$ pour les mêmes entrées m_i que précédemment avec l'exposant secret d cette fois-ci. L'attaquant calcule enfin la différence des moyennes des deux ensembles $\mathcal{T}_i^d$ et $\mathcal{T}_i^k$.

Si, à un instant t correspondant au traitement de k_i et d_i, la même opération est effectuée, alors la courbe de la différence des moyennes est à cet instant proche de zéro et on en déduit que $d_i = k_i$. Dans le cas contraire, la différence est non nulle et on en déduit que $d_i = 1 - k_i$. À la fin de cette étape de comparaison, l'attaquant a donc la succession des opérations effectuées avec l'exposant secret d pour un algorithme irrégulier et retrouve ainsi l'exposant d.

1.3.5.2 Attaque multiple exposants une donnée [19]

L'attaque multiple exposants une donnée ou *Multiple Exponent Single Data* (MESD) [19] suppose que l'attaquant est capable de réaliser des exponentiations m^k pour chaque valeur de k souhaitée, avec un m message fixe. Pour chaque exponentiation, il enregistre alors la courbe de consommation correspondante $\mathcal{T}^k$. Soit d l'exposant secret cherché par l'attaquant et $\mathcal{T}^d$ la courbe correspondant à l'exponentiation m^d. On suppose que l'adversaire connaît déjà les $(j-1)$ bits de poids forts de d. Il cherche donc le bit j du secret. Pour cela, il choisit un exposant k tel que les $(j-1)$ bits de poids forts correspondent à ceux déjà connus de d et il fixe le bit k_j à 0 et à 1. Il réalise ces deux exponentiations et enregistre les courbes de consommations respectives $\mathcal{T}^k_{k_j=0}$ et $\mathcal{T}^k_{k_j=1}$. Il calcule ensuite la différence entre $\mathcal{T}^d$

et $\mathcal{T}^k_{k_j=0}$ et puis celle entre $\mathcal{T}^d$ et $\mathcal{T}^k_{k_j=1}$.

Si la valeur du bit est correcte, les calculs intermédiaires à cette étape correspondent à ceux effectués lors du calcul avec le secret d. La différence entre les deux courbes sera donc proche de zéro à cet instant. Au contraire, si la valeur est erronée, on observe une différence plus importante. L'attaquant retrouve ainsi les bits de d au fur et à mesure.

1.3.5.3 Attaque zéro exposant multiple données [19]

Contrairement à la MESD, l'attaque zéro exposant multiple données ou *Zero Exponent Multiple Data* (ZEMD) [19] ne requiert pas d'exposant connu. La ZEMD est semblable à la MESD car c'est une attaque itérative qui suit les mêmes étapes mais qui est plus efficace. Cependant, elle requière la capacité à simuler les calculs effectués à l'intérieur du composant et donc de la connaissance de l'algorithme cible. En outre, l'attaquant doit être capable d'effectuer des exponentiations modulaires avec différents messages aléatoires m_i choisis mais ce uniquement avec l'exposant secret d (inconnu de l'attaquant).

On considère que l'attaquant connaît les $(j-1)$ bits de d et cherche le bit d_j qu'il fixe à 1 ou à 0. Il simule l'exécution de l'algorithme et dispose ainsi d'une valeur intermédiaire dépendante du bit d_j. Ensuite, il classe les courbes de consommation des m_i^d en deux ensembles suivant l'importance du poids de Hamming. Si la valeur du bit d_j est correcte, la différence entre les deux ensembles va produire des pics significatifs. Sinon, la différence va être proche de zéro. Comme dans le cas de la MESD, l'attaquant retrouve les bits de d au fur et à mesure. L'avantage de l'attaque ZEMD par rapport aux attaques SEMD et MESD est qu'elle peut être appliquée aux algorithmes réguliers.

Ces attaques ont été généralisées (et réalisées en pratique) par K. Itoh et al. dans [52]. Dans ce qui suit cette attaque est décrite.

1.3.5.4 Attaque sur les bits d'adressage de la mémoire [52]

L'attaque sur les bits d'adressage de la mémoire ou *Address-bit* DPA [52], contrairement aux attaques SEMD, MESD et ZEMD, n'est pas basée sur la valeur des données mais plutôt sur la valeur des registres accédés. Ainsi, les contremesures de masquage des données (section 1.4.5) n'ont pas d'effet. Si une même donnée est chargée dans deux registres différents, l'attaquant peut le distinguer dans le canal caché cible grâce au poids de Hamming des adresses des registres. L'attaque réalise des moyennes pour réduire l'influence des données sur les courbes. Elle fonctionne donc lorsqu'il y a une relation nette entre la clé et les adresses des registres accédés. L'attaque a été proposée en 2002 par K. Itoh et al. pour montrer qu'il est possible, dans l'algorithme 7, de distinguer le registre (R_0 ou R_1) utilisé et ainsi d'extraire l'exposant.

1.4 Attaques par collision

Les attaques par collision ou attaques par messages choisis sont apparues dans la littérature au début des années 2000. Ces attaques comparent deux segments de consommation (avec une ou plusieurs exécutions différentes) et utilisent le résultat pour déterminer si les valeurs mises en jeu dans le calcul sont les mêmes ou différentes. Par exemple, lorsqu'on effectue deux multiplications, un attaquant peut espérer que les segments de traces correspondent (on parle alors de collision) dans le canal auxiliaire observé quand les opérandes sont les mêmes, et sont différents autrement. Cette idée peut se traduire dans différentes variantes d'attaques par collision qui font l'objet de cette section.

Le principe de fonctionnement des attaques par collision est illustré par la figure 1.1. Dans cette figure, deux segments issus de deux traces avec des messages d'entrées différents sont comparés en utilisant la différence point à point. Si cette différence est proche du vecteur zéro, les segments correspondent au même calcul (collision), sinon ils correspondent à deux calculs différents (non-collision).

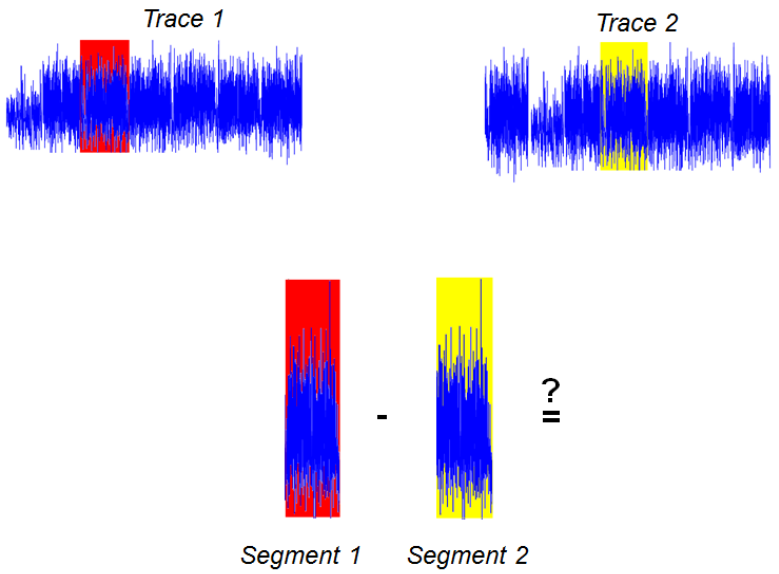


FIGURE 1.1 – Principe des attaques par collision

Les premiers articles sur ce type d'attaque concernaient la cryptographie symétrique. En 1998, H. Dobbertin dans [53] exploite les collisions internes sur les fonctions de hachages comme le MD4. Il sera imité quelques années plus tard par K. Schramm et al. dans [54]. Dans cet article, les auteurs montrent l'apparition de collisions sur le premier round du DES. En 2004, ces mêmes auteurs publient un article sur les collisions lors de l'exécution de l'AES.

Les premières publications d'*attaques par collision* sur la cryptographie asymétrique ont été introduites par P.-A. Fouque et F. Valette dans [1] et ciblaient

l'algorithme carré-et-multiplication systématique. S. Yen et al. ont ensuite proposé une variante de cette attaque en 2005 dans [2]. En 2006, ces mêmes auteurs proposent un autre type d'attaque par collision dans [8], qui cette fois fonctionne sur l'échelle de Montgomery.

Finalement, c'est à partir de 2008 que N. Homma et al. dans [23, 9] proposent une généralisation des attaques par collision, qui jusque-là ne visaient que des implémentations traitant l'exposant du bit de poids fort au bit de poids faible. Cette généralisation s'applique sur la plupart des implémentations d'exponentiation modulaire mais nécessite des messages choisis pour retrouver chaque bit de l'exposant.

Dans les parties qui suivent, sont détaillées différentes attaques par collision. Le fonctionnement de celles-ci sur les différents algorithmes présentés dans les sections 1.2.2.1 et 1.3.3, y est aussi expliqué.

1.4.1 L'attaque par doublement [1]

L'attaque par doublement ou *doubling attack* [1] a été proposée en 2003 par P.-A. Fouque et F. Valette. Ces derniers démontrent pourquoi les implémentations *Droite-Gauche* sont plus résistantes que les implémentations *Gauche-Droite*. Leur attaque est y est définie initialement sur l'équivalent de l'exponentiation carré-et-multiplication systématique (algorithme 6) pour les courbes elliptiques (*Double-and-Add-Always*), implémentation qui est résistante aux attaques SPA classiques. Ainsi bien que dans la suite les illustrations sont faites sur une exponentiation, cette attaque sera toujours appelée attaque par doublement dans le reste du document.

L'idée générale de cette attaque est de comparer la trace relative au calcul de m^d mod N et celle de $(m^2)^d$ mod N. À titre d'exemple, considérons $d = (1, 0, 0, 1, 1, 0, 1)$. En calculant m^d et $(m^2)^d$ avec l'algorithme 6, qui est supposé régulier, on obtient alors les différentes étapes représentées dans le tableau 1.1. En observant ce tableau, on remarque des résultats de calculs intermédiaires identiques. Plus précisément, on observe que l'opération d'élévation au carré (étape 3 de l'algorithme 6) fournit des résultats, au rang $i-1$ dans le calcul de m^d, identiques à ceux au rang i dans le calcul de $(m^2)^d$ si et seulement si $d_i = 0$.

En effet, soit le produit partiel $S_k(m) = \prod_{i=0}^{i=k} m^{2^{k-i} \cdot d_{n-1-i}}$. Cette valeur est égale à R_0 dans l'algorithme 6 après k itérations. Cette valeur $S_k(m)$ est une valeur intermédiaire de l'algorithme 6 car on a :

$$S_k(m) = \prod_{i=0}^{i=k} m^{2^{k-i} \cdot d_{n-1-i}} \tag{1.6}$$

$$= (\prod_{i=0}^{k-1} (m^2)^{2^{k-i-1} \cdot d_{n-1-i}}) m^{d_{n-1-k}} \tag{1.7}$$

$$= S_{k-1}(m^2) \cdot m^{d_{n-1-k}} \tag{1.8}$$

i	d_i	Calcul de m^d	Calcul de $(m^2)^d$
6	1	$R_0 = 1^2$ $R_0 = 1 \cdot m$	$R_0 = 1^2$ $R_0 = 1 \cdot m^2$
5	0	$R_0 = m^2$ $R_1 = m^2 \cdot m$	$R_0 = (m^2)^2$ $R_1 = m^4 \cdot m^2$
4	0	$R_0 = (m^2)^2$ $R_1 = m^4 \cdot m$	$R_0 = (m^4)^2$ $R_1 = m^8 \cdot m^2$
3	1	$R_0 = (m^4)^2$ $R_0 = m^8 \cdot m$	$R_0 = (m^8)^2$ $R_0 = m^{16} \cdot m^2$
2	1	$R_0 = (m^9)^2$ $R_0 = m^{18} \cdot m$	$R_0 = (m^{18})^2$ $R_0 = m^{36} \cdot m^2$
1	0	$R_0 = (m^{19})^2$ $R_1 = m^{38} \cdot m$	$R_0 = (m^{38})^2$ $R_1 = m^{76} \cdot m^2$
0	1	$R_0 = (m^{38})^2$ $R_0 = m^{76} \cdot m$ Retourner $R_0 = m^{77}$	$R_0 = (m^{76})^2$ $R_0 = m^{152} \cdot m^2$ Retourner $R_0 = m^{154}$

Tableau 1.1 – Valeurs intermédiaires de l'algorithme 6 avec les messages d'entrées m et m^2.

Ainsi le résultat intermédiaire de l'opération d'élévation au carré de l'algorithme 6 avec m à l'étape $k+1$ sera égal à la valeur intermédiaire de l'algorithme 6 avec m^2 à l'étape k si et seulement si d_{n-1-k} est égal à 0.

L'algorithme 9 synthétise les différentes étapes de l'attaque par doublement.

1.4.2 L'attaque de S. Yen et al. [2]

Afin d'attaquer une implémentation réalisée suivant l'exponentiation carré-et-multiplication systématique (algorithme 6), qui est résistante à la SPA, S. Yen et al. proposent d'utiliser des messages choisis particuliers comme $N-1$ où N désigne le modulo utilisé pour l'exponentiation.

En effet dans le cas, par exemple du système RSA, si N est le modulo RSA, on observe $(N-1)^2 = 1 \bmod N$. Cette observation peut se généraliser comme suit :

$$(N-1)^j = \begin{cases} 1 \mod N \text{ si } \exists k, j = 2k, \\ (N-1) \mod N \text{ si } \exists k, j = 2k+1. \end{cases}$$

En choisissant le message d'entrée $m = N-1$, l'algorithme 6 donne donc après l'étape 4 de l'itération i :

$$R_0 = (N-1)^{(d_{n-1}, \cdots, d_i)} \bmod N.$$

Si $R_0 = 1$, alors $(d_{n-1}, \cdots, d_i)$ est un entier pair et donc $d_i = 0$.
Sinon, $R_0 = (N-1)^{(d_{n-1}, \cdots, d_i)} = N-1 \mod N$ et alors $(d_{n-1}, \cdots, d_i)$ est un entier

algorithme 9 Attaque par Doublement [1]

Entrées: m, m^2
Sorties: $d = (d_{n-1}, \cdots, d_1, d_0)$
1: Collecter la trace $\mathcal{T}_1$ obtenue lors l'exponentiation m^d.
2: Collecter la trace $\mathcal{T}_2$ obtenue lors l'exponentiation $(m^2)^d$.
3: Diviser $\mathcal{T}_1$ et $\mathcal{T}_2$ respectivement en segments $\left\{T_1^1, T_2^1, \cdots, T_l^1\right\}$ et $\left\{T_1^2, T_2^2, \cdots, T_l^2\right\}$
4: **pour** chaque position i **faire**
5: Comparer les segments T_{i+1}^1 et T_i^2 en utilisant un distingueur
6: Si collision alors $d_i = 0$
7: Sinon $d_i = 1$
8: **fin pour**
9: **retourner** d

impair et donc $d_i = 1$. Ainsi à l'étape 3 de l'itération $i + 1$ on aura soit 1^2 si le bit précédent était égal à 0 ou $(N - 1)^2$ sinon.

Par conséquent, en observant une seule trace de consommation durant l'exécution de l'algorithme 6, il suffit à l'attaquant d'identifier les bouts de trace correspondant au calcul de 1^2 à la fin de chaque de chaque itération afin d'obtenir les valeurs d_{i-1}. Ainsi, avec une simple SPA, il est possible de retrouver tous les bits de l'exposant.

Une contremesure évidente à cette attaque consiste à interdire l'application du message $m = N - 1$. Cependant, dans ce cas, les auteurs ont montré que cette méthode peut aussi être étendue à l'utilisation de deux messages d'entrées comme dans l'attaque par doublement. Les messages utilisés alors sont m et $-m = (N-1)\cdot m \mod N$. Par exemple avec l'algorithme 6, l'attaquant choisit deux entrées m_1 et $m_2 = m_1 \cdot (N - 1) \mod N$.

L'attaque exploite donc le fait que :

$$m_1{}^d = m_2{}^d \bmod N, \tag{1.9}$$

si l'exposant d est pair.

Dans l'algorithme 6, si d_i est égal à 0, alors l'attaquant observe une collision entre les deux courbes de consommation collectées à la fin des opérations d'élévation au carré respectives de l'étape $i - 1$ car :

$$m_2{}^{(d_{n-1},\cdots,\, d_i)} \bmod N = m_1{}^{(d_{n-1},\cdots,\, d_i)}.(N-1)^{(d_{n-1},\cdots,\, d_i)} \bmod N = m_1{}^{(d_{n-1},\cdots,\, d_i)} \bmod N. \tag{1.10}$$

Un exemple de fonctionnement de l'attaque est donné par le tableau 1.2. L'exposant secret $d = (1, 0, 0, 1, 1, 0, 1)$ est utilisé lors des exponentiations modulaires de m et $-m$. Les calculs sont identiques pour $i = 4, 3, 0$ (marqués en rouge) et donc d_5, d_4 et d_1 valent 0. On déduit donc que les bits d_6, d_3 et d_2 valent 1. L'algorithme 10 résume les différentes étapes de l'attaque proposée dans [2].

i	d_i	Calcul de m^d	Calcul de $(-m)^d$
6	1	$R_0 = 1^2$ $R_0 = 1 \cdot m$	$R_0 = 1^2$ $R_0 = 1 \cdot (-m)$
5	0	$R_0 = m^2$ $R_1 = m^2 \cdot m$	$R_0 = (-m)^2$ $R_1 = m^2 \cdot (-m)$
4	0	$R_0 = (m^2)^2$ $R_1 = m^4 \cdot m$	$R_0 = (m^2)^2$ $R_1 = m^4 \cdot (-m)$
3	1	$R_0 = (m^4)^2$ $R_0 = m^8 \cdot m$	$R_0 = (m^4)^2$ $R_0 = m^8 \cdot (-m)$
2	1	$R_0 = (m^9)^2$ $R_0 = m^{18} \cdot m$	$R_0 = (-m^9)^2$ $R_0 = m^{18} \cdot (-m)$
1	0	$R_0 = (m^{19})^2$ $R_1 = m^{38} \cdot m$	$R_0 = (-m^{19})^2$ $R_1 = m^{38} \cdot (-m)$
0	1	$R_0 = (m^{38})^2$ $R_0 = m^{76} \cdot m$ Retourner $R_0 = m^{77}$	$R_0 = (m^{38})^2$ $R_0 = m^{152} \cdot (-m)$ Retourner $R_0 = m^{154}$

Tableau 1.2 – Valeurs intermédiaires de l'algorithme 6 avec les messages d'entrées m et $-m$

algorithme 10 L'attaque de S. Yen et al. [2]

ENTRÉES: $m, -m$

SORTIES: $d = (d_{n-1}, ..., d_1, d_0)$

1: Collecter la trace $\mathcal{T}_1$ obtenue lors l'exponentiation m^d.
2: Collecter la trace $\mathcal{T}_2$ obtenue lors l'exponentiation $(-m)^d$.
3: Diviser $\mathcal{T}_1$ et $\mathcal{T}_2$ respectivement en segments $\left\{T_1^1, T_2^1, \cdots, T_l^1\right\}$ et $\left\{T_1^2, T_2^2, \cdots, T_l^2\right\}$
4: **pour** chaque position i **faire**
5: Comparer les segments T_i^1 et T_i^2 en utilisant un distingueur
6: Si collision alors $d_{i-1} = 0$
7: Sinon $d_{i-1} = 1$
8: **fin pour**
9: **retourner** d

Les attaques par collisions que nous avons vues jusqu'à présent ciblent principalement les implémentations non protégées de l'algorithme 6 ou bien de l'algorithme 1. Voyons maintenant une autre variante de l'attaque du doublement proposée par S. Yen et al. en 2006 dans [8]. Il s'agit d'une adaptation de l'attaque par doublement, ciblant une implémentation de l'échelle de Montgomery (algorithme 7). Jusqu'à cette attaque, l'échelle de Montgomery semblait être l'algorithme offrant la meilleure résistance aux attaques de ce type.

1.4.3 L'attaque par doublement sur l'échelle de Montgomery

L'attaque par doublement ciblant l'échelle de Montgomery ou *Relative Doubling Attack* [8] est une attaque par doublement proposée exclusivement pour l'algorithme 7. Elle utilise les mêmes messages d'entrées m et m^2 que l'attaque par doublement. Cependant les collisions n'apparaissent pas de la même manière que pour l'algorithme carré-et-multiplication systématique (algorithme 6) comme nous allons le voir dans ce qui suit.

Détaillons d'abord le fonctionnement de l'échelle de Montgomery afin d'expliquer le principe de cette attaque.

Soit $\sum_{j=0}^{n-1} d_j 2^j$ l'écriture binaire de l'exposant d. L'échelle de Montgomery se base sur les observations présentées dans [42] :

Soient $L_i = \sum_{j=i}^{n-1} d_j 2^{(j-i)}$ et $H_i = L_i + 1$, on a alors :

$$(L_i, H_i) = \begin{cases} 2L_{i+1} + d_i = L_{i+1} + H_{i+1} - 1 + d_i, \\ L_{i+1} + H_{i+1} + d_i = 2H_{i+1} - 1 + d_i. \end{cases}$$

D'après ses observations, on obtient :

$$(L_i, H_i) = \begin{cases} (2L_{i+1}, L_{i+1} + H_{i+1}) & \text{si } d_i = 0, \\ (L_{i+1} + H_{i+1}, 2H_{i+1}) & \text{si } d_i = 1. \end{cases}$$

Dans l'algorithme 7, R_0 est utilisé pour contenir le résultat de m^{L_i} et R_1 pour contenir le résultat de m^{H_i}. Afin de rendre impraticable une attaque SPA, les étapes 4 et 5 sont conçues comme suit :

$$(R_1, R_0) = (m^{H_i}, m^{L_i}) = (m^{L_{i+1}} \cdot m^{H_{i+1}}, (m^{L_{i+1}})^2) \quad \text{si } d_i = 0, \tag{1.11}$$

$$(R_0, R_1) = (m^{L_i}, m^{H_i}) = (m^{L_{i+1}} \cdot m^{H_{i+1}}, (m^{H_{i+1}})^2) \quad \text{si } d_i = 1. \tag{1.12}$$

Ces étapes montrent bien que l'échelle de Montgomery est résistante aux attaques SPA du fait de sa régularité. Cependant, d'après les remarques détaillées dans [8], nous pouvons déduire une méthode similaire à l'attaque par doublement, pour défaire cet algorithme.

- **Remarque 1 :** si $d_i = 0$, alors $L_i = 2L_{i+1}$

- **Remarque 2 :** si $d_i = 1$, alors $H_i = 2H_{i+1}$

On déduit de l'équation 1.11 que si $d_i = d_{i-1} = 0$ alors :

$$(L_i, H_i) = \begin{cases} R_0 \leftarrow (m^{L_i})^2 : \text{ étape 5 de l'itération } i-1 \text{ quand } m^d \text{ est calculé,} \\ R_0 \leftarrow ((m^2)^{L_{i+1}})^2 : \text{ étape 5 de l'itération } i \text{ quand } (m^2)^d \text{ est calculé.} \end{cases}$$

seront identiques car $L_i = 2L_{i+1}$ (Remarque 1). Ainsi, une collision apparaît quand $d_i = d_{i-1} = 0$. D'autre part l'équation 1.12 montre que si $d_i = d_{i-1} = 1$ alors :

$$\begin{cases} R_1 \leftarrow (m^{H_i})^2 : \text{étape 5 de l'itération } i-1 \text{ quand } m^d \text{ est calculé} \\ R_1 \leftarrow ((m^2)^{H_{i+1}})^2 : \text{étape 5 de l'itération } i \text{ quand } (m^2)^d \text{ est calculé.} \end{cases}$$

seront identiques du fait de la Remarque 2 car $H_i = 2H_{i+1}$. Ainsi une collision apparaît quand $d_i = d_{i-1} = 1$. Dans les deux autres cas, il vient que :

- **Cas 1** : $d_i = 0$ et $d_{i-1} = 1$

$$\begin{cases} R_1 \leftarrow (m^{H_i})^2 : \text{étape 5 de l'itération } i-1 \text{ quand } m^d \text{ est calculé} \\ R_0 \leftarrow ((m^2)^{L_{i+1}})^2 : \text{étape 5 de l'itération } i \text{ quand } (m^2)^d \text{ est calculé.} \end{cases}$$

- **Cas 2** : $d_i = 0$ et $d_{i-1} = 1$

$$\begin{cases} R_0 \leftarrow (m^{L_i})^2 : \text{étape 5 de l'itération } i-1 \text{ quand } m^d \text{ est calculé} \\ R_1 \leftarrow ((m^2)^{H_{i+1}})^2 : \text{étape 5 de l'itération } i \text{ quand } (m^2)^d \text{ est calculé.} \end{cases}$$

De la définition de l'échelle de Montgomery, il vient que dans le **Cas 1**, $H_i \neq 2L_{i+1}$ et donc qu'il n'y a pas de collision ; de même dans le **Cas 2** car $L_i \neq 2H_{i+1}$. En résumé, en comparant l'exécution de m et m^2 on a :

- si $d_i = d_{i-1}$ alors une collision se produit soit sur R_0 soit sur R_1.

- si $d_i \neq d_{i-1}$ aucune collision ne se produit.

Afin de réaliser cette attaque, un adversaire doit enregistrer seulement deux courbes de consommation, une courbe pour m et une autre courbe pour m^2. S'il observe une collision alors $d_{n-1} = d_{n-2}$, sinon $d_{n-1} \neq d_{n-2}$. Une fois d_{n-2} obtenu l'attaquant cible le bit d_{n-3} puis le suivant jusqu'à d_0. Ainsi, il peut retrouver tous les bits de l'exposant d. Le tableau 1.3 détaille le fonctionnement de l'attaque pour $d = (1, 0, 0, 1, 0, 1, 1)$. L'algorithme 11 résume les différentes étapes de l'attaque proposée dans [8].

i	d_i	Calcul de m^d	Calcul de $(m^2)^d$
6	1	$R_0 = 1 \cdot m$ $R_1 = m^2$	$R_0 = 1 \cdot m^2$ $R_1 = (m^2)^2$
5	0	$R_1 = m^2 \cdot m$ $R_0 = m^2$	$R_1 = m^4 \cdot m^2$ $R_0 = (m^2)^2$
4	0	$R_1 = m^3 \cdot m^2$ $R_0 = (m^2)^2$	$R_1 = m^6 \cdot m^4$ $R_0 = (m^4)^2$
3	1	$R_0 = m^4 \cdot m^5$ $R_1 = (m^5)^2$	$R_0 = m^8 \cdot m^10$ $R_1 = (m^{10})^2$
2	0	$R_1 = m^{10} \cdot m^9$ $R_0 = (m^9)^2$	$R_1 = m^{20} \cdot m^{18}$ $R_0 = (m^{18})^2$
1	1	$R_0 = m^{18} \cdot m^{19}$ $R_1 = (m^{19})^2$	$R_0 = m^{36} \cdot m^{38}$ $R_1 = (m^{38})^2$
0	1	$R_0 = m^{37} \cdot m^{38}$ $R_1 = (m^{38})^2$ Retourner m^{75}	$R_0 = m^{74} \cdot m^{76}$ $R_1 = (m^{76})^2$ Retourner m^{150}

Tableau 1.3 – Valeurs intermédiaires de l'algorithme 7 avec les messages d'entrées m et m^2

algorithme 11 L'attaque par doublement ciblant l'échelle de Montgomery [8]

ENTRÉES: m, m^2
SORTIES: $d = (d_{n-1}, ..., d_1, d_0)$

1: Collecter la trace $\mathcal{T}_1$ obtenue lors l'exponentiation m^d.
2: Collecter la trace $\mathcal{T}_2$ obtenue lors l'exponentiation $(m^2)^d$.
3: Diviser $\mathcal{T}_1$ et $\mathcal{T}_2$ respectivement en segments $\left\{T_1^1, T_2^1, \cdots, T_l^1\right\}$ et $\left\{T_1^2, T_2^2, \cdots, T_l^2\right\}$
4: Prendre que les segments correspondants à un carré pour $\mathcal{T}_1$ et $\mathcal{T}_2$
5: **pour** chaque position i **faire**
6: Comparer les segments T_{i+1}^1 et T_i^2 en utilisant un distingueur
7: Si collision alors $d_{i-1} = d_i$
8: Sinon $d_{i-1} = 1 - d_i$
9: **fin pour**
10: **retourner** d

1.4.4 Vers une généralisation des attaques par collision

Les attaques par collision présentées précédemment ciblent les implémentations *Gauche-Droite* des algorithmes L2R, SMA ou encore de l'échelle de Montgomery. Dans [23, 9], N. Homma et al. généralisent les attaques par collision aux algorithmes du type Droite-Gauche (comme l'algorithme R2L) qui résistent naturellement aux premières attaques par collision.

Cette nouvelle attaque exploite des collisions entre deux traces de consommation à des positions arbitraires (pas nécessairement adjacentes) dans le calcul. L'idée consiste à utiliser une paire de messages d'entrée Y et Z qui satisfont la relation suivante :

$$Y^{\alpha} = Z^{\beta} \bmod N, \quad \text{avec } \alpha \leq \beta, \tag{1.13}$$

où α et β sont des constantes déterminées selon l'algorithme attaqué.

L'équation 1.13 admet toujours une solution. En effet, nous pouvons toujours choisir une valeur arbitraire R et prendre $Y = R^{\beta} \bmod N$ et $Z = R^{\alpha} \bmod N$. Cette attaque requiert donc, pour révéler chaque bit de l'exposant secret, deux traces de consommation.

Au message d'entrée Y correspond une trace contenant une opération cible inconnue (multiplication ou carré). Au message d'entrée Z correspond une trace contenant une opération d'élévation au carré. C'est l'opération de référence. La présence ou non d'une collision entre ces deux traces est utilisée pour identifier l'opération cible (inconnue).

Cette attaque se décline en deux versions : l'estimation directe et la rétro-estimation. L'estimation directe compare simplement les deux opérations pour identifier l'opération cible correspondant au bit cible. Tandis que la méthode de rétro-estimation identifie l'opération cible en fixant tout d'abord le bit cible avant de choisir la paire de messages d'entrée en fonction de ce choix.

1.4.4.1 Estimation directe appliquée au L2R (algorithme 1)

Dans l'estimation directe, il est nécessaire de connaître la partie de clé privé $D^{(j)} = (d_{n-1}, ..., d_{n-j})$. Pour obtenir le bit suivant $d_{n-(j+1)}$, on utilise une paire de messages qui favorise l'apparition d'une collision entre la cible et la référence. Le bit $d_{n-(j+1)}$ est obtenu en comparant la trace de consommation de l'opération cible avec comme message d'entrée Y^{α} avec celle contenant l'opération référence Z^{β}. Si on observe une collision entre ces deux traces alors l'opération cible est un carré ($d_{n-(j+1)} = 0$). Sinon l'opération est une multiplication ($d_{n-(j+1)} = 1$). On répète cette procédure pour obtenir tous les bits de l'exposant. La paire de messages est $Y^{\alpha} = Z^{\beta}$ ($Y \neq Z$), où α et β sont donnés par la relation suivante :

$$\begin{cases} \alpha = 2D^{(j)}, \\ \beta = \left\lfloor \frac{\alpha}{2^t} \right\rfloor, \qquad (0 \leq t \leq j) \end{cases} \tag{1.14}$$

Si $d_{n-(j+1)} = 0$ alors $(Y^{\alpha})^2 = (Z^{\beta})^2$. Dans le cas contraire, l'opération Y^{α} est une multiplication, et est donc différente de Z^{β}.

algorithme 12 Estimation Directe [9]

Entrées: $D^{(j)} = (d_{n-1}, \cdots, d_{n-j})$
Sorties: $d_{n-(j+1)}$
1: Calculer α et β en fonction de $D^{(j)}$ et de l'algorithme ciblé
2: calculer Y^{α} et Z^{β}
3: Comparer les segments de $(Y^{\alpha})^2$ et de $(Z^{\beta})^2$
4: **si** Collision **alors**
5: $d_{n-(j+1)} = 0$
6: **sinon**
7: $d_{n-(j+1)} = 1$
8: **fin si**
9: **retourner** $d_{n-(j+1)}$

1.4.4.1.1 Exemple

Supposons que l'attaquant connaisse quatre bits de l'exposant ($D^{(4)} = 1100_2$). Dans ce cas, α et β sont donnés par : $\alpha = 2 \cdot D^{(4)} = 24$ et $\beta = \left\lfloor \frac{2 \cdot D^{(4)}}{2^t} \right\rfloor$, $(0 \leq t \leq 4) = \{1, 3, 6, 12, 24\}$. La figure1.2 illustre un exemple d'application de la méthode directe sur l'algorithme 1. Pour estimer le prochain bit, la paire de messages (Y, Z) telle que $Y^{24} = Z^3$ est choisie. Y^{24} est la donnée d'entrée de l'opération que l'on cherche à estimer dans la trace cible et Z^3 la donnée d'entrée d'une opération carré dans la trace de référence. Si une collision est observée lors de la comparaison de ces deux opérations, alors le bit cible vaut 0 car l'opération cible est le carré $(Y^{24})^2$. Sinon l'opération cible est une multiplication et donc le bit est égal 1.

Notons que si on choisit $\beta = 12$ $\left(Y^{24} = Z^{12}\right)$ alors l'attaque est identique à celle par doublement et si $\beta = 24$ $\left(Y^{24} = Z^{24}\right)$ l'attaque est identique à celle de S. Yen et al. Ainsi ces deux attaques sont des cas particuliers de cette nouvelle attaque.

1.4.4.2 La méthode de rétro-estimation appliquée au L2R

Contrairement à l'estimation directe qui compare deux traces pour retrouver le bit recherché, la méthode de rétro-estimation fixe tout d'abord le bit $d_{n-(j+1)}$. Pour estimer le bit $d_{n-(j+1)}$, l'opération d'élévation au carré suivant l'opération cible est examinée. La paire de messages d'entrée est alors choisie en fonction de cette hypothèse émise sur le bit recherché.

Supposons par exemple que $d_{n-(j+1)} = 1$, la paire de messages Y et Z est choisie de manière à satisfaire la condition $Y^{\alpha+1} = Z^{\beta}$. Si l'hypothèse est correcte, alors $(Y^{\alpha+1})^2 = (Z^{\beta})^2$, et leurs deux traces sont donc identiques sur la partie correspondant du calcul. Sinon aucune collision ne se produit et donc $d_{n-(j+1)} = 0$.

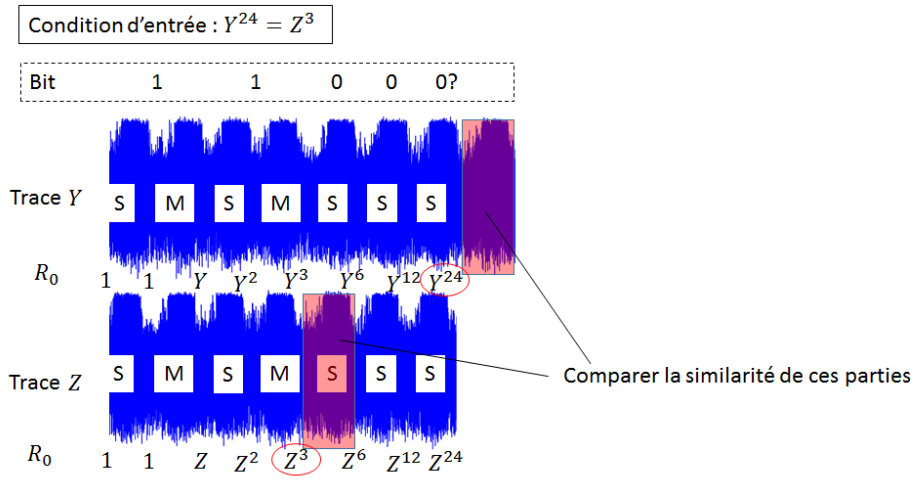


FIGURE 1.2 – Illustration de l'application de l'estimation directe sur l'algorithme L2R (algorithme 1).

algorithme 13 Rétro-estimation [9]

ENTRÉES: $D^{(j)} = (d_{n-1}, ..., d_{n-j})$
SORTIES: $d_{n-(j+1)}$

1: Fixer $d_{n-(j+1)}$ à 0 ou à 1
2: Calculer α et β en fonction de $D^{(j)}$ et de l'algorithme ciblé
3: calculer Y^α et Z^β
4: Comparer les patterns de $(Y^\alpha)^2$ et de $(Z^\beta)^2$
5: **si** Collision **alors**
6: $d_{n-(j+1)}$ est la valeur fixée à l'étape 1
7: **sinon**
8: $d_{n-(j+1)}$ est le complément de la valeur fixée à l'étape 1
9: **fin si**
10: **retourner** $d_{n-(j+1)}$

1.4.4.2.1 Exemple

La figure 1.3 donne un exemple d'application de la méthode rétro-estimation appliquée à l'algorithme 1. Supposons que l'attaquant connaisse toujours la partie de la clé ($D^{(4)} = 1100_2$). Pour estimer le bit cible, l'attaquant suppose qu'il est égal par exemple à 1. Il compare ensuite l'opération cible (la multiplication $Y^{24} \cdot Y$ est attendue si l'hypothèse est correcte) avec l'opération référence (Z^3). Si la supposition est correcte alors $(Y^{25})^2 = (Z^3)^2$ et les deux courbes sont identiques. Dans le cas contraire le bit cible est 0 et l'opération cible est un carré $(Y^{24})^2$.

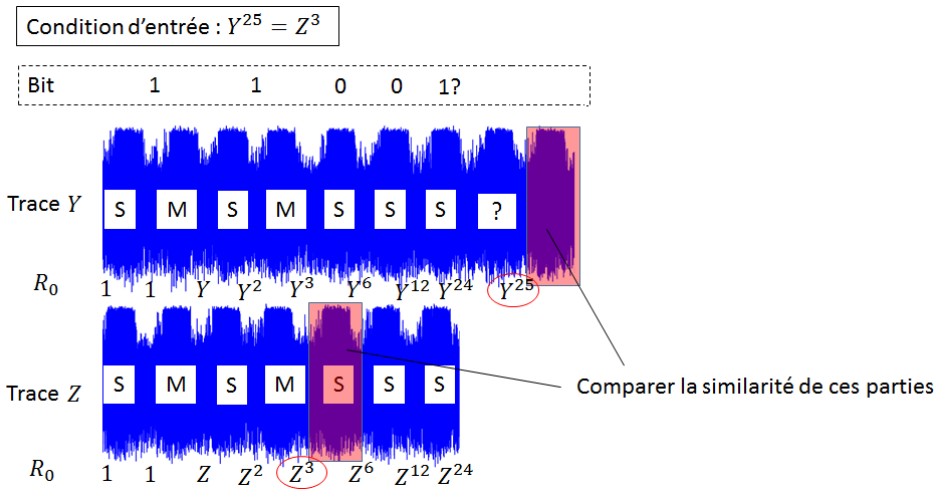


FIGURE 1.3 – Illustration de fonctionnement de la méthode de rétro-estimation sur l'algorithme L2R (algorithme 1).

L'avantage de cette nouvelle attaque est qu'elle peut être appliquée sur les implémentations de type Droite-Gauche, comme l'algorithme R2L (algorithme 2) qui résistent aux autres attaques par collision, comme nous allons le détailler dans le paragraphe qui suit.

1.4.4.3 Estimation directe appliquée au R2L (algorithme 2)

La différence avec l'attaque sur l'algorithme 1 est la façon de déterminer α et β. En effet, les attaques par doublement [1] ou celle de S. Yen et al. [2] exploitent seulement les entrées de l'opération d'élévation au carré, ce qui fait que l'algorithme 2 qui utilise deux variables R_0 et R_1 pour enregistrer respectivement les résultats intermédiaires et le résultat du carré, résiste à ces attaques. Ce n'est pas le cas pour cette nouvelle attaque qui exploite à la fois les entrées et les positions des opérations d'élévation au carré.

Supposons que l'attaquant connaisse déjà les j premiers bits $D^{(j)} = (d_{j-1}, ..., d_0)$ de l'exposant privé d. Pour appliquer l'estimation directe à l'algorithme 2, α et β

sont calculés comme suit :

$$\begin{cases} \alpha = 2^j, \\ \beta = 2^{t-1}, \quad (1 \leq t \leq j) \end{cases} \tag{1.15}$$

L'attaquant enregistre une courbe de consommation correspondant à l'opération de référence ayant comme entrée (Z^β) et une courbe de consommation correspondant à l'opération cible ayant comme entrée Y^α. Si $d_j = 0$, alors $(Y^\alpha)^2$ est égal à $(Z^\beta)^2$. Si $d_j = 1$, alors $(Y^\alpha)^2$ est une multiplication, est donc une opération différente de $(Z^\beta)^2$. Par conséquent l'attaquant obtient le bit cible en comparant la courbe de consommation de l'opération cible manipulant Y^α et celle de l'opération référence manipulant Z^β.

1.4.4.3.1 Exemple

Supposons que l'attaquant connaisse les quatre premiers bits ($D^{(4)} = 0011_2$) de l'exposant. Dans ce cas, α et β sont donnés par $\alpha = 16$ et $\beta = 1, 2, 4, 8$, et ainsi la détermination du bit cible est effectuée en vérifiant si les opérations prenant comme entrées Y^{16} et Z^4 sont identiques. Si le bit cible est égal à 0, l'opération cible est un carré. Dans ce cas une collision se produit entre le carré de Y^{16} et celui de Z^4. Si le bit cible est égal à 1 alors l'opération cible est une multiplication est donc les deux traces sont différentes.

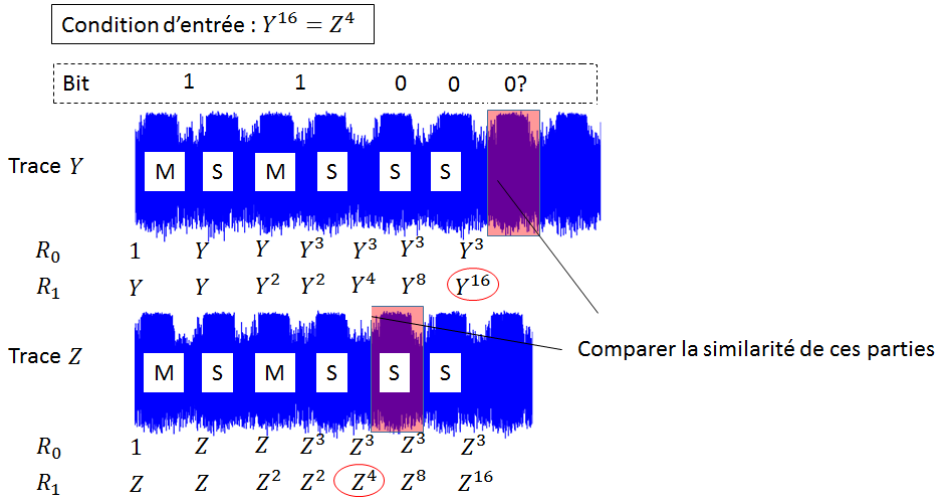


FIGURE 1.4 – Illustration de fonctionnement de la méthode de l'estimation directe sur l'algorithme R2L (algorithme 2)

1.4.4.4 Attaque de la méthode carré-et-multiplication-systématique (algorithme 6)

Une méthode rétro-estimation similaire à celle du L2R (algorithme 1) peut défaire l'exponentiation carré-et-multiplication-systématique (algorithme 6) en identifiant les multiplications fictives ($d_i = 0$) d'une façon similaire aux attaques Safe-Error [41]. En effet, supposons que la partie $D^{(j)} = (d_{n-1}, ..., d_{n-j})$ de la clé est connue, pour retrouver le bit suivant $d_{n-(j+1)}$, les paramètres α et β sont d'abord choisis à l'aide de l'équation 1.15. En supposant que $d_{n-(j+1)} = 1$, la condition d'entrée sera $Y^{\alpha+1} = Z^{\beta}$. Si $d_{n-(j+1)} = 0$, alors la multiplication qui suit le carré est fictive, donc l'entrée de l'opération carré cible est toujours Y^{α}. Ainsi aucune collision ne sera observée entre les deux segments de courbes.

1.4.4.4.1 Exemple

Supposons que l'attaquant connaisse la partie ($D^{(4)} = 1100_2$) de l'exposant. Si la multiplication qui suit le carré n'est pas fausse (i.e $d_{n-5} = 1$), la condition est $Y^{25} = Z^3$. Si l'hypothèse est vraie, une multiplication $Y^{24} \cdot Y$ est effectuée et sera l'entrée de l'opération carré qui suit. Si l'hypothèse est incorrecte alors l'opération cible est une fausse multiplication, et l'entrée de l'opération carré est Y^{24} qui n'est pas égale à Z^3.

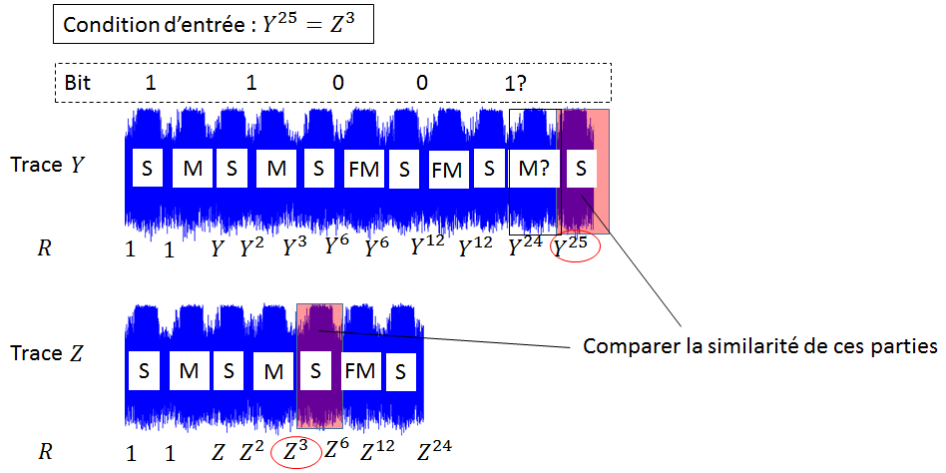


FIGURE 1.5 – Illustration de fonctionnement de la méthode de rétro-estimation sur l'exponentiation carré-et-multiplication-systématique (algorithme 6)

1.4.4.5 Attaque de l'échelle de Montgomery (algorithme 7)

Une variante de la méthode directe utilisée dans le cadre de l'algorithme L2R permet de défaire l'échelle de Montgomery (algorithme 7). Supposons que la partie de la clé $D^{(j)}$ soit déjà connue. Les valeurs de R_0 et R_1 après chaque itération sont $m^{D^{(j)}}$ et $m^{D^{(j)}+1}$ respectivement. Ainsi, nous avons deux possibilités pour l'entrée

de l'opération carré référence. Supposons que $d_{n-(j+1)} = 1$ (resp. 0), nous pouvons choisir les paramètres $\alpha = D^{(j)} + 1$ (resp. $D^{(j)}$). De la même façon, le paramètre β est donné par $D^{(t)} + 1$ (resp. $D^{(t)}$) quand $d_{n-(t+1)} = 1$ (ou 0) avec $1 \leq t \leq j-1$. La paire de messages (Y, Z) est choisie pour satisfaire la condition d'entrée $Y^{\alpha} = Z^{\beta}$. Si l'hypothèse est correcte, Y^{α} est la donnée d'entrée de l'opération carré ciblée qui suit la multiplication, et est égale à Z^{β} qui est l'entrée de l'opération carré référence située à la position $n-(t+1)$, on observe donc une collision entre les deux traces.

1.4.4.5.1 Exemple

La figure 1.6 illustre l'application de l'attaque sur l'algorithme 7. Admettons que l'attaquant connaisse la partie ($D^{(4)} = 1100_2$). En supposant que $d_{n-5} = 1$, on utilise la valeur de $R_1 = Y^{13}$ comme données d'entrée de l'opération carré ciblée (opération cible). Le critère de décision de l'attaque est alors la comparaison des opérations prenant Y^{13} et Z^2 comme données d'entrée car $d_{n-2} = 1$. Si l'hypothèse est correcte, les mêmes entrées Y^{13} et Z^2 sont utilisées pour la cible et la référence et une collision est observable. Dans le cas contraire, le résultat de la multiplication est mis dans R_1 et la valeur $R_0 = Y^{12}$ est la donnée d'entrée de l'opération cible. Ainsi, il n'y a pas de collision entre les deux traces.

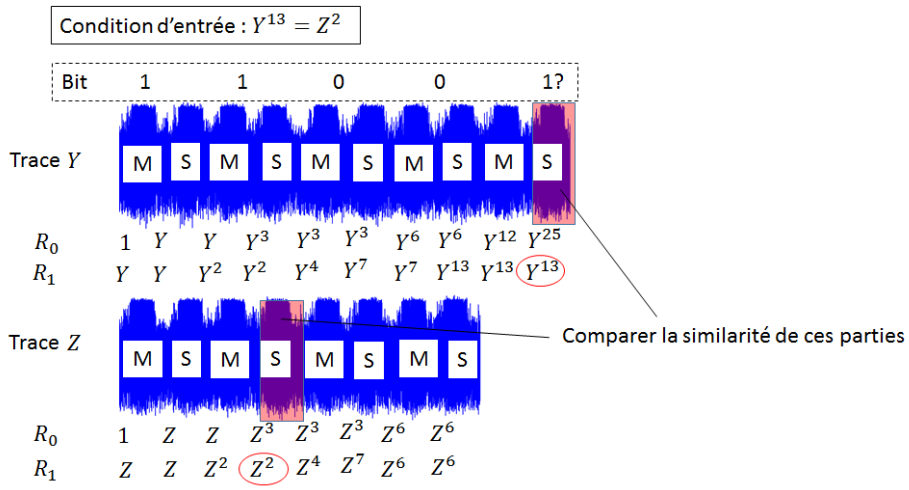


FIGURE 1.6 – Illustration de fonctionnement de la méthode de l'estimation directe sur l'échelle de Montgomery (algorithme 7)

1.4.5 Contremesures visant à lutter contre les attaques verticales

Comme nous l'avons abordé dans ce qui précède, pour contrecarrer les attaques de type SPA ou temporelles, les algorithmes d'exponentiation *irréguliers* ont été abandonnés au profit des algorithmes dits *réguliers*. De tels algorithmes restent

toutefois vulnérables aux attaques verticales.

Pour contrecarrer à la fois les attaques SPA et les attaques verticales, des méthodes de masquage des données utilisées par ces algorithmes réguliers ont été mises au point et sont souvent mises en œuvre. Ces méthodes sont habituellement désignées par le terme de *Masquage*. Il existe plusieurs façons de masquer les données dans un algorithme d'exponentiation modulaire. En effet, nous pouvons :

- masquer le modulo,
- masquer le message,
- masquer l'exposant,
- combiner plusieurs masquages.

1.4.5.1 Masquage du modulo

Le masquage du modulo consiste à remplacer le modulo utilisé lors de l'exponentiation modulaire par un multiple de ce dernier différent pour chaque exponentiation [40]. Ainsi le modulo utilisé dans l'exponentiation modulaire sera non pas N mais $N' = (r.N)$ où r désigne un nombre aléatoire différent pour chaque exécution. Ainsi, on calcule $C' = m^d \bmod r \cdot N$. Pour obtenir le résultat final C, il suffit de réduire le résultat obtenu modulo N.

1.4.5.2 Masquage du message

Dans [40] J.-S. Coron propose un masquage du message. Il consiste à remplacer le message utilisé lors de l'exponentiation modulaire par un multiple de ce dernier différent pour chaque exponentiation. Le nouveau message est alors $M' = M \cdot r$ où r désigne un nombre aléatoire différent pour chaque exécution. Ainsi, on calcule $C' = (M \cdot r)^d$ et $R = r^{-d}$. Le résultat final est obtenu en multipliant C' et R.

Une version optimale consiste à utiliser l'exposant public e pour calculer dans un premier temps $m_1 = r^e \bmod N$ et $m_2 = r^{-1} \bmod N$. Le message d'origine m sera masqué en le multipliant par m_1. Ainsi le nouveau message $m' = m \cdot m_1$ est utilisé pour effectuer l'exponentiation modulaire $C' = m'^d \bmod N$. Pour obtenir le résultat final, il suffit alors de multiplier C' par m_2 . En effet, on a :

$$\begin{aligned} C' \cdot m_2 \bmod N &= [m'^d \bmod N \cdot m_2] \bmod N \\ &= [(m.m_1)^d \cdot m_2] \bmod N \\ &= [m^d \cdot (m_1^d \cdot m_2)] \bmod N \\ &= m^d \cdot (r^{ed} \cdot r^{-1}) \bmod N \\ &= m^d \bmod N \end{aligned}$$

Une autre façon de masquer le message consiste à choisir deux nombres aléatoires r_m et r_n tels que : $r_n < r_m$. Le message masqué sera alors

$m' = m + (r_m N) \bmod r_n N$. Cette méthode permet de masquer simultanément le message et le modulo.

1.4.5.3 Masquage de l'exposant

Pour se prémunir des attaques SPA ou DPA, nous pouvons aussi masquer l'exposant. Pour se faire J.-S. Coron propose dans [40], de remplacer l'exposant d par $d' = d + r \cdot \phi(n)$ où r un nombre aléatoire et $\phi(n)$ l'indicatrice d'Euler. Ainsi, calculer $C = m^d \bmod N$ revient à calculer calculer $m^{d'} \bmod N$. En effet :

$$M^{d'} \bmod N = M^{d+r\cdot\phi(N)} \bmod N = M^d \cdot M^{r\cdot\phi(N)}$$

or d'après le théorème d'Euler :

$$M^{r.\phi(N)} = 1 \bmod N, \ \forall\, r$$

Il vient alors :

$$M^{d'} = M^d \bmod N$$

Une autre façon de masquer l'exposant consiste, comme cela est proposé par C. Clavier et al. dans [55], à le scinder en deux ou plusieurs parties comme suit :

$$d = (d - r) + r$$

et à calculer les exponentiations avec $d - r$ et r respectivement. Dans ce cas, le résultat final est obtenu par la multiplication de l'ensemble des exponentiations. Cette méthode est appelée découpage additif de l'exposant.

Nous pouvons aussi découper l'exposant en utilisant la version multiplicative proposée par M. Ciet et al. dans [56]. En effet, on peut calculer :

$$(m^r)^{\lfloor d/r \rfloor} m^{(d \bmod r)},$$

au lieu de m^d. Cependant, cette méthode nécessite un calcul d'inverse de l'aléa r, ce qui constitue un inconvénient. Ces deux méthodes sont connues sous le nom de découpage de l'exposant ou *exponent splitting* en Anglais.

De ce qui précède, une protection d'une très grande efficacité peut consister à masquer simultanément toutes les données de l'algorithme d'exponentiation modulaire comme suit :

$$m^d \bmod N = \left[(m + r_m \cdot N \bmod r_N \cdot N)^{d+r_d\cdot\phi(N)}\right] \bmod N \tag{1.16}$$

1.4.6 Attaques verticales exploitant la non-homogénéité du masquage additif de l'exposant avec l'indicatrice d'Euler

Avec le masquage de l'exposant (section 1.4.5.3), un attaquant ne doit normalement utiliser qu'une seule courbe pour retrouver l'exposant. Cependant, depuis la publication de D. Boneh et al. [57], des attaques exploitant la non-homogénéité du masquage de l'exposant par $\phi(N)$ ont fait leur apparition dans la littérature [31, 58, 59, 60]. Ces attaques utilisent plusieurs courbes et exploitent le fait que, même en présence du masquage de l'exposant par $\phi(N)$, une partie de ce dernier reste exposée. Elles se basent sur l'observation qu'avec l'équation du système RSA $ed = 1 \bmod \phi(N)$, il existe un entier $k \in \mathbb{Z}$ tel que :

$$ed = 1 + k\phi(N). \tag{1.17}$$

L'indicatrice d'Euler $\phi(N)$, de taille n, est égale à $pq - p - q + 1$. En considérant que p et q sont de tailles $n/2$ (car $p, q \approx \sqrt{N}$), les bits de poids fort (MSB) de $\phi(N)$ sont égaux aux bits de poids fort de $N = pq$. Pour un exposant public petit ($\leq 2^{16} + 1$), cette condition permet de déduire la moitié (MSB) de l'exposant d. Cette observation constitue le point de départ des attaques décrites dans les parties suivantes.

1.4.6.1 Attaque de P. A. Fouque et al. [31]

P. A. Fouque et al. ont été les premiers à montrer comment exploiter la non-homogénéité du masquage de l'exposant utilisant $\phi(N)$ dans leur article paru en 2006. Leur idée est de retrouver l'exposant d à partir de plusieurs bits non consécutifs des exposants obtenus à partir de plusieurs masquages de l'exposant privé d par $\phi(N)$.

En effet, sachant que $d \in [0, \phi(N) - 1]$ et $k\phi(N) = ed - 1 < ed$, il vient que $k < e$, et donc pour $e = 3$, $k \in [1, 2]$. Dans ce cas, $k = 2$ ([31]) et à partir de l'équation 1.17 on a $3d - 2\phi(N) = 1$, comme prouvé dans l'article [31].
On peut donc considérer que :

$$\tilde{d} = \left\lfloor \frac{1 + kN}{e} \right\rfloor = \left\lfloor \frac{1 + 2N}{3} \right\rfloor,$$

est une bonne approximation de d sur les bits de poids fort.

Compte tenu de cela, la première partie de l'attaque consiste à associer un nombre aléatoire r_i à une courbe C_i. Pour cela, on calcule tous les exposants masqués possibles $\tilde{d}'_i = \tilde{d} + r_i\phi(N)$ pour tous les r_i possibles. L'attaque suppose que l'adversaire a récupéré les bits de poids fort (MSB) de l'exposant privé d avec une attaque SPA exploitant les imperfections dans l'implémentation des contremesures. L'association est alors faite en comparant les bits retrouvés via l'attaque SPA avec ceux du MSB de $\tilde{d}'_i$.

La seconde étape de l'attaque consiste à retrouver les bits de poids faible de d. Pour cela, l'adversaire essaie de deviner w bits de $\phi(N)$, en calculant d mod

$2^w = (1 + 2\phi(N))/3$, en commençant par le bit de poids fort d_0. L'idée principale est de retrouver quelques bits à partir des exposants masqués $\tilde{d}'_i$ en appliquant une attaque verticale et les comparer avec $d \bmod 2^w$.

1.4.6.2 Attaque Schindler-Itoh [58]

L'attaque de Schindler-Itoh [58] se base sur l'hypothèse suivante : « *il existera toujours des fuites SPA résiduelles sur des courbes collectées à partir d'un algorithme d'exponentiation protégé, malgré les contremesures tant algorithmiques qu'environnementales*. »

À partir de cette hypothèse, les auteurs supposent que certains bits de l'exposant masqué sont retrouvés avec une attaque, non définie dans l'article, faite en amont.

La première phase de l'attaque consiste à identifier les exposants utilisant les mêmes facteurs de masquage r_i (qui sont inconnus) et regrouper ces derniers dans la même classe. Pour identifier les exposants ayant le même facteur de masquage, on calcule leur distance de Hamming. Si cette distance est inférieure à un seuil fixé par l'attaquant, les exposants sont considérés comme ayant le même facteur de masquage.
La seconde phase de l'attaque consiste à choisir la classe qui contient le plus d'éléments. Pour finir, on applique la décision majoritaire sur les bits des exposants contenus dans cette classe pour déterminer les bits de l'exposant randomisé.

1.4.6.3 Attaque de S. Bauer [60]

L'attaque de S. Bauer [60] est une attaque de type *template*. Elle cible les algorithmes où la séquence d'opérations dépend des bits de l'exposant. Elle suppose la connaissance du MSB de d.

L'idée de l'attaque est de construire des modèles (des templates), des opérations d'élévation au carré et des opérations de multiplication, basés sur leur variance et leur espérance mathématique. Ces modèles sont construits en utilisant une vérification de signature RSA non protégée. Ils fournissent la probabilité pour qu'une opération soit un carré. L'attaquant applique dans la première phase de l'attaque les modèles sur plusieurs courbes. Ainsi, il identifie les facteurs de masquage candidats r pour chaque exposant randomisé (chaque courbe), comme dans le cas de l'attaque de P. A. Fouque et al. [31]. Cette identification est faite en approximant $r\phi(N)$ par rN. Après cette étape, l'attaquant dispose pour chaque courbe, d'un ensemble de candidats pour le facteur de masquage r.

La seconde phase consiste à calculer la valeur de k dans l'équation $\tilde{d} = \left\lfloor \frac{1+kN}{e} \right\rfloor$. Ainsi, l'attaquant teste toutes les valeurs de k possibles et tous les facteurs de masquage candidats pour r et calcule la partie supérieure (MSB) de $\tilde{d}+\tilde{r}N$. Pour chaque couple (k, r), une séquence de carré et de multiplication est déduite en utilisant les modèles de la première étape pour la validation. Après cette seconde étape, l'attaquant a une liste où chaque élément est un couple formé par une courbe et son

facteur de masquage, avec une forte probabilité que cette liste soit correcte.

La dernière étape de l'attaque consiste à découvrir la seconde partie l'exposant d'origine d. Pour cela, l'attaquant devine une portion de d et $\phi(N)$ commençant par les bits de poids faible. Il calcule $d'_i = \tilde{d} + r_i\phi(\tilde{N})$, en considérant uniquement le facteur de masquage obtenu à la seconde étape. En convertissant d'_i en séquence de carrés et de multiplications, il utilise les modèles de la première étape pour valider les portions devinées de d et de $\phi(N)$.

1.4.6.4 Attaque de A. Bauer et E. Jaulmes [59]

L'attaque de A. Bauer et E. Jaulmes proposée dans [59], cible les exponentiations modulaires pour lesquelles le message et l'exposant sont masqués. Contrairement aux précédentes attaques, elle ne suppose pas l'existence de fuites SPA résiduelles, ni la connaissance de l'implémentation des opérations modulaires. Elle suppose uniquement que l'adversaire peut choisir les points d'intérêt pour chaque opération modulaire, comme dans le cas de l'attaque CPA.

Partant de la connaissance du MSB de d, les auteurs proposent deux versions, une avec un message connu, l'autre sans la connaissance du message. Ici nous présentons uniquement la version où le message n'est pas connu de l'attaquant. Dans ce cas, les auteurs proposent d'exploiter *l'analyse par collision-corrélation* comme dans l'attaque *Recovery of Secret Exponent by Triangular Trace Analysis* (ROSETTA) (section 1.5.3).

La première partie de l'attaque consiste à deviner les facteurs de masquage r_i pour obtenir le MSB de chaque d'_i. Pour cela l'attaquant établit une recherche exhaustive de toutes les valeurs possibles de $r_i \in \left\{0, 2^{32}\right\}$ pour chaque courbe $\mathcal{T}^{(i)}$, en exploitant la connaissance du MSB de d. L'attaquant calcule ainsi le coefficient de corrélation entre deux ensembles L_1 et L_2 en fonction de l'algorithme cible.

Par exemple pour l'algorithme 8, L_1 est l'ensemble constitué par les fuites associées aux écritures des résultats des opérations modulaires et L_2 l'ensemble constitué par les fuites associées aux chargements d'un des opérandes des opérations suivantes. Une forte corrélation entre L_1 et L_2, signifie que la valeur supposée de r_i est la bonne. Après cette phase, chaque courbe $\mathcal{T}^{(i)}$ est reliée à une valeur de r_i.

La seconde partie de l'attaque consiste à deviner pas à pas le LSB de d et d'appliquer le même procédé que celui utilisé dans la première partie pour la validation.

1.5 Attaques horizontales

Les attaques par canaux cachés les plus utilisées sont aujourd'hui la DPA, et la CPA. Ces attaques utilisent plusieurs courbes de consommation (des millions de courbes parfois). On appelle *attaques verticales* les attaques nécessitant plusieurs courbes de consommation, donc plusieurs exécutions de l'algorithme.

Comme nous l'avons vu précédemment, un masquage des données permet de contrer ce type d'attaques. Pour contourner ces contremesures, des attaques n'uti-

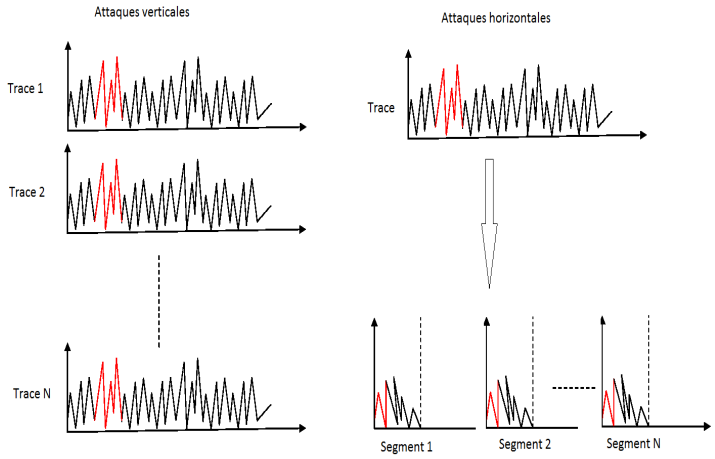


FIGURE 1.7 – Principe des attaques horizontales

lisant qu'une seule courbe de consommation ont fait leur apparition récemment.

Contrairement aux attaques verticales, elles comparent des segments issus d'une même trace de consommation entre eux, afin d'extraire le secret.

Ces nouvelles attaques permettant de contourner les contremesures de masquage, sont appelées *attaques horizontales*. La figure 1.7 illustre le principe des attaques verticales et des attaques horizontales.

L'une des premières attaques horizontales conduite par les cryptanalystes fût l'attaque *Big-Mac* [3] proposée par C. D. Walter en 2001. Mais le terme d'attaque horizontale n'a été utilisé, pour la première fois par C. Clavier et al., qu'en 2010 dans [4]. Ces mêmes auteurs publient en 2012 une attaque améliorée, qui introduit l'attaque *ROSETTA* [5] permettant d'attaquer certains algorithmes réguliers (l'exponentiation atomique par exemple) dont les données sont masquées. D'autres attaques horizontales ciblant notamment l'échelle de Montgomery et utilisant des algorithmes de classification ont été proposées respectivement par J. Heysel et al. [61] et G. Perin et al. [7]. Dans ce qui suit, quelques-unes de ces attaques sont détaillées.

1.5.1 L'attaque Big-Mac [3]

C. D. Walter présente une attaque dans [3] contre la méthode d'exponentiation à fenêtre glissante (*Left-to-Right sliding windows*) et l'exponentiation m-ary [36]. Cependant, bien qu'initialement définie pour ce type d'implémentation, cette attaque est adaptable à l'exponentiation atomique (algorithme 8).

Le principe de cette attaque est d'identifier sur une seule trace d'exponentiation $\mathcal{T}$, les opérations impliquant une multiplication par le message m. Cela équivaut, dans le cas de l'algorithme 8, à retrouver l'exposant secret.

L'idée de l'attaque Big-Mac consiste à construire un modèle (ou "template") caractérisant une opération durant laquelle le message m est manipulé. Pour cela, la deuxième multiplication dans l'algorithme 8 (ou l'étape de pré-calcul d'une exponentiation fenêtre coulissante) peut être exploitée.

Ensuite, l'attaquant calcule la distance euclidienne entre ce modèle et chaque partie de la trace correspondant à une multiplication de deux grands nombres entiers $x \cdot y$. Si cette distance excède un seuil choisi arbitrairement par l'attaquant, alors l'opération est un carré. Sinon l'opération est une multiplication. L'attaquant retrouve ainsi la séquence d'opérations dans un algorithme effectuant des multiplications en fonction du bit de clé.

Cependant, malgré sa pertinence pour certains algorithmes d'exponentiation, l'attaque *Big-Mac* reste inefficace contre des algorithmes où la séquence des opérations ne dépend pas de la valeur d'un bit comme l'échelle de Montgomery (algorithme 7) ou encore l'exponentiation carré-et-multiplication-systématique (algorithme 6).

1.5.2 L'attaque par Corrélation horizontale [4]

L'attaque par corrélation horizontale ou *Horizontal Correlation Analysis* (HCA) [4] a été proposée par C. Clavier et al. en 2010. Comme les attaques DPA et CPA, la HCA révèle les bits de l'exposant privé l'un après l'autre.

Le principe de l'attaque est de retrouver un bit de l'exposant en regardant si l'opération effectuée par ce bit implique ou non une multiplication par m, comme dans le cas de l'attaque *Big-Mac*. Elle vise principalement l'exponentiation atomique (algorithme 8). La différence avec l'analyse verticale classique réside dans la façon dont le test d'hypothèse est construit et vérifié.

Soient $\mathcal{T}$ la trace obtenue avec l'exponentiation et $\mathcal{T}^k$ la partie de la trace correspondant à la k-ème multiplication. Calculer $x \cdot y$ en utilisant l'algorithme 3 nécessite n^2 t-bits multiplications. On note $\mathcal{T}^k_{i,j}$ le segment de trace correspondant à la multiplication interne $x_i \cdot y_j$ dans T^k. Ces n^2 segments sont généralement identifiables par un attaquant sur une trace de consommation quand il a une certaine connaissance du matériel utilisé, en particulier la taille t du multiplieur et la méthode de multiplication utilisée.

Pour déterminer si une opération implique une multiplication par m, l'attaquant calcule le coefficient de Pearson entre les poids de Hamming des mots de t-bit m_j du message m et les segments de trace $\mathcal{T}^k_{i,j}$, $0 \leq i, j < n$. Le coefficient de Pearson est calculé comme suit :

$$\rho(H, \mathcal{T}^k), \quad \text{où} \quad H = (H_0, \cdots, H_{n-1}),$$

avec $H_j = \Big(HW(m_j), \cdots, HW(m_j)\Big)$, avec $HW(m_j)$ le poids de Hamming de m_j et $\mathcal{T}^k = \Big(\mathcal{T}^k_0, \cdots, \mathcal{T}^k_{n-1}\Big)$, avec $\mathcal{T}^k_j = \Big(\mathcal{T}^k_{0,j}, \cdots, \mathcal{T}^k_{n-1,j}\Big)$. Plus la valeur absolue de la corrélation est importante, plus la probabilité que $\mathcal{T}^k$ soit une multiplication est grande.

Contrairement à l'attaque *Big-Mac*, l'attaque HCA peut s'appliquer sur l'exponentiation carré-et-multiplication-systématique (algorithme 6) ou encore sur l'échelle de Montgomery (algorithme 7). Néanmoins, elle suppose la connaissance du message d'entrée ou l'utilisation d'aléa de petite taille pour le masquage du message.

1.5.3 L'attaque ROSETTA [5]

L'attaque ROSETTA [5] peut être considérée comme une amélioration de l'attaque HCA. En effet, l'attaque HCA s'applique avec un message connu tandis que l'attaque ROSETTA suppose que l'attaquant ne connaît aucune des données de l'exponentiation. L'attaque peut donc s'appliquer sur des exponentiations où toutes les données sont masquées et ce quelque soit la taille des aléas utilisés.

Le principe de l'attaque est de différencier les opérations de multiplications des carrés, dans une exponentiation atomique (algorithme 8), en faisant une analyse triangulaire des segments $\mathcal{T}^k$.

Pour cela, C. Clavier et al. observent qu'une multiplication en base $b = 2^t$ faite grâce à l'algorithme 3 peut être représentée par :

$$\mathcal{T}^k = \begin{bmatrix} \mathcal{T}^k_0 \\ \mathcal{T}^k_1 \\ \vdots \\ \mathcal{T}^k_{n-1} \end{bmatrix} = \begin{bmatrix} \mathcal{T}^k_{0,1} & \mathcal{T}^k_{0,1} & \cdots & \mathcal{T}^k_{0,n-1} \\ \mathcal{T}^k_{1,0} & \mathcal{T}^k_{1,1} & \cdots & \mathcal{T}^k_{1,n-1} \\ \vdots & \vdots & \ddots & \vdots \\ \mathcal{T}^k_{n-1,0} & \mathcal{T}^k_{n-1,1} & \cdots & \mathcal{T}^k_{n-1,n-1} \end{bmatrix} = \begin{bmatrix} x_0y_0 & x_0y_1 & \cdots & x_0y_{n-1} \\ x_1y_0 & x_1y_1 & \cdots & x_1y_{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n-1}y_0 & x_{n-1}y_1 & \cdots & x_{n-1}y_{n-1} \end{bmatrix}$$

Le triangle inférieur et le triangle supérieur de cette matrice se correspondent si $\mathcal{T}^k$ est un carré.

Pour comparer ces deux triangles les auteurs proposent deux méthodes. La première méthode consiste à calculer la distance euclidienne entre ces deux parties comme suit :

$$T_{ED} = \frac{2}{n^2 - n} \sum_{0 \le i < j < n} \sqrt{(\mathcal{T}^k_{i,j} - \mathcal{T}^k_{j,i})^2}$$

Si l'opération est un carré alors les multiplications $x_i \cdot y_j$ et $x_j \cdot y_i$ sont égales et donc les fuites générées par ces deux calculs sont similaires. Si maintenant $x \neq y$, les deux produits sont différents et donc les fuites sont moins similaires. Pour un carré on aura $E(T_{ED}) \approx 0$, tandis qu'une multiplication donne une valeur proche de $t/2$, E désigne ici l'espérance mathématique.

L'autre méthode appelée *collision-corrélation*, consiste à définir les deux ensembles de segments de trace suivants :

$$\Theta_0 = \{\mathcal{T}^k_{i,j} \textit{ tel que } \ 0 \le i < j \le n-1\}$$

$$\Theta_1 = \{\mathcal{T}^k_{j,i} \; tel \; que \;\; 0 \leq i < j \leq n-1\}$$

et calculer le *coefficient de Pearson* entre les deux ensembles Θ_0 et Θ_1
Si l'opération effectuée par la LIM est un carré, alors le calcul indique une forte corrélation contrairement à une multiplication.

1.5.4 L'analyse par Corrélation Croisée [6]

Dans [6] M. Witteman et al. proposent l'utilisation de la corrélation croisée ou *Cross-corrélation* pour exploiter la collision interne entre deux opérations consécutives dans l'exponentiation carré-et-multiplication systématique (algorithme 6 et ce même en présence de messages masqués.

Le principe de cette attaque est d'identifier les multiplications inutiles dans l'exponentiation carré-et-multiplication-systématique (algorithme 6). En effet, malgré la présence d'opérations factices dans l'algorithme 6, le contenu de R_0 ne sera pas modifié lorsque le bit d'exposant d_i est égal à 0. Ainsi pour le traitement du bit d_{i-1} suivant, l'algorithme manipulera la même valeur de R_0 utilisée pour l'opération d'élévation au carré lors du traitement de d_i. Une collision se produit alors entre l'opération de multiplication dans le traitement d_{i-1} et l'opération d'élévation au carré dans celui de d_i. En détectant ces collisions internes, l'attaquant récupère alors tous les bits de l'exposant.

Cependant l'attaque nécessite plusieurs courbes d'exponentiation. Cette attaque sera améliorée l'année suivante par N. Hanley et al. dans [24]. Ils généralisent l'attaque de M. Witteman et al. en utilisant une seule courbe d'exponentiation, et ce sur des algorithmes comme l'échelle de Montgomery (algorithme 7) ou encore l'exponentiation atomique (algorithme 8), protégés par le masquage de l'exposant et du message.

1.5.5 Attaques basées sur le Clustering

Les attaques horizontales abordées précédemment visent à identifier la séquence des opérations dans des algorithmes tels que l'exponentiation atomique (algorithme 8). Ainsi l'échelle de Montgomery résiste à la plupart de ces attaques. Pour attaquer des algorithmes tels que l'échelle Montgomery, en utilisant uniquement une courbe, il faut être capable de distinguer le traitement d'un bit à 1 de celui d'un bit à 0.

Un nouveau type d'attaque utilisant des algorithmes de partitionnement (*clustering* algorithms), pour estimer l'exposant secret, ont fait leur apparition dans la littérature [62, 63, 61, 7].

L'utilisation des algorithmes de *clustering* a été proposée dans la littérature des attaques par canaux cachés par L. Batina et al. dans [62]. Dans leur article, ils proposent de remplacer le calcul de la différence des moyennes dans une DPA par l'utilisation des algorithmes de *clustering*. Leur attaque est alors appelée *Differential Cluster Analysis* (DCA).

Dans [63], ce travail est étendu en considérant l'analyse en composante principale (*Principal Component Analysis* (PCA)) pour déterminer les PoIs. Tandis que K. Lemke-Rust et al. proposent dans [64] une attaque verticale supervisée contre des implémentations masquées d'algorithmes symétriques en utilisant l'algorithme *espérance-maximisation*.

algorithme 14 Algorithme k-moyennes non supervisé [10]

Entrées: échantillons $\mathbf{x}_i$, $1 \leq i \leq n$, nombre de classes souhaitées k
Sorties: la classification $\{c_1, \cdots, c_k\}$

1: Initialisation aléatoire de k centres notés $\mathbf{m}_i, \cdots, \mathbf{m}_k$
2: **répéter**
3: assigner chaque observation à la partition la plus proche :
$c_i^{(t)} = \left\{\mathbf{x}_j : \left\|\mathbf{x}_j - \mathbf{m}_i^{(t)}\right\| \leq \left\|\mathbf{x}_j - \mathbf{m}_{i^*}^{(t)}\right\| \forall\ i^* = 1, \ldots, k\right\}$
4: mettre à jour la moyenne de chaque classe :
$\mathbf{m}_i^{(t+1)} = \frac{1}{|c_i^{(t)}|} \sum_{\mathbf{x}_j \in c_i^{(t)}} \mathbf{x}_j$
5: **jusqu'à** $\mathbf{m}_i^{(t+1)} = \mathbf{m}_i^{(t)}$
6: **retourner** $\{c_1, \cdots, c_k\}$

1.5.5.1 Attaque de J. Heyszl et al. [61]

J. Heyszl et al. ont été les premiers à proposer [61] l'utilisation d'algorithmes de partitionnement non supervisés, pour casser les algorithmes à clé publique. Le principe de leur attaque est d'utiliser l'algorithme des k-moyennes (algorithme 14) [10] pour identifier l'exposant utilisé dans une implémentation de l'échelle de Montgomery (algorithme 7).

L'idée de leur attaque est de découper la trace d'exponentiation en segments de traces correspondant chacun au traitements d'un bits de l'exposant. Ainsi, ils construisent une matrice composée de segments de traces.

Pour identifier les bits de l'exposant, les auteurs placent leurs sondes d'acquisition à plusieurs positions simultanément. Pour chaque emplacement (chaque sonde), ils acquièrent la trace correspondante à l'exécution de l'algorithme 7. Pour améliorer le rapport signal sur bruit (ou SNR), ils combinent les traces obtenues. En d'autres termes, les nouveaux segments sont la concaténation des segments obtenus, pour les différents emplacements de la sonde.

Finalement l'algorithme 14 est appliqué pour classifier ces nouveaux segments, ce qui permet de retrouver tous les bits de l'exposant secret.

Cette attaque sera améliorée l'année suivante par R. Specht et al. dans [65]. Les auteurs proposent de remplacer l'algorithme 14 par l'utilisation de l'algorithme *espérance-maximisation* après la réduction de la dimension des segments en utilisant la PCA.

Cependant, les attaques présentées dans [61, 65] souffrent de l'utilisation de plusieurs sondes pour améliorer le SNR, ce qui est souvent impossible dans la

pratique en raison de la taille réduite des circuits intégrés modernes.

1.5.5.2 Attaque de G. Perin et al. [7]

Une autre attaque utilisant le même principe que les attaques précédentes ([61, 65]) a été proposée par G. Perin et al. [7]. L'avantage de cette nouvelle attaque est l'utilisation d'une seule sonde pour extraire l'information secrète.

L'idée originelle de l'attaque de G. Perin et al. est l'utilisation de deux algorithmes de classification : l'algorithme des *k-moyennes* et l'algorithme des *k-moyennes floues* [10]. Le premier sert à identifier les PoIs et le second sert à déterminer les bits de l'exposant.

La première partie de l'attaque consiste à découper la courbe d'exécution de l'algorithme d'exponentiation modulaire en segments de trace correspondant au traitement d'un bit d'exposant. Ces segments de trace sont ensuite mis dans une matrice P. Pour un exposant de n bits on aura une matrice de taille $n \times l$, où l est la taille des segments de trace obtenus après le découpage.

$$P = \begin{bmatrix} P_1 \\ P_2 \\ \vdots \\ P_n \end{bmatrix} = \begin{bmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,l} \\ p_{2,1} & p_{2,2} & \cdots & p_{2,l} \\ \vdots & \vdots & \ddots & \vdots \\ p_{n,1} & p_{n,2} & \cdots & p_{n,l} \end{bmatrix}$$

Suite à ce découpage, l'algorithme des k-moyennes (algorithme 14) est appliqué sur les colonnes de la matrice P. Pour chaque colonne, l'algorithme de k-moyennes donne une estimation de l'exposant. Ensuite, pour chaque estimation la différence des moyennes, entre l'ensemble contenant les segments de trace correspondant aux bits estimés 0 et l'ensemble contenant ceux estimés à 1, est alors calculée :

$$DoM = \sum_i (P_i)_{d_i=0} \; - \; \sum_i (P_i)_{d_i=1} \tag{1.18}$$

Pour une colonne correspondant à un PoI, cette différence des moyennes comporte des pics, contrairement à une colonne ne véhiculant pas d'information sur l'exposant.

Après l'identification des PoIs, l'étape suivante de l'attaque consiste à appliquer un autre algorithme de classification : l'algorithme des k-moyennes floues, sur la matrice P où sont conservées uniquement que les colonnes constituant des PoIs. Cette étape donne pour chaque colonne une nouvelle estimation de l'exposant.

Finalement, l'attaque s'achève par une prise de décision majoritaire appliquée sur les estimations de l'exposant pour en déduire la valeur de l'exposant privé.

1.5.6 Quelques contremesures existantes aux attaques horizontales

Les attaques horizontales ciblent les algorithmes dans lesquels les données sont masquées (avec un même masque pour chaque donnée). Cependant selon l'attaque, certains masquages restent efficaces.

Par exemple, l'attaque présentée dans [44] peut être contrecarrée en rendant aléatoire l'exposant car l'attaque bien qu'horizontale nécessite plusieurs courbes qui auront des exposants différents. Cependant, le masquage du message ou du modulo n'a pas d'effet sur l'efficacité de cette attaque et la rend même plus facile à mettre en œuvre comme l'ont montré les auteurs.

Pour l'attaque HCA [4], le masquage de l'exposant par $\phi(N)$ ne suffit pas. Comme l'a souligné C.D. Walter dans [3], plus la taille des nombres manipulés est grande ou plus la taille du multiplieur est petite, plus le nombre de segments de trace est important. Ainsi, plus l'exposant secret est long, plus l'algorithme est exposé aux attaques horizontales. Par exemple, dans un système RSA de 2048 bits, si la multiplication est effectuée à l'aide d'un multiplieur 32 bits, on obtient $(2048 \div 32)^2 = 4096$ segments de trace $T^k_{i,j}$ (correspond à la multiplication interne $x_i \cdot y_i \bmod N$ dans la k-ème multiplication) par courbe de consommation. Par contre, comme l'attaque HCA nécessite la connaissance du message, la randomisation du message et/ou du modulo avec des aléas suffisamment longs suffit à contrer l'attaque.

Pour l'attaque ROSETTA [5], comme l'attaquant n'a besoin d'aucune information sur le message ou le modulo et que l'attaque fonctionne en une seule trace, les techniques de masquage ne changent pas à son efficacité. Pour contrer cette attaque, on peut masquer les entrées avant chaque calcul de LIM [5]. Par exemple, quand $x = y$ on choisit deux nombres aléatoires r_1 et r_2, et les entrées deviennent $x^* = x + r_1 \cdot N$ et $y^* = y + r_2 \cdot N$. Ainsi si $r_1 \neq r_2$, LIM(x^*,y^*) ne sera pas un carré même si $x = y$.

1.6 Conclusion

Dans ce chapitre, les différentes techniques d'implémentation de l'exponentiation modulaire ont été abordées. Un panorama de l'évolution des attaques par canaux auxiliaires en fonction de l'évolution des algorithmes d'exponentiation modulaire a également été proposé. Un focus particulier a été donné aux attaques par collision et aux attaques horizontales qui constituent des menaces théoriques sur la sécurité des circuits intégrés.

Bien que les attaques verticales soient largement appliquées dans l'industrie, l'application des attaques plus sophistiquées, comme les attaques par collision ou horizontales, est plus rare car surement plus difficiles à maitriser.

Dans les chapitres qui suivent, la mise en pratique de ce type d'attaques est étudiée. La difficulté du passage de la théorie à la pratique y est abordée, avant la proposition de solutions pratiques et génériques.

Chapitre 2

Attaques par canaux cachés verticales en Pratique

Les attaques par collision sur l'exponentiation modulaire visent à induire et à détecter des opérations identiques pendant le déroulement des calculs modulaires. La détection de collisions est difficile dans un environnement réel. Détecter de manière automatisée des collisions dans la pratique est encore plus difficile. Dans

ce chapitre, nous proposons et comparons quelques méthodes et critères permettant de détecter automatiquement (sans inspection visuelle) la présence de collisions dans des mesures de canaux cachés acquises sur des circuits modernes (et donc sur des acquisitions bruitées). Les résultats de ces travaux ont été publiés à DSD 2015 [66].

2.1 Introduction

Comme abordé dans le chapitre précédent, les attaques par canaux cachés constituent de réelles menaces pesant sur les circuits intégrés sécurisés. De nos jours, tous les fabricants de circuits intégrés ont besoin de vérifier la robustesse de leurs composants aux attaques par canaux cachés. L'objectif est notamment de vérifier si des fuites d'information exploitables subsistent sur les canaux cachés.

Deux approches existent. La première est l'approche pratique qui consiste à évaluer après fabrication s'il y a des fuites d'information. Cependant, cette solution impose une grande maîtrise de l'environnement d'acquisition (oscilloscope, sonde, amplificateur d'instrumentation, etc) ainsi qu'une grande expérience pour la mise en place de cet environnement. Enfin, cette approche est consommatrice de temps.

La seconde approche est une approche basée sur la simulation. Elle repose sur l'hypothèse qu'il est toujours possible, avec plus ou moins de précision, de corréler les instructions exécutées par un microcontrôleur avec la grandeur physique manipulée par l'attaque considérée. Un avantage de cette approche est qu'elle est très simple à automatiser, qu'elle ne requiert aucun équipement et est relativement indépendante de la conception matérielle des circuits. L'inconvénient majeur est qu'il faut interpréter des données de la simulation en modélisant la transformation entre l'exécution simulée et la grandeur physique considérée.

Dans ce contexte, afin de maîtriser les attaques par canaux cachés, mais également de vérifier et comparer leurs efficacités respectives, nous avons développé un environnement de simulation d'attaques par canaux cachés sur les algorithmes cryptographiques et notamment l'exponentiation modulaire. Notre stratégie a donc consisté durant ma thèse à valider les attaques par canaux cachés sur ce simulateur avant de considérer des traces réelles provenant de composants sécurisés modernes.

Ce chapitre est organisé comme suit. La section 2.2 détaille l'environnement de simulation développé. La section 2.3 présente quelques résultats d'attaque par simulations. Dans la section 2.4 des techniques de détection de collision en vue d'automatiser les attaques par collision menées sur des composants modernes sont étudiées.

2.2 Environnement de simulation d'attaque

Cette section détaille l'environnement de simulation d'attaques par canaux cachés sur l'exponentiation modulaire qui a été développé afin d'évaluer la pertinence

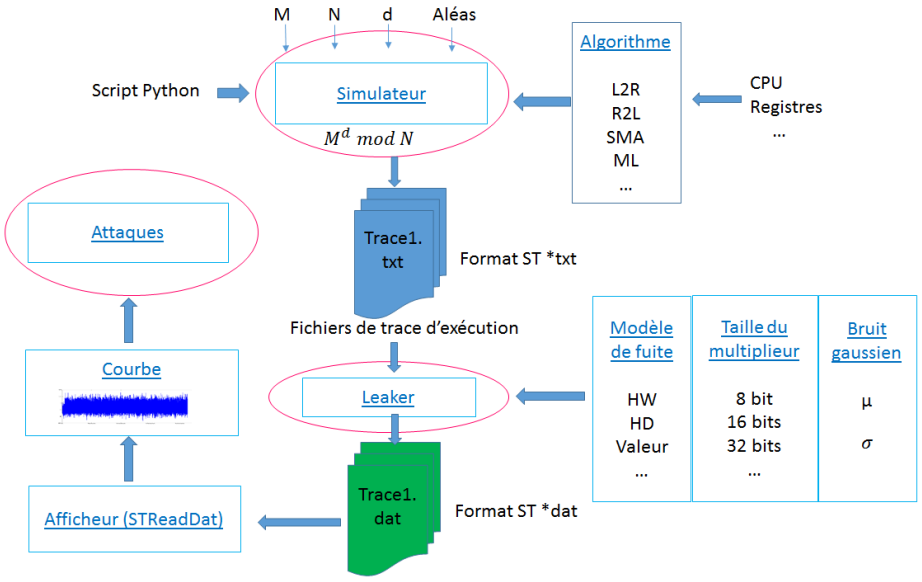


FIGURE 2.1 – Architecture générale de l'environnement de simulation

des attaques et la robustesse des diverses implémentations de l'exponentiation modulaire à celles-ci.

2.2.1 Implémentation des différents algorithmes

Durant les premiers mois de ma thèse, les différentes méthodes d'exponentiation modulaire décrites précédemment (dans le chapitre 1) ont été implémentées dans un logiciel appelé **MultiExp**. Afin d'avoir des simulations (des traces de fuites d'une exponentiation modulaire) se rapprochant du fonctionnement d'un circuit intégré, nous avons implémenté les différentes méthodes de multiplication modulaires dont la multiplication scolaire classique ou LIM (algorithme 3) et la multiplication de Montgomery (algorithme 5), qui sont souvent utilisées dans l'industrie.

Les différentes implémentations ont été décrites en langage *Python*. La bibliothèque *numpy*, destinée à manipuler des matrices, tableaux multidimensionnels ainsi que des fonctions mathématiques opérant sur ces tableaux, a été utilisée. La figure 2.1 présente l'architecture générale de l'environnement de simulation d'attaques par canaux cachés.

Pour construire une trace d'exécution d'un calcul d'exponentiation une fonction a été développée. Elle prend en paramètre le modulo, le message et l'exposant ainsi que les algorithmes d'exponentiation et de multiplication modulaires choisis parmi ceux implémentés dans **MultiExp**. Cette fonction génère des fichiers textes

dans un répertoire. Chaque fichier texte contient le déroulement de l'algorithme d'exponentiation étape par étape, c'est-à-dire toutes les valeurs intermédiaires calculées lors de l'exécution de l'exponentiation modulaire.

La figure 2.2 donne un extrait d'un fichier texte, résultat de cette fonction pour l'algorithme R2L (algorithme 2). Les trois premières lignes correspondent aux données d'entrée (message, modulo, exposant). À partir de la quatrième ligne nous avons les valeurs intermédiaires correspondantes à chaque étape du R2L précédées du temps de latence (l'activité de la CPU), c'est-à-dire le temps entre le traitement de deux opérations modulaires consécutives qui comporte les données relatives à la CPU ; contenues des valeurs des registres impliqués, les résultats de test, etc.

À partir des fichiers textes (*.txt), des fichiers de traces de simulation de l'algorithme d'exponentiation choisi sont générés. Pour cela, une classe développée au sein de *STmicroelectronics*, appelée **Leaker**, est utilisée. Cette classe se charge de collecter les données et la taille du support de fuite (8 bits, 16 bits, 32 bits, etc) depuis les fichiers de traces d'exécution (fichiers textes générés précédemment), selon le modèle de fuite choisit parmi ceux abordés dans la section 1.3.4.1. Un bruit gaussien est finalement ajouté pour plus de réalisme et afin de pouvoir tester le comportement de différentes attaques face à un bruit important.

Pour créer un objet de cette classe, nous avons besoin du répertoire contenant les fichiers des traces (*.txt), du répertoire de sortie où seront enregistrés les résultats, d'un booléen qui permet de rajouter ou non du bruit (bruit gaussien de moyenne μ et écart type σ). Nous avons également besoin du modèle de fuite considéré et les paramètres de modélisation de la fuite comme la taille des données, etc.

Une fois les fichiers *.dat créés, la prochaine étape consiste à utiliser une fonction appelée **StReadDat**. Elle permet de lire des fichiers (*.dat) de traces au format interne de *STMicroelectronics* mais aussi de les afficher en utilisant la bibliothèque *matplotlib* de python.

La figure 2.3 montre une portion de la courbe de simulation obtenue pour l'algorithme R2L (algorithme 2) implémenté avec la multiplication de Montgomery (algorithme 5) de taille 16×16 bits. Le modèle de fuite choisi est le poids de Hamming et le bruit est une fonction gaussienne de moyenne 0 et d'écart type 2. On peut y observer la succession de carrés suivis ou non d'une multiplication selon la valeur du bit de l'exposant traité.

La figure 2.4 reporte une portion d'une trace d'émission électromagnétique (*Électromagnétique* (EM)) obtenue avec un circuit intégré moderne exécutant le même algorithme. On remarque que la courbe obtenue avec le simulateur est similaire à la courbe de consommation d'un vrai circuit intégré. Toutefois le bruit est, dans ce cas, plus important dans la trace mesurée que dans la trace simulée.

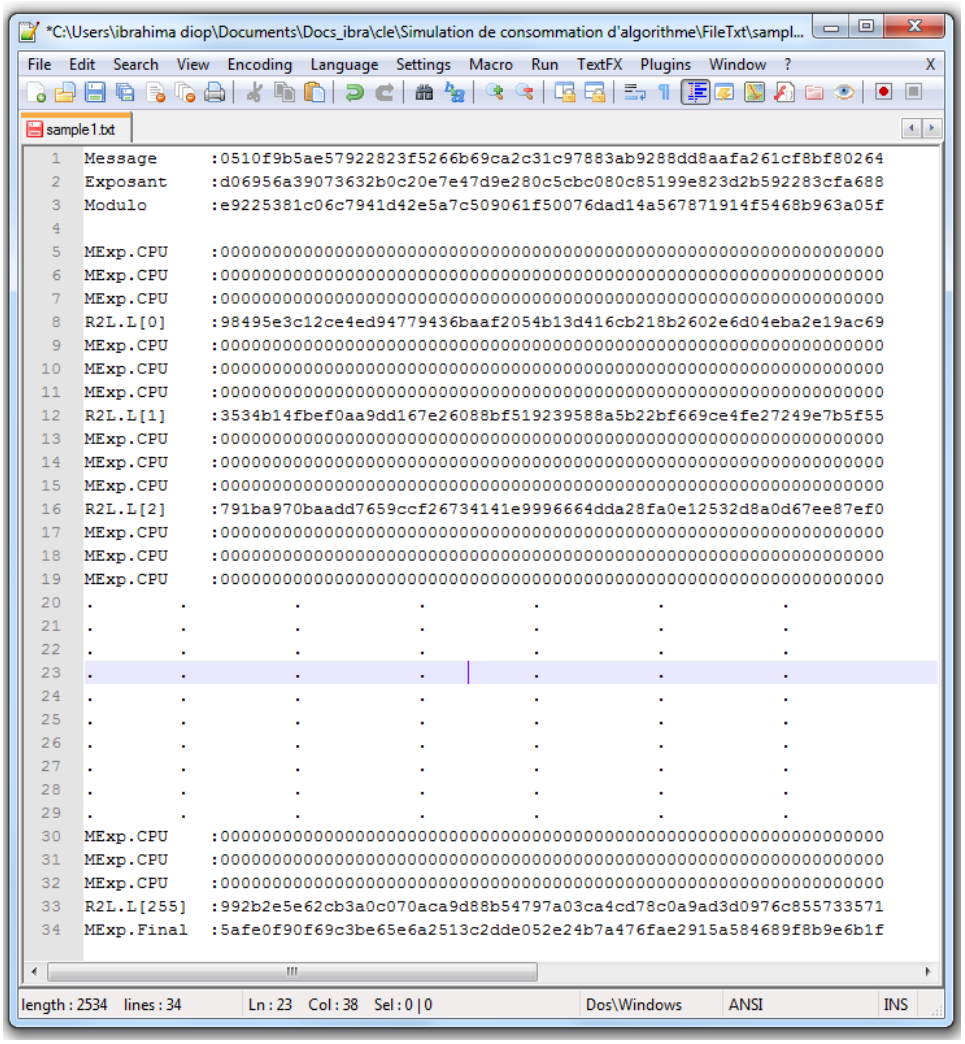


FIGURE 2.2 – Exemple de résultat fourni par l'environnement de simulation dans le cas de l'algorithme R2L (algorithme 2) avec CPU activé

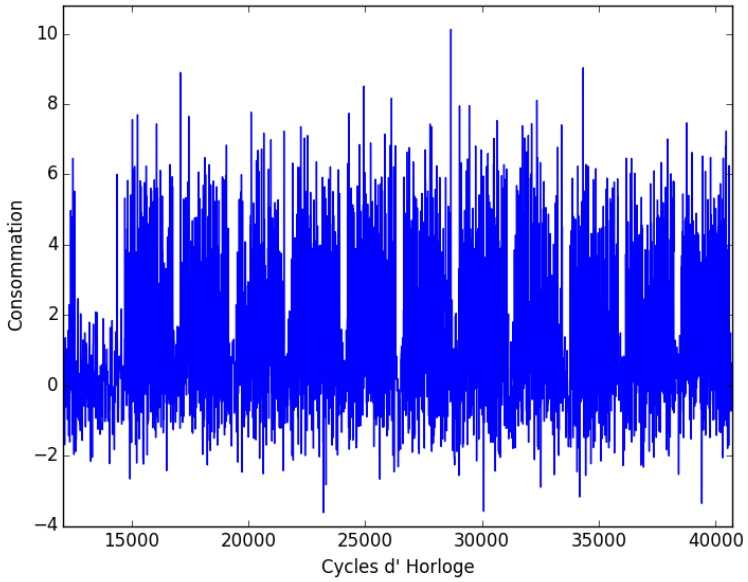


FIGURE 2.3 – Une partie de la trace de simulation de l'algorithme R2L (algorithme 2).

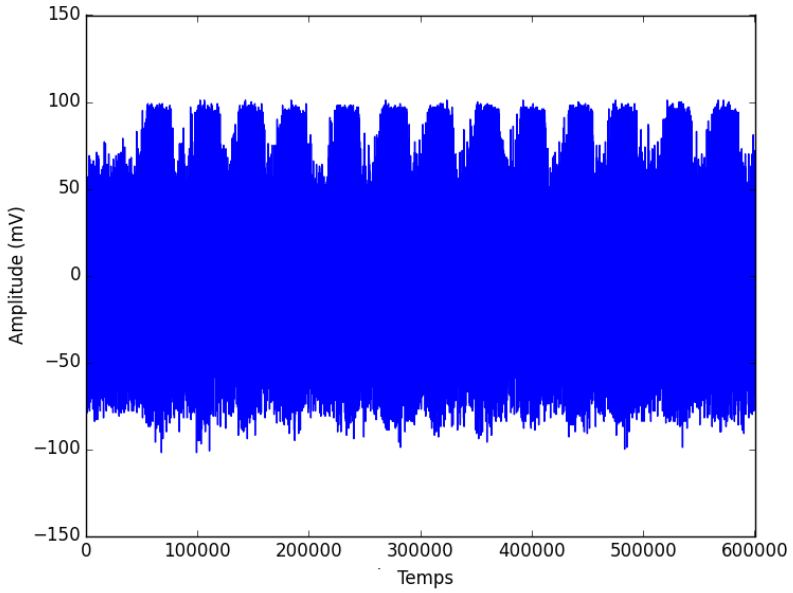


FIGURE 2.4 – Une partie d'une trace EM mesurée sur un vrai composant exécutant l'algorithme R2L (algorithme 2).

2.3 Évaluation de quelques attaques par la simulation

Dans cette section l'efficacité de quelques attaques par canaux cachés est illustrée par simulation.

2.3.1 Attaques SPA appliquées sur courbes de simulation

Les différentes méthodes d'implémentation d'une exponentiation modulaire ont été abordées dans le chapitre 1. Les algorithmes d'exponentiation modulaire classiques comme l'algorithme R2L ou L2R font parfois appel à des fonctions différentes les différentes opérations nécessaires. La multiplication de deux variables identiques de grande taille étant exécutée au moyen d'une fonction "carré", tandis que la multiplication de deux variables différentes de grande taille est exécutée au moyen d'une fonction "multiplication". Cette distinction est due au fait qu'il est possible de calculer plus rapidement $x \cdot y$ lorsque $x = y$ que dans le cas contraire, au moyen de la fonction "carré".

Le ratio entre le temps d'exécution de la fonction "carré" et le temps d'exécution de la fonction "multiplication" est généralement inférieur à 1 suivant la taille des nombres considérés et la façon dont la multiplication est exécutée.

Le fait d'utiliser alternativement la fonction "carré" ou la fonction "multiplication", en fonction du type de calcul à effectuer, permet d'optimiser le temps de calcul global de chiffrement, de signature ou de déchiffrement.

Toutefois, l'utilisation de deux fonctions différentes "carré" et "multiplication" conduit à une fuite d'information détectable par une attaque SPA. La fonction "carré" ayant un temps d'exécution plus court que la fonction "multiplication", il est possible de différencier ces deux opérations en observant la courbe de consommation ou le rayonnement électromagnétique du composant.

Les figures 2.5 et 2.6 représentent respectivement une courbe de consommation d'un composant exécutant l'algorithme L2R (algorithme 1) et l'algorithme R2L (algorithme 2). On distingue clairement le profil de consommation de la fonction "carré" et celui de la fonction "multiplication" avec une différence visible de durée des opérations. Cette discontinuité permet à un attaquant de récupérer tous les bits de l'exposant. Un "carré" suivie d'une opération "multiplication" révèle que le bit de l'exposant d est égal à 1 puisque le branchement conditionnel vers l'étape 5 nécessite que la condition $d_i = 1$ soit réalisée. Inversement, une opération "carré" suivi d'une autre opération "carré" (étape 3 suivie d'une autre étape 3) révèle que le bit de l'exposant est égal à 0. Les bits de l'exposant d peuvent ainsi être découverts les uns après les autres par une simple observation de la courbe de consommation.

Pour supprimer cette fuite d'information, les étapes 3 à 5 de l'algorithme 1 (ou 5 à 7 de l'algorithme 2) peuvent être effectuées au moyen de la fonction "multiplication" uniquement, sans utiliser la fonction "carré". Toutefois, une analyse plus fine de la courbe de consommation permet dans ce cas de distinguer l'étape 3 de l'étape 5 (ou de distinguer l'étape 5 de l'étape 7) car les algorithmes 1 et 2 ne sont pas réguliers. En effet, dans ce cas, le temps s'écoulant entre deux multiplica-

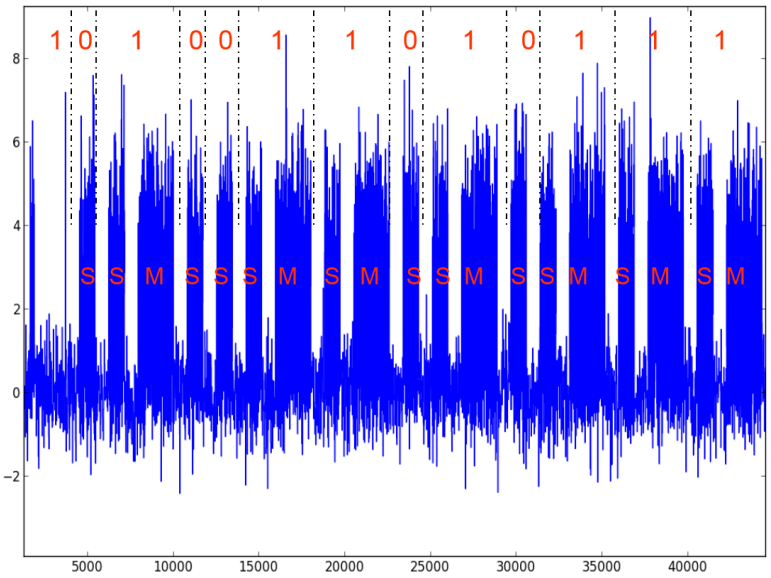


FIGURE 2.5 – Attaque SPA par simulation d'un circuit exécutant l'algorithme L2R (algorithme 1)

tions successives n'est pas le même lorsque les deux multiplications correspondent à l'exécution successive de deux étapes 3 (bit de l'exposant égal à 0) ou correspondent à l'exécution d'une étape 3 suivie d'une étape 5 (bit de l'exposant égal à 1). Un attaquant peut ainsi se concentrer sur la partie de la courbe de consommation s'étendant entre les multiplications et observe une dissymétrie temporelle révélatrice du branchement conditionnel et donc de la valeur du bit de l'exposant.

La figure 2.7 illustre le cas où la même fonction est utilisée pour réaliser les multiplications et les carrés. Les flèches rouges montrent les passages d'un traitement d'un bit à un autre. Ainsi, lorsqu'un traitement d'un bit nécessite deux opérations le bit est égal à 1. Dans le cas contraire, le bit est égal à 0.

L'exponentiation carré-et-multiplication-systématique (algorithme 6) permet de supprimer cette fuite d'information. Cet algorithme effectue une multiplication factice, insérée après un carré lorsque le bit de l'exposant d est égal à 0. On suppose donc que l'ajout de multiplications factices ne produit pas de fuite détectable par analyse SPA, car on ne peut pas distinguer si la condition est vraie ou fausse, puisqu'une multiplication est toujours exécutée.

La figure 2.8 montre une courbe générée par notre simulateur exécutant l'algorithme 6. On constate une régularité des motifs correspondant à une succession d'étapes 3 à 4, qui protège l'algorithme contre une attaque SPA.

Une autre solution pour se prémunir des attaques SPA est d'utiliser l'échelle de

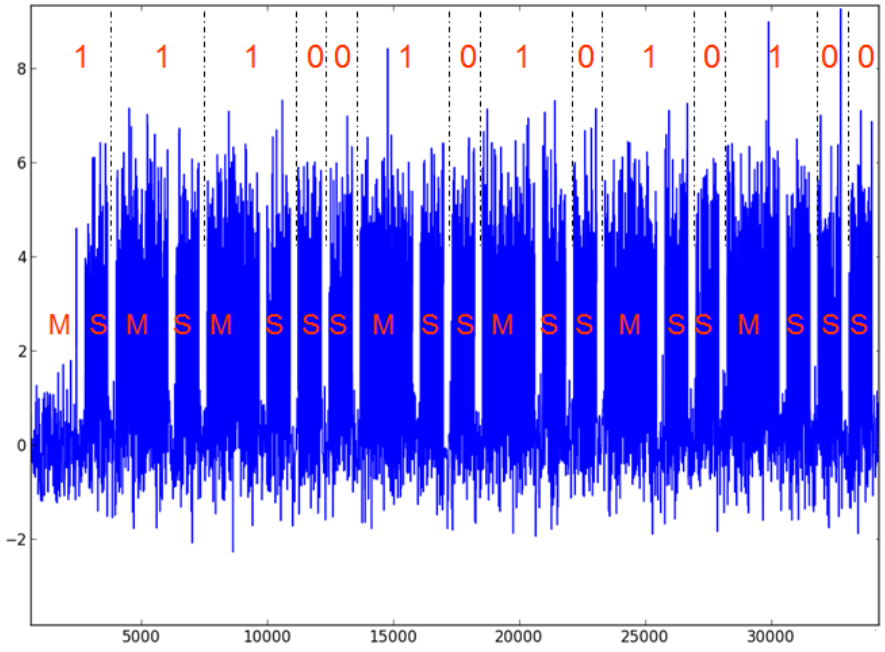


FIGURE 2.6 – Attaque SPA par simulation d'un circuit exécutant l'algorithme R2L (algorithme 2).

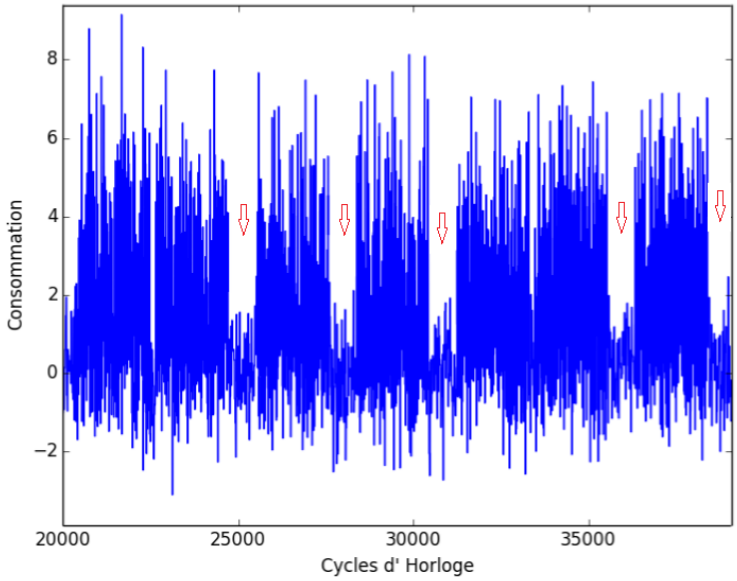


FIGURE 2.7 – Attaque SPA par simulation d'un circuit exécutant l'algorithme L2R (algorithme 1) lorsque une même fonction est utilisée pour réaliser les multiplications et les carrés.

Montgomery (l'algorithme 7). En effet, contrairement à l'exponentiation carré-et-multiplication-systématique (algorithme 6) qui utilise des fausses multiplications (dans le cas d'un bit égal à 0), il utilise plutôt des calculs non redondants. Ces calculs permettent d'obtenir comme avec l'algorithme 6, une succession régulière d'opérations (voir la figure 2.9).

Un autre moyen de procéder est d'utiliser l'exponentiation atomique (algorithme 8). Cet algorithme est plus régulier que les méthodes décrites précédemment. En effet, l'algorithme 8 n'effectue, quel que soit le bit de l'exposant, que des multiplications. Certaines multiplications sont des multiplications de variables différentes et d'autres sont des multiplications de variables identiques. La figure 2.10 montre un zoom sur une trace de simulation d'un circuit exécutant l'algorithme 8. On peut y vérifier que toutes les opérations ont la même durée.

2.3.2 Attaques par collision appliquées sur courbes de simulation

Comme cela a été expliqué dans le chapitre 1 (section 1.4), les attaques par collision sont sensées défaire de nombreux algorithmes d'exponentiation modulaires intégrés aujourd'hui dans les cartes à puces. Notre simulateur nous a permis de vérifier cette assertion et nous a aussi permis de nous convaincre que ce type d'attaque constitue une menace importante même pour les algorithmes les plus robustes.

En effet, de nombreux tests ont été effectués avec notre simulateur avec di-

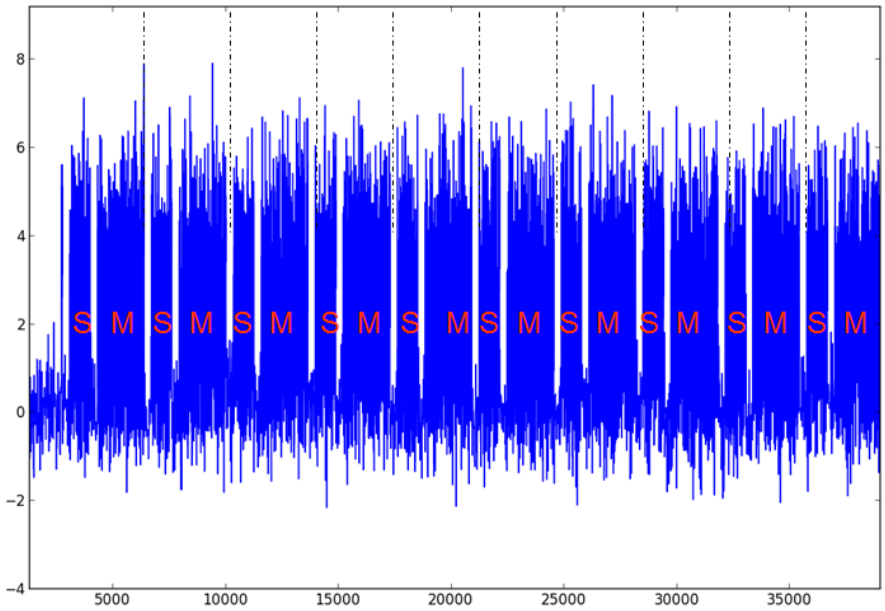


FIGURE 2.8 – Zoom sur une trace (générée par notre outil) d'un circuit exécutant l'exponentiation carré-et-multiplication-systématique (algorithme 6).

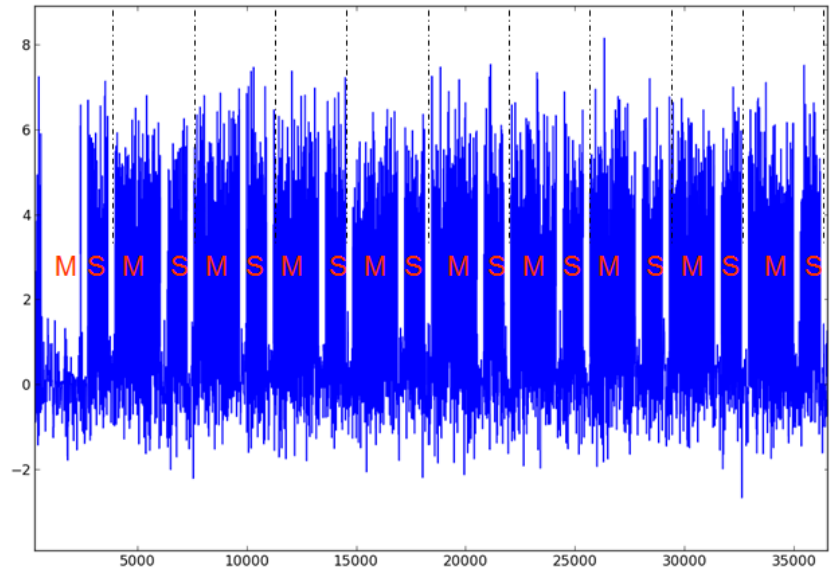


FIGURE 2.9 – Zoom sur une trace (générée par notre outil) d'un circuit exécutant l'échelle de Montgomery (algorithme 7).

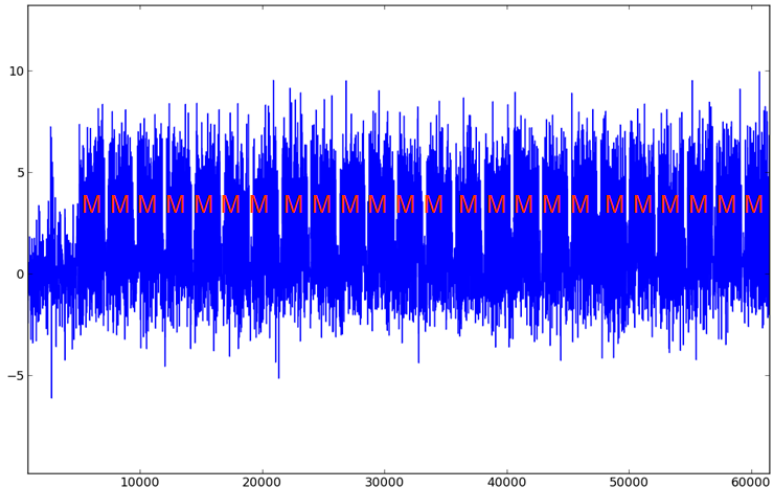


FIGURE 2.10 – Zoom sur une trace (générée par notre outil) d'un circuit exécutant l'exponentiation atomique (algorithme 8).

vers niveaux de bruit, diverses méthodes de multiplications modulaires, différents modèles de fuite et à chaque fois, même sur des algorithmes tels que l'échelle de Montgomery (algorithme 7), ce type d'attaque nous a permis de récupérer la totalité ou une grande partie de l'exposant privé.

En guise d'illustration, la figure 2.11 reporte le résultat d'une attaque par doublement [1] effectuée sur deux courbes obtenues avec le simulateur. Les entrées respectives des deux exponentiations sont m et m^2. Le bruit est une fonction gaussienne de moyenne nulle et de variance égale à 1. La figure du haut représente la courbe obtenue avec m^2 comme entrée, tandis que celle du bas représente la différence point par point des deux courbes. Les parties où des collisions se produisent sont plus proches de zéros et sont facilement distinguables sur la courbe différentielle (différence des deux courbes). Ainsi, d'après la figure 2.11, l'exposant d est égal à 101001011101.

La même attaque, avec le même paramétrage et les mêmes données, a été appliquée à l'exponentiation carré-et-multiplication-systématique (algorithme 6). La figure 2.12 donne le résultat obtenu. Des collisions se produisent entre les carrés de la trace correspondante à l'entrée m et celle (décalée d'une opération) correspondant à l'entrée m^2. Des collisions discernables se produisent lorsque le bit de l'exposant traité est égal à 0.

Pour attaquer par simulation l'échelle de Montgomery (l'algorithme 7), l'attaque par doublement ciblant l'échelle de Montgomery [8] a été appliquée. À la différence de l'attaque par doublement originale [1], cette attaque permet de détecter des bits consécutifs égaux.

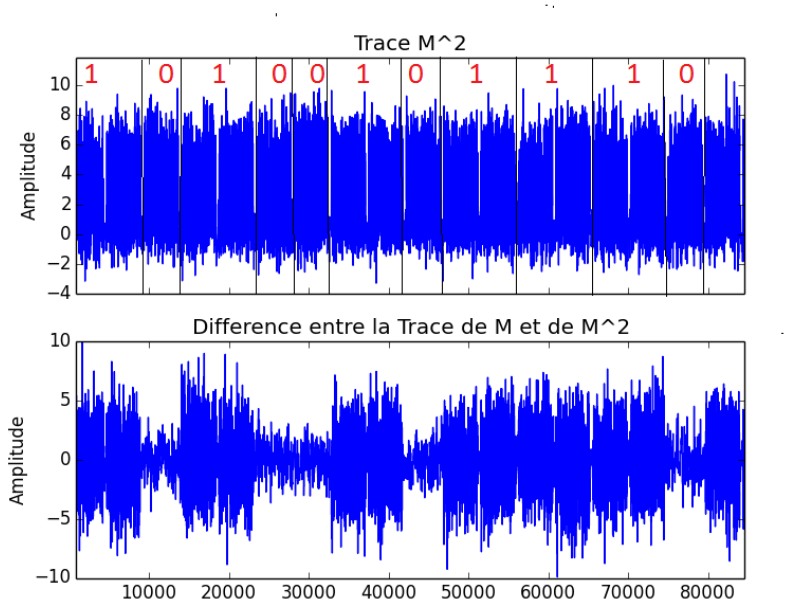


FIGURE 2.11 – Résultat de l'attaque par doublement effectuée par simulation sur l'algorithme L2R (algorithme 1).

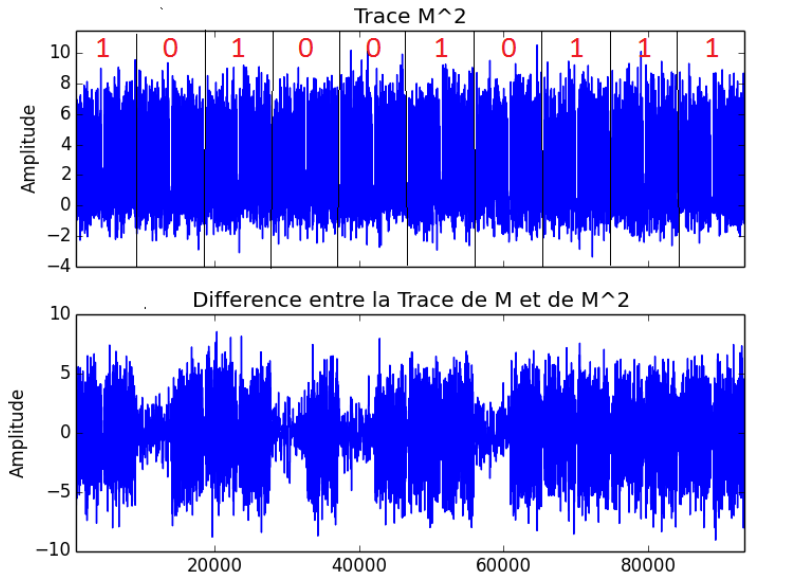


FIGURE 2.12 – Résultat de l'attaque par doublement réalisée par simulation sur l'exponentiation carré-et-multiplication-systématique (algorithme 6).

La figure 2.13 donne le résultat de cette attaque. Une analyse de la courbe de différence (courbe du bas) entre la courbe obtenue avec m comme entrée et celle avec m^2, montre que les bits aux postions 4 et 5 sont égaux, de même que les bits aux positions 8, 9 et 10.

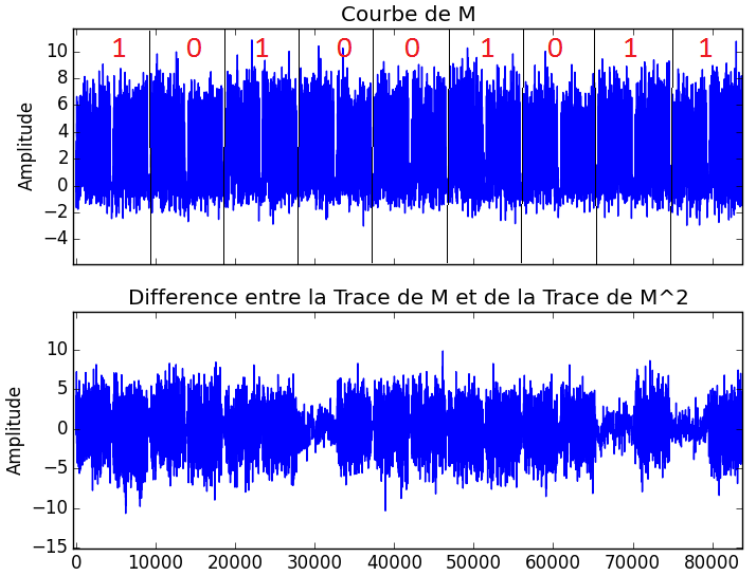


FIGURE 2.13 – Résultat de l'attaque par doublement effectuée par simulation sur une implémentation de l'échelle de Montgomery.

L'attaque présentée par S. Yen et al. [2] utilise $-1 \bmod N$ comme entrée de l'exponentiation carré-et-multiplication-systématique (algorithme 6). La figure 2.14 montre le résultat de cette attaque effectuée par simulation sur un circuit exécutant l'algorithme 6. Nous pouvons observer à l'œil nu les différentes étapes 3 de l'algorithme 6 correspondant au calcul de $R_0 = 1 * 1$. Ces calculs, comme nous l'avons déjà évoqué, correspondent aux cas où le bit précédent est égal à 0. Ces opérations consomment moins d'énergie que les autres opérations et sont facilement repérables sur une trace. Ainsi l'attaquant permet de récupérer la valeur de l'exposant utilisé en inspectant la trace obtenue avec $-1 \bmod N$ comme entrée.

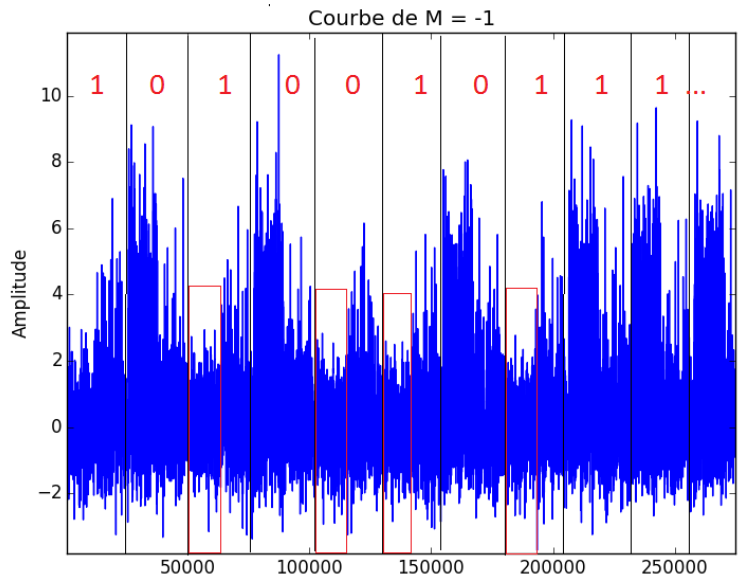


FIGURE 2.14 – Résultat de l'attaque de S. Yen et al. effectuée par simulation avec $m = -1$ comme entrée sur l'exponentiation carré-et-multiplication-systématique (algorithme 6) par simulation.

2.4 Attaques par collision en pratique

Les paragraphes précédents ont montré qu'un adversaire peut retrouver l'exposant privé avec uniquement deux traces de simulations, et ce de façon visuelle. Qu'en est-il sur de véritables traces acquises au-dessus d'un composant moderne ?

Avant de répondre à cette question nous allons d'abord présenter la plateforme d'acquisition utilisée dans le reste de ce chapitre.

2.4.1 Plateforme d'acquisition

Ce paragraphe présente la plateforme utilisée pour l'acquisition de courbes sur plusieurs composants modernes. Pour des raisons de confidentialité les références des composants utilisés dans le cadre de cette thèse ne sont pas données. Tous ces composants sont ouverts, c'est-à-dire que les routines cryptographiques peuvent être appelées avec des données choisies. Pour tester l'efficacité des attaques par collision mais aussi des critères de détection de collision, de nombreux tests ont été effectués en utilisant la plateforme d'acquisition représentée sur la figure 2.15. Durant nos analyses, un oscilloscope type LeCroy TELEDYNE HDO4096 a été utilisé pour collecter les traces de consommation de courant. Les traces ont été acquises avec un taux d'échantillonnage de 2.5 GS/s (respectivement 500 MS/s) donnant des segments de trace (opérations) de 24.000 points (ou respectivement

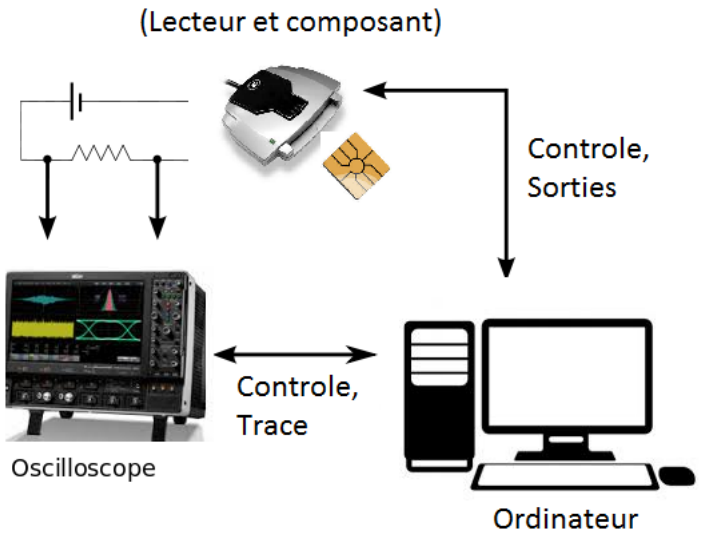


FIGURE 2.15 – Plateforme d'acquisition utilisée pour les analyses

4850 points). Ce choix a été fait pour montrer plus loin l'impact des méthodes de prétraitement choisies en fonction du taux d'échantillonnage.

2.4.2 Problèmes rencontrés dans la pratique

De nos jours, les concepteurs de circuits implémentent de plus en plus de contremesures environnementales. Ces contremesures, dont l'objectif est de dégrader les fuites d'information pouvant être exploitées par l'attaquant, s'avèrent pertinentes pour rendre l'ensemble des attaques par canaux cachés plus difficiles à mettre en œuvre. Les principales techniques utilisées sont présentées ci-après.

- La désynchronisation du processus [67, 68] : elle consiste à déplacer aléatoirement les instructions du programme. Ainsi, d'une exécution d'algorithme à une autre, un même événement ne se produit pas au même instant, ce qui peut entrainer l'absence de collision entre les traces lors d'une attaque par doublement ou encore une absence de PoIs lors d'une attaque horizontale. Ce type de contremesure est généralement assez efficace. Cependant, en effectuant une phase très fine de resynchronisation sur les courbes, les événements peuvent être réalignés et ainsi permettre la l'application avec succès des attaques par canaux cachés.

- Ajout de bruit : Elle consiste à brouiller la consommation électrique du composant en rendant l'extraction des fuites d'information difficile [69]. Par exemple, un circuit filtrant peut être mis en place lors de la fabrication d'un

composant. Ce qui aura pour effet de lisser les variations de consommation de courant d'un composant.

Une autre façon de rajouter du bruit dans la consommation électrique d'un composant, est l'activation de diverses sources de consommation du composant qui ne sont pas liées à l'opération ou alors ont un lien rendu aléatoire. Ce type de contremesures peut dans certains cas être contourné en moyennant les courbes de consommation mesurées ou en observant le rayonnement électromagnétique [25] au lieu de la consommation électrique.

- La technologie duale ou *dual-rail logic* : elle vise à rendre la consommation indépendante des données manipulées par une conception équilibrée au niveau des portes logiques [70, 71, 72] en doublant tous les bus du composant. L'idée est que lorsqu'un signal prend une valeur, le second signal prend la valeur inverse (complément à un). Ainsi, la consommation dynamique de l'ensemble est toujours la même quelles que soient les données. Toutefois, en pratique, de petites variations de consommation peuvent être observées [73]. Notamment, en observant les fuites émises par rayonnement électromagnétique ; des déséquilibres peuvent être en effet distingués par la manière d'observer le signal (position de la sonde de mesure par exemple).

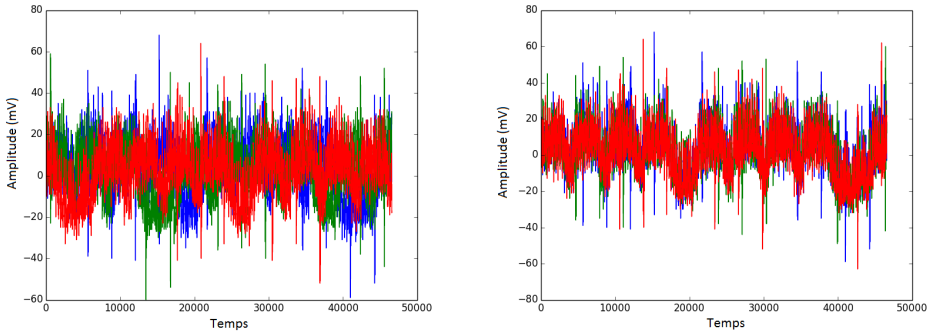
Ces différentes contremesures rendent l'application des attaques par collision, et plus généralement des attaques par canaux cachés, difficile. En effet, la désynchronisation causée par certaines de ces contremesures induit une phase, souvent délicate, de resynchronisation. Un autre problème, causé par ces contremesures, est la présence élevée de bruits de mesure qui caractérise les courbes obtenues sur des composants modernes.

Pour résoudre ces problèmes dans la pratique, plusieurs techniques ont été proposées dans la littérature pour améliorer l'efficacité des attaques par canaux cachés en présence de bruits et de désynchronisation.

Concernant la resynchronisation, dans [74], N. Homma et al. ont proposé une méthode de reconnaissance de motifs basée sur la corrélation de phase (la fonction *Phase-Only Correlation* (POC)). Cette méthode est détaillée dans le chapitre 4.

A titre d'illustration, le résultat de l'application de cette méthode sur des mesures de consommation de courant d'une exponentiation est présentée sur la figure 2.16. La figure 2.16a reporte un ensemble de trois courbes collectées audessus d'un composant moderne exécutant une exponentiation modulaire avant application de la procédure de resynchronisation. Tandis que la figure 2.16b montre ces mêmes courbes après application de la méthode de resynchronisation basée sur la fonction POC. Comme nous pouvons l'observer, même si les courbes obtenues au-dessus d'un composant moderne sont bruitées, la fonction POC semble visuellement efficace pour réaligner des traces de manière globale.

Toutes les attaques sur composants réels présentées dans le reste de ce chapitre ont été réalisées en utilisant la méthode basée sur la POC comme méthode de resynchronisation.



(a) 3 Courbes brutes avant application de la resynchronisation

(b) 3 Courbes après resynchronisation par la méthode basée sur la fonction POC

FIGURE 2.16 – Illustration de la méthode de resynchronisation basée sur la fonction POC

2.4.3 Techniques de détection de collisions

La détection de collisions sur des traces mesurées au-dessus de composants modernes est une tâche difficile. Des méthodes de détection de collisions ont été proposées dans la littérature.

Toutefois, elles n'ont été appliquées que sur des traces de simulation ou sur des composants FPGA [1, 2, 9, 75, 76]. En outre, elles requièrent une inspection visuelle empêchant l'automatisation des attaques. Cette inspection visuelle n'est possible que lorsque le bruit est peu élevé. Cependant, en présence de composants modernes le bruit est souvent important dans les mesures. Ceci rend impossible cette solution dans la pratique. D'autre part une inspection visuelle sur des courbes d'exponentiation, qui sont très longues, est une tâche fastidieuse. Détecter des collisions de manière automatique sur de véritables composants constitue donc un véritable challenge.

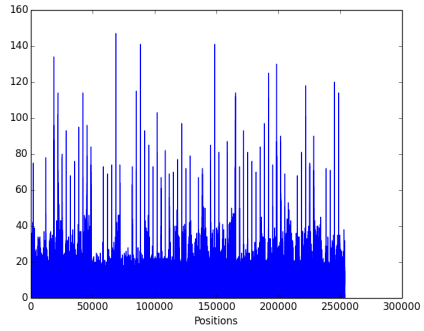
Malheureusement, il n'existe que peu de travaux dans cette direction. Récemment, C. Aidong et al. ont proposé dans [75] une solution automatisée de détection de collision, tandis que G. Perin et al. [76] ont défini un critère qui facilite la détection visuelle de collision dans la pratique.

Avant d'aborder ces techniques, nous allons rappeler une des plus utilisées dans la littérature ; à savoir la différence des courbes point à point.

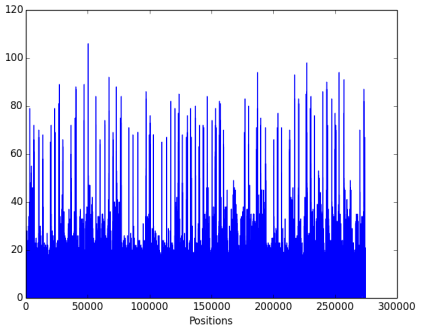
2.4.3.1 La différence des traces comme distingueur

Cette méthode, souvent utilisée dans les premières publications relatives aux attaques par collision [1, 2, 9], consiste à calculer la différence point à point de deux segments de trace. Ainsi, une différence proche de zéro (du vecteur nul) est considérée comme dévoilant une collision.

Pour cela, des couples de traces sont collectées en fonction de l'attaque consi-



(a) Valeurs absolues des différences de traces point à point collectées pour mener l'attaque par doublement sur le premier octet de l'exposant.



(b) Valeurs absolues des différences de traces point à point collectées pour mener l'attaque de S. Yen et al. sur le premier octet de l'exposant.

FIGURE 2.17 – Résultats de l'attaque par doublement [1] et celle de S. Yen et al. [2] sur les traces resynchronisées avec la fonction POC.

dérée. Par exemple, des traces correspondant aux exponentiations avec comme messages d'entrées m et m^2 sont collectées pour l'attaque par doublement [1] ou encore des traces correspondant aux entrées m et $-m$ pour celle de S. Yen et al. [2].

À titre d'illustration, les résultats sur un de nos circuits test pour le premier octet (égal à $0x8A = 10001010_2$) de l'exposant sont représentés sur la figure 2.17. Après la resynchronisation des traces utilisant la méthode basée sur la fonction POC, la différence point par point est directement calculée. Il peut être remarqué sur les figures 2.17a et 2.17b que même si la différence est petite sur certains intervalles de temps (du fait des moments où le composant fait des calculs qui ne sont pas liés au secret), il est difficile de détecter, avec une grande certitude, les parties correspondant à des collisions et des non-collisions.

2.4.3.2 Technique de C. Aidong et al. [75]

Tous les algorithmes de détection de collision nécessitent une méthode qui mesure la présence ou non d'une collision. La réussite de ces méthodes dépend essentiellement du bruit.

C. Aidong et al. [75] ont proposé une solution automatisée de détection de collision basée sur l'algorithme des k-moyennes (algorithme 14). Leur idée est d'appliquer directement l'algorithme des k-moyennes sur une matrice construite en fonction de l'attaque appliquée. Par exemple, pour l'attaque de S. Yen et al., chaque ligne de la matrice représente la différence (point à point) entre les segments de la trace de m et ceux de $-m$ à la même position. Cependant, leur solution nécessite la définition d'un seuil avant même l'application de l'algorithme de classification. Cette définition de seuil est faite par une inspection visuelle. Ce qui constitue donc une limitation de leur proposition dans la pratique.

2.4.3.3 Critères de détection de collision

Récemment G. Perin et al. ont proposé un critère de détection de collision dans [76]. Ce critère appelé *Perin Collision Detection Criteria* (PCDC) est reporté équation 2.1. Il se base sur une idée simple : si une collision se produit, la valeur du PCDC doit être grande. Dans le cas contraire, la valeur du PCDC est petite.

Le critère PCDC entre deux traces T_i et T_j est défini comme suit :

$$\mathrm{PCDC}(T_i, T_j) = \frac{\sigma_{(T_i)}}{\sigma_{(T_i - T_j)}}. \tag{2.1}$$

Dans cette expression, $\sigma_{(T_i)}$ est l'écart-type de la trace T_i et $\sigma_{(T_i - T_j)}$ est l'écart type de la différence entre les traces ou segments de traces T_i et T_j.

Si les résultats reportés dans [76] sont convaincants (facilitant la détection visuelle), démontrant l'intérêt de ce critère, il convient de noter que ces résultats ont été obtenus avec des traces de fuites moyennées. Ceci est souvent impossible en raison des contremesures environnementales ou même des contremesures de masquage. De plus le PCDC n'est pas borné et donc une inspection visuelle est toujours nécessaire pour décider d'une apparition de collision ou non. Ces derniers points constituent des limitations importantes du PCDC. Pour contourner ces problèmes, en suivant l'intuition des auteurs de [76], nous avons dérivé de l'équation 2.1 un nouveau critère appelé *Bounded Collision Detection Criterion* (BCDC), dont les valeurs sont comprises dans l'intervalle]0, 1] et donc bornées. Ce critère est défini par l'équation 2.2 :

$$\mathrm{BCDC}(T_i, T_j) = \frac{1}{\sqrt{2}} \cdot \frac{\sigma_{(T_i - T_j)}}{\sigma_{(T_i)}} \tag{2.2}$$

En d'autres termes, le critère BCDC est défini comme le rapport de l'écart type de la trace différentielle et de l'écart type d'une des traces. Pour avoir un critère borné entre 0 et 1, il est normalisé par $\frac{1}{\sqrt{2}}$.

2.4.3.3.1 Propriétés du BCDC

Soit T_i (respectivement T_j) une mesure, collectée avec un oscilloscope et représentant un signal $S(t)^i$ (respectivement un $S(t)^j$). Nous avons les propriétés suivantes pour le critère BCDC :

1. $0 < \mathrm{BCDC}(T_i, T_j) \leq 1$.

2. En cas de collision, $\mathrm{BCDC}(T_i, T_j) \to 0$, quand le rapport signal sur bruit est élevé.

3. En cas de non-collision, $\mathrm{BCDC}(T_i, T_j) \to 1$.

2.4.3.3.2 Démonstration

Soient $T_i = [t_i^1, \cdots, t_i^q]$ et $T_j = [t_j^1, \cdots, t_j^q]$ deux vecteurs de tailles q et d'écart-type respectifs $\sigma_{(T_i)}$ et $\sigma_{(T_j)}$. En supposant que tous les échantillons t_i^k (respectivement t_j^k) de cette mesure (avec $k \in [1, q]$) ont pour expression $t_i^k = s_i^k + \eta_i^k$ (respectivement $t_j^k = s_j^k + \eta_j^k$), c'est à dire sont la somme :

- d'une valeur déterministe s_i^k (respectivement s_j^k) formant un échantillon du signal $S(t)$ émis par le circuit sous test ou DUT lors de son fonctionnement,
- et d'une valeur aléatoire η_i^k (respectivement η_j^k), représentant le bruit, tirée dans une distribution normale de valeur moyenne nulle et d'écart-type inconnu σ_N,

il est alors possible d'exprimer la différence entre T_i et T_j :

$$\Delta T = T_i - T_j = [\delta_{t_1}, ..., \delta_{t_q}] \tag{2.3}$$

avec

$$\delta_{t_k} = t_i^k - t_j^k = s_i^k + \eta_i^k - (s_j^k + \eta_j^k). \tag{2.4}$$

En considérant alors les termes de l'équation 2.4, il apparaît que ΔT est une réalisation d'une variable aléatoire dont l'écart-type est égal à :

$$\sigma_{(T_i - T_j)} = \sqrt{\sigma^2([s_i^1 - s_j^1, ..., s_i^q - s_j^q]) + \sigma^2([\eta_i^1 - \eta_j^1, ..., \eta_i^q - \eta_j^q])}, \tag{2.5}$$

expression qui peut être ré-écrite comme suit :

$$\sigma_{(T_i - T_j)} = \sqrt{2 \cdot \sigma_S^2 + 2 \cdot \sigma_N^2} = \sqrt{2} \cdot \sqrt{\sigma_S^2 + \sigma_N^2} \tag{2.6}$$

Cette équation nous permet de démontrer les propriétés 1, 2 et 3.

1. À partir de l'équation 2.5, on peut alors exprimer la valeur asymptotique du BCDC dans le cas où T_i et T_j sont deux mesures du même signal, i.e. lorsqu'il y a collision. En effet, dans ce cas, parce que $s_i^k = s_j^k$ pour tout $k \in [1, q]$, et donc parce que $\sigma_{(T_i - T_j)} = \sqrt{2} \cdot \sqrt{\sigma_N^2}$, l'équation 2.2 devient :

$$BCDC(T_i, T_j) = \frac{1}{\sqrt{2}} \frac{\sigma(\Delta T)}{\sigma(T_i)} = \frac{1}{\sqrt{2}} \frac{\sqrt{2} \cdot \sigma_N}{\sqrt{\sigma_S^2 + \sigma_N^2}} = \frac{1}{\sqrt{1 + \frac{\sigma_s^2}{\sigma_N^2}}}, \tag{2.7}$$

   Cette expression est toujours compris entre 0 et 1.

2. Elle tend vers 0 (mais n'est jamais égale à 0) lorsque le rapport signal à bruit (${\sigma_S}^2/{\sigma_N}^2$) est élevé ($\sigma_S >> \sigma_N$).

3. Dans le cas contraire, correspondant au cas où T_i et T_j ne sont pas des mesures du même signal (non-collision), l'expression du BCDC est :

$$BCDC(T_i, T_j) = \frac{1}{\sqrt{2}} \cdot \frac{\sqrt{2}.\sqrt{\sigma_s^2 + \sigma_N^2}}{\sqrt{\sigma_s^2 + \sigma_N^2}} = 1 \tag{2.8}$$

2.4.4 Seuil du BCDC

L'interprétation des valeurs du BCDC (équation 2.2) peut sembler difficile, surtout si l'on considère que les caractéristiques de sa distribution ne sont pas connues (ce point est en cours d'étude). Cependant, en pratique, nous pouvons nous fier à certaines observations spécifiques.

D'une part, dans tous les calculs précédents il n'est pas tenu compte des autres activités électriques du circuit. Il est donc possible d'observer des valeurs supérieures à 1 dans la pratique. Ceci peut arriver même en l'absence d'activités supplémentaires du circuit car chaque couple de segments de trace donne une estimation de $\sigma(T_i)$ et $\sigma(T_j)$ et donc du BCDC. Le critère BCDC doit donc être considéré comme asymptotiquement borné entre 0 et 1.

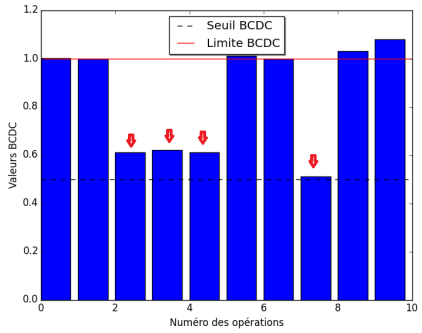
D'autre part, comme indiqué par l'équation 2.7, dès que σ_S^2 devient supérieur à trois fois σ_N^2 (la variance du signal devient supérieure à celle du bruit), les valeurs du BCDC deviennent inférieures à $1/2$ en cas de collision. Ainsi, $1/2$ apparaît comme un seuil de confiance pour distinguer les collisions. Cette valeur est donc considérée dans le reste de ce chapitre comme le seuil séparant l'occurrence (classe c_1) ou non (classe c_2) de collisions. En dessous $1/2$, le BCDC indique qu'une collision s'est produite, sinon qu'il n'y a pas de collision entre les deux traces.

2.4.4.1 BCDC comme distingueur

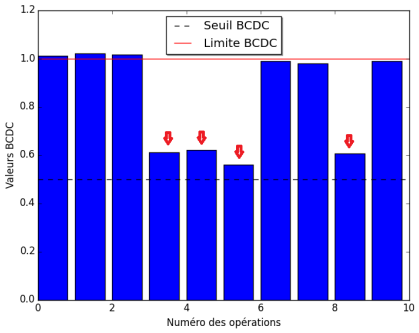
Pour évaluer l'efficacité de notre critère de détection, le BCDC a été appliqué sur les mêmes traces brutes utilisées dans la section 2.4.3.1. Les figures 2.18a et 2.18b reportent les résultats obtenus. Elles sont à comparer avec les figures 2.17a et 2.17b Les flèches rouges montrent les parties où les collisions se produisent. Cependant, en raison du bruit élevé sur les traces utilisées, le seuil expérimental du BCDC (1/2) est dépassé, même en cas de collision. La définition de ce seuil ne permet donc pas d'automatiser la détection de collision. Dans le paragraphe suivant nous verrons comment il est possible de classifier ces valeurs de manière automatique.

2.4.4.2 Phase de décision : collision ou non

À ce point, nous avons vu qu'en cas de présence de forts bruits, les valeurs du BCDC dépassent le seuil même en cas de collision. La classification des valeurs du BCDC en classes correspondant à l'apparition ou non de collision, est donc nécessaire. Pour cela, l'algorithme des k-moyennes (algorithme 14), qui permet



(a) Résultats de l'application du BCDC pour mener l'attaque par doublement sur le premier octet.



(b) Résultats de l'application du BCDC pour mener l'attaque de S. Yen et al. sur le premier octet.

FIGURE 2.18 – Résultats de l'attaque par doublement [1] et celle de S. Yen et al. [2] en utilisant le BCDC sur les traces brutes.

de classifier automatiquement un ensemble de données, peut être utilisé. Cet algorithme renvoie, lorsque $k = 2$, deux classes selon les données. Elles sont respectivement notées c_1 (classe collision) et c_2 (classe non-collision). Comme les valeurs du BCDC sont a priori plus petites lorsqu'une collision se produit, la distinction se fait naturellement et ce sans aucune inspection visuelle.

L'avantage de cette méthode en comparaison de celle proposée par C. Aidong et al.[75] est que l'algorithme des k-moyennes est ici appliqué sur des scalaires au lieu de vecteurs. Ce qui permet d'avoir moins d'erreurs de classements.

L'algorithme 15 résume les différentes étapes pour retrouver l'exposant d dans le cadre de l'attaque par doublement [1].

2.4.4.2.1 Résultats expérimentaux

Dans ce paragraphe, sont donnés des résultats correspondant à une application directe de la méthode de détection de collision, basée sur le critère BCDC et de l'algorithme des k-moyennes (algorithme 14). Le BCDC est appliqué sur les mêmes traces brutes puis l'algorithme des k-moyennes est utilisé pour classifier automatiquement les valeurs de BCDC obtenues. Les figures 2.19a et 2.19b montrent les résultats des deux attaques considérées, appliquées sur les courbes brutes. Sur ces figures, chaque barre représente une valeur du BCDC entre deux segments de trace selon l'attaque considérée. Les couleurs des barres montrent leur classe d'appartenance. Les barres bleues correspondent à la classe $\widehat{c_2}$ et les barres rouges à la classe $\widehat{c_1}$. La comparaison de ces figures avec les figures 2.17a et 2.17b montre l'avantage à utiliser le critère BCDC. En effet, il facilite la détection automatique des occurrences de collisions.

Dans la figure 2.19a, les valeurs du BCDC, en l'absence de collision, sont très proches de 1, comme cela a été démontré dans la section 2.4.3.3. On peut

algorithme 15 Attaque par doublement appliquée à l'algorithme L2R en utilisant le critère BCDC et l'algorithme des k-moyennes (algorithme 14)

Entrées: $\mathcal{T}_1 = \left\{T_1^1, T_2^1, \cdots, T_l^1\right\}$: la trace collectée lors du calcul de m^d, $\mathcal{T}_2 = \left\{T_1^2, T_2^2, \cdots, T_l^2\right\}$: la trace collectée lors du calcul de m^{2^d} et l : le nombre d'opérations

Sorties: d

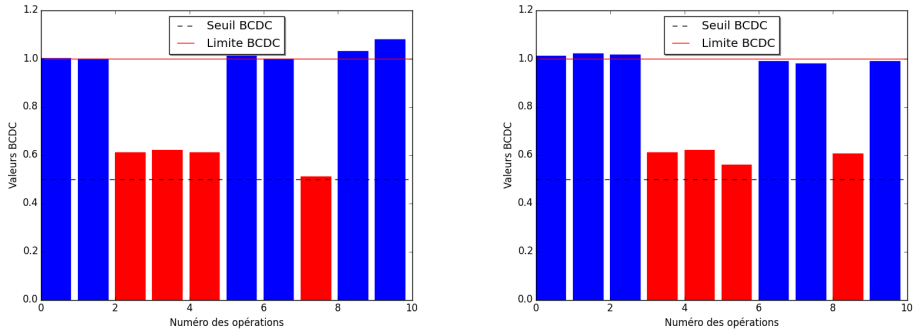
1: **pour** $i = 1$ **à** l **faire**
2: $A_i = \text{BCDC}(T_{i+1}^1, T_i^2)$
3: **fin pour**
4: $R = k$-moyennes $(A = [A_1, \cdots, A_l])$, avec $k = 2$, $R_i \in \{c_1, c_2\}$
5: **pour** $i = 2$ **à** $l - 1$ **faire**
6: **si** $(R_i \in c_2)$ **et** $(R_{i+1} \in c_2)$ **alors**
7: $d_{i-1} = 1$
8: **sinon**
9: $d_{i-1} = 0$
10: **fin si**
11: **fin pour**
12: **retourner** d

aussi noter que lorsque des collisions se produisent, les valeurs sont aux alentours de 0.6. Ainsi, la détection automatique des collisions devient simple en utilisant l'algorithme des k-moyennes (algorithme 14) car le BCDC crée deux populations bien distinctes. La figure 2.19a permet de déduire, en utilisant l'algorithme 15, tous les bits de l'exposant.

2.4.5 Amélioration du SNR

La qualité des mesures est importante pour la réussite des attaques par canaux cachés. Le SNR est souvent estimé avec des analyses verticales très sensibles aux désynchronisations globales. Comme évoqué précédemment, plus le SNR est élevé, plus les valeurs du BCDC sont petites en cas de collision. Il est ainsi intéressant d'appliquer des techniques de traitement de signal avant l'application du critère BCDC.

Il existe plusieurs méthodes proposées dans la littérature, notamment le moyennage des courbes, le filtrage ou encore la compression. Du fait des contremesures environnementales (section 2.4.2), le moyennage est souvent difficile à appliquer lorsque l'on considère des composants modernes. Ainsi, dans cette section seul le filtrage et la compression sont considérés comme applicables pour améliorer le SNR.



(a) Résultats de la combinaison du BCDC et de l'algorithme des k-moyennes pour mener de manière automatique l'attaque par doublement sur le premier octet.

(b) Résultats de la combinaison du BCDC et de l'algorithme des k-moyennes pour mener de manière automatique l'attaque de S. Yen et al. sur le premier octet.

FIGURE 2.19 – Résultats de l'attaque par doublement [1] et de celle de S. Yen et al. [2] en combinant BCDC et l'algorithme des k-moyennes sur les traces brutes.

2.4.5.1 La compression de trace

Une autre méthode pour améliorer le SNR est la compression des traces [77]. La méthode de compression choisie dans ce chapitre est celle qui consiste à sommer un ensemble de c points consécutifs et de prendre cette somme pour point de la nouvelle trace. Elle permet de réduire la taille des traces sans perte d'information sur la fuite tout en réduisant l'impact des bruits gaussiens.

Le choix du facteur de compression c (nombre de points consécutifs sommés) est crucial pour une compression efficace. En effet, un mauvais choix de c peut noyer les fuites relatives à l'information secrète et plus précisément mélanger cette fuite à d'autres informations. En général, c ne doit pas être tel que les points sommés correspondent à une durée supérieur à un cycle d'horloge. Il peut être choisi plus petit mais pas plus grand.

Montrons maintenant pourquoi la compression améliore le SNR et donc facilite la détection de collision du BCDC. Soit $T_i = \left[t_i^1, \cdots, t_i^q\right]$ un segment de taille q. La compression de T_i par un facteur de compression c est donnée par :

$$C^c(T_i) = \left\{ t'^z_i = \sum_{k=zc}^{(z+1)c-1} t_i^k, \quad z \in [0, q/c] \cap \mathbb{N} \right\} \tag{2.9}$$

En supposant que toutes les réalisations du bruit sont indépendantes, le théorème central limite permet d'estimer l'écart type de $C^c(T_i)$ par :

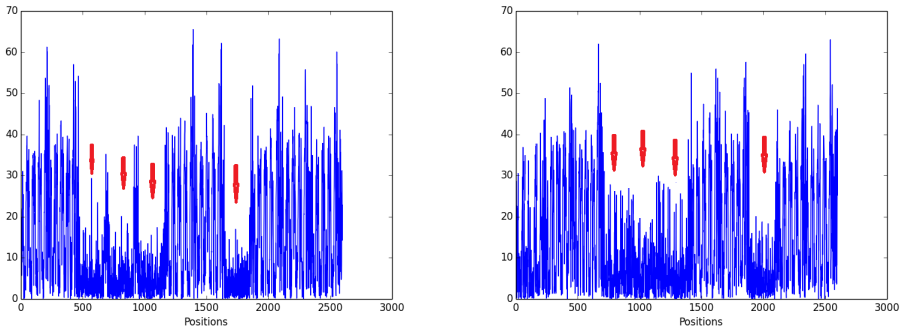
$$\sigma_{C^c(T_i)} = \sqrt{\sigma'^2_{s'} + \frac{{\sigma_N}^2}{c}} \tag{2.10}$$

où σ'^2_s est la variance de la partie déterministe et compressée s'^k_i de T_i. Ainsi la

compression divise l'écart-type σ_N du bruit par un facteur égal à $\frac{1}{\sqrt{c}}$. Cette réduction n'affecte pas les valeurs du BCDC en cas de non-collision car la réduction de l'écart type du bruit s'applique sur le numérateur et le dénominateur de l'équation 2.8. Cependant, quand une collision se produit, la compression réduit significativement la valeur du BCDC. Par conséquent, la compression permet d'augmenter le contraste entre les valeurs du BCDC correspondant à l'apparition ou non d'une collision. Ainsi, la compression peut être utilisée pour faciliter la classification effectuée par l'algorithme des k-moyennes (algorithme 14).

2.4.5.1.1 Résultats expérimentaux

Les résultats précédents montrent que la méthode de référence (différences directes) sur les courbes brutes n'est pas suffisante pour détecter l'apparition de collision lorsque les traces sont acquises sur des composants modernes. Dans ce paragraphe, la méthode de compression est appliquée avant d'effectuer la différence point par point. Le facteur de compression est $c = 100$ (taille correspondant à un cycle d'horloge). La figure 2.20a montre le résultat de l'attaque par doublement [1] lorsque les traces sont compressées avec un facteur $c = 100$. Tandis que la figure 2.20b montre le résultat avec l'attaque de S. Yen et al. [2]. Les parties correspondant aux collisions, marquées sur la figure par des flèches rouges, sont maintenant distinguables. Ainsi, les bits de l'exposant sont retrouvés facilement. Ces résultats montrent que la technique de compression améliore significativement les résultats. Cependant, une inspection visuelle reste nécessaire pour identifier les collisions.



(a) Valeurs absolues des différences point par point de traces collectées pour mener l'attaque par doublement sur le premier octet de l'exposant lorsque une compression avec un facteur $c = 100$ est appliquée.

(b) Valeurs absolues des différences point par point de traces collectées pour mener l'attaque de S. Yen et al. sur le premier octet de l'exposant lorsque une compression avec un facteur $c = 100$ est appliquée.

FIGURE 2.20 – Résultats de l'attaque par doublement [1] et celle de S. Yen et al. [2] sur des traces compressées.

2.4.5.2 Le filtrage

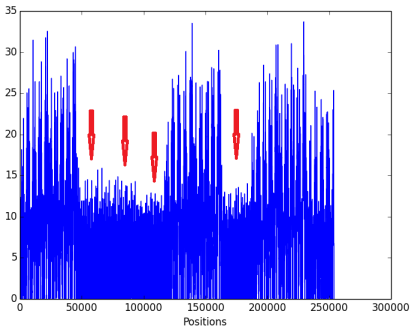
Les techniques de filtrage sont généralement efficaces lorsque l'attaquant a une certaine connaissance des fréquences utiles (porteuses de signaux) pour une attaque par canaux cachés. Ainsi, nous pouvons supprimer de la trace les informations porteuses essentiellement de bruit. S. Tiran et al. ont proposé dans [78], une méthode permettant de choisir les fréquences (utiles) devant être conservées lors d'attaques par canaux cachés verticales. Cette méthode est basée sur un critère appelé *Leakage-to-Noise Ratio* (LNR). Le LNR est défini par l'équation 2.11 :

$$LNR(f) = \frac{1}{f^2} \frac{\langle PSD(f) \rangle}{\sigma_{PSD(f)}} \tag{2.11}$$

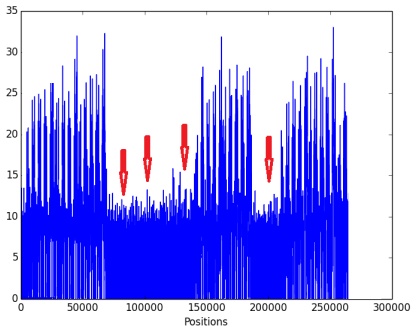
où $\langle\rangle$ désigne une moyenne, $PSD(f)$ la densité spectrale de puissance ou *Power Spectral Density* (PSD) du signal à la fréquence f et σ son écart-type. Il sera montré dans la paragraphe suivant que cette méthode peut être utilisée pour améliorer la détection de collision.

2.4.5.2.1 Résultats expérimentaux

Les mêmes expériences que celles conduites pour évaluer l'efficacité de la méthode de compression ont été conduites en utilisant la méthode de filtrage basée sur le LNR. La figures 2.21a et 2.21b montrent les résultats obtenus. Comme dans le cas de la compression, une inspection visuelle permet d'identifier les parties où des collisions se produisent.



(a) Valeurs absolues des différences point par point de traces collectées pour mener l'attaque par doublement sur le premier octet de l'exposant en utilisant le filtrage LNR.



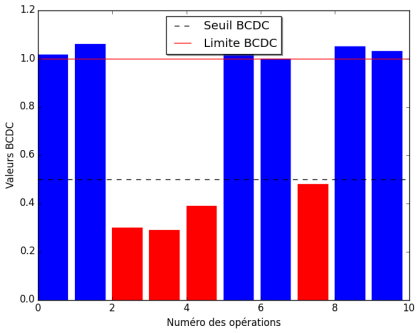
(b) Valeurs absolues des différences point par point de traces collectées pour mener l'attaque de S. Yen et al. sur le premier octet de l'exposant en utilisant le filtrage LNR.

FIGURE 2.21 – Résultats de l'attaque par doublement [1] et celle de S. Yen et al. [2] sur les traces filtrées.

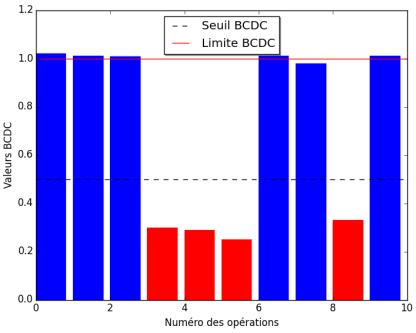
La compression et le filtrage peuvent être utilisés en conjonction avec le BCDC comme expliqué plutôt. Dans ce cas, on s'attend à une amélioration significative lorsque le bruit est très important.

2.4.5.3 Combinaison du BCDC et de la méthode compression

Les résultats introduits dans le paragraphe précédent valident la méthode proposée de détection de collision. Lorsque le critère BCDC est combiné avec la méthode de compression sur les mêmes traces (avec un facteur de compression $c = 100$), le contraste entre les collisions et les non-collisions, devient plus net. La comparaison entre les figures 2.22a et 2.22b avec les figures 2.19a et 2.19b respectivement, montre clairement que l'utilisation de la compression améliore nettement la détection. Les valeurs du BCDC sont maintenant comprises entre 0.2 et le seuil de décision lors des collisions et restent proches de 1 dans le cas contraire.



(a) Résultats du BCDC pour mener de manière automatique l'attaque par doublement sur le premier octet lorsque les traces sont compressés ($c = 100$).



(b) Résultats du BCDC pour mener de manière automatique l'attaque de S. Yen et al. sur le premier octet lorsque les traces sont compressés ($c = 100$).

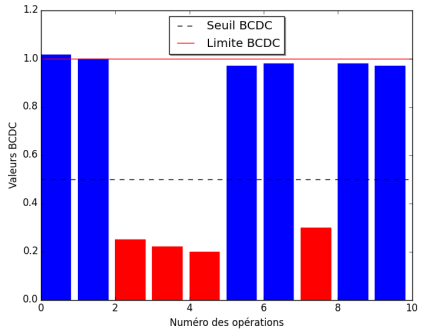
FIGURE 2.22 – Résultats de l'attaque par doublement [1] et celle S. Yen et al. [2] en combinant BCDC et compression.

2.4.5.4 Combinaison du BCDC et le filtrage LNR

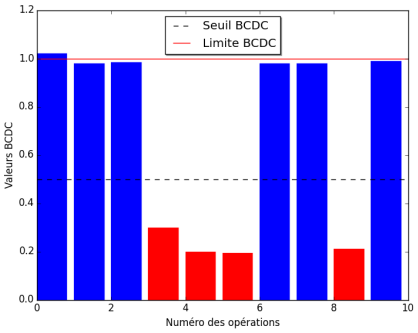
Les expériences décrites dans la section précédente ont été reprises en utilisant la méthode de filtrage basée sur le LNR au lieu de la méthode de compression. Les figures 2.23a et 2.23b donnent les résultats obtenus pour l'attaque par doublement et celle de S. Yen et al. respectivement. Comme évoqué dans [78], garder uniquement les fréquences basses permet d'améliorer significativement le SNR.

2.4.6 Taux de Succès

Pour valider définitivement l'intérêt de la méthode de détection de collisions basée sur la combinaison du BCDC et de l'algorithme des k-moyennes en cas de courbes bruitées, et du BCDC seul lorsque le SNR des traces est amélioré, plusieurs campagnes ont été conduites. Ces campagnes ont eu pour but d'estimer la probabilité de détecter des collisions connaissant l'exposant. Pour cela, 1000 couples de



(a) Résultats du BCDC pour mener de manière automatique l'attaque par doublement [1] sur le premier octet lorsque les traces sont filtrées avec la méthode LNR.



(b) Résultats du BCDC pour mener de manière automatique l'attaque de S. Yen et al. [2] sur le premier octet lorsque les traces sont filtrées avec la méthode LNR.

FIGURE 2.23 – Résultats de l'attaque par doublement [1] et celle de S. Yen et al. [2] en combinant BCDC et filtrage basé sur le LNR.

traces ont été collectées pour mener l'attaque par doublement [1]. Cette probabilité est appelée taux de succès.

Suite à la collecte des traces et en guise de comparaison les attaques ont été effectuées avec les critères BCDC et le PCDC, en considérant la méthode de compression et le filtrage LNR. Les tableaux 2.1 et 2.2 montrent les résultats obtenus, pour différents facteurs de compression, mais aussi pour deux taux d'échantillonnage différents. On peut observer que la compression ou le filtrage basé sur le LNR ont un effet considérable sur le taux de succès.

Nous pouvons également observer que l'utilisation du critère BCDC, couplé à l'algorithme des k-moyennes est efficace pour détecter automatiquement les collisions en présence de courbes bruitées, avec des valeurs de taux de succès supérieurs à 80% lorsque le taux d'échantillonnage est égal à 2.5 GS/s. Lorsque les courbes sont pré-traitées, le seuil expérimental du BCDC suffit pour détecter automatiquement les collisions. Le tableau 2.2 montre également que lorsque le taux d'échantillonnage est faible (500 MS/s), le filtrage et la compression améliorent les résultats et le taux de succès atteint les 100%.

Taux d'échantillonnage	500 MS/s			
Methode	Attaque par doublement		Attaque de S. Yen et al.	
	PCDC	BCDC	PCDC	BCDC
Traces brutes	77%	89%	75%	87%
c = 10	85%	**100%**	86%	**100%**
c = 20	90%	**100%**	90%	**100%**
c = 100	40%	46%	37%	51%
LNR	82%	**100%**	80%	**100%**

Tableau 2.2 – Taux de succès SR obtenus avec les critères PCDC et BCDC lorsque les traces sont collectées avec un taux d'échantillonnage de 500 MS/s avec et sans application de filtrage ou de compression..

Taux d'échantillonnage	2.5 GS/s			
Methode	Attaque par doublement		Attaque de S. Yen et al.	
	PCDC	BCDC	PCDC	BCDC
Traces brutes	72%	81%	69%	83%
c = 10	74%	85%	72%	85%
c = 50	80%	90%	78%	91%
c = 100	83%	**98%**	82%	**97%**
c = 500	53%	54%	34%	68%
LNR	73%	**94%**	70%	**96%**

Tableau 2.1 – Taux de succès obtenus avec les critères PCDC et BCDC lorsque les traces sont collectées avec un taux d'échantillonnage de 2.5 GS/s avec et sans application de filtrage ou de compression.

2.5 Conclusion

Dans ce chapitre nous avons présenté l'outil de simulation d'attaque développé durant ma thèse. Cet outil permet de valider en amont les attaques par canaux cachés. Quelques résultats d'attaques obtenus avec cet outil ont été présentés. Cet outil et les résultats qu'il fournit permettent de jauger de manière rapide la dangerosité des attaques en s'affranchissant des difficultés expérimentales liées à leur pratique. Toutefois, ces difficultés sont bien réelles.

Pour appliquer les attaques par collision sur des courbes obtenues sur de vrais composants, nous avons introduit un nouveau critère de détection de collision appelé BCDC. Ce critère a été proposé pour rendre les attaques par collision beaucoup plus efficaces dans la pratique. Certaines expériences permettant de comparer ce critère avec celui déjà publié dans [7] ont été détaillées. Le nouveau critère est efficace pour trouver et exploiter les attaques par collision présentées dans la litté-

rature [1, 2, 8, 9]. Même si les résultats présentés dans ce chapitre se concentrent sur l'exponentiation modulaire, ce critère est également efficace pour détecter des collisions sur des cryptosystèmes symétriques.

En outre, certaines techniques de prétraitement de signal (filtrage LNR et compression) ont été étudiées dans la pratique montrant des résultats concrets.

Ces travaux ont in fine conduit à mettre en place une méthodologie permettant l'application automatique des attaques par collision dans des environnements bruités. Elle permet l'obtention de taux de succès élevés sur des composants modernes.

Chapitre 3

La collision pour estimer la qualité de mesures et applications associées

Si le SNR est un critère couramment utilisé dans de nombreux domaines pour jauger de la qualité de mesures analogiques, son utilisation dans le contexte des attaques par canaux cachés reste toutefois très délicate étant donné que la nature et les caractéristiques des signaux ne sont pas a priori connues. Par voie de conséquence, ce critère est peu utilisé dans un domaine où la qualité des mesures joue un rôle primordial. Dans ce contexte, ce chapitre introduit une méthode d'estimation rapide et pragmatique du SNR ainsi que quelques-unes des applications qu'il

permet d'effectuer avec efficacité, comme l'identification des fréquences porteuses de signal ou encore l'analyse temporelle de l'activité des circuits.

Les résultats de ce chapitre ont été publiés à CARDIS 2015 [79].

3.1 Introduction

Ces dernières années, de nombreux efforts ont été consentis pour localiser les fuites d'information dans l'espace [80, 81, 82, 83] et le temps [84], accroître notre compréhension de ces fuites ou encore augmenter nos capacités d'extraction de celles-ci [18, 21, 45]. Des efforts ont également été dévolus à la définition de méthodes permettant de standardiser certains savoir-faire ou bien de comparer leurs performances. Parmi ces méthodes, on trouve par exemple les notions de taux de succès (*Success Rate* (SR)) et de l'entropie résiduelle ou *Guessing Entropy* (GE) [85, 86] qui ont été largement adoptées pour comparer l'efficacité de diverses attaques. Parallèlement peu d'efforts ont été fournis pour comparer de manière standard les pratiques expérimentales qui sont au cœur de toute analyse sécuritaire. Pourtant, le succès d'une attaque par canal auxiliaire, qui consiste somme toute à retrouver une information au milieu du bruit, repose grandement sur la qualité des mesures effectuées.

On peut bien sûr mettre en avant que comparer la qualité de mesures n'est pas un problème spécifique au contexte des attaques par canaux auxiliaires et que des outils existent comme le SNR.

Certes ! La notion de SNR est effectivement un critère parfaitement adapté pour jauger de la qualité de mesures. Toutefois, son exploitation dans le domaine des attaques par canaux cachés est particulièrement difficile compte tenu que le signal n'est pas a priori connu. Des méthodes permettant de contourner ce problème ont cependant été proposées. Toutefois, la majorité de ces techniques visent à quantifier la quantité de fuite dans un jeu de traces plutôt que la qualité des mesures elles mêmes.

Parmi celles-ci on peut citer l'approche proposée en 2011 dans [87] qui s'appuie sur un modèle linéaire de la fuite d'information mais également sur des considérations empiriques liées à l'utilisation d'oscilloscope pour proposer une méthode d'estimation du SNR. Toutefois, la définition du SNR qui est considérée dans ces travaux est relativement éloignée de la définition usuelle, et est finalement proche de la notion de LNR.

Cette notion de LNR apparaît explicitement en 2013 dans [78] où une méthode d'estimation de celui-ci est proposée. Elle s'appuie sur une modélisation des fuites dans le domaine fréquentiel. Parallèlement à l'introduction du critère LNR, S. Bhasin et al. ont introduit en 2013 le critère *Normalized Inter-Class Variance* (NICV) [84]. Ce critère repose sur une utilisation astucieuse et efficace de l'analyse de la variance et permet d'identifier les points des traces véhiculant potentiellement de la fuite. Outre cette application importante, il est également suggéré que le critère NICV fournit également une estimation du SNR. Toutefois, là encore la définition

du SNR considérée dans ces travaux est plus une estimation du rapport quantité de fuite d'information sur bruit qu'une véritable mesure du SNR. Le critère NICV ne renseigne donc en rien sur la qualité des mesures ou sur la rigueur du protocole expérimental suivi pour les effectuer.

Dans ce contexte, la contribution essentielle de ce chapitre à l'état de l'art est une méthode d'estimation du SNR, basée sur des collisions de signaux ne nécessitant que peu de courbes : en théorie deux, dans la pratique une ou deux dizaines.

Le seconde contribution est de montrer que cette estimation du SNR constitue bien une méthode efficace pour jauger de la qualité de traces de canaux cachés. De plus, il est ensuite montré comment cette méthode d'estimation du SNR peut être appliquée pour :

- analyser le comportement temporel d'un circuit à partir de quelques mesures,
- distinguer les fréquences porteuses de signal des fréquences véhiculant essentiellement du bruit. Être capable d'effectuer une telle distinction est capital. Cela permet en effet de guider de manière adaptative et rationnelle les étapes de filtrage couramment utilisées avant une application d'analyses par canaux cachés.

Le reste de ce chapitre est organisé comme suit. En section 3.2 quelques notions sont présentées. En section 3.3 est rappelée la définition stricte du SNR et une déclinaison de celle-ci dans le cas de signaux de moyenne nulle. Ces définitions posées, la section 3.4 montre comment obtenir une estimation du SNR à partir de collisions de signaux et plus particulièrement du BCDC introduit dans la section 2.4.3.3. Il est ensuite expérimentalement montré en section 3.5 que l'estimation du SNR ainsi obtenue est effectivement un facteur de mérite permettant de jauger de la qualité de traces dans le contexte des attaques par canaux cachés. De la même manière, la section 3.6 montre comment cette méthode d'estimation rapide du SNR permet d'analyser les traces de canaux cachés de manière simple, efficace et complémentaires aux approches proposées dans [84], [78]. La section 3.7 montre ensuite comment distinguer, de manière aussi simple et précise que dans [78], les fréquences porteuses de signal de celles véhiculant essentiellement du bruit. Enfin, une conclusion est proposée en section 3.8.

3.2 Préambule

Dans ce chapitre, le terme *signal* désigne l'évolution déterministe (répétée et mesurée n fois avec des équipements parfaits en l'absence de toute source de bruit, le signal est unique) d'une grandeur physique dans le temps, en réponse à un stimulus unique. Par exemple, l'évolution du champ électromagnétique au niveau des coordonnées (X, Y, Z) au-dessus d'un circuit intégré calculant le chiffrement d'un texte donné, est un signal. Le changement d'un de ces paramètres entraine un nouveau signal différent.

Dans la littérature [77, 84, 88, 78], une mesure de puissance de consommation (trace) est définie comme la somme d'une partie exploitable par une attaque par canaux cachés (P_{exp}), d'un bruit (considéré dans [77] comme la somme du bruit électronique ($P_{el.noise}$) et du bruit provenant d'un changement d'état) et d'une partie constante (P_{const}). Dans ce chapitre, une mesure de puissance de consommation est considérée comme étant la somme de la puissance du signal P_S à mesurer et de la puissance du bruit P_N. Ainsi, P_S contient P_{exp} (mais n'est pas réduite à P_{exp}). Ce choix est fait dans le but d'évaluer la qualité des protocoles de mesures, indépendamment du fait que les traces contiennent ou non des informations exploitables.

Ces deux définitions de signal sont différentes car, elles ont été adoptées à des fins différentes. Dans notre cas, la définition du signal est adoptée pour évaluer la qualité d'une mesure. Cela permet également d'évaluer la qualité du protocole expérimental connexe, comme alternative à la quantification de la fuite présente dans un ensemble de traces [84]. Cependant, nous ferons un parallèle entre notre approche et ce point plus tard dans le document.

3.3 Définition et Problèmes liés

Avec la définition adoptée dans ce chapitre, le SNR est un facteur de mérite permettant de jauger de la qualité de la mesure $T(t) = [t_1, \cdots, t_q]$ d'un signal $S(t) = [s_1, \cdots, s_q]$, qualité qui dépend du bruit régnant lorsque celle-ci est effectuée, mais aussi des équipements utilisés pour collecter le signal. La définition standard stricte du SNR est le rapport entre la puissance P_S du signal $S(t)$ que l'on cherche à mesurer et la puissance P_N du bruit $N(t) = [\eta_1, \cdots, \eta_q]$:

$$SNR = \frac{P_S}{P_N} = \frac{\frac{1}{q} \cdot \sum_{i=1}^{q} (s_i)^2}{\frac{1}{q} \cdot \sum_{i=1}^{q} (\eta_i)^2} \tag{3.1}$$

Dans le cas où le signal que l'on cherche à mesurer a une valeur moyenne nulle, le numérateur et le dénominateur de l'équation 3.1 correspondent à la variance du signal et du bruit, respectivement. Ainsi, en considérant $S(t)$ et $N(t)$ comme étant des variables aléatoires dans le temps (variables aléatoires horizontales), le SNR peut également être exprimé par l'équation 3.2 :

$$SNR = \frac{\sigma_S^2}{\sigma_N^2} = \frac{\frac{1}{q} \cdot \sum_{i=1}^{q} (s_i - 0)^2}{\frac{1}{q} \cdot \sum_{i=1}^{q} (\eta_i - 0)^2} \tag{3.2}$$

avec σ_S et σ_N représentant respectivement l'écart type (calculé horizontalement) du signal $S(t)$ et du bruit $N(t)$. Il faudrait noter que même si le SNR prend la forme d'un rapport de variances, c'est en fait un ratio de puissances qui doit être considéré comme tel.

Cette définition du SNR est particulièrement intéressante dans le contexte des attaques exploitant le canal électromagnétique car la moyenne des signaux collectés est toujours très proche de zéro, ou peut être forcée à zéro en utilisant des

amplificateurs de tension travaillant en mode petit signal (cela vaut alors pour les analyses en consommation). C'est donc cette définition qui est considérée dans le reste de ce chapitre. Toutefois, compte tenu du fait que dans le contexte des attaques par canaux cachés le signal et ses caractéristiques ne sont pas a priori connus, cette définition est difficile à exploiter en l'état. Un moyen doit donc être trouvé pour extraire des mesures les valeurs du SNR (eq. 3.2) ou bien obtenir un facteur de mérite caractérisée par les mêmes propriétés.

3.4 De la collision à l'estimation du SNR

Si dans le contexte des attaques par canaux cachés le signal $S(t)$ n'est pas connu, d'autres degrés de liberté s'offrent aux praticiens. Parmi ces degrés de liberté, on trouve la possibilité d'émuler les circuits de manière répétitive et donc de répéter n fois la mesure associée à une exécution quelconque donnée. Toutefois, ce degré de liberté n'est pas illimité. En effet, certains protocoles de sécurité veillent à limiter ce degré de liberté à quelques unités et si cela n'est pas le cas, il est certes théoriquement possible de répéter de manière illimitée une même exécution mais dans la pratique cela s'avère difficile à cause des contremesures environnementales qui requièrent la mise en place de processus de ré-alignement des traces qui peuvent être couteux en temps et ressources de calcul. Toutefois, il semble possible d'exploiter les collisions de signaux, en utilisant quelques traces (de 1 si on est capable de générer des collisions lors d'un même calcul, à quelques dizaines si ce n'est pas le cas), pour extraire la valeur du SNR, i.e. extraire σ_S et σ_N, comme cela est expliqué ci-dessous.

3.4.1 Du BCDC à l'estimation du SNR

L'exploitation de collisions de signaux est maintenant une technique reconnue efficace qui permet d'attaquer des algorithmes de chiffrement symétriques [54] ou asymétriques [1, 2, 23, 9] comme expliqué dans le chapitre 1 (ces attaques sont détaillées dans la section 1.4) et mise en pratique dans le chapitre 2, en choisissant des textes clairs induisant ou non, selon la valeur d'un secret, l'exécution de calculs intermédiaires identiques. L'occurrence ou non de ces calculs identiques permet alors d'inférer la valeur du secret ciblé comme nous l'avons vu dans le chapitre précédent. Toutefois, ceci nécessite l'utilisation de techniques de traitement des traces permettant de détecter l'occurrence de ces collisions dans le bruit de mesure. Dans le chapitre précédent (section 2.4.3.3), un critère permettant d'automatiser la détection de collision a été proposé. Ce critère, appelé BCDC, prend ses valeurs dans]0, 1] et a pour expression (en gardant les mêmes notations que dans le chapitre 2 :

$$BCDC(T_i, T_j) = \frac{1}{\sqrt{2}} \times \frac{\sigma_{(T_i - T_j)}}{\sigma_{(T_j)}} \tag{3.3}$$

À partir de cette équation, on peut exprimer la valeur asymptotique du BCDC dans le cas où T_i et T_j sont deux mesures du même signal, i.e. lorsqu'il y a collision (chapitre 2, section 2.4.3.3.2 pour la preuve) par l'équation 3.4.

$$BCDC(T_i, T_j) = \frac{1}{\sqrt{1 + \frac{\sigma_S^2}{\sigma_N^2}}}, \tag{3.4}$$

Dans celle-ci, le SNR, $\frac{\sigma_S^2}{\sigma_N^2}$, exprimé par l'équation 3.2, apparaît explicitement.

L'estimation rapide du SNR devient donc possible dans le contexte des attaques par canaux cachés ou tout autre contexte d'ailleurs. Pour ce faire, il suffit de collecter n (> 1) traces correspondant à l'exécution d'un même traitement par le DUT ; puis de calculer les $\frac{1}{2} \cdot n \cdot (n-1)$ valeurs de BCDC pour enfin en déduire autant de valeurs de SNR_{ij} :

$$SNR_{ij} = \frac{1}{BCDC^2(T_i, T_j)} - 1 \tag{3.5}$$

Finalement, le SNR d'un ensemble de traces est estimé en moyennant les $\frac{1}{2} \cdot n \cdot (n-1)$ valeurs de SNR_{ij}. Dans le reste de ce chapitre cette estimation est notée $\widehat{SNR}$

$$\widehat{SNR} = \frac{2}{n \cdot (n-1)} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \left(\frac{1}{BCDC^2(T_i, T_j)} - 1\right) \tag{3.6}$$

La donnée de cette valeur moyenne et de sa distribution constitue une figure de mérite permettant de jauger de la qualité d'un jeu de mesures et de la qualité des protocoles expérimentaux suivis pour collecter ces mesures : plus la valeur du $\widehat{SNR}$ est élevée, meilleur est le protocole expérimental.

En guise d'illustration, deux exemples de distributions typiques de valeurs de SNR_{ij} sont donnés figures 3.3c et d. Les paragraphes suivants donnent une illustration dans un cas d'école.

3.4.2 Exemples

Afin d'amener des éléments pratiques sur l'efficacité et la simplicité de mise en œuvre de cette méthode d'estimation rapide du SNR dans le cas où le signal n'est pas connu, des données ont été générées. Elles sont relatives au cas de la mesure d'un signal sinusoïdal d'amplitude égale à 1 dont la variance horizontale sur une période vaut 2 ($\sigma_S = 1/\sqrt{2}$ sur une période), dans un environnement introduisant un bruit gaussien de moyenne nulle et d'écart-type σ_N.

La table 3.1 donne la valeur exacte du SNR, ainsi que son évolution lorsque l'écart-type et le nombre de mesures varient. Comme on peut le constater, avec 5 mesures (soit 10 réalisations de BCDC et de SNR_{ij}), l'estimation $\widehat{SNR}_{n=5}$ est très proche de la valeur exacte du SNR. En effet, l'erreur relative absolue ne dépasse pas 5%.

$\sigma_{\mathbf{N}}$	**0.1**	**0.3**	**0.5**	**0.7**	**0.9**	**1.1**	**1.3**	**1.5**	**1.7**	**1.9**
SNR	**50**	**5.56**	**2.00**	**1.02**	**0.62**	**0.41**	**0.30**	**0.22**	**0.17**	**0.14**
$\widehat{SNR_{n=2}}$	49.38	5.58	2.03	1.00	0.64	0.41	0.28	0.24	0.17	0.16
$\widehat{SNR_{n=5}}$	49.76	5.51	1.99	1.03	0.60	0.41	0.29	0.21	0.17	0.14
$\widehat{SNR_{n=10}}$	50.41	5.56	1.99	1.02	0.62	0.41	0.29	0.21	0.18	0.14
$\widehat{SNR_{n=20}}$	49.91	5.52	2.00	1.03	0.62	0.41	0.30	0.22	0.17	0.14
$\widehat{SNR_{n=50}}$	50.08	5.56	2.01	1.01	0.62	0.41	0.30	0.22	0.17	0.14

Tableau 3.1 – Évolution de la valeur moyenne de $\widehat{SNR}$ lorsque l'écart-type et le nombre de mesures varient

Ce résultat est d'autant plus précis que l'écart-type σ_{SNR} des valeurs de SNR obtenues pour chaque paire de mesure est relativement faible. Celui-ci a été mesuré avec 50 mesures (soit 1275 paires de mesures). Pour $\sigma_N = 1.9$, la valeur obtenue est égale à 0.0048. Ceci représente 3.45% de la valeur moyenne ($\widehat{SNR} = 0.1385$). Ces résultats démontrent la pertinence de l'approche proposée.

3.4.3 Lien entre le BCDC et le SNR estimé en cas de collision

Pour illustrer le lien entre le SNR estimé et le critère BCDC pratique, nous avons calculé l'évolution du SNR estimé en fonction des valeurs prises par le BCDC.

La figure 3.1 reporte le résultat obtenu. Nous pouvons observer que plus le SNR est petit, plus les valeurs du BCDC sont grandes (et tendent donc vers le cas de non-collision).

Le SNR est égal à 3 au seuil de collision. Ainsi, cette valeur peut être utilisée comme indicateur de qualité de mesure. Un jeu de traces avec un SNR estimé supérieur ou égal 3 peut être considéré comme de bonne qualité et potentiellement exploitable par une attaque par canaux cachés. Lorsque le SNR estimé est compris entre 1 et 3 les traces sont considérées comme étant de qualité moyenne. Enfin, lorsque le SNR estimé est inférieur à 1 les traces sont de mauvaise qualité.

Ce calcul nous permet de confirmer les observations faites dans le chapitre 2, à savoir plus le SNR des traces est élevé plus les collisions sont faciles à détecter.

3.4.4 Discussions

À ce point, une méthode efficace d'estimation du SNR, pouvant être appliquée lorsque le signal n'est pas connu, a été introduite. Il a également été démontré que cette méthode, qui est basée sur la définition du SNR utilisée pour vérifier la qualité de mesures analogiques ou de la qualité d'un canal de communication, est très précise.

La définition considérée dans ce chapitre est différente de celles adoptées dans [88, 77, 84]. Dans ces derniers articles les auteurs quantifient la fuite exploitable

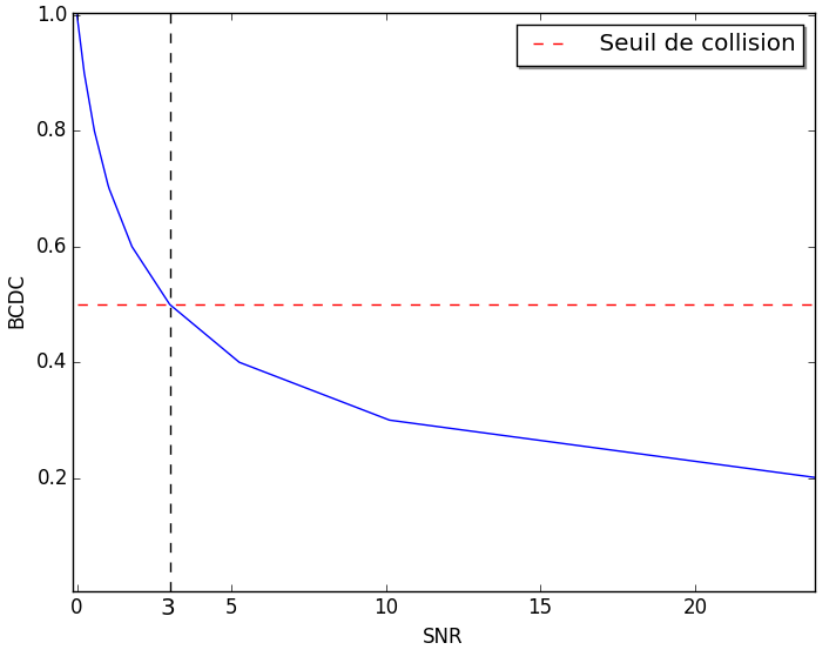


FIGURE 3.1 – Évolution du SNR en fonction des valeurs de BCDC

par une attaque par canaux cachés verticale (une attaque CPA par exemple). Cependant rien n'a été proposé pour vérifier si la définition considérée dans ce chapitre est aussi une méthode indirecte pour quantifier la fuite présente dans une trace. Après tout, les mesures de grande qualité sont plus susceptibles de contenir plus d'informations permettant de réaliser une attaque par canaux cachés que celles de mauvaise qualité.

Si la définition de SNR considérée ici est en effet un facteur de mérite de la qualité des traces de canaux cachés, l'idée que nous proposons pour évaluer la qualité d'un ensemble de traces de canaux cachés est simple. Elle consiste à acquérir une petite quantité de traces (2 à 50 par exemple) d'un même calcul (ou paire de traces du même calcul) afin d'estimer le $\widehat{SNR}$ et pour tracer la distribution de SNR_{ij}. Cette valeur (ou sa loi de répartition estimée) sera considérée comme une mesure de la qualité des traces mais aussi du protocole expérimental appliqué pour collecter les traces.

3.5 SNR estimé et CPA

La méthode proposée ci-dessous permet d'obtenir une estimation du SNR avec un ensemble très réduit de paires (10 à 50) de mesures effectuées lorsque le circuit cible effectue un même travail calculatoire. Pour mener des analyses par canaux

cachés, par exemple sur un AES, de nombreux messages différents sont utilisés. Il semble donc judicieux de vérifier que cette méthode d'estimation du SNR est une figure de mérite de la qualité des traces en vue de mener des attaques par canaux cachés et plus particulièrement d'analyser l'évolution de la *global Guessing Entropy* gGE en fonction du SNR. Ceci est d'autant plus nécessaire que dans les travaux reportés dans [88], le SNR est défini comme :

$$\frac{\sigma_Q^2}{\sigma_N^2} \tag{3.7}$$

où σ_Q^2 est la variance du signal imputable aux valeurs intermédiaires cibles de l'attaque, quantité qui peut être petite devant la variance totale du signal qui sert de numérateur (voir eq. 3.2) dans la définition physique du SNR. Comme évoqué dans l'introduction, l'expression du SNR proposée dans [88] doit donc être perçue comme un rapport fuite à bruit (LNR) et non pas comme une valeur de SNR. C'est la raison pour laquelle cette expression, qui est reprise dans [84], est appelée dans le reste de ce papier LNR.

Il existe toutefois un cas dans lequel le LNR est égal au SNR. Il s'agit du cas où la seule activité calculatoire régnant dans le circuit est effectivement le calcul de ces valeurs intermédiaires. Ce cas est malheureusement peu réaliste et on peut donc penser que de manière générale :

$$SNR = \frac{\sigma_S^2}{\sigma_N^2} \geq \frac{\sigma_Q^2}{\sigma_N^2} = LNR. \tag{3.8}$$

On peut même penser que $SNR = \alpha \cdot LNR$ avec α qui dépend de l'architecture du circuit considéré. En effet, si les valeurs intermédiaires ciblées correspondent à un octet, α devrait être plus élevé si la cible est un produit 32 bits plutôt que 8 bits.

3.5.1 Résultats Expérimentaux

Afin de faire varier le SNR, différentes campagnes de mesures électromagnétiques ont été conduites. Elles ont consisté à mesurer le rayonnement électromagnétique d'un AES implémenté dans un FPGA. La tête de la sonde de mesure électromagnétique est placée à une distance Z croissante de la surface du FPGA et cela sans modifier les réglages de l'oscilloscope afin de réduire la variance du signal tout en augmentant celle du bruit.

Pour chaque distance Z considérée, 50 traces électromagnétiques de 20000 points correspondant au chiffrement d'une même entrée, sont collectées pour estimer le SNR avec notre méthode ; 5000 traces supplémentaires, correspondant chacune au chiffrement d'une entrée aléatoire, sont aussi collectées dans le but d'appliquer une attaque CPA.

La figure 3.2 illustre le principe de ces expérimentations et reporte 25 mesures EM obtenues au contact du composant ($Z = 0$) et à une distance de $500\mu m$ ($Z = 500\mu m$) lorsque l'AES effectue le même chiffrement. Sur ces figures, la

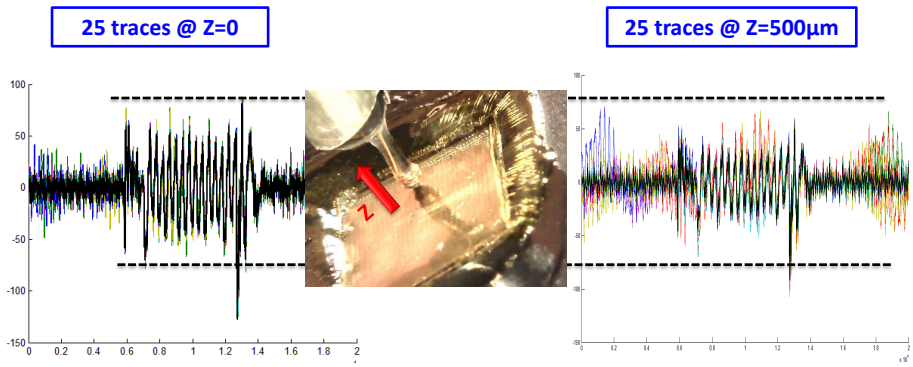


FIGURE 3.2 – 25 Mesures EM prises à $Z = 0$ et $Z = 500\mu m$ au dessus d'un AES mappé dans un FPGA

courbe noire en trait épais correspond à la trace moyenne des 25 mesures. On peut constater, sur les traces mesurées à $Z = 500\mu m$ que l'amplitude des courbes est plus faible mais également et surtout l'apparition de bruits de mesure importants. Ce bruit a pour effet de rendre les mesures obtenues moins intelligibles. En effet, contrairement à celles relatives aux mesures faites au contact du composant ($Z = 0$), les rondes de l'AES sont moins visibles, et disparaissent complètement à $Z = 4000\mu m$.

3.5.2 Analyses des résultats expérimentaux

Une CPA a ensuite été appliquée sur chaque jeu de 5000 traces collectées à des valeurs de Z comprises entre 0 et $5mm$. Dans ces jeux de mesures, 50 d'entre-elles ont été effectuées alors que le circuit traite le même texte clair de manière à pouvoir calculer le SNR. La figure 3.3a reporte les évolutions avec Z de σ_S et $\sqrt{\widehat{SNR}}$. Comme attendu le $\widehat{SNR}$ décroît rapidement quand Z augmente ; les distributions du SNR obtenues pour les mesures à $Z = 0$ et $Z = 2000\mu m$ sont données à titre d'illustration sur les figures 3.3c et d respectivement. Pour $Z = 0$, $\sqrt{\widehat{SNR}} = 15.11$ et $\sigma_{\sqrt{SNR}} = 4.80$ alors que pour $Z = 2000\mu m$, $\sqrt{\widehat{SNR}} = 7.31$ et $\sigma_{\sqrt{SNR}} = 1.64$.

La figure 3.3b donne l'évolution de gGE en fonction du $\sqrt{\widehat{SNR}}$ calculé selon l'expression 3.2. Comme on peut le constater, plus la valeur du SNR estimé est faible moins la CPA est efficace. On peut observer que dès que $\sqrt{\widehat{SNR}} < 5$, la CPA ne permet plus de retrouver la clé avec 5000 traces et la gGE reste proche de 128 (256 hypothèses de clés pour l'AES).

Quoi qu'il en soit, la figure 3.3b met en évidence l'existence d'un lien entre la gGE et le SNR. La méthode d'estimation du SNR basée sur la définition stricte de ce ratio (équation 3.2) constitue donc bien un facteur de mérite pour jauger de la qualité de mesures EM, tout comme la définition (équation 3.7). Plus la valeur de

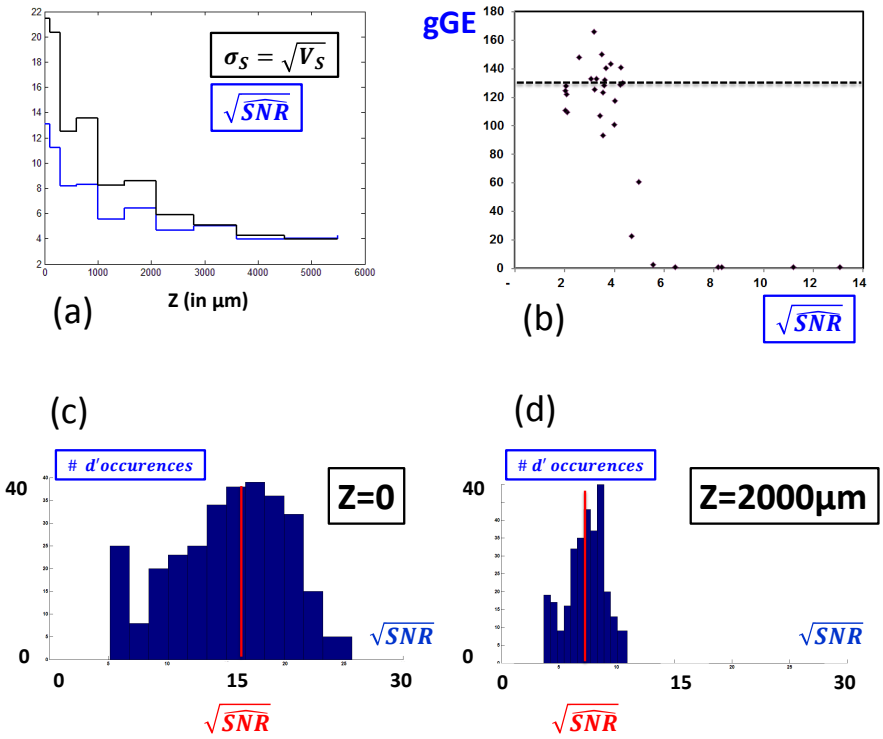


FIGURE 3.3 – (a) Évolutions de σ_S et de $\sqrt{\widehat{SNR}}$ en fonction de la distance entre la sonde et le composant (b) Évolution du gGE en fonction $\sqrt{\widehat{SNR}}$ (c) histogramme des valeurs du SNR estimé avec les traces collectées au contact du composant (d) histogramme des valeurs du SNR estimé avec les traces collectées à une distance de $Z = 2000\mu m$

SNR estimé est élevée plus les traces sont susceptibles de porter de l'information utile pour les attaques par canaux cachés. Toutefois, il convient de noter que ce facteur de mérite n'indique en rien la possible difficulté que peut rencontrer un adversaire pour exploiter cette information. Ces difficultés peuvent par exemple être :

- la présence ou non de contremesures comme le masquage [89],
- le choix d'un distingueur plus ou moins efficace [21] et générique [45],
- l'identification du modèle de fuite [77],

qui sont des problèmes indépendants de la qualité des mesures.

3.6 Exploitation du BCDC pour l'analyse de fuite

À ce stade, le BCDC a été utilisé pour estimer le SNR (équation 3.2). Il a ensuite été montré que ce critère constituait un facteur de mérite de la qualité des traces, i.e. de leur potentiel informatif en termes d'attaque par canaux cachés. Lors de ces travaux, une estimation du SNR, $\widehat{SNR}$, a été calculée pour différents jeux de traces.

Le BCDC peut également être utilisé pour analyser l'activité calculatoire dans le temps. En effet, il est tout à fait possible de calculer une valeur de BCDC pour différentes fenêtres temporelles ($[t_i, \cdots, t_k]$ avec $i > 1$ et $k \leq q$)

$$SNR([t_i, \cdots, t_k]) = \frac{1}{BCDC^2(t_i, \cdots, t_k)} - 1 \tag{3.9}$$

et d'en déduire une estimation du SNR ($\widehat{SNR}$) pour chaque fenêtre. en faisant glisser celle-ci sur l'ensemble de la trace. On obtient alors l'évolution de $\widehat{SNR}$ dans le temps.

Avant d'appliquer cette méthode sur de véritables traces, nous allons d'abord la valider sur des traces obtenues par simulation.

3.6.1 Application sur des traces de simulation

Dans ce paragraphe nous allons vérifier l'efficacité du BCDC pour l'analyse de fuites par simulation. Pour cela nous avons collecté 25 traces de simulation. Ces traces correspondent à 25 exécutions d'une exponentiation modulaire basée sur de l'échelle de Montgomery (algorithme 7), utilisant les mêmes données de 128 bits. La multiplication modulaire est effectuée à l'aide de la multiplication de Montgomery (algorithme 5), tandis que le modèle de fuite est le poids de Hamming. Plusieurs expériences ont été effectuées en faisant varier le bruit.

Pour toutes ces expériences, seuls les segments de trace correspondant au traitement du premier du bit de l'exposant sont considérés. Ainsi chaque segment consiste en une multiplication suivie d'un carré. Chaque opération est précédée d'une phase de chargement des opérandes et est suivie d'une phase d'activité du CPU.

Une première expérience avec un bruit gaussien de moyenne nulle et d'écart type égal à 1 a été faite. La figure 3.4 reporte le résultat. La figure du haut montre les 25 segments de traces et leur moyenne, la figure du milieu montre la variance, calculée verticalement et enfin la figure du bas montre l'évolution de $\sqrt{\widehat{SNR}}$ calculée par fenêtre glissante de 32 échantillons. Le bruit étant faible, la moyenne des segments de traces en noir dans le premier cadre et le $\widehat{SNR}$ permettent de distinguer les différentes parties des segments, tandis que la variance n'est pas suffisante pour distinguer ces parties.

Une seconde expérience avec un bruit gaussien élevé, toujours de moyenne nulle mais d'écart type égal à 5, a été effectuée. La figure 3.5 reporte le résultat de cette expérience. Comme le montre cette figure, lorsque le bruit est élevé, $\widehat{SNR}$

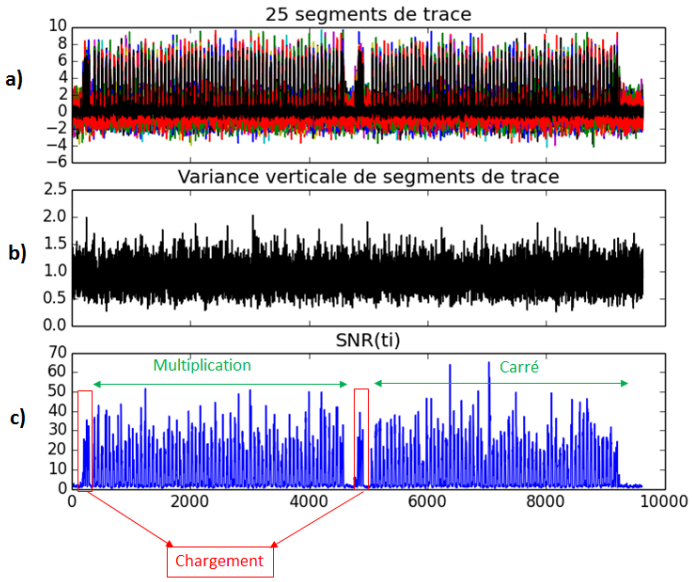


FIGURE 3.4 – (a) 25 traces de simulation de l'échelle de Montgomery (b) évolution $\sigma(T(t_i))$ et (c) évolution de $\sqrt{\widehat{SNR}}$.

permet de distinguer les différentes parties des segments. Toutefois, les parties sont moins évidentes à distinguer que dans le cas où le bruit est faible. Par contre ni la moyenne, ni la variance ne permettent de discerner ces parties lorsque le bruit est élevé.

3.6.2 Application sur des traces EM d'un AES

Le paragraphe précédent a permis de vérifier l'efficacité du BCDC pour l'analyse de traces par simulation. Nous allons vérifier si cela est avéré pour de véritables traces obtenues sur des composants réels. Pour cela, nous avons utilisé un composant exécutant un AES. La figure 3.6 reporte respectivement 25 traces avec la sonde EM placée au contact du composant FPGA, la courbe de variance associée à ces 25 traces et enfin l'évolution du $\sqrt{\widehat{SNR}}$ estimée avec le BCDC calculée sur une fenêtre de 250 échantillons, ce qui représente une durée de 0.625 fois la période de l'horloge.

Comme on peut le constater, même si la sonde est au contact du circuit et que le SNR est élevé, 25 courbes EM ne permettent pas de localiser les rondes de l'AES, ni de pouvoir interpréter le comportement du circuit, en utilisant la variance verticale des échantillons ou encore le critère NICV. En effet, le calcul du critère NICV requiert la répartition des traces dans des classes (ou clusters) définis selon les valeurs d'un octet avant de calculer des moyennes. 25 traces sont alors trop peu

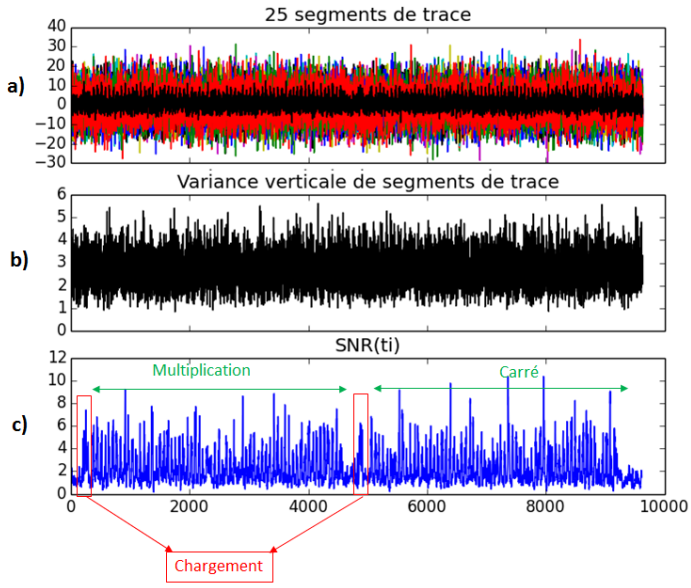


FIGURE 3.5 – (a) 25 traces de simulation de l'échelle de Montgomery (b) évolution $\sigma(T(t_i))$ et (c) évolution de $\sqrt{\widehat{SNR}}$.

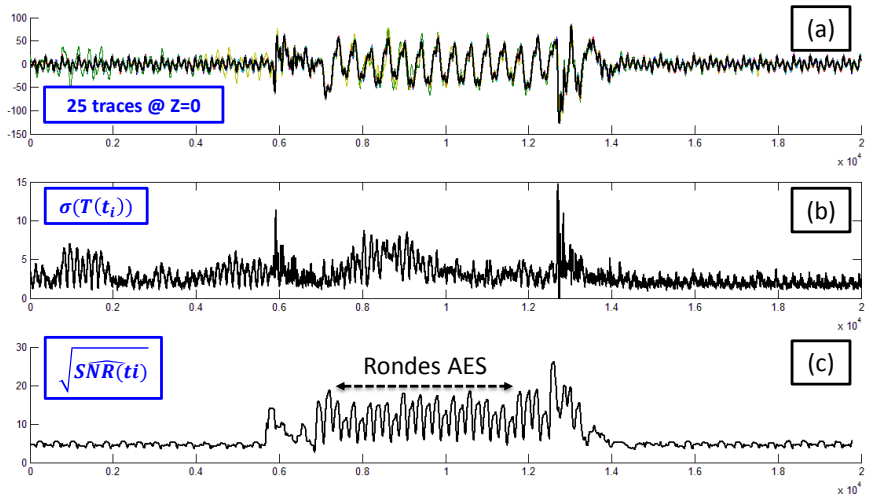


FIGURE 3.6 – (a) 25 traces EM d'un AES collectées à Z=0 (b) évolution de $\sigma(T(t_i))$ et (c) évolution de $\sqrt{\widehat{SNR}}$.

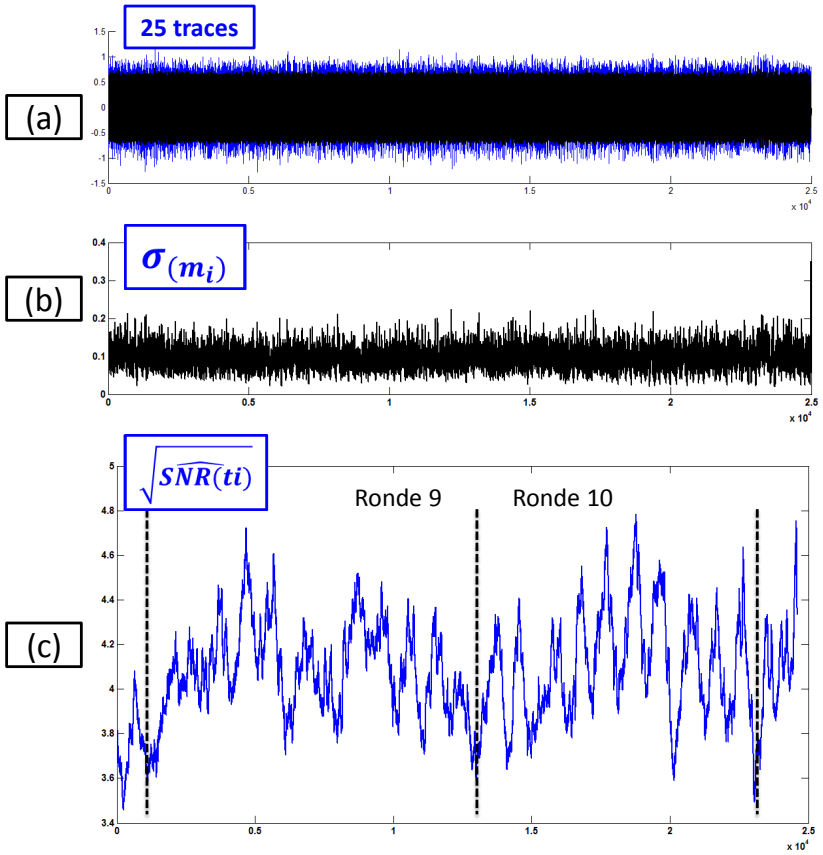


FIGURE 3.7 – (a) 25 traces EM (et leur moyenne en noir) collectées à travers un micro-contrôleur 32-bit exécutant un AES (b) évolution $\sigma(T(t_i))$ et (c) évolution de $\sqrt{\widehat{SNR}}$ dans le temps.

pour obtenir des estimations précises de ces moyennes.

En revanche, 25 traces sont suffisantes pour localiser, à l'aide du BCDC, les rondes de l'AES et identifier les différentes phases d'activité (chargement de la clé et du texte dans les registres, émission d'un signal de déclenchement sur un IO) du DUT. On notera que la différence entre les valeurs de $\sqrt{\widehat{SNR}}$ associées à des phases d'activité et celles associées à des phases d'inactivité sont très importantes.

Ce résultat peut être expliqué par deux raisons. La première est que le critère BCDC travaille sur un ensemble d'échantillons. Ceci permet de limiter l'impact du bruit. Ce n'est pas le cas avec la variance ou encore la NICV [84] qui travaillent sur chaque échantillon de manière indépendante et ce en fonction des textes appliqués en entrée dans le cas de la NICV. La seconde tient également à la réduction de l'impact du bruit et plus particulièrement d'outliers. En effet, 25 traces permettent

d'obtenir une valeur moyenne de $\widehat{SNR}$ calculée avec 300 réalisations alors que la variance et la NICV ne peuvent pas être estimées correctement avec seulement 25 valeurs.

Si les résultats, présentés ci-dessus, obtenus sur un AES matériel embarqué dans un FPGA sont intéressants, la pleine mesure de l'efficacité de la technique s'apprécie lors de l'analyse de l'activité d'un circuit effectuant un calcul logiciel. Afin d'illustrer cela, nous reportons sur la figure 3.7, 25 traces EM (et leur moyenne) collectées lors du calcul de deux rondes par un processeur cortex M4 (cadencé à 40 MHz) d'un micro-contrôleur 32 bits (technologie 90 nm), la variance estimée avec ces 25 traces et enfin l'évolution du $\widehat{SNR}$. Comme on peut le constater, ni les traces EM (ni leur moyenne en noir), ni leur variance ne permettent de distinguer les deux rondes de l'AES. Le calcul du SNR quant à lui permet de distinguer les deux rondes et même des motifs particuliers dans ces rondes.

3.6.3 Exploitation du BCDC pour l'analyse des fuites d'une trace d'exponentiation réelle

La même procédure que précédemment a été appliquée sur des segments de traces EM obtenues lors d'exécutions d'un algorithme d'exponentiation basé sur l'échelle de Montgomery par un composant moderne. La sonde est placée de telle sorte à ne distinguer que les parties correspondant au chargement des données avant chaque opération et à l'écriture du résultat après chaque opération. Ainsi les parties correspondant aux multiplications modulaires sont négligées.

Les résultats obtenus sont reportés sur la figure 3.8. Les segments de trace correspondent chacun à une multiplication suivie d'un carré. Chacune des ces opérations est composée de 3 parties : le chargement des opérandes, le calcul et l'écriture du résultat. Sur la courbe du haut sont représentés 25 segments de trace. La figure du milieu représente la variance associée à ces 25 segments de traces. Tandis que la figure du bas montre l'évolution du $\widehat{SNR}$ estimé en fonction du temps. Cette estimation est calculée sur les 25 segments traces et par fenêtre glissante de 20 échantillons qui représente un quart du cycle d'horloge du DUT utilisé. Comme nous pouvons l'observer, la variance n'est pas suffisante dans notre cas pour identifier les différentes parties des segments de trace, tandis que 25 segments de trace sont suffisants pour identifier les instants correspondant aux chargements des opérandes, au calcul et à l'écriture du résultat à l'aide du SNR estimé.

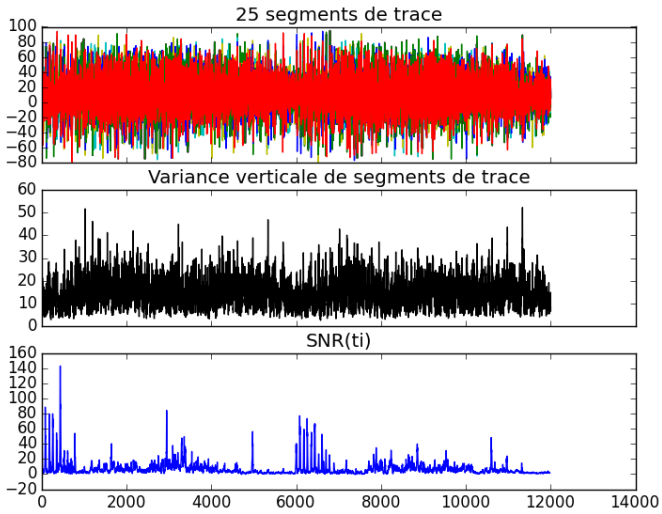


FIGURE 3.8 – 25 segments de trace EM collectés au dessus d'un circuit moderne exécutant l'échelle de Montgomery (algorithme 7) (figure du haut) la variance des segments (figure du milieu) et l'évolution du $\widehat{SNR}$ (figure du bas).

3.7 BCDC pour un filtrage adaptatif

3.7.1 Application sur des traces EM d'un AES

Dans le paragraphe précédent, le BCDC a été exploité pour analyser l'activité du circuit dans le temps grâce à l'adoption d'une fenêtre temporelle glissante. Une telle approche peut également être exploitée dans le domaine fréquentiel afin d'identifier les fréquences porteuses d'information (signal).

Pour ce faire une largeur de bande de fréquences δf est choisie. Ceci fait, les paires de traces induisant des collisions sont traduites dans le domaine fréquentiel en utilisant par exemple la transformée de Fourier rapide ou *Fast Fourier Transformation* (FFT). Pour l'ensemble des fréquences f des spectres obtenus, les harmoniques ne tombant pas dans $f \pm \delta f$ sont supprimées (mises à zéro) avant de transformer à nouveau les traces dans le domaine temporel et de finalement calculer $\widehat{SNR(f)}$ pour chaque bande de fréquences souhaitée.

Notons que la bande de fréquences adoptée peut être égale à la résolution spectrale de la FFT appliquée qui dépend du nombre de points constituant les traces et de la fréquence d'échantillonnage de l'oscilloscope utilisée pour leur acquisition. Cette approche à résolution fréquentielle maximale, qui a un coût calculatoire plus élevé, revient à faire le calcul du SNR en ne gardant dans δf qu'une fréquence.

La figure 3.9 donne l'évolution spectrale de $\widehat{SNR(f)}$ estimée à partir de 8 traces EM au-dessus d'un AES implémenté sur FPGA et ce avec $\delta f = 2$MHz. L'évolution fréquentielle de l'écart-type du bruit $\sigma_N(f)$ y est également reportée. Comme on

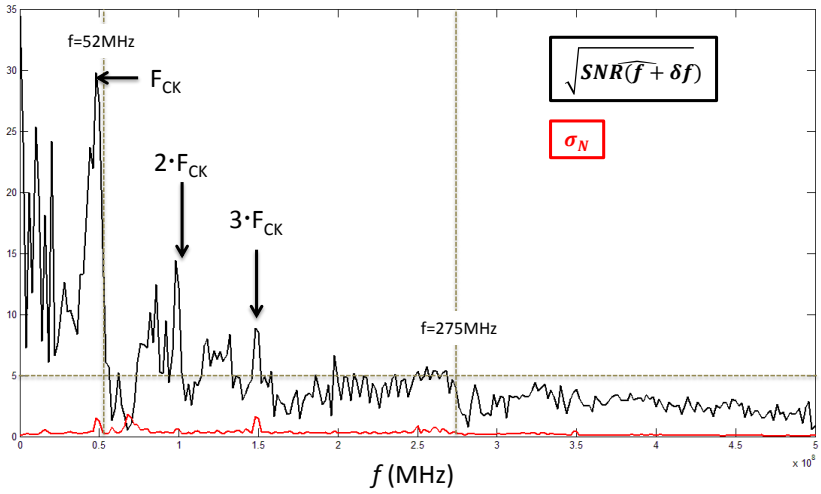


FIGURE 3.9 – $\widehat{SNR(f)}$ et $\sigma_N(f)$ calculés avec 8 traces EM.

peut le constater, cette dernière est assez plate avec des valeurs de $\sigma_N(f)$ comprises entre 1.8 (à 68MHz) et 0.05 (à 500MHz).

Contrairement à l'évolution de $\sigma_N(f)$, $\widehat{SNR(f)}$ semble décroître en $\frac{1}{f}$ et plus particulièrement en $\frac{sin(f)}{f}$ comme indiqué par la modélisation de la fuite introduite dans [78] et comme expérimentalement observé dans [90]. Toutefois, on notera que des pics de $\widehat{SNR(f)}$ apparaissent aux fréquences multiples de la fréquence d'horloge du circuit (50 MHz). Cela est probablement dû au rayonnement de l'arbre d'horloge du circuit, grand consommateur d'énergie à ces fréquences-là.

Comme on peut le constater, avec le matériel d'acquisition utilisé, les valeurs de $\widehat{SNR(f)}$ restent élevées (> 5) pour $f < 275$MHz. Compte tenu des résultats présentés en section 3.5, il semble donc pertinent de filtrer toutes les harmoniques au-delà de 275MHz ou encore de ne mener l'attaque CPA en ne conservant que les fréquences inférieures à 52MHz, bande de fréquences dans laquelle les valeurs de $\widehat{SNR(f)}$ sont particulièrement élevées.

Afin de valider que $\widehat{SNR(f)}$ est un facteur de mérite pertinente pour adapter au mieux d'éventuelles étapes de filtrage, nous avons conduit des attaques CPA sur les courbes brutes, puis sur ces mêmes courbes après suppression des harmoniques supérieures à 52MHz et 275MHz respectivement. La figure 3.10 donne l'évolution de la gGE pour ces trois attaques CPA qui ont été répétées 10 fois, et ce pour des traces collectées avec une sonde EM placée à $Z = 0$ et $Z = 4200\mu m$ de la surface du circuit.

Comme on peut le constater, pour le jeu de traces avec des valeurs de $\widehat{SNR(f)}$ élevées (mesures effectuées à Z=0), le filtrage n'a qu'un intérêt très limité. En revanche, pour les traces collectées à $Z = 4200\mu m$, caractérisées par un $\widehat{SNR}$ de 3

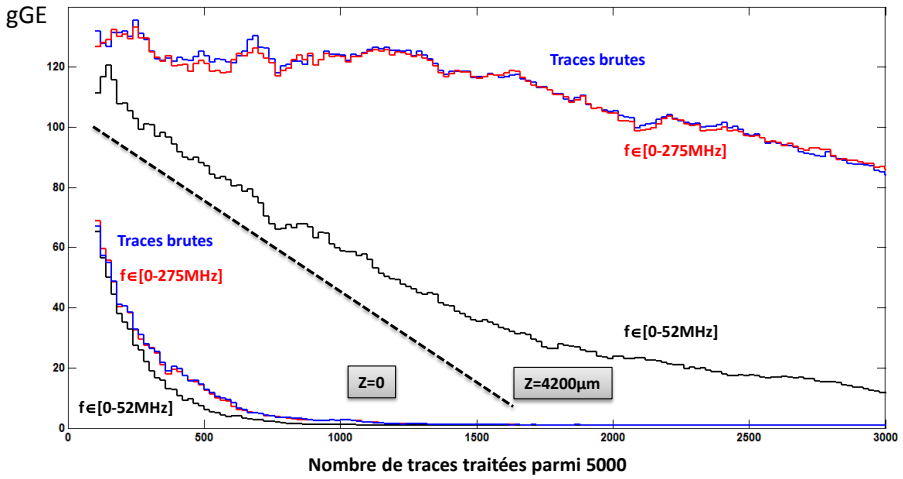


FIGURE 3.10 – Évolution du gGE avec le nombre de traces traitées (collectées soit à Z=0 ou à $Z = 4200\mu m$) gardant respectivement toutes les harmoniques (en bleu), les harmoniques en dessous de 275MHz (en rouge) et les harmoniques inférieures à 52MHz (en noir).

donc des traces relativement bruitées, ne conserver que les harmoniques ayant les valeurs $\widehat{SNR(f)}$ les plus grandes permet d'améliorer très significativement l'efficacité de la CPA. En effet, la gGE atteint la valeur 5 après le traitement des traces filtrées de sorte à ne conserver que les fréquences inférieures à 52MHz. Cette valeur est à comparer avec 60 en l'absence de filtrage. Ce résultat est en adéquation avec [90] et [78], qui soulignent l'importance des fuites en basses fréquences lors des analyses par canaux cachés. On notera également qu'avec une CPA dans le domaine fréquentiel (CPFA), la gGE vaut 63.3 après traitement de toutes les traces, résultat équivalant à celui obtenu avec la CPA.

3.7.2 Filtrage adaptatif et collision

Dans le chapitre 2, pour rendre plus efficace la méthode détection de collision basée sur le BCDC, les traces avaient été filtrées en utilisant soit la méthode compression, soit la méthode de filtrage basée sur le LNR. Nous avions constaté que le filtrage basé sur le LNR améliorait les résultats.

Toutefois, ce filtrage a été effectué à la volée et de manière empirique. Dans ce paragraphe, nous avons utilisé ces mêmes traces mais en utilisant l'évolution de $\widehat{SNR(f)}$ estimé, pour déterminer les fréquences porteuses d'information et guider le filtrage.

Pour cela, la même expérience que dans la section précédente a été effectuée sur les mêmes traces d'exponentiation modulaire avec un taux d'échantillonnage à 2.5GS/s utilisées dans le chapitre 2. La figure 3.11 présente l'évolution de $\widehat{SNR(f)}$

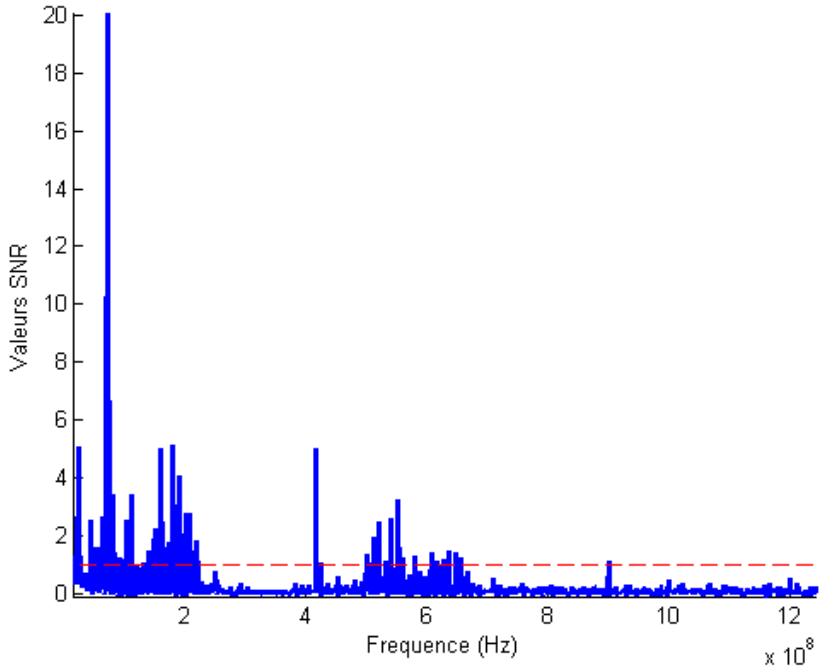


FIGURE 3.11 – Évolution $\widehat{SNR}(f)$ en fonction de f.

en fonction des fréquences avec un $\delta f = 5MHz$.

Dans cette figure, les valeurs de $\widehat{SNR}(f)$ comportent plusieurs pics d'amplitude supérieurs à 1 à certaines fréquences. Ces pics sont notamment plus présents aux fréquences f inférieures à 200MHz. En outre, aux alentours de 72MHz, le SNR présente de grandes valeurs (>8). D'après ces observations, nous avons effectué trois attaques par collisions, en gardant respectivement les fréquences 70MHz $\leq f \leq$ 75MHz, $f \leq$ 200MHz, et $\widehat{SNR}(f) \geq 1$.

La figure 3.12 montre le résultat obtenu en gardant uniquement les fréquences 70MHz $\leq f \leq$ 75MHz. Dans ce cas le seuil du BCDC n'est pas suffisant pour détecter les collisions. Il faut le combiner avec l'algorithme des k-moyennes pour déterminer les collision.

Pour améliorer les résultats, nous avons pris en compte toutes les fréquences qui sont inférieures ou égales à 200MHZ. La figure 3.13 donne le résultat de l'attaque par collision. Cette fois-ci, le seuil du BCDC permet de détecter de façon automatique toutes les occurrences de collisions.

Lorsque les fréquences telles que $\widehat{SNR}(f) \geq 1$ sont gardées, les valeurs du BCDC deviennent plus petites, ce qui rend la détection encore plus facile.

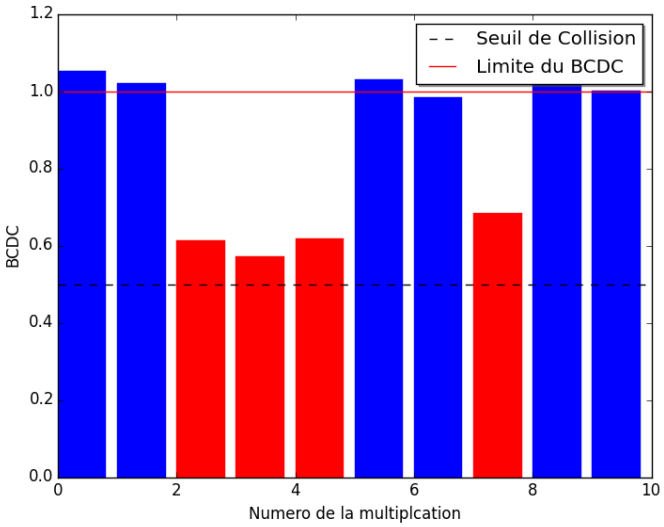


FIGURE 3.12 – Résultats de l'attaque par doublement [1] en appliquant le BCDC sur les traces filtrées par la méthode basée sur le SNR en gardant $f \in [70{,}75\text{MHz}]$.

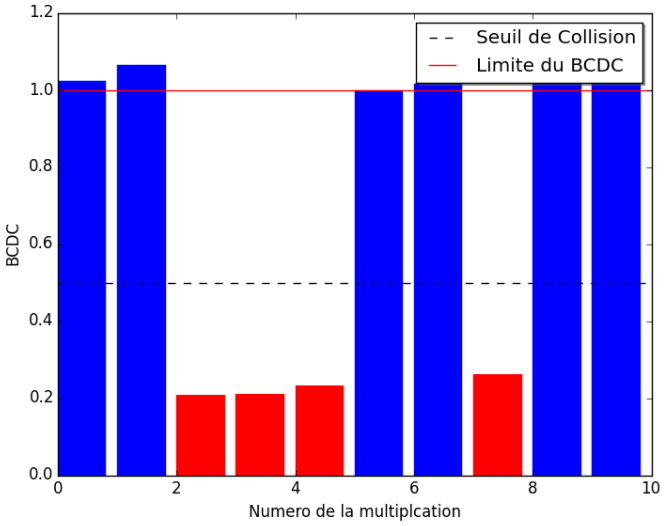


FIGURE 3.13 – Résultats de l'attaque par doublement [1] en appliquant le BCDC sur les traces filtrées par la méthode basée sur le SNR en gardant $f \leq 200\text{MHz}$.

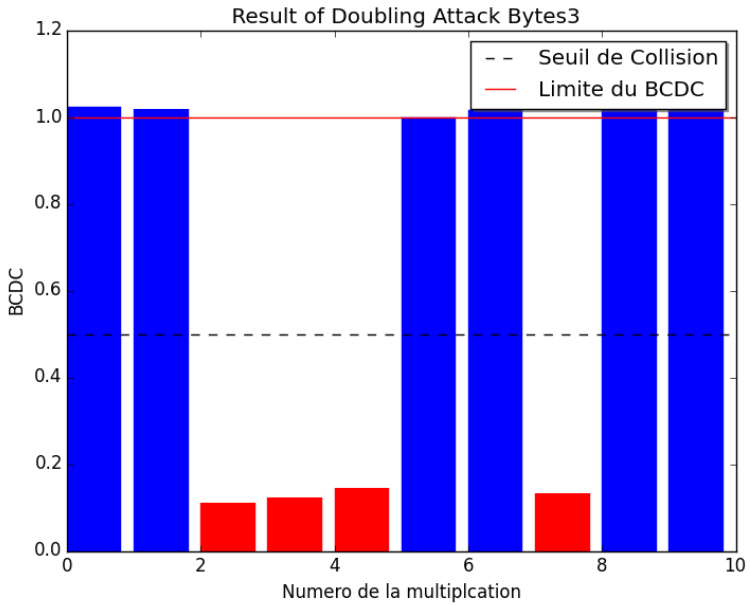


FIGURE 3.14 – Résultats de l'attaque par doublement [1] en appliquant le BCDC sur les traces filtrées par la méthode basée sur le SNR en gardant que les fréquences où $\widehat{SNR(f)} \geq 1$.

3.8 Conclusion

Le SNR est un facteur incontournable pour jauger de la qualité d'une mesure. Son utilisation requiert toutefois la connaissance du signal que l'on cherche à mesurer. Dans le contexte des analyses par canaux cachés, la nature et les caractéristiques des signaux mesurés ne sont pas connues. L'utilisation du SNR est donc particulièrement difficile. Des techniques ont été proposées pour essayer de contourner ce problème. Toutefois, celles-ci considèrent des définitions du SNR plus ou moins éloignées de la définition usuelle, et cherchent le plus souvent non pas à quantifier la qualité d'une mesure mais la proportion de variance liée au bruit de mesure et celle liée aux fuites d'informations exploitées par les analyses par canaux cachés.

Dans ce contexte une méthode permettant d'estimer le SNR lorsque le signal n'est pas connu a été proposée dans ce chapitre. Elle est basée sur le critère de détection de collision BCDC défini dans le chapitre 2. Cette technique d'estimation du SNR présente de nombreux avantages pratiques dont les plus importants sont sa simplicité de mise en œuvre, son adaptation au contexte des canaux cachés.

Il a été également montré dans ce chapitre que le SNR est un facteur de mérite intéressante pour jauger de la qualité de mesures en vue des attaques par canaux cachés, pour analyser temporellement le comportement d'un circuit, ou bien pour contrôler de manière raisonnée et efficace les étapes de filtrage couramment utili-

sées en prétraitement.

Toutefois, nous pensons que le principal intérêt de cette technique est qu'elle constitue un moyen simple, peu coûteux et pratique pour établir le SNR estimé comme technique permettant de comparer les différents protocoles expérimentaux des attaques par canaux cachés et ce tant dans les laboratoires académiques et industriels que dans les centres d'évaluation.

Chapitre 4

De la théorie à la pratique des attaques horizontales

Les attaques par canaux cachés horizontales se déroulent en plusieurs étapes. Parmi ces étapes, les plus importantes sont : l'**acquisition de la courbe complète**, **le découpage de la courbe** en segments de trace, l'**alignement** des segments de trace, **la réduction du bruit** dans la mesure du possible, l'**identification des points d'intérêt** (PoIs) et le choix d'un distingueur efficace.

Chacune de ces étapes est cruciale et conduit, si certains critères de qualité ne sont pas remplis, à une attaque infructueuse.

Dans ce contexte, ce chapitre détaille des solutions efficaces pour mener les différentes étapes d'une attaque horizontale dans la pratique, ainsi qu'un environnement permettant d'évaluer la qualité du travail effectué à chacune de ces étapes.

Les résultats de ce chapitre ont été soumis au Journal of Cryptographic Engineering (JCEN).

4.1 Introduction

Les différentes publications d'attaques horizontales décrites dans le chapitre 1 (section 1.5) reportent des expérimentations effectuées soit sur des simulations soit sur des circuits FPGA, où le SNR est souvent très élevé. Toutefois, en pratique, aucun résultat concret n'a été publié à notre connaissance sur des micro-contrôleurs modernes, où le SNR est généralement faible.

Le taux de succès d'une attaque horizontale dépend fortement de la manière dont sont effectuées les étapes successives constituant une attaque horizontale. Dans un premier temps, la courbe entière doit être acquise. Puis, la courbe doit être découpée et les différentes parties (ou segments de courbe) correspondantes aux activités ciblées doivent être resynchronisées entre elles. Enfin, après une étape de réduction du bruit et d'identification des PoIs, différentes méthodes statistiques peuvent être utilisées pour récupérer le secret. Ici, PoIs désignent les moments (points) d'une courbe contenant les informations pertinentes (informations dépendant du secret ciblé).

Toutes ces étapes sont cruciales et doivent être effectuées de manière efficace en raison de la présence limitée de fuite dans une unique trace d'exécution, contrairement aux attaques exploitant plusieurs traces d'exécution. Par exemple, la synchronisation n'a pas besoin d'être parfaite lors d'une attaque verticale car l'utilisation de plusieurs traces permet d'atténuer l'effet de la désynchronisation sur le résultat de l'attaque. Ainsi, une seule méthode de resynchronisation globale peut suffire. Au contraire, pour mener une attaque horizontale, la synchronisation est cruciale et peut nécessiter comme nous le verrons plus tard dans ce chapitre, plusieurs étapes.

Ainsi, dans un environnement réel (dans la pratique), l'application d'une attaque horizontale, et donc de ces différentes étapes, constitue une tâche difficile. Le fait qu'un attaquant doit récupérer le secret avec une seule trace rend le défi encore plus difficile. Cette difficulté fait que les attaques horizontales sont pour le moment moins exploitées dans la pratique

Dans ce contexte, le but de ce chapitre est de fournir des solutions génériques, basées sur les méthodes proposées dans les chapitres 1 et 2, permettant d'effectuer de manière efficace non supervisée et avec une précision quantifiable chaque étape d'une attaque horizontale dans la pratique (sur des (ICs) modernes).

Le reste du chapitre est organisé comme suit. Dans la section 4.2, pour chaque étape d'une attaque horizontale par canaux cachés, les problèmes que nous avons

rencontrés dans la pratique sont mis en évidence et des solutions génériques basées sur le critère BCDC sont données. La section 4.3 présente un bilan des attaques horizontales et quelques contremesures existantes. Dans la section 4.3.1, de nouvelles propositions de contremesures pour lutter contre les attaques horizontales sont introduites. Finalement, une conclusion est tirée en fin de chapitre.

4.2 Attaque horizontale en pratique

Cette section souligne les différents problèmes susceptibles d'être rencontrés durant une attaque horizontale. Des solutions sont également proposées pour surmonter ces problèmes, afin de permettre une mise en œuvre efficace des attaques horizontales en pratique.

4.2.1 L'acquisition de la trace

Une attaque horizontale sur une implémentation de l'exponentiation modulaire protégée commence par l'acquisition d'une trace entière avec un oscilloscope. Comme l'exposant est masqué (ici par un multiple de $\phi(N)$), l'acquisition de la trace doit être effectuée en une seule mesure. Il s'agit d'un premier problème car la longueur d'une trace correspondante à une exécution d'une exponentiation modulaire complète est généralement trop importante (plusieurs millions de points) et ce même pour de faibles valeurs du taux d'échantillonnage, comme indiqué figure 4.1, qui donne l'évolution de la longueur d'une trace en fonction de la taille de l'exposant et de la valeurs du taux d'échantillonnage utilisé pour la collecte.

Comme fréquemment mentionné dans la littérature, il est préférable d'utiliser le taux d'échantillonnage maximal disponible pour un oscilloscope donné, pour faciliter l'exploitation de la fuite. Malheureusement, en raison de la limitation de la profondeur mémoire des oscilloscopes mais aussi pour des raisons financières, l'acquisition de traces entières est souvent impossible. Par conséquent, la plupart des attaques sur l'exponentiation modulaire présentées dans la littérature sont faites avec des données de longueurs raisonnables (512 bits ou moins).

Pour un cryptosystème basé sur le système RSA, la longueur des données est supérieure ou égale à 1024 bits. Pour surmonter ce problème, une solution naturelle serait de réduire le taux d'échantillonnage. Cependant, selon la figure 4.2 tirée d'expériences conduites sur notre DUT et sur des données à 128 bits, pour qu'une attaque horizontale réussisse, la fréquence d'échantillonnage doit être supérieure ou égale à 2.5GS/s ; de préférence 5GS/s pour minimiser le nombre de bits erronés. Ainsi, la réduction de la fréquence d'échantillonnage n'est pas une solution pertinente pour surmonter le problème de longueur de trace car l'attaque ne sera plus effective.

Par conséquent, des solutions alternatives doivent être envisagées pour appliquer une attaque horizontale sur des exponentiations modulaires utilisant de longs exposants. Parmi les autres solutions possibles, la plus naturelle semble être l'uti-

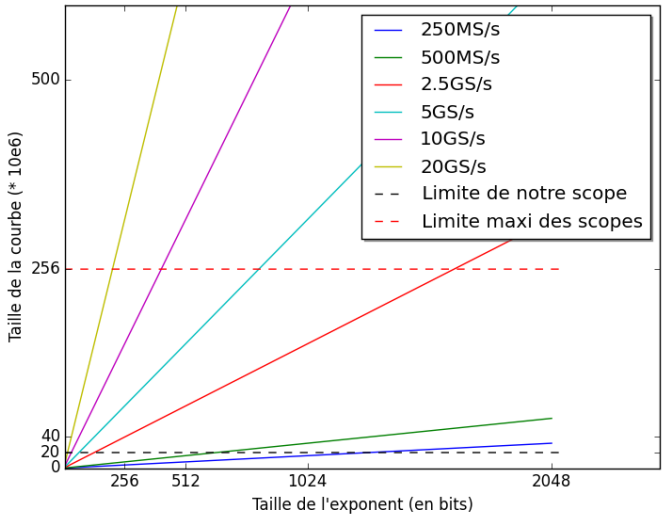


FIGURE 4.1 – Longueur de la trace en fonction de la longueur de l'exposant pour différents taux d'échantillonnage.

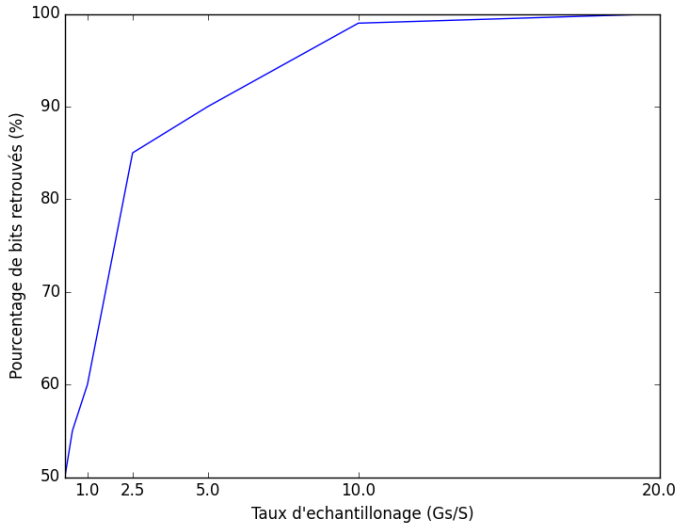


FIGURE 4.2 – Pourcentage de bits retrouvés avec l'attaque de G. Perin et al. [7] en fonction du taux d'échantillonnage.

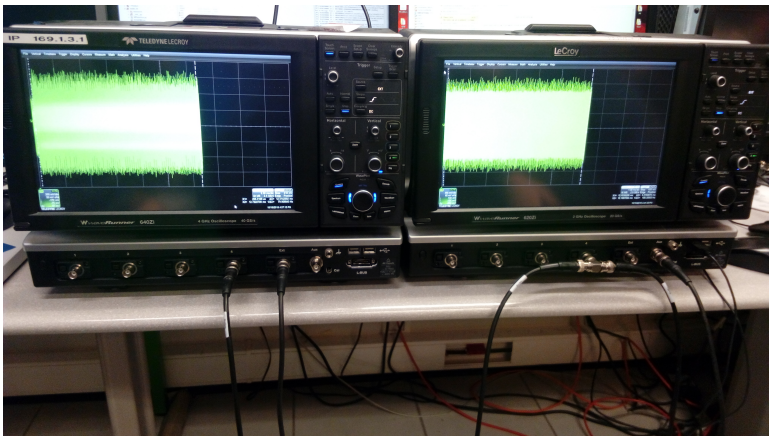


FIGURE 4.3 – Acquisition avec plusieurs oscilloscopes en serie.

lisation de plusieurs cartes d'acquisition identiques en parallèle avec des déclencheurs retardés afin de pouvoir collecter toute la trace d'exécution.

Pour cela, nous avons utilisé plusieurs oscilloscopes connectés en cascade. La sortie auxiliaire du premier oscilloscope est reliée à l'entrée externe du deuxième oscilloscope, la sortie auxiliaire du second à l'entrée externe du troisième, ainsi de suite... Une impulsion est générée par chaque sortie auxiliaire et cette impulsion déclenche l'acquisition de l'oscilloscope suivant. Un tel montage est illustré par la figure 4.3 où deux oscilloscopes de type *Lecroy WaveRunner* ont été utilisés. Une fois l'acquisition terminée, il convient de récupérer les traces acquises par les différents oscilloscopes et de concaténer ces traces afin de reconstruire la trace représentant l'ensemble de l'exponentiation modulaire. Ces traces comportent des parties en commun (recouvrement) comme le montre la figure 4.4. Ces recouvrements sont importants pour ne pas perdre de bits lors de l'acquisition.

Il convient toutefois de noter que cette limitation peut ne pas exister sur des composants à haute performance comme ceux présents dans les téléphones portables par exemple.

4.2.2 Découpage de la trace d'exécution

Dans toute attaque horizontale, le découpage de la trace ciblée en segments de trace correspondant chacun à une opération (en cas d'algorithmes irréguliers) ou à un traitement d'un bit d'exposant (algorithmes réguliers) est déterminant et doit être suffisamment précis pour permettre l'identification et la localisation des informations (PoIs) dans la trace.

Généralement, la méthode appelée corrélation-croisée (ou cross-corrélation, en anglais) est utilisée [24] pour l'extraction des segments correspondants aux traitements des bits (ou des opérations). L'idée est de sélectionner, par une inspection visuelle, un segment comme segment de référence et d'appliquer une corrélation croisée (par fenêtre glissante) de ce dernier avec la trace cible complète. Ceci per-

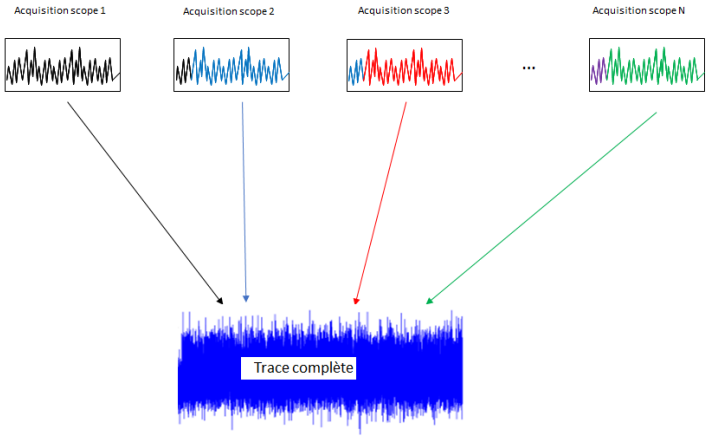


FIGURE 4.4 – Illustration de la méthode d'acquisition de la trace complète avec plusieurs oscilloscopes.

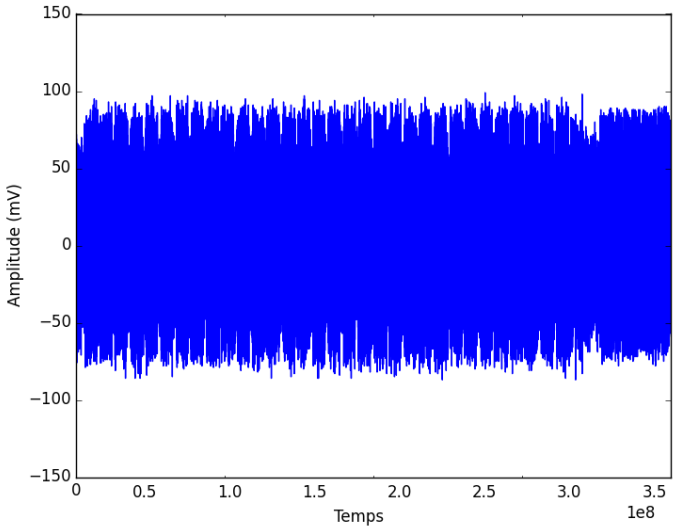


FIGURE 4.5 – Trace EM acquise avec un taux d'échantillonnage de 5GS/s durant l'exécution de l'échelle de Montgomery(Algorithme 7). La longueur de la trace est de 350 000 000 points.

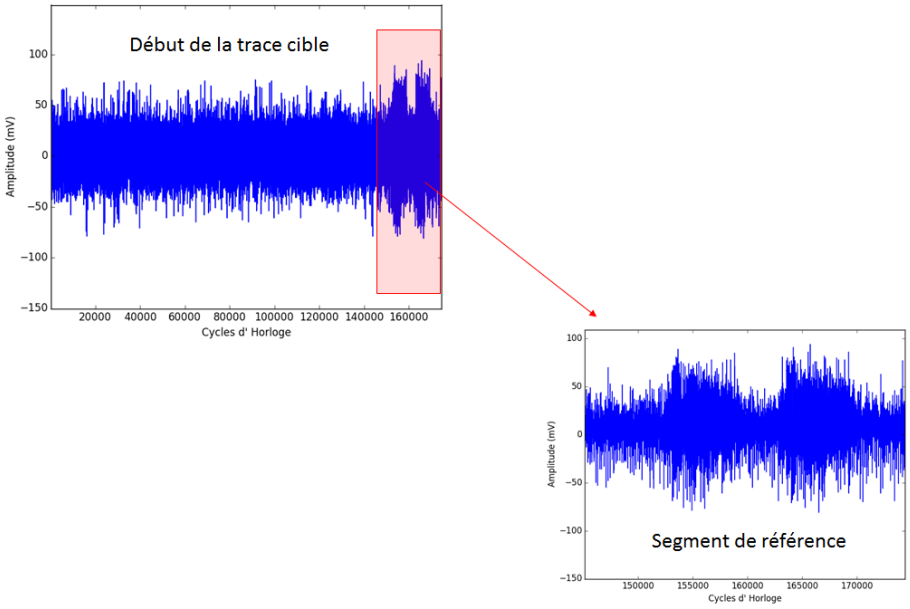


FIGURE 4.6 – Segment de trace choisi comme segment référence pour le découpage de la trace complète.

met d'obtenir la trace de corrélation croisée montrant un pic à la position de chaque interprétation d'un bit d'exposant. Ces pics apparaissent au démarrage de chaque traitement d'un bit de l'exposant lorsque le segment de référence est correct.

En guise d'illustration, cette technique a été appliquée sur la trace cible représentée sur la figure 4.5. Le début de cette trace est reporté en haut de la figure 4.6. Puisque l'algorithme ciblé est l'échelle de Montgomery, qui est régulier, le segment de référence est choisie comme étant le traitement du premier bit de l'exposant (multiplication suivie d'un carré). Ce segment est représenté par le cadre rouge de la figure du haut. Il est constitué 20000 points et est reporté sur le bas de la figure 4.6.

Le résultat de la corrélation-croisée du segment de référence et de la trace cible est représenté par la figure 4.7. En haut de cette figure, se trouve la trace de la corrélation-croisée où sont représentées 24 croix rouges. La figure du bas représente une partie de la trace cible correspondant au traitement des trois premiers octets de l'exposant. A ce stade, la question est : quels sont, parmi les pics (24 croix rouges) ceux correspondant réellement à un début de traitement d'un bit de l'exposant ?

Habituellement, une inspection visuelle est nécessaire pour définir un seuil arbitraire. Ce seuil permet de déterminer chaque position (pic) correspondant au début du traitement d'un bit. L'exactitude de ce seuil dépend fortement de la qualité des paramètres d'acquisition et du protocole expérimental utilisé. Par conséquent,

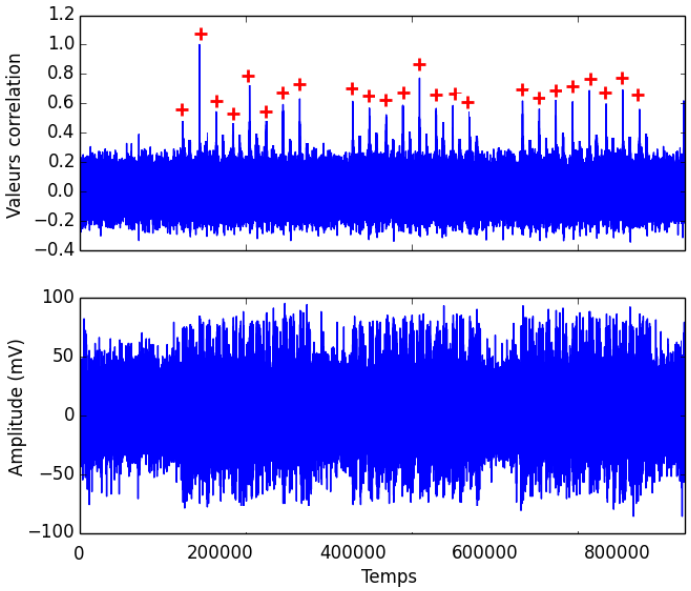


FIGURE 4.7 – Résultat de l'application de la corrélation croisée entre le segment de référence et la trace cible (uniquement le traitement des 3 premiers octets) (figure du haut), les 3 premiers octets de la trace cible (figure du bas)

il diffère d'un cas à l'autre. Si le seuil n'est pas valide, des erreurs dans l'identification des segments se produisent et l'attaque échoue.

A titre d'illustration, en raison de la variance significative de l'amplitude des pics sur la trace de corrélation croisée, la définition d'un seuil conduit soit à l'omission de parties correspondantes au traitement de bits, soit à un ajout de segments de trace qui ne constituent pas l'interprétation d'un bit d'exposant.

Afin d'éliminer cette question de définition du seuil, nous proposons d'exploiter une technique basée sur le BCDC plutôt que sur la corrélation croisée. L'idée est de remplacer le coefficient de Pearson par le BCDC, dans le calcul de la corrélation-croisée. Ce choix est justifié par le fait qu'un seuil fixe peut être considéré comme une valeur "universelle" lorsque le BCDC est appliqué. Sa valeur est égale $1/2$ comme expliqué dans la section 2.4.4. Ce seuil évite ainsi toute inspection visuelle susceptible d'introduire des erreurs dans la segmentation de la trace d'exécution.

La figure 4.8 montre le résultat lorsque l'on utilise le BCDC au lieu de la corrélation-croisée. Le BCDC est appliqué par fenêtre glissante sur une longueur de 20000 points. Le résultat est donné sur les 3 premiers octets de la trace représentée sur la figure 4.5. La trace du haut correspond à la trace d'évolution de valeurs du BCDC. On peut observer 24 pics (et seulement 24 pics) en dessous du seuil de collision $(1/2)$. Contrairement à la méthode de la corrélation-croisée la variance de la trace représentant l'évolution des valeurs BCDC est faible.

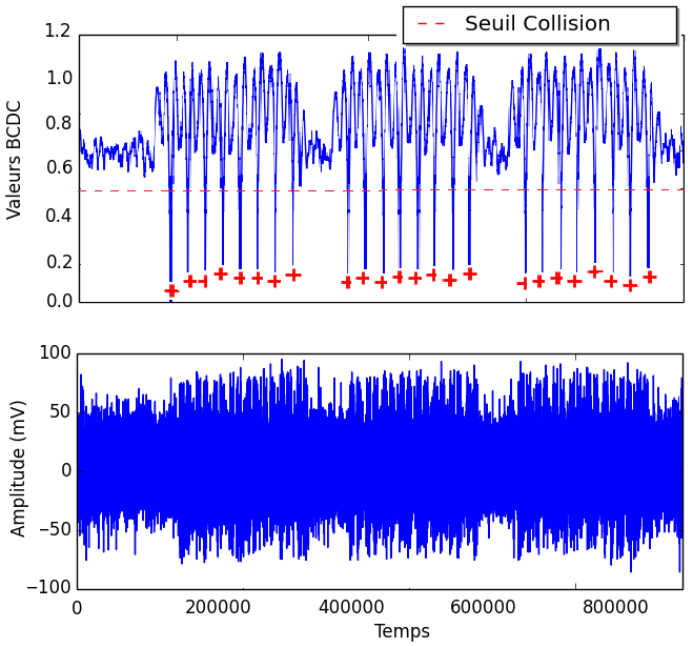


FIGURE 4.8 – Résultat de l'application du BCDC par fenêtre glissante entre le segment de référence et la trace cible (uniquement les 3 premiers octets) (figure du haut), les 3 premiers octets de la trace cible (figure du bas)

L'identification des segments de trace avec la technique basée sur le BCDC devient plus facilement automatisable que celle basée sur la corrélation croisée. Elle consiste simplement à sélectionner parmi les pics d'une amplitude inférieure à $1/2$, les n pics ayant les plus basses amplitudes dans la trace d'évolution du BCDC, où n est le nombre de bits l'exposant d utilisé dans l'algorithme d'exponentiation modulaire.

La figure 4.9 montre un exemple d'un segment de trace typique obtenu après la découpage de la courbe d'exécution avec la méthode basée sur le BCDC. Comme l'implémentation ciblée dans ce chapitre est l'échelle de Montgomery (algorithme 7), chaque segment correspondant au traitement d'un bit d_i, et chaque traitement d'un bit consiste en une multiplication suivie d'un carré (lignes 4 à 5 de l'algorithme 7).

La figure 4.10 montre l'ensemble des segments de trace obtenus après découpage de la courbe avec la méthode basée sur le BCDC. Après cette étape de découpage de la courbe, l'attaquant obtient n segments P_i (avec $1 \leq i \leq n$), où P_i correspond au traitement du bit d_i.

Dans cet exemple, en raison des contremesures environnementales implémentées dans l'IC moderne ciblé, les segments de trace obtenus, correspondant aux activités cibles (interprétation des bits), ne sont pas alignés et n'ont pas toujours la même longueur. Il est donc nécessaire d'appliquer une procédure de resynchroni-

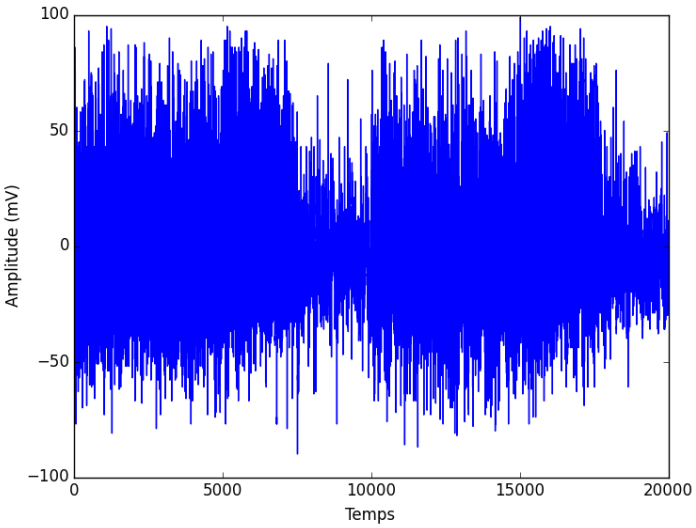


FIGURE 4.9 – Un segment obtenu après découpage de la trace cible en utilisant la méthode basée sur le BCDC.

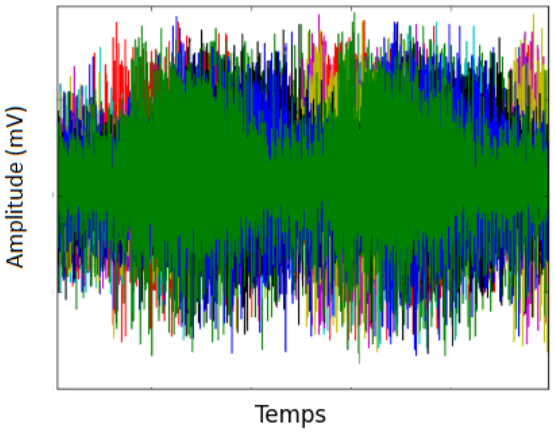


FIGURE 4.10 – 1024 segments obtenus après découpage en utilisant la méthode basée sur le BCDC.

sation.

4.2.3 Phase de Resynchronisation

Lors d'une attaque horizontale, les segments obtenus après le découpage de la courbe doivent être réalignés avec un soin afin d'identifier les PoIs, c'est-à-dire les instants portant le plus de fuite, exploités par l'outil statistique (ou distingueur) pour déterminer l'exposant secret. La resynchronisation temporelle est donc un facteur crucial pour une attaque horizontale.

La technique de resynchronisation et les algorithmes associés sont donc des éléments-clés lors d'une attaque par canaux cachés (plus particulièrement lors d'une attaque horizontale). Plusieurs méthodes ont été proposées dans la littérature [74, 91] pour améliorer les attaques par canaux cachés en présence de traces désynchronisées. Pour cette étape, nous allons procéder de la manière suivante : l'application d'une synchronisation globale suivie d'une synchronisation locale si nécessaire.

Dans ce chapitre, nous considérons uniquement la méthode de resynchronisation globale basée sur la fonction POC et l'alignement élastique basé sur l'algorithme *Dynamic Time Warping* (DTW).

4.2.3.1 Alignement statique basé sur la fonction POC

Dans [74], N. Homma et al. ont proposé une méthode de reconnaissance de motifs basée sur la fonction POC. La fonction POC [92] calcule la différence de phase entre deux traces en utilisant la phase croisée spectrale de la transformée de Fourier discrète (DFT) des traces. Considérons deux signaux (traces) $f(l)$ et $g(l)$, de taille $L = 2M + 1$. Leurs DFT respectives sont $F(k)$ et $G(k)$:

$$F(k) = \sum_{l=-M}^{M} f(l) W_L^{kl} = A_F e^{j\theta_F(k)} \tag{4.1}$$

$$G(k) = \sum_{l=-M}^{M} g(l) W_L^{kl} = A_G e^{j\theta_G(k)} \tag{4.2}$$

où $W_L = 2^{-j\frac{2\pi}{L}}$, $A_F(k)$, $A_G(k)$ sont les composantes relatives aux amplitudes et $e^{j\theta_F(k)}$ et $e^{j\theta_G(k)}$ les composantes relatives aux phases. Ainsi, la phase croisée spectrale est définie comme suit :

$$R(k) = \frac{F(k)\overline{G(k)}}{\left|F(k)\overline{G(k)}\right|} = e^{j(\theta_F(k)-\theta_G(k))} = e^{j\theta} \tag{4.3}$$

La fonction POC est l'inverse de la DFT de $R(k)$ et est donnée par $r(l)$:

$$r(l) = \frac{1}{L} \sum_{k=-M}^{M} R(k) W_L^{-kl} \tag{4.4}$$

La fonction $r(l)$ présente un pic distinct lorsqu'il existe une similarité entre $f(l)$ et $g(l)$. Le déplacement entre les deux signaux est indiqué par l'index l où se trouve le pic.

À titre d'illustration, la figure 4.11 donne le résultat de la fonction POC appliquée sur deux traces translatées de 200 points (figure 4.11a). Cette dernière comporte un pic maximal à l'index $l = 200$.

L'avantage de la fonction POC est que l'on peut disposer d'un équipement dédié qui la calcule à la volée. Cependant, l'inconvénient de cette méthode statique est de ne pas prendre en compte les différences temporelles activement induites dans les composants modernes. Par conséquent, la fonction POC n'est souvent pas suffisante pour aligner correctement les traces collectées sur des composants modernes, en particulier pour des traces EM, et tout particulièrement dans le cas de longues traces de plusieurs millions de points.

Lorsque les contremesures induisent une désynchronisation temporelle comme dans le cas d'utilisation d'une fréquence d'horloge variable, une méthode d'alignement dynamique doit être utilisée. L'alignement élastique proposé dans [91] est l'une des méthodes d'alignement dynamique les plus connues.

4.2.3.2 Alignement élastique basé sur le DTW

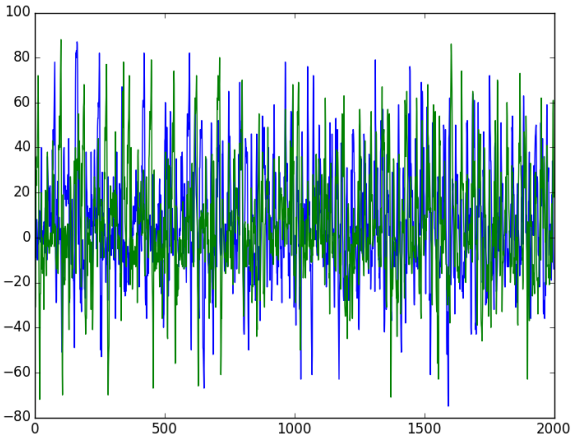
L'alignement élastique proposé dans [91] est basé sur l'algorithme de déformation temporelle dynamique ou DTW. Couramment utilisé pour la reconnaissance vocale, l'algorithme DTW permet de trouver l'alignement optimal entre deux séries temporelles, si une des séries temporelles peut être "déformée" de manière non linéaire par étirement ou rétrécissement le long de l'axe temporel. La déformation est ensuite utilisée pour trouver les similitudes entre les deux séries temporelles.

Soient la trace $X = (x_1, x_2, \cdots, x_n)$ de référence et la trace à réaligner $Y = (y_1, y_2, \cdots, y_m)$. On considère une matrice D de dimension $(n * m)$ où n et m sont respectivement le nombre d'éléments dans la trace de référence et dans la trace à réaligner. À chaque entrée (i, j) de cette matrice, on associe la distance locale $d(x_i, y_j)$. Pour rechercher la meilleure distance $D(X, Y)$ entre la trace à réaligner Y et la trace de référence X, il suffit alors de rechercher le "chemin" dans cette matrice D pour aller du point initial (1, 1), correspondant au début des deux traces, au point final (n, m), correspondant à la fin des deux traces en progressant de proche en proche de façon à minimiser la somme des distances.

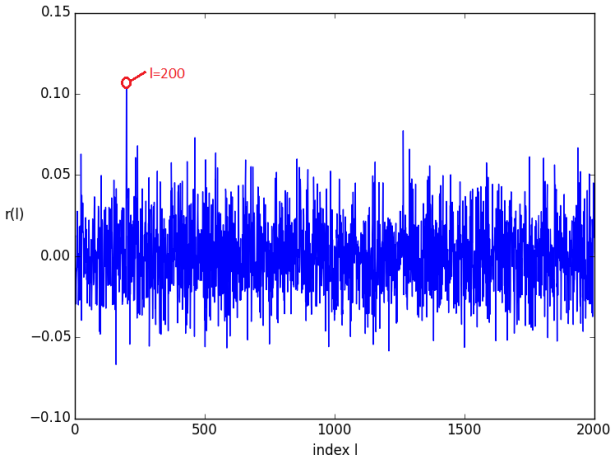
La mise en œuvre de cet algorithme se fait de façon simple en calculant, pour chaque entrée (i, j) la distance accumulée $D(i, j)$ correspondant à la distance optimale que l'on obtient au fur et à mesure de la progression. On peut montrer que cette distance peut se calculer en utilisant la formule de récurrence suivante développée dans ([93]) :

$$D(i, j) = d(i, j) + min_{p(i,j)} \{D(p(i, j))\} \tag{4.5}$$

avec :



(a) Traces translatées de 200 points.



(b) Resultat de la fonction POC.

FIGURE 4.11 – Illustration du calcul de la fonction POC sur deux courbes translatées de 200 points.

- $p(i, j)$: ensemble des prédécesseurs possibles de l'élément (i, j)
- $D(i, j)$: distance globale
- $d(i, j)$: distance locale

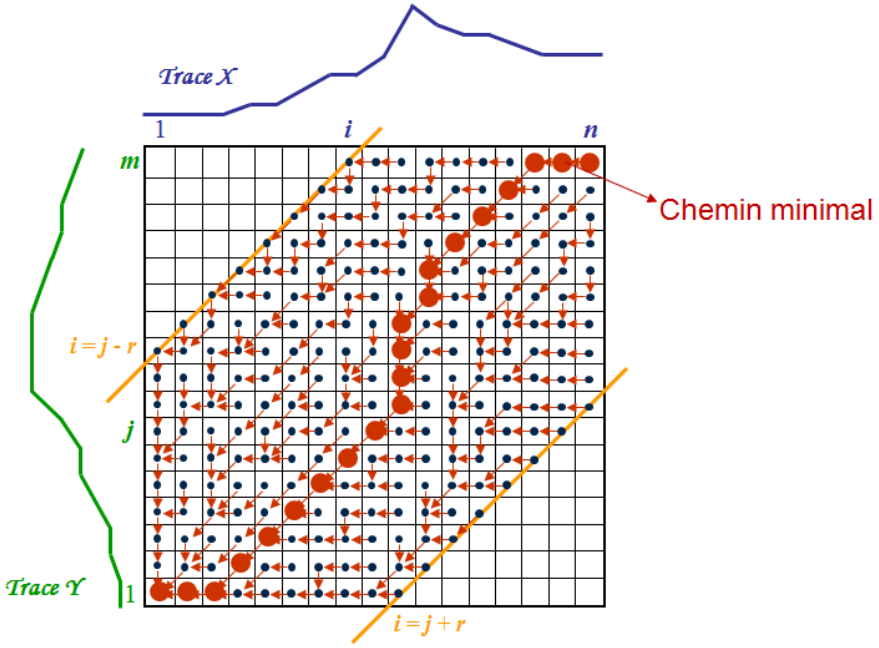


FIGURE 4.12 – Figure illustrant un chemin parcouru entre deux traces de longueurs différentes.

Les prédécesseurs peuvent être choisis afin d'obtenir une trajectoire monotone et plausible. Celle-ci doit rester le plus près possible de la diagonale. On peut également associer des pénalités aux prédécesseurs les moins probables afin de favoriser certains profils de chemin. L'algorithme implémenté tient compte des zones où le calcul des distances ne fait pas sens. C'est pour cette raison en particulier que pour la zone supérieure gauche et la zone inférieure droite les distances ne sont pas calculées. Les distances locales associées sont initialisées à une valeur très élevée afin que le chemin trouvé ne puisse pas y passer. Cette procédure permet d'obtenir un chemin minimal $F = (c(1), c(2), \cdots, c(K))$ (avec $\min(n, m) \leq K \leq n + m$).

Un exemple d'illustration est donné par la figure 4.12 dans laquelle le chemin minimal est représenté en rouge.

Une fois le chemin minimal obtenu, les nouvelles traces, notées $\dot{X}$ et $\dot{Y}$, peuvent être calculées comme suit :

$$\dot{X}[i] = X[i] \tag{4.6}$$

$$\dot{Y}[j] = \frac{1}{|\{k\ ,\ x(k) = j\}|} \sum_{x(k)=j} Y\left[y(k)\right] \tag{4.7}$$

avec $c(k) = (x(k), y(k))$, $1 \leq k \leq K$, $1 \leq i \leq n$, $1 \leq j \leq m$.

Plus de détails pour l'algorithme DTW peuvent être trouvés dans la thèse de S. Haidar [93].

4.2.3.3 Combiner alignement statique et élastique

Pour une application correcte de l'algorithme DTW, il est nécessaire que les premiers éléments et les derniers des deux traces soient au préalable égaux (c'est à dire $x_1 = y_1$ et $x_n = y_m$). Malheureusement, après le découpage de la trace d'exécution d'un composant moderne (figure 4.10), cette condition n'est pas toujours remplie. La question à ce stade est alors : comment pouvons-nous faire pour que cette hypothèse soit toujours réalisée ?

Pour résoudre ce problème, nous proposons l'application conjointe d'une méthode de synchronisation statique et d'une méthode de synchronisation élastique. En effet, un alignement statique, comme la méthode basée sur la fonction POC ou d'autres méthodes d'alignement statiques existantes, permettent de resynchroniser globalement les segments. Cette première étape permet ainsi de réaliser la condition initiale de l'alignement élastique pour une application efficace de l'algorithme DTW.

La figure 4.13 montre le résultat de l'alignement global appliqué sur les segments représentés sur la figure 4.10 en utilisant la méthode basée sur la fonction POC. Visuellement, les segments semblent parfaitement alignés, mais en observant de plus près (figure 4.14), on remarque que les segments sont encore localement non-alignés. Afin d'affiner la synchronisation, l'alignement local et élastique basé sur l'algorithme DTW est appliqué. Après cette étape, les segments deviennent globalement et localement alignés comme le montrent les figures 4.15 et 4.16.

Visuellement, l'utilisation conjointe des techniques d'alignement statique (global) et élastique (local) fournit une procédure d'alignement efficace en présence de contremesures environnementales. Cependant, puisque ce travail vise à automatiser les différentes étapes d'une attaque horizontale, une question importante se pose : comment est-il possible de valider (lors d'une attaque horizontale ou plus généralement d'une attaque par canaux cachés) que cette procédure de synchronisation est efficace ?

4.2.3.4 Validation de la procédure de resynchronisation

Généralement, la variance ou la moyenne des traces sont utilisées pour la validation d'une procédure de resynchronisation. Ces techniques nécessitent une inspection visuelle afin de valider la procédure de resynchronisation. Dans cette section, nous proposons une stratégie différente basée sur les collisions de signaux

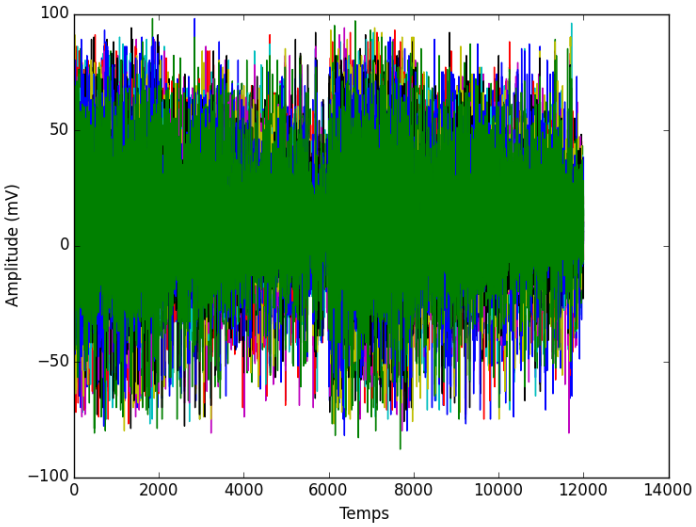


FIGURE 4.13 – Segments alignés avec la méthode basée sur la fonction POC seulement.

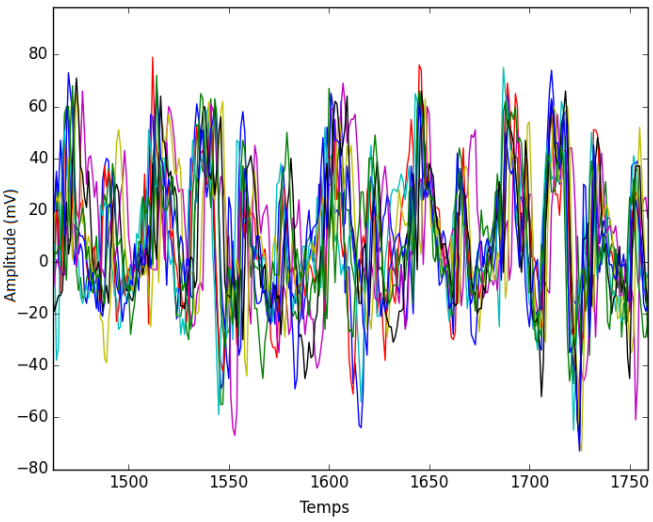


FIGURE 4.14 – Zoom sur une partie de la figure 4.13.

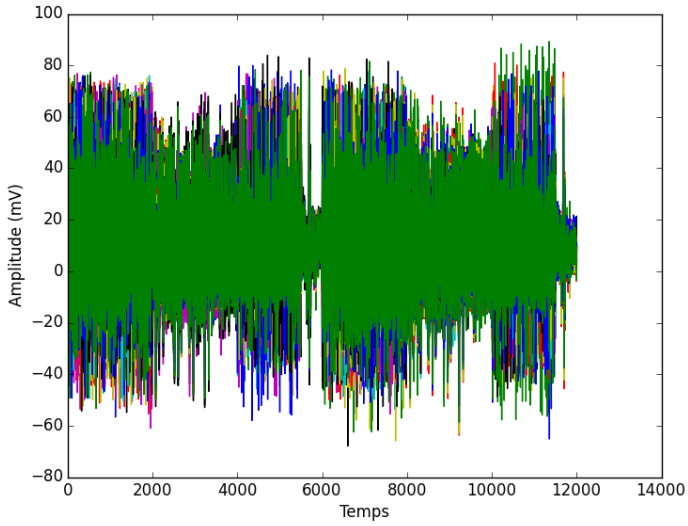


FIGURE 4.15 – Segments alignés avec la combinaison des méthodes basées sur la fonction POC puis sur le DTW.

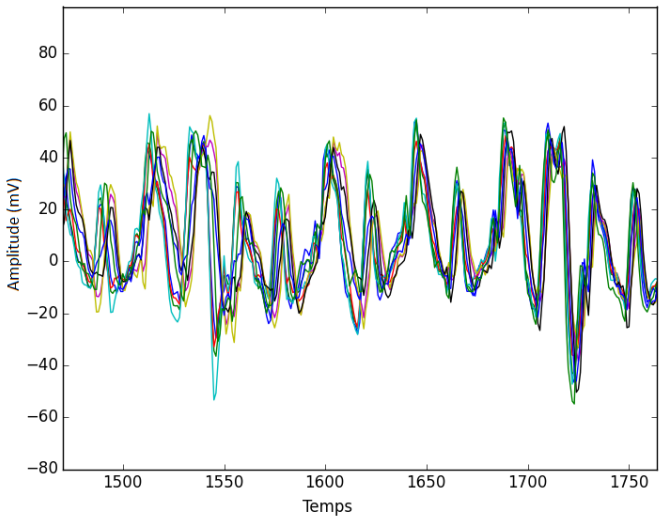


FIGURE 4.16 – Zoom sur une partie de la figure 4.15.

pour automatiser la procédure de resynchronisation. En effet, si deux traces représentant le même signal sont parfaitement synchronisées, une collision doit apparaitre entre les deux traces et cela en tout point.

Pour cela, nous avons acquis n mesures M_i avec $i = 1, \cdots, n$. Ces mesures correspondent à l'exécution de l'échelle de Montgomery (algorithme 7) avec les mêmes entrées et sans masquage des données. L'idée ici est d'utiliser les collisions entre ces mesures afin de vérifier la qualité de la procédure de resynchronisation après chaque étape de la procédure de resynchronisation (POC puis DTW). Cela se fait en calculant pour chaque couple de mesures, les valeurs du BCDC par fenêtres glissantes. Cette étape fournit un ensemble de $\frac{1}{2} \cdot n \cdot (n-1)$ traces d'évolution du BCDC. Ensuite, la moyenne de ces $\frac{1}{2} \cdot n \cdot (n-1)$ traces d'évolution du BCDC est calculée. Ainsi on obtient la trace de l'évolution moyenne des valeurs du BCDC. Cette trace moyenne est représentée pour chaque étape de la procédure de resynchronisation respectivement par les figures 4.17a, 4.17b et 4.17c. Plus les valeurs contenues dans la trace moyenne d'évolution du BCDC sont faibles, meilleure est la procédure de resynchronisation. En outre, si ces valeurs sont supérieures à $1/2$, les traces sont considérées comme désynchronisées.

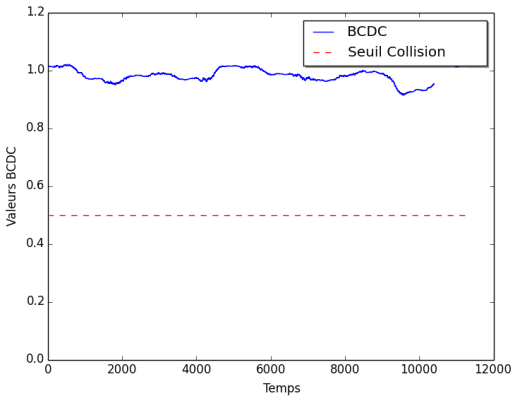
La figure 4.17a donne les valeurs du BCDC obtenues en utilisant 10 segments de trace parmi ceux obtenues après l'étape de découpage, et avant l'application de toute technique de ré-alignement. Les valeurs moyennes de BCDC oscillent autour de 1. Ainsi d'après notre critère les traces sont désalignées.

En appliquant sur ces mêmes segments, la technique d'alignement basée sur la POC, i.e une technique de resynchronisation globale et statique, les valeurs de la trace moyenne d'évolution du BCDC diminuent significativement, mais restent autour de 0.65 (figure 4.17b). Ces valeurs sont donc supérieures au seuil de collision et donc les segments sont toujours considérés comme désynchronisés.

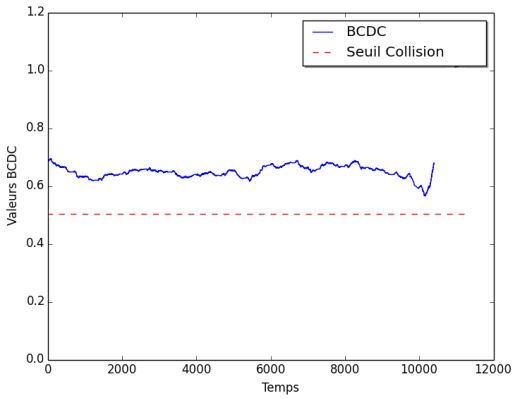
La combinaison des techniques basées sur la fonction POC et sur l'algorithme DTW conduit à des valeurs moyennes BCDC oscillant autour de 0.38 (figure 4.17c). Les valeurs du BCDC sont devenues inférieures au seuil de collision. Cela signifie que des collisions se produisent entre les segments. Les segments sont donc considérés resynchronisés d'après notre critère. Cela indique également que le SNR des mesures est égal à 1.6, les traces sont donc relativement bruitées.

Ces expériences valident l'efficacité de la procédure de resynchronisation proposée sur notre DUT. Cette procédure implique l'utilisation conjointe d'une technique d'alignement statique et global (POC) et d'une technique d'alignement élastique et local (DTW). Une telle procédure est consommatrice en temps mais efficace sur les circuits intégrés modernes embarquant des contremesures environnementales.

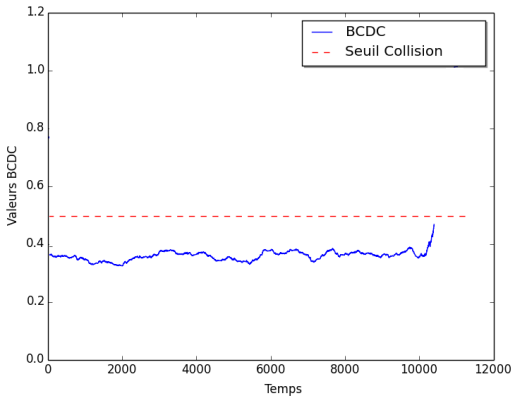
Après la resynchronisation des segments de trace, l'étape suivante d'une attaque horizontale consiste à réduire autant que possible le bruit. C'est une étape qui peut être cruciale pour la réussite d'une attaque horizontale.



(a) Évolution du BCDC avant la procédure de synchronisation.



(b) Évolution du BCDC après synchronisation avec la méthode basée sur la POC.



(c) Évolution du BCDC après synchronisation avec la POC et le DTW.

FIGURE 4.17 – Évolution du BCDC pour les différentes étapes de la méthode de resynchronisation.

4.2.4 Réduction du bruit

La réduction du bruit est largement utilisée dans le domaine des attaques par canaux cachés dans le but d'augmenter le taux de succès de celles-ci. Plusieurs méthodes de réduction du bruit sont proposées dans la littérature. Habituellement, les méthodes de filtrage (passe-bas, passe-haut, passe-bande, etc) sont utilisées pour réduire la présence du bruit dans les traces brutes. D'autres méthodes comme la soustraction de la trace moyenne (consistant à calculer la différence point à point entre chaque trace et de la trace moyenne), le filtrage basé sur la moyenne mobile (ou moving-average) ou encore les techniques de compression [77] sont déjà utilisées dans les attaques par canaux cachés. Presque toutes ces méthodes nécessitent une certaine connaissance du DUT.

Généralement la connaissance de la fréquence d'horloge utilisée est nécessaire pour un filtrage efficace. Concernant les techniques de compression, la durée d'un cycle d'horloge est également requise. Malheureusement, ces connaissances sont généralement incomplètes, ainsi les techniques de filtrage ou de compression ne sont pas optimales et peuvent même, si elles sont utilisées avec de mauvais paramètres, conduire à une réduction du taux de succès.

Dans cette section, l'estimation du SNR grâce au BCDC (équation 3.6) présentée dans les chapitres précédents est utilisée à cette fin. Afin de supprimer les informations non pertinentes dans la trace (porteuses essentiellement de bruits), la même stratégie que dans la section 3.7 est effectuée.

La figure 4.18 donne l'évolution avec f de $\widehat{SNR(f \pm \delta f)}$ estimée avec seulement 10 segments de trace parmi les segments obtenus après la phase de découpage de la courbe (section 4.2.2). La largeur de bande de fréquence, δf, est égale à $5MHz$. On observe des valeurs $\widehat{SNR}$ élevées (>5) à certains multiple de la fréquence f_{ck} du DUT mais également pour les basses fréquences.

La figure 4.19 montre deux segments filtrés de sorte à ne garder respectivement que les fréquences pour lesquelles $SNR(f) \geq 1$ (segment noir) et $SNR(f) \geq 3$ (segment vert). On peut remarquer que les résultats (segment en vert et segment en noir) semblent moins bruités que le segment brut en bleu.

4.2.4.1 Validation de la méthode de réduction du bruit

La validation de l'efficacité de la méthode de filtrage est effectuée à l'aide du critère BCDC comme dans la section 4.2.3.4. La figure 4.20 donne le résultat obtenu. Après filtrage en gardant les fréquences pour lesquelles $SNR(f) \geq 3$, les valeurs moyennes du BCDC sont de l'ordre de 0.15, ce qui signifie que le SNR des traces filtrées est égal à 5.6 au lieu de 1.6 (valeur obtenue figure 4.17c). Ces valeurs de BCDC, indiquent que les segments filtrés contiennent maintenant environ 5 fois moins de bruits et sont donc prêts pour la phase suivante consistant à identifier les PoIs.

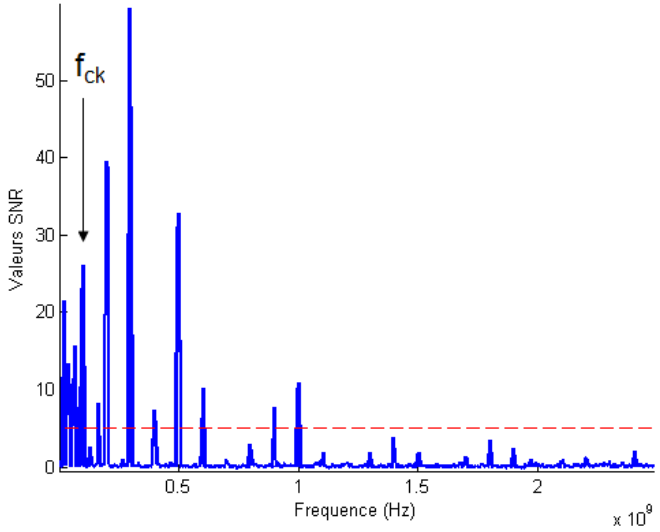


FIGURE 4.18 – Évolution du $\widehat{SNR}$ en fonction des fréquences f.

4.2.5 Identifications des PoIs

Identifier les échantillons de temps contenant les fuites pertinentes constitue une tâche importante dans une attaque horizontale non supervisée. Différentes méthodes d'identification des PoIs ont été présentées dans la littérature : la méthode basée sur la différence des moyennes (DPA) [18, 22], sur l'analyse par corrélation (CPA) [21], sur le test t de Welch [94, 50], sur l'analyse de la variance (NICV) [84], sur le rapport signal sur bruit (SNR) [77], ou encore sur l'analyse en composant principale (PCA) [95] etc.

Cependant, ces méthodes sont souvent utilisées de manière supervisée (lors d'une attaque par template [22] par exemple), où un échantillon ouvert est disponible. En d'autres termes, ces méthodes nécessitent la connaissance de la clé pour identifier les PoIs. Par exemple, nous pouvons effectuer une CPA au préalable pour connaître les PoIs.

Dans ce qui suit, nous allons nous attacher à l'étude du cas non supervisé en améliorant la méthode présentée dans la sous-section 1.5.5.2

4.2.5.1 Amélioration de la méthode proposée par G. Perin et al.

Dans [7] l'algorithme de classification non-supervisé des k-moyennes (algorithme 14) est utilisé pour l'identification les PoIs. Cette technique commence par la définition d'une matrice P dont les lignes sont les n segments de trace obtenus après les phases de découpage et de resynchronisation :

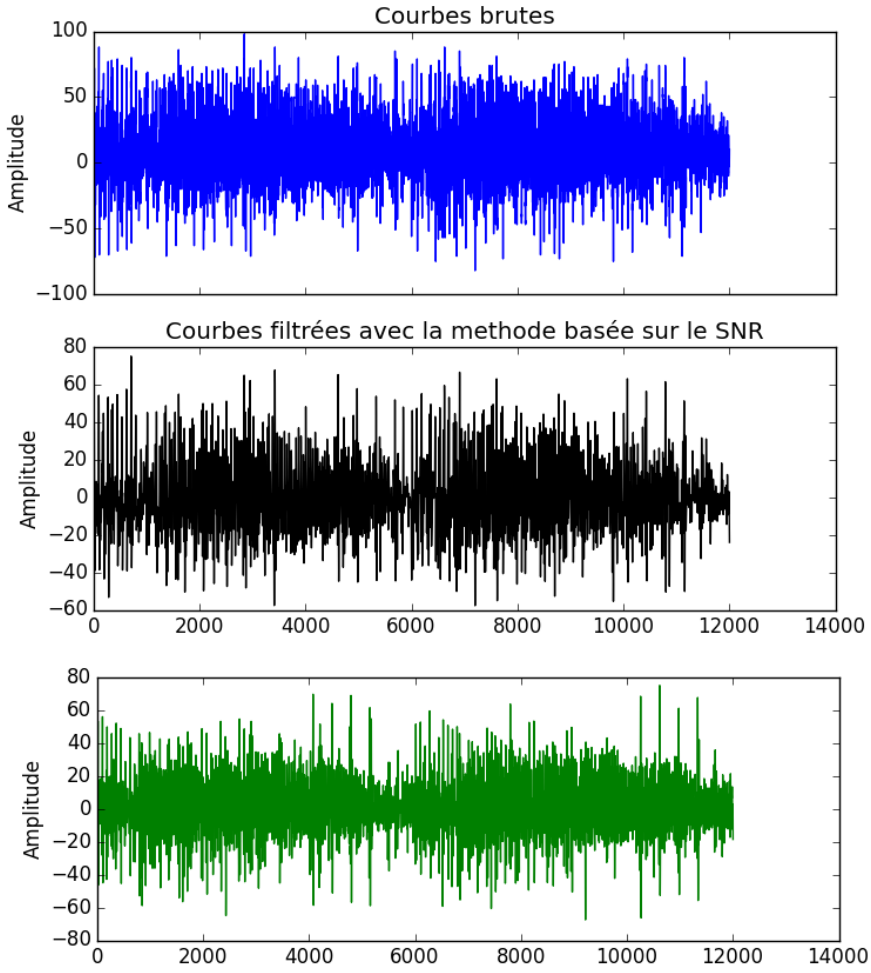


FIGURE 4.19 – Courbe brute (en bleu) et courbes après filtrage (en vert et en noir) avec la méthode basée sur le SNR estimé.

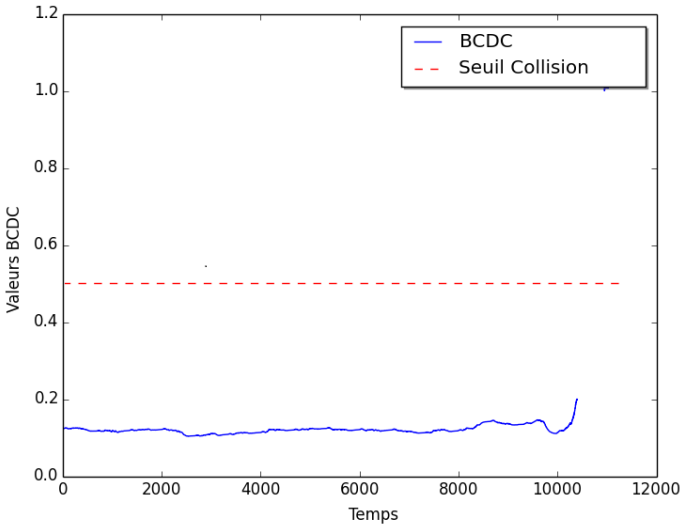


FIGURE 4.20 – Évolution du BCDC après filtrage avec la méthode basée sur le SNR estimé.

$$P = \begin{bmatrix} P_1 \\ P_2 \\ \vdots \\ P_n \end{bmatrix} = \begin{bmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,q} \\ p_{2,1} & p_{2,2} & \cdots & p_{2,q} \\ \vdots & \vdots & \ddots & \vdots \\ p_{n,1} & p_{n,2} & \cdots & p_{n,q} \end{bmatrix}$$

où $p_{i,j}$ est l'échantillon associé à l'instant j du segment P_i ; ce dernier étant constitué de q points consécutifs, correspondant au traitement du bit d'exposant d_i (ligne 4 à ligne 5 de l'algorithme 7).

En appliquant l'algorithme des k-moyennes avec $k = 2$ sur chaque colonne de P, l'attaquant obtient q exposants estimés $\widehat{d_{1:n,j}}$ (avec $1 \leq j \leq q$). Ici, $\widehat{d_{1:n,j}}$ est l'exposant estimé obtenu avec l'algorithme des k-moyennes (algorithme 14) sur la colonne j.

Pour chaque exposant estimé, l'attaquant calcule la différence des moyennes (DoM), entre les segments correspondant aux bits d'exposant estimés égaux à 0 (appelé T_{r_0}) et ceux estimés égaux à 1 (appelé T_{r_1}). La différence des moyennes est alors sensée fournir des pics aux positions où les deux moyennes diffèrent significativement, et ainsi permettre l'identification des points (PoIs) qui distinguent deux classes T_{r_0} et T_{r_1}.

Cependant, en raison du bruit, il est souvent difficile, avec peu de segments de décider ce qui est un pic significatif et ce qui n'est pas un, et donc de sélectionner les points d'intérêt et de déterminer quel est le bon nombre de PoI. Une autre façon de procéder, avec l'avantage d'une automatisation possible, est d'utiliser le test t

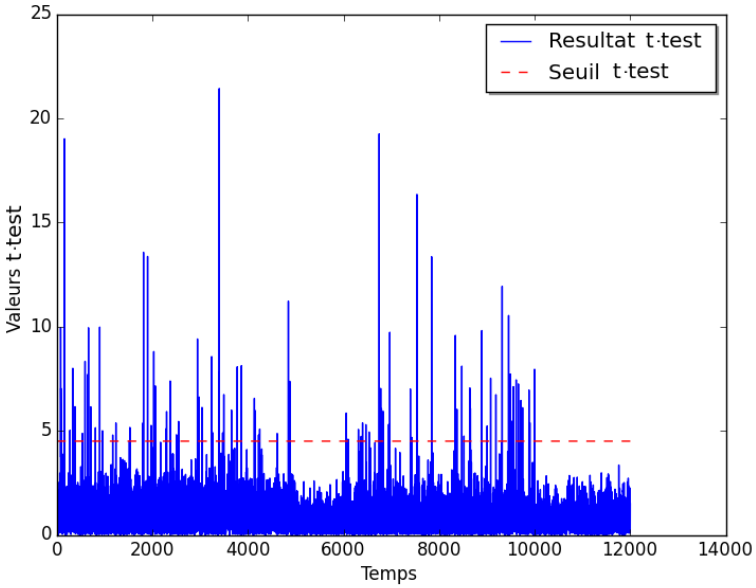


FIGURE 4.21 – Résultat du test de Welch en considérant les estimations de l'algorithme des k-moyennes sur la colonne h correspondant à un PoI.

de Welch [46] (comme détaillé dans la section 1.3.4.2). Toutes les instants i avec une valeur de t supérieure à 4.5 sont considérées comme des positions où les deux familles ont des moyennes différentes. Ces positions sont sélectionnées comme PoI. Nous noterons l par la suite le nombre de PoIs.

La figure 4.21 donne les valeurs du test t de Welch, obtenues à une position h en utilisant l'algorithme des k-moyennes, où h est une position dans laquelle le secret est manipulé, tandis que la figure 4.22 donne les valeurs du test de Welch sachant le vrai exposant. Les deux figures sont très similaires et comportent plusieurs échantillons de temps pour lesquelles les deux ensembles T_{r_0} et T_{r_1} sont différents.

Inversement, la figure 4.23 montre les valeurs de la statistique t obtenues à un instant pour lequel la trace ne contient pas d'information sur le secret manipulé, c'est-à-dire une position qui ne constitue pas un PoI. Comme attendu, il n'y a aucune valeur au-dessus 4.5. Ces résultats montre la pertinence de cette méthode même sur notre circuit de test.

La solution proposée dans [61] en utilisant la PCA pour identifier les PoIs peut également être utilisée. Mais l'efficacité de cette méthode dépend fortement du DUT. En effet, selon les DUT, les composantes principales ne sont pas forcement celles qui véhiculent de l'information. Le problème de la sélection des composantes d'intérêt n'est pas résolu dans la littérature et une étape de profilage est souvent nécessaire pour contourner ce problème.

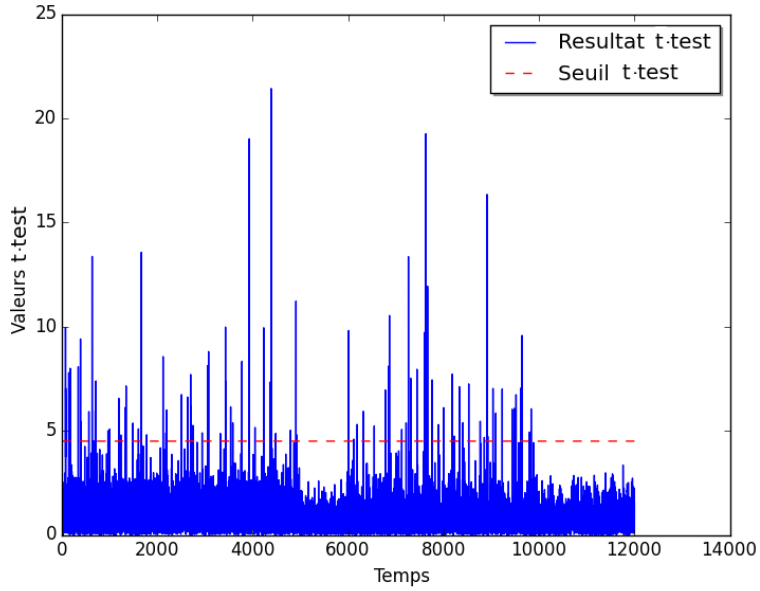


FIGURE 4.22 – Résultat du test de Welch en considérant l'exposant réel.

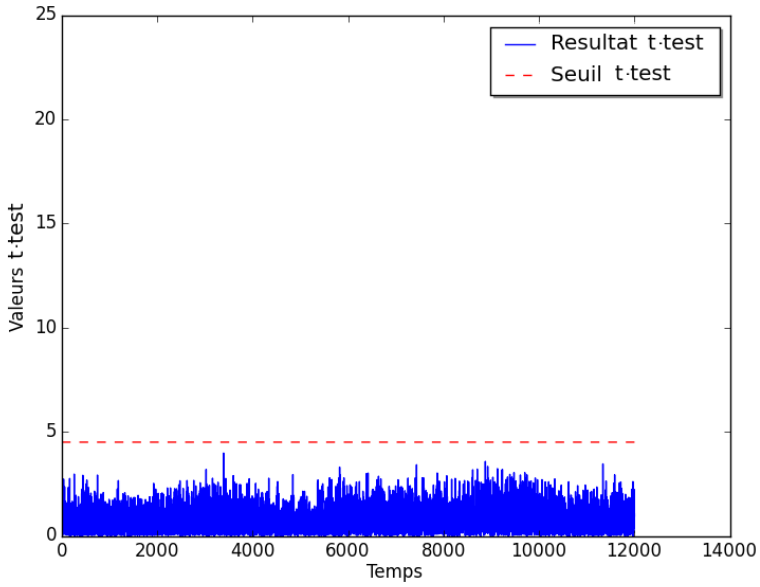


FIGURE 4.23 – Résultat du test de Welch en considérant les estimations de l'algorithme des k-moyennes sur une colonne ne correspondant pas à un PoI.

4.2.6 Choix d'un distingueur efficace

Après la sélection des PoIs, la phase finale d'une attaque horizontale consiste à retrouver les valeurs des bits de l'exposant. Pour cela, l'attaquant doit transformer chaque segment P_i en $\widetilde{P_i}$ en ne conservant que les PoIs et les comparer avec un segment référence noté ici $\widetilde{P_{ref}}$. A l'issue de ce travail, il obtient deux ensembles : les segments de traces proches de P_{ref} et les autres segments. Ainsi, il en déduit les valeurs des bits d'exposant. La comparaison peut être effectuée en utilisant différents distingueurs couramment considérés dans la littérature.

On peut utiliser par exemple le coefficient de corrélation de Pearson [21], ou la distance euclidienne, selon les données disponibles.

Cependant, on retrouve la problématique de l'établissement d'un seuil qui diffère d'un cas à l'autre, comme nous l'avons déjà vu dans la section 4.2.2 en appliquant la corrélation-croisée pour le découpage des traces.

Le critère BCDC peut être utilisé pour décider des valeurs binaires de l'exposant d_i pour $1 \leq i \leq n$ selon l'équation 4.8 suivante :

$$d_i = \begin{cases} d_{ref} & \text{Si } BCDC(\widetilde{P_{ref}}, \widetilde{P_i}) \leq 0.5 \\ \neg d_{ref} & \text{Sinon} \end{cases} \tag{4.8}$$

Dans la figure 4.24, est représenté le résultat lorsque BCDC est utilisé comme distingueur. Le segment référence P_{ref} choisi est P_7. Ce choix est fait de façon arbitraire. Pour plus de clarté, nous n'avons montré le résultat que sur les 3 premiers octets (24 premiers bits de d). Avec le seuil 0.5 relatif à l'utilisation du BCDC (marqué avec la ligne rouge), on peut voir les bits de l'exposant égaux au bit de référence d_7 (inférieurs au seuil de collision) et ceux différents de d_7. Par conséquent, les 24 premiers bits de d sont égaux soit à 111111001111111100011110 soit à son complément. La première solution correspond effectivement aux 3 premiers octets de l'exposant $0xFCFF1E$. Les résultats obtenus avec l'utilisation du coefficient de Pearson mais aussi de la distance euclidienne sont également donnés respectivement par les figures 4.25 et 4.26. La distance euclidienne donne le même résultat que BCDC dans cet exemple. Cependant, un seuil arbitraire doit être choisi pour cela. En ce qui concerne le coefficient de Pearson, l'attaque n'a pas trouvé le bit d'exposant d_{16}.

4.2.7 Validation finale

Toute la méthodologie proposée dans ce chapitre pour une mise en pratique efficace et maitrisée d'une attaque horizontale, a été appliquée plusieurs fois en utilisant les trois distingueurs considérés. Le Tableau 4.1 donne le nombre moyen de bits retrouvés pour un exposant de 1024 bits en fonction des distingueurs. Pour chaque étape, 1000 attaques horizontales ont été effectuées. Plus précisément, le tableau 4.1 montre que notre méthodologie permet, en moyenne, de récupérer 99.6% des 1024 bits de l'exposant. Par conséquent, le nombre de bits erronés est très faible (environ 5 bits en moyenne).

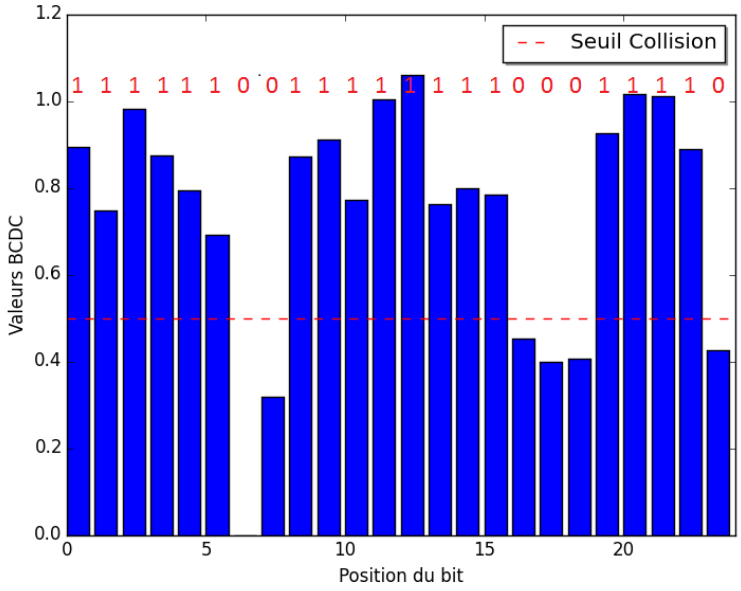


FIGURE 4.24 – Classification des bits de l'exposant en utilisant le critère BCDC comme distingueur.

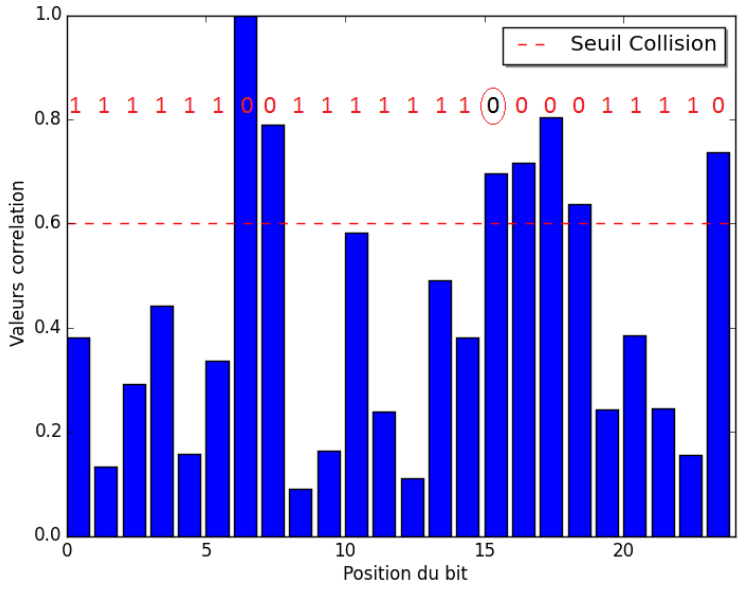


FIGURE 4.25 – Classification des bits de l'exposant en utilisant le coefficient de Pearson comme distingueur.

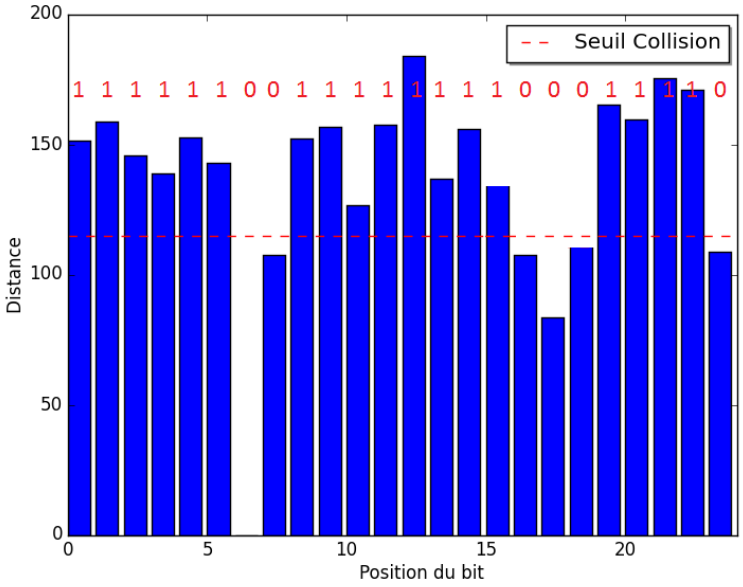


FIGURE 4.26 – Classification des bits de l'exposant en utilisant la distance euclidienne comme distingueur.

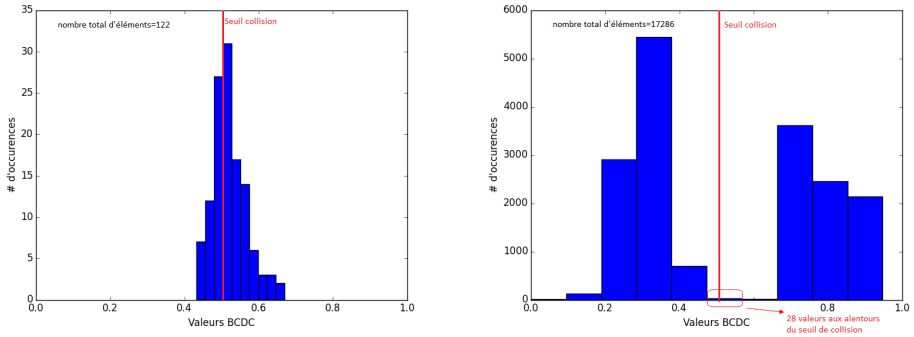
D'autre part, lorsque la corrélation de Pearson est utilisée comme distingueur au lieu du BCDC, on obtient seulement 75% des 1024 bits d'exposant. Ce n'est pas suffisant car l'attaquant ne connaît pas les positons des bits erronés. Il reste ainsi $\sum_{i=0}^{256} C_i^{1024}$ possibilités à explorer pour retrouver l'exposant entier, ce qui est impossible en pratique.

Le tableau 4.1 montre également l'importance des étapes de pré-traitement (découpage de la courbe, application d'une resynchronisation hiérarchique POC puis DTW, réduction adaptative du bruit) et le choix d'un distingueur efficace, dans le contexte d'une attaque horizontale. Plus particulièrement, l'importance de la procédure de resynchronisation ainsi que la réduction du bruit apparaît nettement. Ces deux dernières étapes sont probablement les plus importantes car l'attaque est appliquée sur de longues traces.

À ce stade, on peut s'interroger sur le temps consacré à l'application de la procédure proposée. La répartition du temps relatif nécessaire pour conduire chaque étape est de : 85% pour le pré-traitement des traces (découpage de la courbe, réduction du bruit et resynchronisation), 10% pour l'acquisition des traces et 5% pour l'identification des PoIs suivie de la récupération des bits de l'exposant. Le temps total consacré à l'application d'une attaque horizontale est d'environ deux heures pour une clé de 1024 bits.

	Distingueur		
Méthode de pré-traitement	Coefficient de Pearson	BCDC	Distance
Segments non alignés	50.8%	51%	50%
POC uniquement	53%	52.7%	50%
POC & DTW	70%	84%	63%
POC & DTW & Filtrage	75%	99.6%	70%

Tableau 4.1 – Pourcentage du nombres moyen de bits retrouvés en fonction des étapes effectuées et du distingueur considéré.



(a) Histogramme des valeurs du BCDC équivalentes aux bits erronés.

(b) Histogramme des valeurs du BCDC équivalentes aux bits retrouvés.

FIGURE 4.27 – Distribution des valeurs du BCDC.

4.2.8 Distribution des valeurs du BCDC

Pour analyser les valeurs BCDC correspondant aux bits erronés et comprendre pourquoi le critère BCDC ne distingue pas ces bits, la distribution des valeurs du BCDC obtenues pour les bits erronés mais aussi celle pour les bits retrouvés sont données en guise d'illustration sur la figure 4.27. Ces valeurs de BCDC correspondants aux résultats de 17 attaques horizontales, parmi les 1000 effectuées au total. Ceci nous donne 122 valeurs de BCDC pour les bits erronés et 17286 valeurs de BCDC pour les bits retrouvés. Comme le montre la figure 4.27a les valeurs du BCDC sont très proches du seuil de collision 0,5 lorsque les bits sont erronés, tandis qu'elles sont comprises entre 0,2 et 0,35 ou encore entre 0,65 et 0,95 pour les bits correctement identifiés, comme indiqué sur la figure 4.27b.

Ceci permet de connaître les positions présumées des bits potentiellement mal estimés, ce qui facilite l'identification finale de l'exposant en force brute.

4.3 Contremesures attaques horizontales

L'objectif des attaques horizontales est d'exploiter les liens qui peuvent exister entre les parties correspondant au traitement des bits de l'exposant privé. Ainsi, la suppression de ces liens, ou leur diminution, peut aider à se prémunir de ces attaques.

Nous avons montré dans ce chapitre que les attaques horizontales constituent une véritable menace pesant sur les composants modernes sécurisés. Pour cela chaque étape doit être effectué avec soin.

Après la collecte de la trace cible, celle-ci doit être découpée en évitant les méthodes nécessitant une inspection visuelle comme évoqué dans la section 4.2.2. Si cette étape n'est pas faite avec suffisamment de précision, l'attaque ne fonctionnera pas. Ainsi, pour rendre la tâche de l'attaquant difficile dans la pratique, des techniques permettant d'avoir des formes de fuites aléatoires peuvent être envisagées.

L'étape de resynchronisation, consiste généralement à combiner plusieurs techniques de resynchronisation. Une direction visant à augmenter significativement la difficulté de la resynchronisation peut être de varier de façon aléatoire la durée des segments de trace ou encore leurs formes.

4.3.1 Nouvelles contremesures

Dans cette section différentes propositions de contremesures contre les attaques par canaux cachés et plus particulièrement contre les attaques horizontales, sont détaillées. Ces propositions ont toutes fait l'objet d'un brevet déposé par *STMicroelectronics*.

4.3.1.1 Contremesure Infective

Pour se prémunir des attaques horizontales, nous avons proposé une technique appelée *contremesure infective*. Elle consiste à introduire une erreur sur chacun des bits de l'exposant d, autrement dit un vecteur erreur e (noté e mais différent de l'exposant public) d'un nombre de bit égal à celui de l'exposant est appliqué (par ou exclusif) à l'exposant pour obtenir l'exposant fauté d''. Puis on calcule l'exponentiation modulaire avec cet exposant d''. Le résultat obtenu n'est évidemment pas correct mais a été effectué avec un exposant aléatoire. Afin de rétablir le bon résultat, un algorithme de correction permet de calculer un facteur de correction multiplicatif à partir du type d'erreur en chaque bit modifié ($0 \rightarrow 1$ ou $1 \rightarrow 0$).

Formellement, on suppose que l'exposant d'origine d est protégé par le masquage utilisant l'indicatrice d'Euler $\phi(N)$ (section 1.4.5). Ce choix est fait pour se prémunir des attaques par faute. Le nouvel exposant masqué est alors $d' = d + r \cdot \phi(N)$. Ensuite, l'exposant erroné est obtenu par : $d'' = d' \oplus e$.

Soit C le résultat de l'exponentiation avec l'exposant d''. Pour corriger C chaque bit de l'aléa e est testé. Dans le cas où $e_i = 1$ (d'_i est fauté), la transition induite par la faute (de $0 \rightarrow 1$ ou de $1 \rightarrow 0$) est prise en compte. Si $d'_i - e_i = 0$

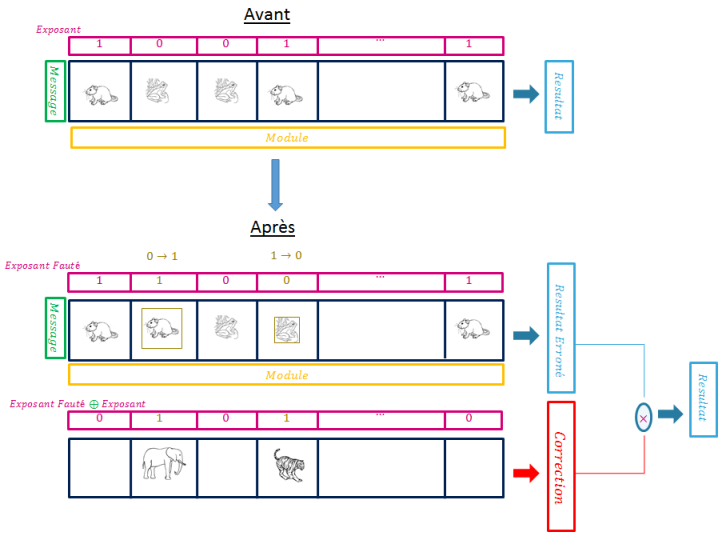


FIGURE 4.28 – Description de la méthode de contremesure infective

(sens $1 \rightarrow 0$), on calcule :

$$C = C \cdot m \bmod N.$$

Dans le cas où $d'_i - e_i = 1$ $(0 \rightarrow 1)$ le calcul sera :

$$C = C \cdot m^{-1} \bmod N.$$

À la fin de chaque étape, m et $m^{-1} \bmod N$ sont actualisés respectivement par $m^2 \bmod N$ et $(m^{-1})^2 \bmod N$. L'algorithme 16 résume les différentes étapes de la contremesure proposée. Tandis que la figure 4.28 illustre cette contremesure.

La sécurité de cette technique repose sur le fait que l'algorithme de correction (étapes 7 à 16) doit être inconnu de l'adversaire. Ainsi, même s'il est capable d'appliquer un attaque horizontale sur l'algorithme d'exponentiation modulaire effectué à l'étape 4, l'exposant retrouvé sera erroné et ne pourra pas être utilisé pour retrouver l'exposant d'origine sans connaître l'algorithme de correction et la position des bits erronés.

En outre, l'exposant étant protégé par un premier masquage, les attaques par fautes ne pourront pas fonctionner sur cette proposition. Ainsi, elle constitue une protection aussi contre les attaques par faute sur l'exposant.

Le temps d'exécution d'un algorithme implémentant cette technique équivaut à l'exécution de deux algorithmes d'exponentiation. Cependant la manière de faire l'algorithme de correction reste parfaitement optimisable et ainsi le surcoût en performances peut être fortement réduit.

algorithme 16 Infective contremesure

Entrées: $m, N, d = (d_{n-1}, ..., d_1, d_0),\ d_i \in \{0, 1\}$ pour tout $i \in \{0, \cdots, n-1\}$
Sorties: $C = m^d \bmod N$

1: Choisir deux aléas r et e
2: $d' = d + r \cdot \phi(N)$
3: $d'' = d' \oplus e$
4: Exécuter $C = m^{d''}$
5: $M_1 = m$
6: $M_2 = m^{-1} \bmod N$
7: **pour** $i = n - 1$ **à** 0 **faire**
8: **si** $e_i = 1$ **alors**
9: **si** $d'_i - e_i = 0$ **alors**
10: $C = C \cdot M_1 \bmod N$
11: **sinon**
12: $C = C \cdot M_2 \bmod N$
13: **fin si**
14: **fin si**
15: $M_1 = M_1^2 \bmod N$
16: $M_2 = M_2^2 \bmod N$
17: **fin pour**
18: **retourner** C

4.3.1.2 Masquage du modulo pas-à-pas

Une méthode pour se prémunir des attaques par canaux cachés utilisant plusieurs courbes consiste à masquer le modulo comme détaillé dans la section 1.4.5. Ce masquage se fait en dehors de la boucle de l'algorithme d'exponentiation modulaire. Ainsi, le même modulo masqué est utilisé dans toute la boucle de l'algorithme. Ceci permet l'application des attaques horizontales car on retrouve les mêmes liens (collision sur les entrées des opérations successives par exemple) généralement exploités par les attaques horizontales en cas de non-masquage.

Pour améliorer cette méthode, nous proposons une contremesure qui exploite la remarque précédente, pour se prémunir des attaques par canaux cachés. Elle consiste à masquer le modulo pas-à-pas avec un aléa différent pour chaque produit modulaire de l'exponentiation. Cependant, il suffit de changer d'aléa à chaque itération correspondant au traitement d'un bit égal à 1.

L'algorithme 17 résume les différentes étapes de la contremesure du masquage pas-à-pas du modulo. À chaque itération de cet algorithme un aléa r_i (étape 4) est choisi et multiplié par le modulo N. Le modulo intermédiaire $r_i \cdot N$ est utilisé pour les produits modulaires de l'itération. Pour retrouver le résultat correct de l'algorithme d'exponentiation modulaire la sortie de la dernière itération est réduite modulo N.

La figure 4.29 illustre cette contremesure. La figure du haut représente la ver-

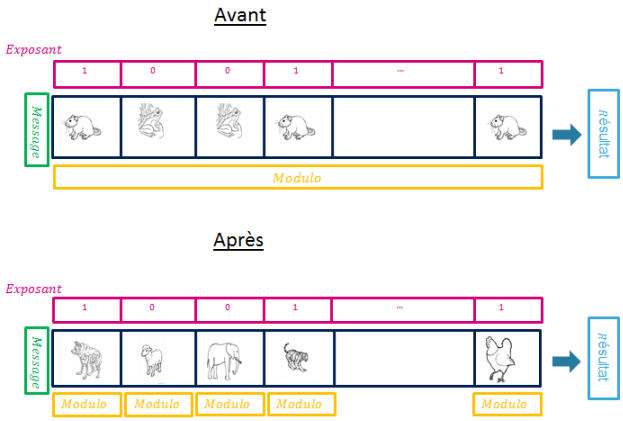


FIGURE 4.29 – Description de la méthode de masquage du modulo pas à pas

sion classique du masquage du modulo tandis que la figure du bas représente notre proposition du masquage du modulo pas-à-pas.

Le temps d'exécution de l'algorithme 17 est supérieur à celui de l'échelle de Montgomery du fait de la sélection d'un aléa à chaque étape mais aussi du fait du calcul d'une nouvelle constante de Montgomery à chaque nouveau modulo.

algorithme 17 Échelle de Montgomery avec randomisation du modulo pas à pas

ENTRÉES: $m, N, d = (d_{n-1}, ..., d_1, d_0)$, $d_i \in \{0, 1\}$ pour tout $i \in \{0, \cdots, n-1\}$
SORTIES: $S = m^d \bmod N$

1: $R_0 = 1$
2: $R_1 = m$
3: **pour** $i = n - 1$ **à** 0 **faire**
4: Choisir un aléa r_i
5: $R_{\neg d_i} = R_{d_i} \cdot R_{\neg d_i} \bmod r_i \cdot N$
6: $R_{d_i} = R_{d_i} \cdot R_{d_i} \bmod r_i \cdot N$
7: **fin pour**
8: **retourner** $R_0 \bmod N$

4.4 Conclusion

Dans ce chapitre, nous avons proposé, pour chacune des étapes d'une attaque horizontale, des solutions efficaces et pratiques permettant d'améliorer l'extraction

de l'information secrète dans un environnement réel. Des méthodes pour valider chaque étape ont également été proposées. La plupart des méthodes proposées sont basées sur notre critère borné de détection de collision (BCDC) décrit dans le chapitre 2, section 2.4.3.3.

Si nous proposons des moyens d'automatiser et d'améliorer les attaques horizontales, nous proposons également dans ce chapitre deux contremesures qui à notre connaissance sont nouvelles dans le domaine. Ces deux contremesures, l'une portant sur un masquage innovant de l'exposant, l'autre sur la randomisation du modulo au sein de l'algorithme d'exponentiation peuvent être déclinées en plusieurs variantes afin d'être adaptées à l'implémentation du produit modulaire ou de l'exponentiation utilisée et optimisées afin de tenir compte du facteur de performance. Elles peuvent également être combinées car elles sont indépendantes.

Conclusion et Perspectives

Dans cette thèse, nous nous sommes intéressés à l'étude pratique des attaques par canaux cachés et plus particulièrement à la famille des attaques par collision et des attaques horizontales appliquées à l'exponentiation modulaire.

Les outils que nous avons proposés dans cette thèse ont pour but de faciliter l'automatisation et sont des réponses pratiques aux problèmes qu'un attaquant peut rencontrer en pratique (ou un fondeur voulant tester ses propres produits). Ces problèmes sont par exemple : l'effet des contremesures environnementales qui résultent souvent en des traces trop bruitées (et donc un faible rapport signal à bruit lorsque l'on utilise une trace seulement pour extraire l'information secrète) et désynchronisées, l'utilisation de méthodes nécessitant la définition d'un seuil visuel ou encore l'absence d'outils permettant de valider si une étape est bien faite ou non. Pour toutes ces problématiques, nous avons essayé de proposer des solutions pratiques et des méthodes de validation de ces solutions. Pour cela, nous avons dans un premier temps rappelé l'évolution des attaques par canaux cachés sur l'exponentiation modulaire. Nous avons choisi parmi la multitude d'attaques par canaux cachés celles qui nous ont semblé les plus pertinentes et nous les avons décrites dans le chapitre 1 avec le souci de les rendre faciles à implémenter.

Puis, nous avons étudié les attaques par canaux cachés sur un simulateur développé pour la vérification de leur efficacité. Ce simulateur nous a permis de valider les méthodologies proposées dans cette thèse avant l'application de ces dernières sur des traces réelles.

Ensuite, nous avons étudié les attaques par collision dans un environnement réel. Pour cela, nous nous sommes intéressés à l'automatisation de la détection de collision dans la pratique. Ces travaux ont permis de dégager un nouveau critère de détection de collision, appelé BCDC. Ce critère a l'avantage d'être borné et de tendre vers 0 en cas de collision lorsque le bruit diminue. Il permet de détecter l'occurrence de collisions entre deux traces dans un environnement réel, s'il est couplé à un algorithme de classification bien que les traces soient bruitées. Lorsque le SNR des traces est amélioré via une méthode de prétraitement, le critère permet de détecter l'occurrence de collisions en utilisant un seuil défini dans le chapitre 2. Ces travaux ont fait l'objet d'une publication à DSD 2015 [66].

Nous nous sommes ensuite intéressés à l'estimation du SNR d'un jeu de traces dans le contexte des attaques par canaux cachés. Ainsi, nous avons présenté une façon d'estimer le SNR, qui permet de valider les protocoles expérimentaux suivis (et donc la qualité de mesures), et de faciliter notre capacité à extraire l'information du bruit lors d'une attaque par canaux cachés. Cette estimation du SNR est déduite à partir du critère de détection de collision BCDC. En outre, nous avons montré que ce SNR estimé peut être exploité pour analyser les traces mais aussi pour effectuer

un filtrage adaptatif ne nécessitant pas la connaissance de certains paramètres du composant testé, contrairement aux techniques de filtrage classiques. Ces travaux ont fait l'objet d'une publication à Cardis 2015 [79].

Dans la dernière partie de ce manuscrit, une étude détaillée des principales étapes d'une attaque horizontale est exposée. Ainsi des solutions génériques ont été détaillées pour chaque étape. Ces solutions nous ont permis de mener avec succès des attaques horizontales sur des composants modernes exécutant l'échelle de Montgomery (algorithme 7), algorithme qui a l'avantage d'avoir été étudié en profondeur dans la littérature et d'être considéré comme l'un des plus résistants aux attaques par canaux cachés. Ces solutions ont été comparées aux méthodes classiquement utilisées pour montrer leur efficacité. Ces travaux ont été soumis à JCEN.

Finalement deux nouvelles contremesures aux attaques par canaux cachés et plus particulièrement des attaques horizontales ont été proposées. Ces deux contremesures, l'une portant sur un masquage innovant de l'exposant, l'autre sur la randomisation du modulo au sein de l'algorithme d'exponentiation peuvent être déclinées en plusieurs variantes afin d'être adaptées à l'implémentation du produit modulaire ou de l'exponentiation utilisée et optimisées. Elles peuvent également être combinées car elles sont indépendantes. Grâce à notre simulateur, ces contremesures ont été validées avec les différentes attaques décrites en détail dans l'état de l'art. Ces travaux ont fait l'objet de plusieurs brevets déposés par *STMicroelectronics*.

Dans notre étude nous n'avons pas pris en compte les cryptosystèmes à base de courbes elliptiques (ECC). Il serait ainsi intéressant de vérifier l'efficacité des méthodes proposées dans cette thèse sur des algorithmes tels que la multiplication scalaire qui est l'opération principale de nombreux ECC et qui peut être vue comme une opération d'exponentiation modulaire dans un groupe additif. Cependant, le doublement et l'addition dans une multiplication scalaire sont composés de plusieurs opérations. Ces opérations sont ainsi plus longues que dans le cas de l'exponentiation modulaire et comportent généralement une partie "software" bien plus importante. Cette partie permet d'introduire beaucoup plus de bruit dans les traces. Il serait ainsi intéressant de vérifier l'efficacité du BCDC dans ces cas.

Concernant le seuil du BCDC, nous l'avons défini de façon expérimentale durant cette thèse, il serait ainsi intéressant de démontrer l'exactitude de ce seuil mais aussi de calculer ces intervalles de confiance. Ce qui pourrait permettre, notamment lors des attaques par collision, de se passer de l'utilisation de l'algorithme des k-moyennes ou des techniques de filtrages pour une détection automatique de collisions.

L'algorithme des k-moyennes est au sein de nos travaux un algorithme clé sur l'automatisation de la distinction des bits de l'exposant. Parmi les techniques émergentes du "pattern matching", certaines comme l'apprentissage automatique profond ou "Deep Learning" nous semblent prometteuses afin de supprimer l'étape d'identification des points d'intérêt, vecteur support de comparaison nécessaire à

une application de l'algorithme des k-moyennes efficace.

Des techniques de filtrage basées sur la démodulation ont fait leur apparition dans la littérature. Évaluer si ces techniques permettent d'améliorer les attaques horizontales dans la pratique, constitue une piste également intéressante.

Bibliographie

[AFT+09] Frederic Amiel, Benoit Feix, Michael Tunstall, Claire Whelan, and William P Marnane. Distinguishing multiplications from squaring operations. In *Selected Areas in Cryptography*, pages 346–360. Springer, 2009.

[ANS14] ANSSI. Mecanismes cryptographiques - regles et recommandations. February 2014. Rev. 2. 03.

[APcXSQ06] Cédric Archambeau, Eric Peeters, François Xavier Standaert, and Jean-Jacques Quisquater. Template attacks in principal subspaces. In *Cryptographic Hardware and Embedded Systems - CHES 2006, 8th International Workshop*, volume 4249 of *Lecture Notes in Computer Science*, pages 1–14. Springer, 2006.

[ASYZ13] Chen Aidong, Xu Sen, Chen Yun, and Qin Zhiguang. Collision-based chosen-message simple power clustering attack algorithm. *Communications, China*, 10(5) :114–119, 2013.

[Bar86] Paul Barrett. Implementing the rivest shamir and adleman public key encryption algorithm on a standard digital signal processor. In *Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings*, volume 263 of *Lecture Notes in Computer Science*, pages 311–323. Springer, 1986.

[Bau12] Sven Bauer. Attacking exponent blinding in rsa without crt. In *Constructive Side-Channel Analysis and Secure Design*, pages 82–88. Springer, 2012.

[BCO04] Eric Brier, Christophe Clavier, and Francis Olivier. Correlation power analysis with a leakage model. In *Cryptographic Hardware and Embedded Systems - CHES 2004 : 6th International Workshop Cambridge, MA, USA, August 11-13, 2004. Proceedings*, volume 3156 of *Lecture Notes in Computer Science*, pages 16–29. Springer, 2004.

[BDF98] Dan Boneh, Glenn Durfee, and Yair Frankel. An attack on rsa given a small fraction of the private key bits. In *Advances in Cryptology - ASIACRYPT '98, International Conference on the Theory and Applications of Cryptology and Information Security, Beijing, China, October 18-22, 1998, Proceedings*, volume 1514 of *Lecture Notes in Computer Science*, pages 25–34. Springer, 1998.

[BDGN14] Shivam Bhasin, Jean-Luc Danger, Sylvain Guilley, and Zakaria Najm. Nicv : normalized inter-class variance for detection of side-channel leakage. In *Electromagnetic Compatibility, Tokyo (EMC'14/Tokyo), 2014 International Symposium on*, pages 310–313. IEEE, 2014.

[BGLR09] Lejla Batina, Benedikt Gierlichs, and Kerstin Lemke-Rust. Differential cluster analysis. In *Cryptographic Hardware and Embedded Systems - CHES 2009, 11th International Workshop, Lausanne, Switzerland, September 6-9, 2009, Proceedings*, volume 5747 of *Lecture Notes in Computer Science*, pages 112–127. Springer, 2009.

[BGLT06] Marco Bucci, Luca Giancane, Raimondo Luzzi, and Alessandro Trifiletti. Three-phase dual-rail pre-charge logic. In *Cryptographic Hardware and Embedded Systems - CHES 2006, 8th International Workshop*, volume 4249 of *Lecture Notes in Computer Science*, pages 232–241. Springer, 2006.

[BHvW12] Lejla Batina, Jip Hogenboom, and Jasper GJ van Woudenberg. Getting more from pca : First results of using principal component analysis for extensive power analysis. In *Topics in Cryptology–CT-RSA 2012*, pages 383–397. Springer, 2012.

[BJ13] Aurélie Bauer and Éliane Jaulmes. Correlation analysis against protected sfm implementations of rsa. In *Progress in Cryptology–INDOCRYPT 2013*, pages 98–115. Springer, 2013.

[CCD00] Christophe Clavier, Jean-Sébastien Coron, and Nora Dabbous. Differential power analysis in the presence of hardware countermeasures. In *Cryptographic Hardware and Embedded Systems - CHES 2000, Second International Workshop, Worcester, MA, USA, August 17-18, 2000, Proceedings*, volume 1965 of *Lecture Notes in Computer Science*, pages 252–263. Springer, 2000.

[CDMG+13] Jeremy Cooper, Elke De Mulder, Gilbert Goodwill, Joshua Jaffe, Gary Kenworthy, Pankaj Rohatgi, et al. Test vector leakage assessment (tvla) methodology in practice. In *International Cryptographic Module Conference*, 2013.

[CFG+10] Christophe Clavier, Benoit Feix, Georges Gagnerot, Mylène Roussellet, and Vincent Verneuil. Horizontal correlation analysis on exponentiation. In *Information and Communications Security*, pages 46–61. Springer, 2010.

[CFG+12] Christophe Clavier, Benoit Feix, Georges Gagnerot, Christophe Giraud, Mylene Roussellet, and Vincent Verneuil. Rosetta for single

trace analysis. In *Progress in Cryptology-INDOCRYPT 2012*, pages 140–155. Springer, 2012.

[CJ01] Christophe Clavier and Marc Joye. Universal exponentiation algorithm. In *Cryptographic Hardware and Embedded Systems - CHES 2001, Third International Workshop, Paris, France, May 14-16, 2001, Proceedings*, volume 2162 of *Lecture Notes in Computer Science*, pages 300–308. Springer, 2001.

[CJ03] Mathieu Ciet and Marc Joye. (virtually) free randomization techniques for elliptic curve cryptography. In *International Conference on Information and Communications Security*, pages 348–359. Springer, 2003.

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

[CK09] Jean-Sébastien Coron and Ilya Kizhvatov. An efficient method for random delay generation in embedded software. In *Cryptographic Hardware and Embedded Systems - CHES 2009, 11th International Workshop, Lausanne, Switzerland, September 6-9, 2009, Proceedings*, volume 5747 of *Lecture Notes in Computer Science*, pages 156–170. Springer, 2009.

[CMCJ04] Benoît Chevallier-Mames, Mathieu Ciet, and Marc Joye. Low-cost solutions for preventing simple side-channel analysis : Side-channel atomicity. *Computers, IEEE Transactions on*, 53(6) :760–768, 2004.

[Cop96] Don Coppersmith. Finding a small root of a bivariate integer equation ; factoring with high bits known. In *Advances in Cryptology - EUROCRYPT '96, International Conference on the Theory and Application of Cryptographic Techniques, Saragossa, Spain, May 12-16, 1996, Proceeding*, volume 1070 of *Lecture Notes in Computer Science*, pages 178–189. Springer, 1996.

[Cor99] Jean-Sébastien Coron. Resistance against differential power analysis for elliptic curve cryptosystems. In *Cryptographic Hardware and Embedded Systems, First International Workshop, CHES'99, Worcester, MA, USA, August 12-13, 1999, Proceedings*, volume 1717 of *Lecture Notes in Computer Science*, pages 292–302. Springer, 1999.

[CRR02] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. Template attacks. In *Cryptographic Hardware and Embedded Systems - CHES*

2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers, volume 2523 of *Lecture Notes in Computer Science*, pages 13–28. Springer, 2002.

[CZ06] Zhimin Chen and Yujie Zhou. Dual-rail random switching logic : A countermeasure to reduce side channel leakage. In *Cryptographic Hardware and Embedded Systems - CHES 2006, 8th International Workshop*, volume 4249 of *Lecture Notes in Computer Science*, pages 242–254. Springer, 2006.

[DCO+15] Ibrahima Diop, Mathieu Carbone, Sébastien Ordas, Yanis Linge, Pierre-Yvan Liardet, and Philippe Maurine. Collision for estimating SCA measurement quality and related applications. In *Smart Card Research and Advanced Applications - 14th International Conference, CARDIS 2015, Bochum, Germany, November 4-6, 2015. Revised Selected Papers*, pages 143–157, 2015.

[DH76] Whitfield Diffie and Martin E Hellman. Multiuser cryptographic techniques. In *Proceedings of the June 7-10, 1976, national computer conference and exposition*, pages 109–112. ACM, 1976.

[Dhe98] Jean-Francois Dhem. *Design of an efficient public-key cryptograc library for RISC-based smart cards*. PhD thesis, PhD thesis, Universit e Catholique de louvain-Belgica, 1998.

[DHS01] RO Duda, PE Hart, and DG Stork. Pattern classification, edition wiley interscience. *New York*, 2001.

[DKL+98] Jean-Francois Dhem, Francois Koeune, Philippe-Alexandre Leroux, Patrick Mestré, Jean-Jacques Quisquater, and Jean-Louis Willems. A practical implementation of the timing attack. In *International Conference on Smart Card Research and Advanced Applications*, pages 167–182. Springer, 1998.

[DLLM15] Ibrahima Diop, Pierre-Yvan Liardet, Yanis Linge, and Philippe Maurine. Collision based attacks in practice. In *Digital System Design (DSD), 2015 Euromicro Conference on*, pages 367–374. IEEE, 2015.

[DLM+09] Amine Dehbaoui, Victor Lomne, Philippe Maurine, Lionel Torres, and Michel Robert. Enhancing electromagnetic attacks using spectral coherence based cartography. In *IFIP/IEEE International Conference on Very Large Scale Integration-System on a Chip*, pages 135–155. Springer, 2009.

[Dob98] Hans Dobbertin. Cryptanalysis of md4. *J. Cryptology*, 11 :253–271, 1998.

[DS16] François Durvaux and François-Xavier Standaert. From improved leakage detection to the detection of points of interests in leakage traces. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 240–262. Springer, 2016.

[FKJM+06] Pierre-Alain Fouque, Sébastien Kunz-Jacques, Gwenaëlle Martinet, Frédéric Muller, and Frédéric Valette. Power attack on small rsa public exponent. In *Cryptographic Hardware and Embedded Systems - CHES 2006, 8th International Workshop*, volume 4249 of *Lecture Notes in Computer Science*, pages 339–353. Springer, 2006.

[FV03] P. A. Fouque and F. Valette. The doubling attack - why upwards is better than downwards. In *Cryptographic Hardware and Embedded Systems - CHES 2003, 5th International Workshop, Cologne, Germany, September 8-10, 2003, Proceedings*, volume 2779 of *Lecture Notes in Computer Science*, pages 269–280. Springer, 2003.

[Gam84] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Advances in Cryptology, Proceedings of CRYPTO '84, Santa Barbara, California, USA, August 19-22, 1984, Proceedings*, volume 196 of *Lecture Notes in Computer Science*, pages 10–18. Springer, 1984.

[GBTP08] Benedikt Gierlichs, Lejla Batina, Pim Tuyls, and Bart Preneel. Mutual information analysis. In *Cryptographic Hardware and Embedded Systems - CHES 2008, 10th International Workshop, Washington, D.C., USA, August 10-13, 2008. Proceedings*, volume 5154 of *Lecture Notes in Computer Science*, pages 426–442. Springer, 2008.

[GcXS12] Benoˆ Gérard and François Xavier Standaert. Unified and optimized linear collision attacks and their application in a non-profiled setting. In *CHES*, volume 7428, pages 175–192. Springer, 2012.

[GMO01] Karine Gandolfi, Christophe Mourtel, and Francis Olivier. Electromagnetic analysis : Concrete results. In *Cryptographic Hardware and Embedded Systems - CHES 2001, Third International Workshop, Paris, France, May 14-16, 2001, Proceedings*, volume 2162 of *Lecture Notes in Computer Science*, pages 251–261. Springer, 2001.

[GMS+11] Sylvain Guilley, Houssem Maghrebi, Youssef Souissi, Laurent Sauvage, and J Danger. Quantifying the quality of side-channel acquisition. *COSADE, February*, 2011.

[Hai05] Siba Haidar. *Comparaison des documents audiovisuels par matrice de similarité*. PhD thesis, Université Paul Sabatier-Toulouse III, 2005.

[HIM+14] Johann Heyszl, Andreas Ibing, Stefan Mangard, Fabrizio De Santis, and Georg Sigl. Clustering algorithms for non-profiled single-execution attacks on exponentiations. In *Smart Card Research and Advanced Applications*, pages 79–93. Springer, 2014.

[HKT12] Neil Hanley, HeeSeok Kim, and Michael Tunstall. Exploiting collisions in addition chain-based exponentiation algorithms using a single trace. Technical report, Cryptology ePrint Archive, Report 2012/485, 2012.

[HMA+08] Naofumi Homma, Atsushi Miyamoto, Takafumi Aoki, Akashi Satoh, and Adi Shamir. Collision-based power analysis of modular exponentiation using chosen-message pairs. In *Cryptographic Hardware and Embedded Systems - CHES 2008, 10th International Workshop, Washington, D.C., USA, August 10-13, 2008. Proceedings*, volume 5154 of *Lecture Notes in Computer Science*, pages 15–29. Springer, 2008.

[HMA+10] Naofumi Homma, Atsushi Miyamoto, Takafumi Aoki, Akashi Satoh, and Adi Samir. Comparative power analysis of modular exponentiation algorithms. *Computers, IEEE Transactions on*, 59(6) :795–807, 2010.

[HMH+12] Johann Heyszl, Dominik Merli, Benedikt Heinz, Fabrizio De Santis, and Georg Sigl. Strengths and limitations of high-resolution electromagnetic field measurements for side-channel analysis. In *International Conference on Smart Card Research and Advanced Applications*, pages 248–262. Springer, 2012.

[HNI+06] Naofumi Homma, Sei Nagashima, Yuichi Imai, Takafumi Aoki, and Akashi Satoh. High-resolution side-channel attack using phase-based waveform matching. In *Cryptographic Hardware and Embedded Systems - CHES 2006, 8th International Workshop*, volume 4249 of *Lecture Notes in Computer Science*, pages 187–200. Springer, 2006.

[HS09] Nadia Heninger and Hovav Shacham. Reconstructing rsa private keys from random key bits. In *Advances in Cryptology - CRYPTO 2009, 29th Annual International Cryptology Conference*, volume 5677 of *Lecture Notes in Computer Science*, pages 1–17. Springer, 2009.

[IIT02] Kouichi Itoh, Tetsuya Izu, and Masahiko Takenaka. Address-bit differential power analysis of cryptographic schemes ok-ecdh and ok-ecdsa. In *Cryptographic Hardware and Embedded Systems - CHES*

2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers, volume 2523 of *Lecture Notes in Computer Science*, pages 129–143. Springer, 2002.

[JM06] Ellen Jochemsz and Alexander May. A strategy for finding roots of multivariate polynomials with new applications in attacking rsa variants. In *Advances in Cryptology - ASIACRYPT 2006, 12th International Conference on the Theory and Application of Cryptology and Information Security*, volume 4284 of *Lecture Notes in Computer Science*, pages 267–282. Springer, 2006.

[JY02] Marc Joye and Sung-Ming Yen. The montgomery powering ladder. In *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*, volume 2523 of *Lecture Notes in Computer Science*, pages 291–302. Springer, 2002.

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.

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

[Kug75] CD Kuglin. The phase correlation image alignment method. In *Proc. Int. Conf. on Cybernetics and Society, 1975*, pages 163–165, 1975.

[LCC+06] Thanh-Ha Le, Jessy Clédière, Cécile Canovas, Bruno Robisson, Christine Servière, and Jean-Louis Lacoume. A proposition for correlation power analysis enhancement. In *Cryptographic Hardware and Embedded Systems - CHES 2006, 8th International Workshop*, volume 4249 of *Lecture Notes in Computer Science*, pages 174–186. Springer, 2006.

[Len96] Arjen K. Lenstra. Generating standard dsa signatures without long inversion. In *Advances in Cryptology - ASIACRYPT '96, International Conference on the Theory and Applications of Cryptology and Information Security, Kyongju, Korea, November 3-7, 1996, Proceedings*, volume 1163 of *Lecture Notes in Computer Science*, pages 57–64. Springer, 1996.

[LRP07] Kerstin Lemke-Rust and Christof Paar. Gaussian mixture models for higher-order side channel analysis. In *Cryptographic Hardware and Embedded Systems - CHES 2007, 9th International Workshop, Vienna, Austria, September 10-13, 2007, Proceedings*, volume 4727 of *Lecture Notes in Computer Science*, pages 14–27. Springer, 2007.

[Man04] Stefan Mangard. Hardware countermeasures against dpa–a statistical analysis of their effectiveness. In *Topics in Cryptology–CT-RSA 2004 : The Cryptographers' Track at the RSA Conference 2004, San Francisco, CA, USA, February 23-27, 2004, Proceedings*, volume 2964, page 222. Springer Science & Business Media, 2004.

[May03] Alexander May. *New RSA vulnerabilities using lattice reduction methods.* PhD thesis, University of Paderborn, 2003.

[MDS99] Thomas S. Messerges, Ezzy A. Dabbish, and Robert H. Sloan. Power analysis attacks of modular exponentiation in smartcards. In *Cryptographic Hardware and Embedded Systems, First International Workshop, CHES'99, Worcester, MA, USA, August 12-13, 1999, Proceedings*, volume 1717 of *Lecture Notes in Computer Science*, pages 144–157. Springer, 1999.

[Mes00] Thomas S. Messerges. Using second-order power analysis to attack dpa resistant software. In *Cryptographic Hardware and Embedded Systems - CHES 2000, Second International Workshop, Worcester, MA, USA, August 17-18, 2000, Proceedings*, volume 1965 of *Lecture Notes in Computer Science*, pages 238–251. Springer, 2000.

[MG10] Edgar Mateos and Catherine H Gebotys. A new correlation frequency analysis of the side channel. In *Proceedings of the 5th Workshop on Embedded Systems Security*, page 4. ACM, 2010.

[MOBW13] Luke Mather, Elisabeth Oswald, Joe Bandenburg, and Marcin Wójcik. Does my device leak information ? an a priori statistical power analysis of leakage detection tests. In *Advances in Cryptology - ASIACRYPT 2013*, volume 8269 of *Lecture Notes in Computer Science*, pages 486–505. Springer, 2013.

[Mon85] Peter L Montgomery. Modular multiplication without trial division. *Mathematics of computation*, 44(170) :519–521, 1985.

[MOP08] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power analysis attacks : Revealing the secrets of smart cards*, volume 31. Springer Science & Business Media, 2008.

[MVOV96] Alfred J Menezes, Paul C Van Oorschot, and Scott A Vanstone. *Handbook of applied cryptography*. CRC press, 1996.

[PITM14a] Guilherme Perin, Laurent Imbert, Lionel Torres, and Philippe Maurine. Attacking randomized exponentiations using unsupervised learning. In *Constructive Side-Channel Analysis and Secure Design*, pages 144–160. Springer, 2014.

[PITM14b] Guilherme Perin, Laurent Imbert, Lionel Torres, and Philippe Maurine. Practical analysis of rsa countermeasures against side-channel electromagnetic attacks. In *Smart Card Research and Advanced Applications*, pages 200–215. Springer, 2014.

[PM05] Thomas Popp and Stefan Mangard. Masked dual-rail pre-charge logic : Dpa-resistance without routing constraints. In *Cryptographic Hardware and Embedded Systems - CHES 2005, 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005, Proceedings*, volume 3659 of *Lecture Notes in Computer Science*, pages 172–186. Springer, 2005.

[QS01] Jean-Jacques Quisquater and David Samyde. Electromagnetic analysis (ema) : Measures and counter-measures for smart cards. In *Smart Card Programming and Security*, pages 200–210. Springer, 2001.

[Qui92] Jean-Jacques Quisquater. Encoding system according to the so-called rsa method, by means of a microcontroller and arrangement implementing this system, November 24 1992. US Patent 5,166,978.

[RSA78] Ronald L Rivest, Adi Shamir, and Len Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2) :120–126, 1978.

[RVD09] Denis Réal, Frédéric Valette, and Mhamed Drissi. Enhancing correlation electromagnetic attack using planar near-field cartography. In *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 628–633. European Design and Automation Association, 2009.

[Sch70] Henry Scheffé. Practical solutions of the behrens-fisher problem. *Journal of the American Statistical Association*, 65(332) :1501–1508, 1970.

[Sch02] Werner Schindler. A combined timing and power attack. In *Public Key Cryptography - PKC 2003, 6th International Workshop on Theory and Practice in Public Key Cryptography, Miami, FL, USA, January 6-8, 2003, Proceedings*, volume 2274 of *Lecture Notes in Computer Science*, pages 263–279. Springer, 2002.

[SGF+12] Laurent Sauvage, Sylvain Guilley, Florent Flament, Jean-Luc Danger, and Yves Mathieu. Blind cartography for side channel attacks : Cross-correlation cartography. *International Journal of Reconfigurable Computing*, 2012 :15, 2012.

[Sha00] Adi Shamir. Protecting smart cards from passive power analysis with detached power supplies. In *Cryptographic Hardware and Embedded Systems - CHES 2000, Second International Workshop, Worcester, MA, USA, August 17-18, 2000, Proceedings*, volume 1965 of *Lecture Notes in Computer Science*, pages 71–77. Springer, 2000.

[SHKS15] Robert Specht, Johann Heyszl, Martin Kleinsteuber, and Georg Sigl. Improving non-profiled attacks on exponentiations based on clustering and extracting leakage from multi-channel high-resolution em measurements. In *International Workshop on Constructive Side-Channel Analysis and Secure Design*, pages 3–19. Springer, 2015.

[SI11] Werner Schindler and Kouichi Itoh. Exponent blinding does not always lift (partial) spa resistance to higher-level security. In *Applied Cryptography and Network Security*, pages 73–90. Springer, 2011.

[SMBY04] Danil Sokolov, Julian Murphy, Alexandre V. Bystrov, and Alexandre Yakovlev. Improving the security of dual-rail circuits. In *Cryptographic Hardware and Embedded Systems - CHES 2004 : 6th International Workshop Cambridge, MA, USA, August 11-13, 2004. Proceedings*, volume 3156 of *Lecture Notes in Computer Science*, pages 282–297. Springer, 2004.

[ST93] National Institute Standards and Technology. Data encryption standard (des), Publication 46-2, 1993.

[ST01] National Institute Standards and Technology. Advanced encryption standard (aes), Publication 197, 2001.

[Stu08] Student. The probable error of a mean. *Biometrika*, pages 1–25, 1908.

[SWP03] Kai Schramm, Thomas J. Wollinger, and Christof Paar. A new class of collision attacks and its application to des. In *Fast Software Encryption, 10th International Workshop, FSE 2003, Lund, Sweden, February 24-26, 2003, Revised Papers*, volume 2887 of *Lecture Notes in Computer Science*, pages 206–222. Springer, 2003.

[TOT+14] Sébastien Tiran, Sébastien Ordas, Yannick Teglia, Michel Agoyan, and Philippe Maurine. A model of the leakage in the frequency domain and its application to cpa and dpa. *Journal of Cryptographic Engineering*, 4(3) :197–212, 2014.

[TPR13] Adrian Thillard, Emmanuel Prouff, and Thomas Roche. Success through confidence : Evaluating the effectiveness of a side-channel attack. In *CHES*, pages 21–36. Springer, 2013.

[vWWB11] Jasper GJ van Woudenberg, Marc F Witteman, and Bram Bakker. Improving differential power analysis by elastic alignment. In *Topics in Cryptology–CT-RSA 2011*, pages 104–119. Springer, 2011.

[Wal01] Colin D. Walter. Sliding windows succumbs to big mac attack. In *Cryptographic Hardware and Embedded Systems - CHES 2001, Third International Workshop, Paris, France, May 14-16, 2001, Proceedings*, volume 2162 of *Lecture Notes in Computer Science*, pages 286–299. Springer, 2001.

[Wel47] BL Welch. The generalization of student's problem when several different population variances are involved. *Biometrika*, 1947.

[WvWM11] Marc F Witteman, Jasper GJ van Woudenberg, and Federico Menarini. Defeating rsa multiply-always and message blinding countermeasures. In *Topics in Cryptology–CT-RSA 2011*, pages 77–88. Springer, 2011.

[YJ00] Sung-Ming Yen and Marc Joye. Checking before output may not be enough against fault-based cryptanalysis. *IEEE Transactions on computers*, 49(9) :967–970, 2000.

[YKMH06] Sung-Ming Yen, Lee-Chun Ko, SangJae Moon, and JaeCheol Ha. Relative doubling attack against montgomery ladder. In *Information Security and Cryptology-ICISC 2005*, pages 117–128. Springer, 2006.

[YLMH05] Sung-Ming Yen, Wei-Chih Lien, SangJae Moon, and JaeCheol Ha. Power analysis by exploiting chosen message and internal collisions–vulnerability of checking mechanism for rsa-decryption. In *Progress in Cryptology–Mycrypt 2005*, pages 183–195. Springer, 2005.

École Nationale Supérieure des Mines
de Saint-Étienne

NNT : *Communiqué le jour de la soutenance*

DIOP Ibrahima

FROM THEORY TO PRACTICE OF COLLISION BASED ATTACKS AND HORIZONTAL ATTACK MODULAR EXPONENTATION

Speciality: Cryptology

Keywords : Side channel Analysis, Collision based attacks, Horizontal attacks.

Abstract :

Nowadays side channel attacks against protected implementation of modular exponentiation constitute a threat to the secure devices. Nevertheless, in practice their application on modern devices remains very difficult because of their complexity but also because designers integrate environmental countermeasures.

This thesis is focused on the study of two sub-families of side channel attacks applied on modular exponentiation called respectively, collision based attacks and horizontal attacks.

This study is made according to two axes: their applications and the possible countermeasures.

Firstly, we study side channel attacks on a simulator developed during this thesis.

This simulator allows to validate the good implementation of a any side channel attack before its application in a real environment.

Secondly, we study collision based attacks in a real environment.

For this purpose, we study the automation of collision detection in practice. Then, we introduce a new collision detection criterion and show its practical interest. Afterwards, we study the estimation of the signal to noise ratio in the context of side channel attacks.

So, we introduce a fast and accurate method for its estimation during a side channel analysis.

From our method we derive pragmatic and efficient methods for the daily tasks of evaluators.

Among them the analysis of the electrical activity of integrated circuit or the identification of the frequencies carrying usable information or information leakage.

Finally, through a detailed description of the main stages of an horizontal attack, we propose effective and practical solutions to improve secret information extraction in real environment and on the other hand possible countermeasures against the horizontal attacks applied on modular exponentiation.

École Nationale Supérieure des Mines
de Saint-Étienne

NNT : *Communiqué le jour de la soutenance*

Ibrahima DIOP

METHODOLOGIE ET OUTILS POUR LA MISE EN PRATIQUE DES ATTAQUES PAR COLLISION ET ATTAQUES HORIZONTALES SUR L'EXPONENTATION MODULAIRE

Spécialité: Cryptologie

Mots clefs : Analyse par canaux cachés, Attaques par collision, Attaques horizontales

Résumé :

Dans cette thèse, nous étudions deux sous-familles d'attaques par canaux cachés sur l'exponentiation modulaire appelées respectivement attaques par collision et attaques horizontales.

Cette étude est faite selon deux axes : leurs mises en pratique et les possibles contremesures.

Dans un premier temps, nous étudions les attaques par canaux cachés sur un simulateur développé durant cette thèse. Ce simulateur permet de vérifier la bonne implémentation d'une attaque par canaux cachés avant sa mise en pratique dans un environnement réel.

Dans un deuxième temps, nous étudions les attaques par collision dans un environnement réel. Pour cela, nous nous sommes intéressés à l'automatisation de la détection effective de collision. Ainsi, nous proposons un nouveau critère de détection de collision.

Dans un troisième temps, nous étudions l'estimation du rapport signal à bruit d'un jeu de traces dans le contexte des attaques par canaux cachés.

Ainsi, nous proposons une nouvelle façon d'estimer le rapport signal à bruit lors d'une attaque par canaux cachés. En outre, nous montrons que cette estimation du rapport signal à bruit peut être exploitée pour l'analyse des fuites d'un canal caché mais aussi pour effectuer un filtrage adaptatif ne nécessitant pas la connaissance de certains paramètres du composant analysé.

Dans un quatrième temps, au travers d'une étude détaillée des principales étapes d'une attaque horizontale, nous montrons les problèmes pouvant intervenir dans la pratique et comment les résoudre. Ainsi des solutions génériques sont proposées.

Nous proposons finalement de possibles contremesures aux attaques horizontales sur l'exponentiation modulaire.