Méthodologie de conception de composants intégrés protégés contre les attaques par corrélation

Selma Laabidi

▶ **To cite this version:**

Selma Laabidi. Méthodologie de conception de composants intégrés protégés contre les attaques par corrélation. Autre. Ecole Nationale Supérieure des Mines de Saint-Etienne, 2010. Français. NNT : 2010EMSE0562 . tel-00488013

HAL Id: tel-00488013
https://theses.hal.science/tel-00488013

Submitted on 31 May 2010

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Ecole Nationale Supérieure des Mines
SAINT-ETIENNE

N° d'ordre : 2010 EMSE 0562

THÈSE

présentée par

Selma LAABIDI

pour obtenir le grade de
Docteur de l'École Nationale Supérieure des Mines de Saint-Étienne

Spécialité : Microélectronique

Méthodologie de conception de composants intégrés protégés contre les attaques par corrélation

soutenue à Gardanne, le 19 Janvier 2010

Membres du jury

Rapporteurs :	David NACCACHE	Professeur, Ecole Normale Supérieure, Paris
	Lionel TORRES	Professeur, LIRMM, Montpellier
Examinateur(s) :	Pierre PARADINAS	Professeur, CNAM, Paris
	Philippe MAURINE	Chargé de recherche, LIRMM, Montpellier
	Nathalie FEYT	Ingénieur, CESTI/CEACI (Thales), Toulouse
	Assia TRIA	Ingénieur chercheur, CEA/ENSMSE, Gardanne
	Bruno ROBISSON	Ingénieur chercheur, CEA/ENSMSE, Gardanne
Directeur(s) de thèse :	Philippe COLLOT	Professeur, ENSMSE, Gardanne

Spécialités doctorales :	**Responsables :**
SCIENCES ET GENIE DES MATERIAUX	J. DRIVER Directeur de recherche – Centre SMS
MECANIQUE ET INGENIERIE	A. VAUTRIN Professeur – Centre SMS
GENIE DES PROCEDES	G. THOMAS Professeur – Centre SPIN
SCIENCES DE LA TERRE	B. GUY Maître de recherche – Centre SPIN
SCIENCES ET GENIE DE L'ENVIRONNEMENT	J. BOURGOIS Professeur – Centre SITE
MATHEMATIQUES APPLIQUEES	E. TOUBOUL Ingénieur – Centre G2I
INFORMATIQUE	O. BOISSIER Professeur – Centre G2I
IMAGE, VISION, SIGNAL	JC. PINOLI Professeur – Centre CIS
GENIE INDUSTRIEL	P. BURLAT Professeur – Centre G2I
MICROELECTRONIQUE	Ph. COLLOT Professeur – Centre CMP

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

AVRIL	Stéphane	MA	Mécanique & Ingénierie	CIS
BATTON-HUBERT	Mireille	MA	Sciences & Génie de l'Environnement	SITE
BENABEN	Patrick	PR 2	Sciences & Génie des Matériaux	CMP
BERNACHE-ASSOLANT	Didier	PR 0	Génie des Procédés	CIS
BIGOT	Jean-Pierre	MR	Génie des Procédés	SPIN
BILAL	Essaïd	DR	Sciences de la Terre	SPIN
BOISSIER	Olivier	PR 2	Informatique	G2I
BOUCHER	Xavier	MA	Génie Industriel	G2I
BOUDAREL	Marie-Reine	MA	Génie Industriel	DF
BOURGOIS	Jacques	PR 0	Sciences & Génie de l'Environnement	SITE
BRODHAG	Christian	MR	Sciences & Génie de l'Environnement	SITE
BURLAT	Patrick	PR 2	Génie industriel	G2I
COLLOT	Philippe	PR 1	Microélectronique	CMP
COURNIL	Michel	PR 0	Génie des Procédés	SPIN
DAUZERE-PERES	Stéphane	PR 1	Génie industriel	CMP
DARRIEULAT	Michel	ICM	Sciences & Génie des Matériaux	SMS
DECHOMETS	Roland	PR 1	Sciences & Génie de l'Environnement	SITE
DESRAYAUD	Christophe	MA	Mécanique & Ingénierie	SMS
DELAFOSSE	David	PR 1	Sciences & Génie des Matériaux	SMS
DOLGUI	Alexandre	PR 1	Génie Industriel	G2I
DRAPIER	Sylvain	PR 2	Mécanique & Ingénierie	SMS
DRIVER	Julian	DR	Sciences & Génie des Matériaux	SMS
FEILLET	Dominique	PR 2	Génie Industriel	CMP
FOREST	Bernard	PR 1	Sciences & Génie des Matériaux	CIS
FORMISYN	Pascal	PR 1	Sciences & Génie de l'Environnement	SITE
FORTUNIER	Roland	PR 1	Sciences & Génie des Matériaux	SMS
FRACZKIEWICZ	Anna	DR	Sciences & Génie des Matériaux	SMS
GARCIA	Daniel	CR	Génie des Procédés	SPIN
GIRARDOT	Jean-Jacques	MR	Informatique	G2I
GOEURIOT	Dominique	MR	Sciences & Génie des Matériaux	SMS
GOEURIOT	Patrice	MR	Sciences & Génie des Matériaux	SMS
GRAILLOT	Didier	DR	Sciences & Génie de l'Environnement	SITE
GROSSEAU	Philippe	MR	Génie des Procédés	SPIN
GRUY	Frédéric	MR	Génie des Procédés	SPIN
GUILHOT	Bernard	DR	Génie des Procédés	CIS
GUY	Bernard	MR	Sciences de la Terre	SPIN
GUYONNET	René	DR	Génie des Procédés	SPIN
HERRI	Jean-Michel	PR 2	Génie des Procédés	SPIN
INAL	Karim	MR	Microélectronique	CMP
KLÖCKER	Helmut	MR	Sciences & Génie des Matériaux	SMS
LAFOREST	Valérie	CR	Sciences & Génie de l'Environnement	SITE
LERICHE	Rodolphe	CR	Mécanique et Ingénierie	SMS
LI	Jean-Michel	EC (CCI MP)	Microélectronique	CMP
LONDICHE	Henry	MR	Sciences & Génie de l'Environnement	SITE
MOLIMARD	Jérôme	MA	Mécanique et Ingénierie	SMS
MONTHEILLET	Frank	DR 1 CNRS	Sciences & Génie des Matériaux	SMS
PERIER-CAMBY	Laurent	PR1	Génie des Procédés	SPIN
PIJOLAT	Christophe	PR 1	Génie des Procédés	SPIN
PIJOLAT	Michèle	PR 1	Génie des Procédés	SPIN
PINOLI	Jean-Charles	PR 1	Image, Vision, Signal	CIS
STOLARZ	Jacques	CR	Sciences & Génie des Matériaux	SMS
SZAFNICKI	Konrad	CR	Sciences & Génie de l'Environnement	SITE
THOMAS	Gérard	PR 0	Génie des Procédés	SPIN
VALDIVIESO	François	MA	Sciences & Génie des Matériaux	SMS
VAUTRIN	Alain	PR 0	Mécanique & Ingénierie	SMS
VIRICELLE	Jean-Paul	MR	Génie des procédés	SPIN
WOLSKI	Krzysztof	CR	Sciences & Génie des Matériaux	SMS
XIE	Xiaolan	PR 1	Génie industriel	CIS

Glossaire :

PR 0	Professeur classe exceptionnelle
PR 1	Professeur 1ère catégorie
PR 2	Professeur 2ème catégorie
MA(MDC)	Maître assistant
DR (DR1)	Directeur de recherche
Ing.	Ingénieur
MR(DR2)	Maître de recherche
CR	Chargé de recherche
EC	Enseignant-chercheur
ICM	Ingénieur en chef des mines

Centres :

SMS	Sciences des Matériaux et des Structures
SPIN	Sciences des Processus Industriels et Naturels
SITE	Sciences Information et Technologies pour l'Environnement
G2I	Génie Industriel et Informatique
CMP	Centre de Microélectronique de Provence
CIS	Centre Ingénierie et Santé

Dernière mise à jour le : 22 juin 2009

École Nationale Supérieure des Mines
de Saint-Étienne

N° d'ordre : 2010 EMSE 0562

Selma LAABIDI

A DESIGN METHODOLOGY FOR INTEGRATED COMPONENTS PROTECTED FROM CORRELATION ATTACKS

Speciality: Microelectronics

Keywords: cryptographic algorithms, attacks, DPA (Differential Power Analysis), security, correlation, design flow

Abstract:

The cryptographic circuits, because they contain confidential information, are subject to fraudulent manipulations called attacks from malicious people. Several attacks have been identified and analyzed. Among them DPA (Differential Power Analysis), DEMA (Differential Electromagnetic Analysis), DBA (Differential Behaviour Analysis) and probing attacks form the class of correlation attacks and are considered as the most dangerous because they allow to retrieve, at lower cost, secret keys of cryptographic algorithms. Designers of secure circuits have thus added counter-measures to protect their circuits from these attacks. Counter-measures overhead got to have a minimum of impact on circuit's cost and performances.

In this thesis, we first focus on correlation attacks; the principle of these attacks is described as well as the main counter-measures to address them. A formalism describing these attacks is also proposed. Second, we study the safe evaluation tools to estimate the resistance of integrated circuits towards correlation attacks. After a state of the art on the existing tools, we describe our tool based on a search of correlations between the designer's model and the model which can be predicted by an attacker. The analysis of the correlations determines the most sensitive bits to complete an attack. This tool is integrated into the design flow to asses the strength of cryptographic algorithms at RTL (Register Transfer Level) and gate levels.

An application of our flow on several models of the algorithm AES (Advanced Encryption Standard) with and without counter-measures is proposed. The obtained results have demonstrated the effectiveness of our technique.

École Nationale Supérieure des Mines
de Saint-Étienne

N° d'ordre : 2010 EMSE 0562

Selma LAABIDI

METHODOLOGIE DE CONCEPTION DE COMPOSANTS INTEGRES PROTEGES CONTRE LES ATTAQUES PAR CORRELATION

Spécialité: Microélectronique

Mots clefs : algorithmes cryptographiques, attaques, DPA (Differential Power Analysis), sécurité, corrélation, flot de conception

Résumé :

Les circuits cryptographiques, parce qu'ils contiennent des informations confidentielles, font l'objet de manipulations frauduleuses, appelées communément attaques, de la part de personnes mal intentionnées. Plusieurs attaques ont été répertoriées et analysées. Parmi elles, les attaques DPA (Differential Power Analysis), DEMA (Differential Electromagnetic Analysis), DBA (Differential Behavior Analysis) et les attaques en probing forment la classe des attaques par corrélation et sont considérés comme les plus redoutables car elles permettent de retrouver, à moindre coût, les clefs de chiffrement des algorithmes cryptographiques. Les concepteurs de circuits sécurisés ont été donc amené à ajouter des parades, appelées contre-mesures, afin de protéger les circuits de ces attaques. Ces contre-mesures doivent impacter au minimum les performances et le coût du circuit.

Dans cette thèse, nous nous intéressons dans un premier temps aux attaques par corrélation, le principe de ces attaques est décrit ainsi que les principales contre-mesures pour y parer. Un formalisme décrivant de manière unique ces attaques est aussi proposé. Dans un deuxième temps, nous étudions les outils d'évaluation sécuritaires qui permettent d'estimer la résistance des circuits intégrés face aux attaques par corrélation. Après un état de l'art sur les outils existants, nous décrivons notre outil basé sur une recherche de corrélations entre le modèle du concepteur et le modèle qui peut être prédit par un attaquant. L'analyse de corrélations permet de déterminer les bits les plus sensibles pour mener à bien une attaque. Cet outil est intégré dans le flot de conception permettant ainsi d'évaluer la résistance des algorithmes cryptographiques au niveau RTL (Register Transfer Level) et portes.

Table des matières

Introduction **1**

1 La cryptographie dans le domaine de l'embarqué **5**
1.1 Introduction . . . 5
1.2 La cryptographie . . . 5
1.2.1 La cryptographie symétrique . . . 6
1.2.1.1 Le DES . . . 6
1.2.1.2 L'AES . . . 7
1.2.2 La cryptographie asymétrique . . . 8
1.2.2.1 Le RSA . . . 9
1.2.2.2 Les courbes elliptiques . . . 10
1.3 La cryptanalyse . . . 11
1.3.1 La cryptanalyse linéaire . . . 11
1.3.2 La cryptanalyse différentielle . . . 11
1.4 Présentation de la carte à puce . . . 14
1.4.1 Historique . . . 14
1.4.2 Description . . . 14
1.4.3 Communication . . . 14
1.4.4 Différents types de cartes à puce . . . 15
1.5 Conclusion . . . 15

2 État de l'art sur les attaques matérielles **17**
2.1 Introduction . . . 17
2.2 Les attaques par injection de fautes . . . 17
2.2.1 Méthodes d'injection de fautes . . . 18
2.2.1.1 Attaques par glitch . . . 18
2.2.1.2 Attaques lumières . . . 18
2.2.2 Modèles de fautes . . . 19
2.2.3 Exemples d'attaques en fautes . . . 19
2.2.3.1 DFA sur DES . . . 20
2.2.3.2 Attaque Safe error sur le RSA . . . 20
2.3 Les attaques par canaux cachés . . . 21
2.3.1 Les attaques temporelles . . . 21
2.3.2 Les attaques par analyse du courant . . . 22
2.3.2.1 Puissance consommée par les circuits CMOS . . . 22

2.3.2.2 Modèles de consommation 22
2.3.2.3 Mesure de courant 24
2.3.2.4 Analyse simple de consommation 25
2.3.2.5 DPA, CPA 25
2.3.3 Les attaques électromagnétiques 27
2.3.3.1 Phénomene physique 27
2.3.3.2 Les attaques 27
2.3.4 Attaque par observation de collisions 29
2.3.5 Attaques par caractérisation de bruit ou "Template Attack" 29
2.3.6 Attaque par analyse différentielle de comportement 30
2.3.6.1 Hypothèses 30
2.3.6.2 Déroulement de la DBA mono-bit 31
2.3.7 Attaques en probing 31
2.3.8 Classe des attaques par corrélation 32
2.3.9 Information exploitée par l'attaquant 33
2.4 Conclusion 34

3 Conception des circuits intégrés sécurisés 35
3.1 Introduction 35
3.2 Flot de conception des circuits intégrés 35
3.3 Contre-mesures 37
3.3.1 Hypothèses fondamentales 37
3.3.2 Principes de contre-mesures 38
3.3.3 Principe1 39
3.3.3.1 Équilibrage 39
3.3.3.2 Randomisation 42
3.3.3.3 Filtrage 44
3.3.4 Principe 2 44
3.3.4.1 Masquage 44
3.3.4.2 Partage du secret 45
3.3.5 Principe 3 45
3.3.6 Principe 4 45
3.3.6.1 Capteurs 46
3.3.6.2 Redondances spatiales 46
3.3.6.3 Redondances temporelles 49
3.3.7 Principe 5 49
3.3.8 Principe 6 51
3.4 Outils de conception dédiés aux circuits intégrés sécurisés 51
3.4.1 Analyse fonctionnelle : effets de bord 52
3.4.1.1 Simulation puissance et rayonnement électromagnétique . . 52
3.4.1.2 Estimation de la résistance 53
3.4.1.3 Résumé des travaux 54
3.4.2 Analyse fonctionnelle : attaques en faute 54
3.4.3 Analyse structurelle 55
3.4.3.1 Déséquilibre des capacités de charges 55
3.4.3.2 Déséquilibre des temps de transition 56

3.4.3.3 Déséquilibre des temps d'arrivée des signaux 56
3.5 Approche proposée 56
3.5.1 Recherche heuristique 57
3.5.1.1 Evolution du signal 57
3.5.1.2 Évolution du bruit 59
3.5.1.3 Exemple 60
3.5.2 Stratégie 1 61
3.5.2.1 Description 61
3.5.2.2 Avantages 62
3.5.2.3 Inconvénients 62
3.5.3 Stratégie 2 63
3.5.3.1 Description 63
3.5.4 Avantages 63
3.5.5 Inconvénients 64
3.5.6 Spécificités du travail de thèse 65
3.6 Conclusion 65

4 Méthodologie de conception pour l'évaluation des CIs 67
4.1 Introduction 67
4.2 Principe de corrélations dans les crypto-systèmes 67
4.2.1 Principe de Shannon 67
4.2.2 Corrélations dans les algorithmes cryptographiques 68
4.2.2.1 Cas du DES 68
4.2.2.2 Cas de l'AES 68
4.3 Analyse de corrélations 71
4.3.1 Formalisme 71
4.3.1.1 Circuit et modèle 71
4.3.1.2 Signaux 71
4.3.1.3 Syndromes 72
4.3.1.4 Résumé 72
4.3.2 Espace de recherche 73
4.3.3 Exemple d'attaques 74
4.3.3.1 Exemple : CPA power et EM 74
4.3.3.2 DBA et micro-sondage 75
4.3.3.3 Résumé 75
4.3.3.4 Estimation de la sécurité 76
4.3.4 Position du problème 77
4.3.4.1 Textes clairs, T 77
4.3.4.2 Clefs K 77
4.3.4.3 Syndromes S 78
4.3.4.4 Syndromes S' 78
4.3.4.5 Bases de temps 78
4.3.4.6 Opérateur de corrélation, $OCorr$ 78
4.4 Réduction de l'analyse et définitions associées 79
4.4.1 Bits corrélés 79
4.4.2 Bits d'attaque, bits sensibles 80

4.4.3 Recherche des bits sensibles . . . 81
4.4.4 Recherche des bits d'attaque . . . 81
4.5 Flot de conception sécurisé . . . 82
4.5.1 Automatisation des simulations . . . 82
4.5.2 Récupération des valeurs numériques et calcul des syndromes associés 83
4.5.3 Calcul de la matrice de corrélations . . . 83
4.6 Exemple d'application de l'analyse de corrélations . . . 83
4.6.1 Résultat de la méthode classique . . . 84
4.6.2 Résultat de notre méthode . . . 86
4.6.3 Comparaison entre les approches . . . 86
4.7 Conclusion . . . 89

5 Évaluation d'un crypto-processeur à clef secrète : l'AES 91
5.1 Introduction . . . 91
5.2 Architectures des crypto-processeurs . . . 91
5.2.1 Architecture globale . . . 91
5.2.1.1 Interface . . . 91
5.2.1.2 Le coeur de l'AES . . . 92
5.2.2 Versions sans contre-mesure . . . 93
5.2.2.1 Bloc SubBytes . . . 94
5.2.2.2 Bloc MixColumns . . . 94
5.2.3 Comparaison des versions sans contre-mesure . . . 96
5.2.4 Versions avec contre-mesures . . . 96
5.2.4.1 Dual-rail . . . 96
5.2.4.2 Masquage . . . 97
5.2.4.3 Comparaison des versions avec contre-mesure . . . 97
5.3 Évaluation sécuritaire des différents modèles . . . 98
5.3.1 Paramètres de l'étude . . . 98
5.3.2 Comparaison des versions sans contre-mesure . . . 99
5.3.2.1 Niveau RTL . . . 99
5.3.2.2 Niveau portes . . . 104
5.3.3 Evaluation de la contre-mesure "dual" face aux attaques CPA . . . 106
5.3.3.1 RTL . . . 106
5.3.3.2 Niveau portes rétro- annotée . . . 106
5.3.4 Étude de l'AES masqué . . . 109
5.3.4.1 Syndrome de poids de Hamming . . . 109
5.3.4.2 Syndrome de distance de Hamming . . . 112
5.3.4.3 Amélioration de la contre-mesure masquage . . . 115
5.4 Bits d'attaque . . . 115
5.4.1 S-Box $GF(2^4)$ . . . 116
5.4.2 S-Box LUT . . . 116
5.5 Conclusion . . . 116

Conclusion et perspectives 119

Liste des algorithmes

1 Algorithme DES 7
2 Algorithme AES 8
3 Implémentation *Square & Multiply Always* du RSA 21
4 Implémentation *Square & Multiply* du RSA 21
5 Implémentation *Double-and-add* d'une courbe elliptique 25
6 Recherche exhaustive 73

Table des figures

1.1 Schéma de Feistel 7
1.2 Schéma de la fonction F 8
1.3 Chiffrement de l'AES ($N_r = 10$) 9
1.4 Approximation linéaire d'une ronde du DES 12
1.5 Propagation de la différence ($0x00808200,60000000$) sur 3 tours du DES . . 13
1.6 Disposition des contacts d'une carte à puce suivant le standard ISO 7816-2 14
1.7 Architecture de la carte à puce à microprocesseur 15

2.1 Générateur d'impulsion (“ glitch”) 18
2.2 Laser pour l'injection de fautes 19
2.3 Porte CMOS élémentaire 23
2.4 Schéma de mesure et d'acquisition des courbes de courant 24
2.5 Courbe de courant courbe elliptique 26
2.6 Déroulement de l'attaque DPA [43] 28
2.7 Courbes DBA mono-bit réalisée sur un AES-128 32

3.1 Flot de conception des circuits intégrés sécurisés 36
3.2 Niveaux d'abstraction 37
3.3 Codage double rail 40
3.4 Porte NAND double rail (vue porte, à gauche et transistor, à droite) 40
3.5 Routage différentiel 41
3.6 Insertion d'instructions factices lors d'un calcul 43
3.7 Redistribution aléatoire d'instructions indépendantes 43
3.8 Découplage de l'alimentation à l'aide de condensateurs 44
3.9 Schéma du passage de la parité au niveau des S-Boxes 47
3.10 Schéma de la comparaison à chaque transformation 48
3.11 Schéma de la comparaison au niveau des S-Boxes 48
3.12 Circuits asynchrones et fautes 50
3.13 Schéma du vote majoritaire 51
3.14 Cellule double rail 55
3.15 Porte `XOR` double rail (vue “porte”) 58
3.16 Profil de courant d'une porte `XOR` en double rail (niveau RTL) 59
3.17 Profil de courant d'une porte `XOR` en double rail (niveau porte) 59
3.18 Évolution du bruit dans les niveaux d'abstraction 60
3.19 Simulation de consommation sur un AES dual sans le fichier SDF 61
3.20 Simulation de consommation sur un AES dual avec le fichier SDF 61

3.21 Recherche de modèles basée sur la réduction du SNR 62
3.22 Recherche de modèles basée sur la réduction du signal informatif 64

4.1 Définition des registres dans l'AES 69
4.2 Courbes DPA sur le premier bit de $s\_box[0]$ 70
4.3 Calcul des signaux de R^M 80
4.4 Architecture de l'implémentation de la S-Box 84
4.5 Numérotation des 16 S-Box de l'AES 84
4.6 Courbes CPA de l'AES pendant la première ronde 85
4.7 Courbes CPA de l'AES pendant la deuxième ronde 85
4.8 Répartition des bits du chemin de données en fonction de leurs corrélations 86
4.9 Corrélations du bit $SB_4/inverse/mapping/n2$ 87
4.10 Courbes CPA réalisées sur SB_4 89

5.1 Architecture globale du crypto-processeur 92
5.2 Architecture du bloc de chiffrement et de l'expansion de clef 93
5.3 S-Box (gauche) et MixColumns (droite) utilisant les opérations dans $GF(2^4)$ 95
5.4 Architecture de l'AES masqué 98
5.5 Architecture du MixColumns **V1** 101
5.6 Architecture du MixColumns **V2** 101
5.7 Répartition des bits corrélés 102
5.8 Architecture de l'implémentation de la S-Box 104
5.9 Corrélations pendant la troisième ronde 104
5.10 Vue partielle du bloc MixColumns (haut : chemin normal, bas : chemin dual)108
5.11 Dissymétries AES dual niveau portes 108
5.12 Courbes CPA de l'AES Dual 109
5.13 Valeurs maximales de corrélations pour le bit d'attaque $SB_0[5]$, clef k_0 . . . 110
5.14 Valeurs de corrélations pour le bit $state\_o[123]$ avec $SB_0^0[5]$ à l'instant $683ns$111
5.15 Valeurs maximales de corrélations pour le bit d'attaque $SB_0[5]$, clef k_1 . . . 111
5.16 CPA AES masqué 115
5.17 Bits d'attaques dans la S-Box $GF(2^4)$ 117
5.18 Corrélations S-Box $GF(2^4)$ 117
5.19 Corrélations S-Box LUT 118

Liste des tableaux

2.1 Courant consommé en fonction des transitions 23
2.2 Distance de Hamming en fonction des transitions 23
2.3 Poids de Hamming en fonction des transitions 24

3.1 Etude de la robustesse des circuits face aux attaques en puissance 54

4.1 Complexité des attaques 76
4.2 Correspondance entre les pics CPA et les bits sensibles 88

5.1 Caractéristiques des différentes versions de l'AES après synthèse 96
5.2 Performances des contre-mesures de l'AES après synthèse 97
5.3 Nbr. bits sensibles niveau RTL 99
5.4 Répartition des bits corrélés lors de la première ronde 100
5.5 Nbr. bits sensibles $2^{éme}$ ronde 102
5.6 Nbr. bits sensibles AES niveau RTL 105
5.7 Nbr. bits sensibles AES niveau portes 105
5.8 Nbr. bits sensibles AES dual RTL 106
5.9 Répartition des bits sensibles AES dual 107
5.10 Nbr. bits sensibles AES dual au niveau portes 107
5.11 Corrélations AES masqué 110
5.12 Bits sensibles pour l'AES masqué 112

Introduction

Sécurité et technologies sont intimement associées depuis toujours. Le monde électronique dans lequel nous œuvrons n'a fait qu'implanter le concept du sceau dans une puce, en donnant naissance à la carte à puce et aujourd'hui aux multiples composants sécurisés que nous utilisons quotidiennement.
C'est tout naturellement dans le domaine de la monnaie et du paiement que la sécurité s'est implantée tout d'abord, parce que la carte bancaire apportait un gain considérable de sécurité, de confiance pour l'utilisateur final, en combinant le concept du sceau, c'est à dire de l'authentification, avec celui de la mémorisation des événements ; des fonctions qui étaient jusqu'alors remplies séparément par la signature et un objet sécurisé primitif : le billet de banque ou la pièce de monnaie. Puis, avec le module SIM, est apparu l'usage de la puce électronique pour sécuriser les échanges d'information dans les réseaux de téléphonie cellulaire.
Aujourd'hui, les objets communicants sécurisés ne sont plus seulement la base de la confiance des transactions monétaires et des communications sans fil mais pénètrent aussi le monde physique, avec la sécurisation par exemple des locaux ou des biens matériels les plus variés. Comme leur appellation l'indique, ces circuits intégrés sont d'abord des microcontrôleurs, c'est-à-dire des composants réunissant sur une même puce un cœur de processeur optimisé pour gérer des interruptions, ainsi que des mémoires et autres périphériques utiles à la réalisation d'applications de contrôle. Les microprocesseurs pour carte à puce précisent le type d'applications auxquelles ils sont dédiés et induisent des obligations au niveau de la sécurité.
En effet, en charge du stockage et de la manipulation de données très confidentielles, il est naturel que ces puces soient soumises à de hautes exigences de sécurité concernant l'accès et le traitement des informations qu'elles abritent. Cette condition est le maître mot pour tous les systèmes concernés, qu'il s'agisse de la carte bancaire, de la carte SIM des téléphones mobiles, de la carte d'identité, du passeport électronique ou de la carte d'authentification tant sur Internet (e-commerce) que dans les entreprises (sécurité physique et des moyens informatiques), voire dans les applications citoyennes ou gouvernementales.
Des organismes de certification nationaux et internationaux délivrent des garanties de sécurité après avoir vérifié la conformité aux standards établis et que le niveau de protection est suffisant par rapport aux applications envisagées.
La cryptographie a permis d'introduire au travers d'algorithmes variés un niveau de sécurité élevé dans les systèmes de communication et services associés, et ce, en particulier au travers de l'utilisation de la carte à puce. En confinant et restreignant l'information confidentielle à protéger (clés), la sécurité de l'information transitant dans un système physique est améliorée ; cependant les techniques d'attaques de systèmes physiques visant

à extraire ces clés de cryptage ne cessent de progresser. Parmi elles on peut citer :

- Les attaques par injection de fautes qui exploitent le comportement du circuit en présence de perturbations
- Les attaques par canaux auxiliaires qui consistent à analyser le comportement d'un circuit au travers de constantes physiques observables telles que la consommation de puissance, le temps d'exécution ou le rayonnement émis.

L'un des plus redoutables attaques est l'attaque différentielle de puissance (*Differential Power Analysis*). Celle-ci fait partie d'une classe d'attaque que nous avons baptisée *attaques par corrélations* laquelle est basée sur une recherche de corrélations entre un modèle paramétré par la clef de chiffrement et une mesure du signal compromettant.

Face à ces techniques d'attaques dont l'évolution est particulièrement rapide, s'est développé un arsenal de contremesures multiples susceptibles d'améliorer la résistance aux attaques, en particulier aux deux premiers types décris ci-dessus. Celles-ci peuvent intervenir à différents niveaux tels que le composant, l'algorithme cryptographique et son implémentation. Des capteurs de variation de tension d'alimentation, de signal d'horloge, de température ou de lumière sont généralement rajoutés sur le composant. Les techniques de masquage et de duplication sont utilisées pour sécuriser les algorithmes cryptographiques en rendant les variables intermédiaires non prédictibles. Enfin des logiques dédiées comme la logique double rail et des styles logiques spécifiques (SABL, WDDL...) sont employées dans le but de réduire la corrélation entre les données manipulées et les canaux auxiliaires .

Malheureusement, la course poursuite engagée entre les concepteurs de circuits et systèmes sécuritaires et les hackers, s'accélère avec la diversité des systèmes, leur ouverture et leur multiplicité. Les enjeux nouveaux (contenus, transactions financières en ligne, contrôle d'informations confidentielles) participent également à l'accélération de ce processus. Il y a cependant une notion temporelle à prendre en compte : les hackers se concentrent sur des produits commerciaux correspondants à une ou deux générations antérieures en termes de technologie. Il apparaît donc aujourd'hui comme un enjeu majeur dans la sécurisation des systèmes de communication, et l'amélioration drastiquement de la résistance des composants à ces techniques d'attaques. .

Le développement d'un circuit cryptographique devient ainsi un réel challenge pour le concepteur. En effet, il doit apporter des contre-mesures qui n'impactent ni les performances du circuit ni le coût de sa commercialisation tout en étant confronté à trois problématiques majeures. D'une part, les ressources limitées des cartes à puce conditionnées par un standard réduisent fortement les marges de manoeuvre (taille silicium maximun de 25 mm^2). De plus, les outils de conception existants, non adaptés aux circuits sécurisés, peuvent engendrer un surcoût en surface important voire même rendre le circuit plus vulnérable aux attaques. Enfin,l'évaluation tardive de la résistance des contre-mesures, pendant la phase de caractérisation sécuritaire, risquent d'entraîner des pertes matérielles importantes.

C'est dans ce contexte que nous avons orienté nos travaux de recherche sur l'estimation de la sécurité des circuits intégrés dès la phase de conception. Une étude bibliographique sur les outils d'évaluation sécuritaire est réalisée à cet effet. Cette recherche nous a permis

de proposer une nouvelle approche pour estimer la résistance des circuits intégrés basée sur l'étude du signal utile d'un modèle de circuit décrit au niveau RTL et portes. Un outil a été développé pour démontrer l'intérêt de cette approche dans le cas particulier de l'estimation de la robustesse des contre-mesures vis-à-vis des attaques par corrélation.

Le premier chapitre présente les notions relatives à la cryptographie et la cryptanalyse. Des algorithmes cryptographiques à clef publique et à clef privée sont décrits et les techniques de cryptanalyse linéaire et différentielle sont présentées. Enfin, le système de carte à puce est exposé.

Un état de l'art sur les attaques physiques est dressé dans le deuxième chapitre. La première partie présente les attaques par injection de fautes. Quand à la seconde partie, elle détaille les attaques par canaux cachés et plus particulièrement les attaques par corrélation.

Dans le chapitre 3, le flot de conception des circuits intégrés sécurisés est rappelé. Un état de l'art sur les contre-mesures et les outils d'évaluation associés est détaillé. Nous concluons ce chapitre par l'exposition des limitations de ces outils et l'introduction de notre approche.

Une méthodologie de conception des circuits intégrés résistants aux attaques par corrélation est présentée dans le chapitre 4. Celle-ci est basée sur une analyse de corrélations dans un modèle de circuit décrit au niveau RTL et portes. Les notions de bits sensibles et bits d'attaque sont introduits lesquels définissent notre critère de sécurité.

Une application de cette méthodologie est exposée dans le chapitre 5 où différents modèles avec et sans contre-mesures du crypto-processeur AES sont décrits et évalués. Une analyse des résultats obtenus permet de mettre en évidence la complémentarité de notre approche par rapport aux méthodes existantes.

Chapitre 1

La cryptographie dans le domaine de l'embarqué

1.1 Introduction

Ce chapitre a pour objectif de familiariser le lecteur avec la cryptologie ou *la science des secrets*, constituée de deux disciplines duales qui sont la cryptographie, laquelle consiste à chiffrer les messages, et la cryptanalyse, laquelle consiste à déchiffer ces derniers. Ce chapitre abordera également la problématique liée à l'implémentation des techniques de cryptographie dans les composants intégrés.
Depuis des millénaires, les hommes et en particulier les dirigeants de tous les pays cherchent des moyens pour protéger le secret de leurs communications. La première trace de technique de cryptographie est apparue il y a 4000 ans en Egypte mais la cryptographie est ensuite restée, pendant de nombreuses années, exclusivement réservée au domaine militaire et diplomatique. La littérature sur le sujet était par conséquent peu abondante. Un virage a alors été amorcé par la publication en 1949 d'un article de Shannon qui pose les bases de la cryptographie moderne [64]. L'utilisation de cette dernière s'est ensuite rapidement développée. La cryptographie est actuellement l'un des piliers de la sécurité dans des réseaux de communication comme Internet mais son utilisation emblématique reste sans aucun doute la carte à puce. Ce module portatif offre une multitude de services nécessitant un haut degré de sécurité comme par exemple les transactions bancaires, l'identification et le contrôle d'accès, l'authentification pour l'accès à un service.
Dans ce chapitre, deux techniques fondamentales de cryptographie, à savoir la cryptographie symétrique et la cryptographie asymétrique, vont tout d'abord être présentées. Ensuite des éléments de cryptanalyse linéaire et différentielle sont développés. Enfin, les principaux éléments de la carte à puce seront décrits afin de mieux comprendre son fonctionnement mais aussi afin de mieux comprendre les attaques réalisées sur ce composant.

1.2 La cryptographie

La cryptographie sécurise les échanges d'information entre des personnes en assurant la confidentialité, l'intégrité et l'authenticité de l'information échangée mais également en fournissant aux personnes en contact des mécanismes de non-répudiation.

1. La confidentialité assure que l'information reste secrète de toutes les personnes qui ne sont pas autorisées à y accéder.
2. L'intégrité assure que l'information n'a pas été modifiée.
3. L'authenticité certifie l'identité de la personne qui envoie le message.
4. La non-répudiation permet d'éviter qu'une personne nie l'envoie de données qu'elle a effectivement envoyées.

Pour assurer ces quatre fonctions, la cryptographie met en œuvre diverses techniques, dont celles dites de *chiffrement*. Ces dernières transforment un message, appelé texte clair m, en un texte chiffré c qui ne sera lisible que par le destinataire légitime, grâce à une fonction E_e paramétrée par une clef e dite de chiffrement. Le destinataire du message ne pourra déchiffrer le message que s'il connait la clef d, dite de déchiffrement, qui paramètre une fonction D_d.

$$\begin{aligned} E_e &: \mathcal{M} \rightarrow \mathcal{C} \\ D_d &: \mathcal{C} \rightarrow \mathcal{M} \end{aligned}$$

Deux types d'algorithmes de chiffrement sont distingués : Si $e = d$ (et $D_d = E_e^{-1}$), le chiffrement est dit *symétrique* ou *à clef secrète*. Si, e et d sont différents, le chiffrement est *asymétrique* ou *à clef publique*.

1.2.1 La cryptographie symétrique

Les algorithmes de chiffrement symétriques sont ceux pour lesquels l'émetteur et le destinataire partagent une même clef. L'emploi de ce type d'algorithme lors d'une communication nécessite donc un échange préalable de la clef secrète entre les deux protagonistes à travers un canal supposé sécurisé (réalisable, par exemple, à l'aide de techniques de chiffrement asymétrique).
Deux types de chiffrements symétriques sont classiquement distingués : Ceux *par flot* et ceux *par blocs*. Les premiers traitent les données bit à bit et les seconds par blocs (typiquement de 64 ou 128 bits). Dans la suite de cette partie, les deux algorithmes de chiffrement par blocs actuellement les plus utilisés sont présentés. Il s'agit du DES (Data Encryption Standard) [67] et de son successeur l'AES (Advanced Encryption Standard) [68].

1.2.1.1 Le DES

Il a été élaboré par le NBS (National Bureau of Standards devenu le NIST), à partir d'un algorithme développé par IBM appelé *Lucifer*. Il a été standardisé en 1977. Le DES est un algorithme de chiffrement par blocs qui permet de chiffrer des données de 64 bits avec une clef de 56 bits.

Il est constitué d'une permutation initiale IP, de 16 itérations appelés **rondes** et d'une permutation finale P. Après la permutation initiale, le bloc de 64 bits est divisé en deux parties : droite (R pour *right*) et gauche (L pour *left*). Ces deux parties sont traitées alternativement. Ce croisement est connu sous le nom de fonction de Feistel. Cette fonction remplace la partie droite et gauche des variables intermédiaires par les valeurs suivantes :

$$\begin{aligned} L_i &= R_{i-1} \\ R_i &= L_{i-1} \oplus F(R_{i-1}, K_i) \end{aligned}$$

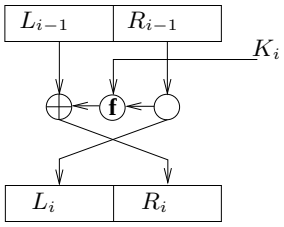

Fig. 1.1 – Schéma de Feistel

Algorithme 1 Algorithme DES

Entrées: P, K
Sorties: C
  $L_0R_0 = IP(P)$
  pour $1 \leq i \leq 16$ **faire**
    $L_i = R_{i-1}$
    $R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$
  fin pour
  $C = P(R_{16}L_{16})$

La fonction F est constituée d'une fonction d'expansion de 32 bits vers 48 bits, d'un XOR avec une sous clef K_i de 48 bits, de 8 fonctions de substitutions (boîte $SBOX_i$ avec $1 \leq i \leq 8$).

$$F(R_{i-1}, K_i) = P(S(E(R_{i-1}) \oplus K_i))$$

1.2.1.2 L'AES

L'AES est un algorithme de chiffrement symétrique qui opère sur des données de taille 128 bits avec une clef de taille 128, 192 ou 256 bits. Les données sont organisées en une matrice $4 * 4$ appelée état (*state*).

$$\begin{pmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{pmatrix}$$

Pour le chiffrement ainsi que pour le déchiffrement, l'AES utilise une fonction ronde. Le nombre de rondes N_r dépend de la longueur de la clef ($N_r = 10$ pour 128 bits, $N_r = 12$

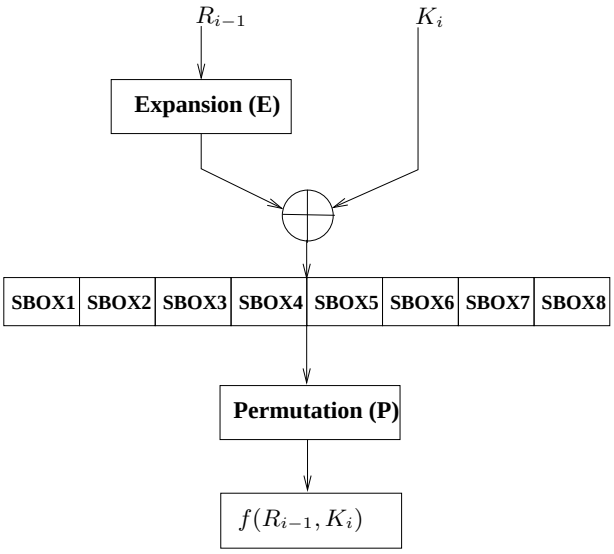

Fig. 1.2 – Schéma de la fonction F

pour 192 bits et $N_r = 14$ pour 256 bits). Chaque ronde comporte quatre transformations : `SubBytes`, `ShiftRows`, `MixColumns` et `AddRoundKey`. L'opération `AddRoundKey` est un XOR bit à bit entre la clef et l'état. La transformation `SubBytes` est une transformation non linéaire qui substitue chaque octet de l'état en utilisant une table de substitution, appelée *SBox*. L'opération `ShiftRows` est un décalage cyclique des lignes de l'état. Enfin, l'opération `MixColumns` est une multiplication matricielle dans $GF(2^8)$ de l'état avec une matrice constante.

Algorithme 2 Algorithme AES

Entrées: Etat, K
Sorties: C
`AddRoundKey`(Etat, K_0)
pour $1 \leq i \leq N_r - 1$ **faire**
`SubBytes`(Etat)
`ShiftRows`(Etat)
`MixColumns`(Etat)
`AddRoundKey`(Etat, K_i)
fin pour
P = Etat

1.2.2 La cryptographie asymétrique

C'est en 1976 que Diffie et Hellman [20] introduisent le concept de cryptographie asymétrique pour résoudre le problème d'échange de clefs posé par les chiffrements symétriques. Dans les protocoles de cryptographie asymétriques, chaque utilisateur dispose d'un couple de clefs : une clef secrète, connue de lui seul et une clef publique pouvant

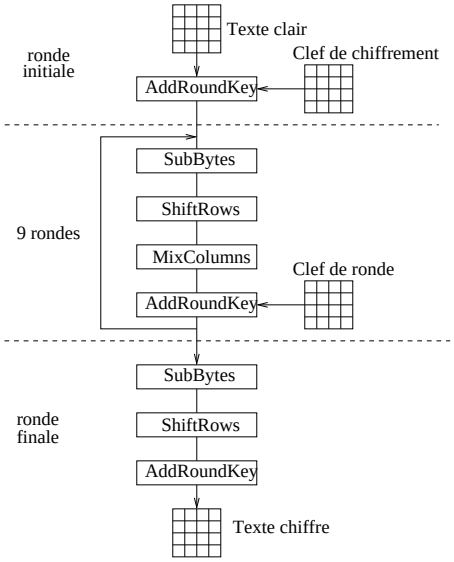

Fig. 1.3 – Chiffrement de l'AES ($N_r = 10$)

être communiquée à toute personne susceptible de communiquer avec lui. La première est généralement choisie par l'utilisateur et la seconde est calculée à partir de la clef publique grâce à des fonctions *à sens unique*. Celles-ci reposent sur des problèmes mathématiques réputés difficiles comme la factorisation de grands nombres. L'utilisation de ces problèmes mathématiques impliquent que les algorithmes de chiffrement à clef publique sont beaucoup plus lents que ceux à clef secrète, d'une part à cause de la complexité des opérations mises en jeux mais aussi à cause de la longueur des clefs utilisées. Ceci implique en particulier que la cryptographie à clef publique et la cryptographie à clef secrète sont souvent combinées en pratique, la première servant à transmettre la clef qui sera utilisée par la seconde pour effectuer le chiffrement des données à transmettre.
Dans cette partie, deux des principaux algorithmes utilisés en cryptographie asymétrique sont décrits : le RSA et les courbes elliptiques.

1.2.2.1 Le RSA

Le RSA a été inventé en 1997 par Rivest, Shamir et Adelman [59]. Sa sécurité est basée sur la difficulté de l'inversion modulaire.
Soit N le modulo public du RSA tel que N est le produit de deux grands nombres premiers secrets p et q. Soit e un entier tel que e est premier avec $(p-1)\cdot(q-1)$. Le couple (N, e) est appelé clef publique. L'exposant privée d est calculé tel que $ed = 1mod((p-1)\cdot(q-1))$.

Dans le RSA, les blocs de message sont représentés par des entiers compris entre 0 et $N-1$. Pour envoyer un message m à Bob, Alice va chercher la clef publique de Bob et elle calcule le message chiffré c correspondant par $C = M^e \bmod N$. Lorsqu'il reçoit le chiffré, Bob retrouve le texte clair en calculant $M = C^d \bmod N$.

1.2.2.2 Les courbes elliptiques

Les courbes elliptiques ont été étudiées en mathématiques depuis le milieu du dix-neuvième siècle mais ce n'est qu'en 1985 que N. Koblitz [33] et V. Miller [49] les ont introduites en cryptographie. Les courbes elliptiques permettent de travailler, à niveau de sécurité égal, avec des clefs de longueur significativement plus courte que celle du RSA. En effet, une clef de 147 bits utilisée dans le cadre des courbes elliptiques fournit le même niveau de sécurité qu'une clef RSA de 1077 bits[1]. Cet avantage est particulièrement important pour le déploiement d'algorithmes asymétriques en environnement contraint en ressources tel que la carte à puce.

Définition d'une courbe elliptique Soit E une courbe elliptique définie sur un corps $\mathcal{K}$. La courbe E peut être définie par son équation affine de Weierstrass :

$$y^2 + a_1xy + a_3y \quad = \quad x^3 + a_2x^2 + a_4x + a_6$$

avec a_1, a_2, a_3, a_4, $a_6 \in \mathcal{K}$. La courbe elliptique E est l'ensemble des points $(x, y) \in \mathcal{K}^2$ satisfaisant l'équation précédente et d'un point imaginaire $\mathcal{O}$ appelé point à l'infini.

Loi de groupe L'ensemble $E \bigcup \mathcal{O}$ est équipée de l'opération + dont l'élément neutre est $\mathcal{O}$. En cryptologie, les courbes elliptiques sont utilisées dans le corps $\mathbb{F}_{2^n}$ ou dans le corps $\mathbb{F}_p$ avec p un nombre premier strictement supérieur à 3.
Soit E une courbe elliptique définie sur $\mathbb{F}_p$. L'équation affine de Weierstrass de E peut être décrite par l'équation suivante :

$$y^2 \quad = \quad x^3 + ax + b$$

où a, $b \in \mathbb{F}_p$ tels que $4a^3 + 27b^2 \not\equiv 0 mod p$.
Soient $P = (x_1, y_1)$ et $Q = (x_2, y_2)$ deux points sur E :

- Le point $(x_1, -y_1)$ est l'opposé du point P et il est noté $-P$.
- Si $Q \neq P$ et $Q \neq -P$, alors le point $P + Q = (x_3, y_3)$ est défini par :

$$\begin{aligned} x_3 &= (\frac{y_2 - y_1}{x_2 - x_1})^2 - x_1 - x_2 \\ y_3 &= \frac{(x_1 - x_3)(y_2 - y_1)}{x_2 - x_1} - y_1 \end{aligned}$$

- Si $P \neq -P$, alors le point $2P = (x_3, y_3)$ est défini par :

$$\begin{aligned} x_3 &= (\frac{3x_1^2 + a}{2y_1})^2 - 2x_1 \\ y_3 &= \frac{(x_1 - x_3)(3x_1^2 + a)}{2y_1} - y_1 \end{aligned}$$

Crypto-système basé sur les courbes elliptiques Etant donné E une courbe elliptique sur un corps fini $\mathbb{F}_p$ et P et Q deux points de E, le problème du logarithme discret, sur lequel est basé la sécurité des cryptosystèmes à courbes elliptiques, consiste à trouver un nombre k tel que $Q = kP$.

[1] selon le site www.keylength.com

1.3 La cryptanalyse

Les techniques de cryptanalyse visent à retrouver le message clair à partir d'un chiffré sans bien sûr connaître la clef utililisée pour le chiffrement. Généralement, le cryptanalyste, ou *attaquant* est supposé connaître l'algorithme cryptographique utilisé (conformément au principe de A. Kerckhoffs). Deux types de cryptanalyse sont classiquement distinguées : la cryptanalyse linéaire et la cryptanalyse différentielle.

1.3.1 La cryptanalyse linéaire

Introduite par Mastui et al. [45] en 1993 sur DES, cette technique de cryptanalyse essaie d'approximer un algorithme de chiffrement par blocs par une fonction linéaire, c'est-à-dire s'exprimant comme une somme (XOR) des bits du texte clair, du chiffré et de la clef. Si l'approximation obtenue est vraie avec une probabilité différente de 0.5, alors elle peut être exploitée pour retrouver des bits de la clef. Mastui et al. montrent, par exemple, qu'une des S-Box du DES, la cinquième, s'approxime particulièrement bien grâce à une fonction linéaire. En effet, le 2nd bit en entrée de cette S-Box est égal au XOR des 4 bits en sortie de la S-Box, et ce pour 12 entrées sur les 64 possibles (donc avec une probabilité égale à $12/64 = 0.19$). Avec les notations de la 1.4, on a alors :

$$B[26] \quad = \quad C[17] \oplus C[18] \oplus C[19] \oplus C[20]$$

Comme les équations suivantes sont vérifiées quelque soit le texte :

$$\begin{aligned} B[26] &= A[26] \oplus K_i[26] \\ A[26] &= X[17] \\ C[17] &= Y[8] \\ C[18] &= Y[14] \\ C[19] &= Y[25] \\ C[20] &= Y[3] \end{aligned}$$

On obtient l'équation suivante qui est vérifiée avec une probabilité de 0.19 :

$$K_i[26] \quad = \quad X[17] \oplus Y[3] \oplus Y[8] \oplus Y[14] \oplus Y[25]$$

Pour attaquer une ronde du DES, l'attaquant calculera alors le terme gauche de l'équation précédente pour N paires de textes clairs et chiffrés. Soit T le nombre de paires pour lesquels ce terme est nul. Si $T > N/2$ alors $K_i[26] = 1$ et 0 sinon.

Cette technique a permis aux auteurs d'attaquer un DES de 8 rondes et un DES complet avec respectivement 2^{21} textes clairs connus, et 2^{47} textes clairs connus.

1.3.2 La cryptanalyse différentielle

Développé par Biham et Shamir [6], la cryptanalyse différentielle recherche des relations entre la différence entre une paire de textes clairs et la différences entre la paire de chiffrés associés. Elle peut être mise en oeuvre dès lors que le chiffrement présente la faiblesse

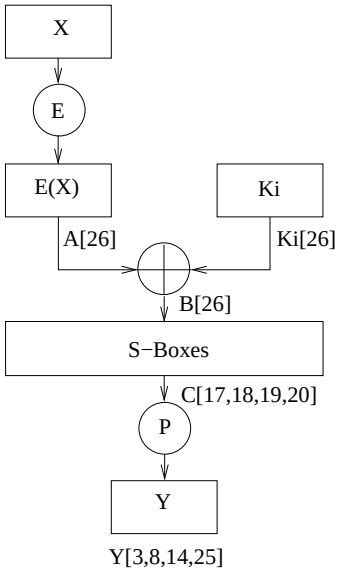

Fig. 1.4 – Approximation linéaire d'une ronde du DES

suivante : il existe un couple de différences (a, b) tel que la différence entre les images par le chiffrement de deux entrées dont la différence vaut a est égale à b avec une probabilité élevée. La différence est généralement le ou exclusif (`XOR`).
Toute la difficulté pour concevoir une attaque différentielle réside dans la recherche d'un couple de différences (a, b) qui soit propagé avec une probabilité élevée.

Attaque Différentielle du DES à 4 tours Dans le but de trouver une différence a qui se propage à travers plusieurs tours du DES, nous commençons par nous intéresser à un seul tour. Considérons deux entrées (L,R) et (L',R') dont les 32 bits de gauche différent d'une constante α et les 32 bits de droite différent de $0x60000000$. Autrement dit, tous les bits R et R' sont égaux sauf les bits en deuxième et troisième position à partir de la gauche. La fonction `Expansion` envoie les bits 2 et 3 de son entrée sur les bits 3 et 4 de sa sortie. Le fait d'ajouter la sous clef k ne change pas la valeur de la différence. Ainsi, à l'entrée des S-Box, les 2 mots que l'on considère ne différent que sur leurs troisième et quatrième bits. Ces 2 bits n'interviennent qu'en entrée de la première SBox. Les entrées des 7 autres SBox sont les mêmes pour les 2 messages, donc leurs sorties seront les mêmes. Maintenant, il faut déterminer la valeur du XOR en sortie de la première S-Box sachant que le XOR en entrée est égale à 001100 (0xB). Une étude de la première S-Box permet de constater que sur les 64 valeurs possibles x, 14 d'entre elles vérifient :

$$S_1(x) \oplus S_1(x \oplus 001100) = 1110$$

Les 2 octets en sortie de la première S-Box ont donc une différence égale à $0xE$ avec une probabilité égale à $\frac{7}{32}$. La `permutation` P envoie les bits 1, 2 et 3 en positions 9, 17 et 23. Donc en sortie de P, les deux mots diffèrent de $0x00808200$. En ajoutant la partie gauche des entrées, on en déduit :

$$(x \oplus L) \oplus (x' \oplus L') = (x \oplus x') \oplus (L \oplus L') = 00808200 \oplus \alpha$$

Grâce à cette étude, il est possible de trouver un couple de différences qui se propage sur 3 tours avec une probabilité élevée.

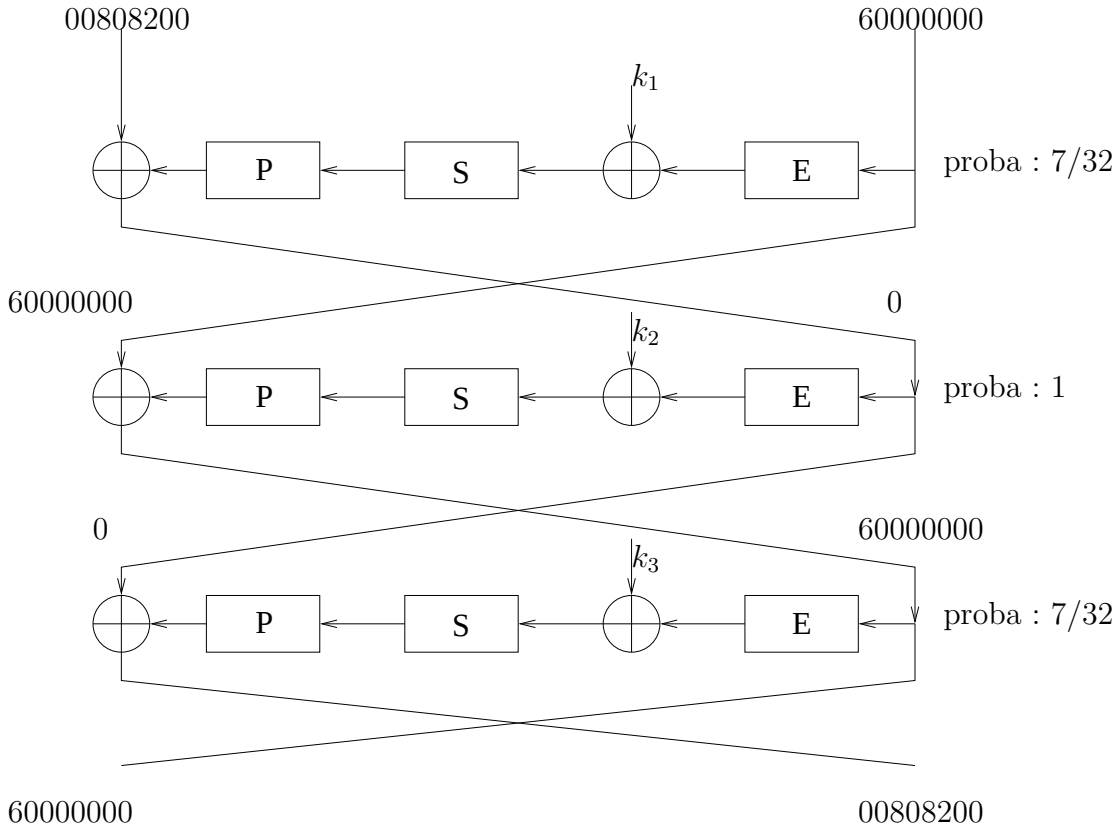

Fig. 1.5 – Propagation de la différence ($0x$00808200,60000000) sur 3 tours du DES

D'après la Figure 1.5, en choisissant une paire de textes clairs qui diffèrent de la valeur ($0x$00808200,$0x$60000000), on obtient une paire de textes à la sortie de la troisième ronde qui différent de la valeur (60000000,$0x$00808200) avec une probabilité égale à :

$$\frac{7}{32} * 1 * \frac{7}{32} = 0.048$$

En considérant 100 couples d'entrées-sorties de la forme $(x_1, y_1), (x_1 \oplus a, y'_1)$, ...$(x_{50}, y_{50}), (x_{50} \oplus a, y'_{50})$ où $a = (0x00808200, 0x60000000)$. Si le nombre de couples (y_i, y'_i) tel que $y_i \oplus y'_i = (0x60000000, 0x00808200)$ est égale à $0,048 * 50 = 2,4$, alors les couples de valeurs sont bien des couples d'entrées-sorties d'un DES à 3 tours. Avec les couples de chiffrés (y_i, y'_i), il est possible de déterminer la clef partielle du 4ième tour du DES.

La cryptanalyse linéaire et différentielle ont permis d'attaquer l'algorithme DES en utilisant un très grand nombre de textes clairs. Toutefois, le DES est encore utilisé dans le domaine de la sécurité comme la carte à puce par exemple. C'est ce que nous allons entre autre abordé dans la prochaine section.

1.4 Présentation de la carte à puce

1.4.1 Historique

Le concept de la carte à puce fut inventé dans les années 1970. En effet, en 1974, Roland Moreno déposa un brevet intitulé *Procédé et dispositif de commande électronique*, qui explique le fonctionnement d'une carte à mémoire permettant de stocker des données. Trois ans plus tard, Michel Ugon développe l'idée de Roland Moreno en y rajoutant un microcontrôleur, créant ainsi la carte à microprocesseur. Cette carte est programmable et son architecture s'apparente à celle d'un (petit) ordinateur.
La carte à puce commença vraiment à exister aux yeux du grand public en 1983, avec l'apparition des cartes téléphoniques. La fonctionnalité de ces premières cartes consiste à décrémenter le nombre d'unités restantes, chaque unité représentant une certaine durée de communication. Ce furent les banques françaises qui introduisirent en premier les cartes à microprocesseur en 1984 pour sécuriser leurs transactions.
Afin de standardiser les caractéristiques physiques des cartes à puce ainsi que leur utilisation lors d'échanges internationaux, plusieurs normes **ISO** (International Organization for Standardization) portant sur la carte à puce furent rédigées à partir de 1987.

1.4.2 Description

Le format de la carte plastique est définie de manière très précise dans les normes ISO/IEC 7816-1 [1] et -2 [2]. Dans ces normes, sont définies également les contraintes mécaniques que doit pouvoir supporter la carte à puce (humidité, chaleur, déformation...).

1.4.3 Communication

Il existe actuellement deux types de communication entre la puce et le monde extérieur : la communication avec contact et la communication sans contact.
Une carte à contact possède huit contacts électriques dont six seulement sont actuellement utilisés pour alimenter la puce ou communiquer avec elle (Figure 1.6) :

- **VCC** : tension d'alimentation
- **RST** : signal de remise à zéro
- **CLK** : signal d'horloge
- **GND** : masse
- **VPP** : tension de programmation qui servait par le passé à alimenter une pompe de charge utilisée pour les écritures en EEPROM (mais inutilisée de nos jours)
- **I/O** : entrées/sorties des données entre la carte et le monde extérieur
- **RFU** : réservé pour un usage futur

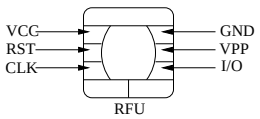

Fig. 1.6 – Disposition des contacts d'une carte à puce suivant le standard ISO 7816-2

Dans le cas d'une carte sans contact, la puce est alimentée et communique avec le lecteur grâce à une antenne. Celle-ci est soit intégrée dans la carte soit sur la puce elle-même. Les cartes à puce sans contact sont souvent utilisées dans les domaines où les transactions sont fréquentes (transports en commun, accès remontées mécaniques, etc..).

1.4.4 Différents types de cartes à puce

Les cartes à puce peuvent être divisées en deux catégories : les cartes à mémoire qui ont une architecture très basique et les cartes à microprocesseur, plus complexes, aussi appelées aussi cartes asynchrones, et principalement composées :

- D'un microprocesseur (de 8, 16 ou 32 bits suivant la puissance requise par l'application).
- De blocs de mémoires volatiles ou non. La ROM est généralement initialisée au début de la vie de la carte pendant l'étape de masquage et sert à stocker le système d'exploitation. La mémoire EEPROM est initialisée lors de la personnalisation de la carte et contient donc des données spécifiques à cette dernière. La RAM est la mémoire de travail du processeur.
- D'interfaces d'entrées/sorties.
- En option, un générateur de nombres aléatoires (RNG) qui permet la génération de clefs pour l'authentification mutuelle de la carte et du terminal, et l'implémentation de certaines contre-mesures contre les attaques par canaux cachés.
- En option, un coprocesseur permettant de faire tout ou partie de certaines opérations cryptographiques comme une multiplication modulaire ou un chiffrement DES ou AES.

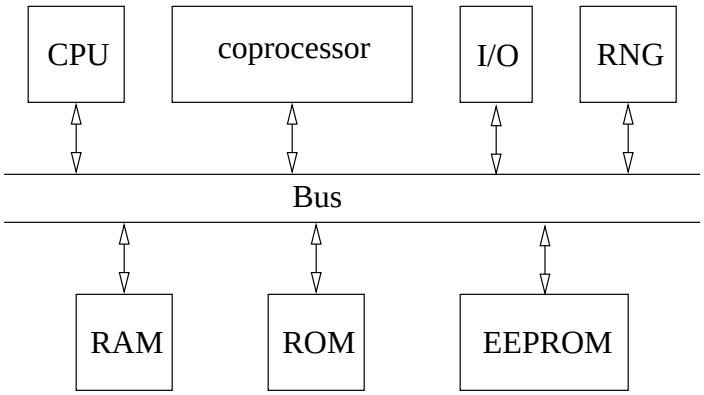

Fig. 1.7 – Architecture de la carte à puce à microprocesseur

1.5 Conclusion

Dans ce chapitre, les deux types de chiffrement utilisés dans le domaine de la cryptographie à savoir le chiffrement symétrique et asymétrique, ont été précisés. Des exemples d'algorithmes de ces deux types ont également été donnés. Nous avons ensuite décrit deux techniques théoriques qui sont classiquement envisagées pour mettre à mal ces algorithmes.

Un exemple de circuit embarquant typiquement ces derniers, la carte à puce, a enfin été présenté.

Dans le chapitre suivant, nous allons nous intéresser à un autre type d'attaques, qui contrairement à celles présentées dans ce chapitre, n'exploitent pas uniquement les propriétés mathématiques des algorithmes mais les faiblesses de l'implémentation et les caractéristiques physiques du composant qui les embarque.

Chapitre 2

État de l'art sur les attaques matérielles

2.1 Introduction

Les composants de sécurité, parce qu'ils contiennent des informations confidentielles, font l'objet d'attaques. Celles-ci tentent généralement de porter atteinte à la confidentialité, à l'intégrité ou à l'authenticité des données protégées par les algorithmes de cryptographie embarqués sur ces composants. Les attaques dites *matérielles* ou *physiques* exploitent les faiblesses de l'implémentation matérielle de ces algorithmes.
Il existe plusieurs techniques, généralement combinées, pour mettre à mal un circuit cryptographique. Un attaquant peut utiliser un microscope optique pour rétro-concevoir le circuit (*reverse engineering*) ou pour déduire la valeur de données sensibles [66]. Il peut également retirer les différentes couches de passivation et d'isolation afin de lire la valeur des signaux transitant sur les bus grâce à des micro-sondes. On parle généralement dans ce cas, d'attaques par "probing". Il peut même envisager, en utilisant, par exemple, un FIB (Focused Ion Beam) [36] de modifier la fonctionnalité du circuit (et rendre inopérante une protection). D'autres méthodes, moins "intrusives" mais très efficaces sont basées sur l'analyse du comportement du circuit en présence de perturbations ou sur l'observation des modifications de l'environnement créées par le circuit lors qu'il réalise des calculs cryptographiques.
Dans ce chapitre, ces deux derniers types d'attaques seront détaillés. Un point commun sera ensuite mis en évidence et utilisé pour proposer une nouvelle technique d'attaque en "probing".

2.2 Les attaques par injection de fautes

C'est en 1975 que des scientifiques de l'industrie aérospatiale rapportent 4 inversions spontanées de bits dans des bascules bipolaires J-K d'un satellite de communication [8]. Ils expliquent ce phénomène par des atomes de fer présents dans le vent solaire. Trois ans plus tard, Intel observe ces mêmes basculements de bits sur des mémoires de type DRAM [46], causées par une contamination radioactive des matériaux du boîtier des mémoires. Afin de rendre leur circuits tolérants à ces perturbations, les scientifiques ont, dans un premier

temps, cherché à reproduire l'effet des rayons cosmiques ou de particules radioactives sur les semi-conducteurs [76] puis à modéliser cet effet. Ils ont ensuite testé et validé des techniques de conception de circuits insensibles à ces effets, techniques essentiellement basées sur de la redondance (temporelle, spatiale ou d'information).
S'inspirant très certainement des résultats obtenus par ces travaux, des perturbations, cette fois-ci intentionnelles, ont été utilisées en septembre 1996 pour retrouver les informations secrètes dans une carte à puce [11]. Depuis, les techniques d'injection de fautes ont été améliorées. Elles seront, ainsi que leur effet sur les circuits, présentées dans cette section.

2.2.1 Méthodes d'injection de fautes

Plusieurs méthodes permettent de perturber un circuit. Les plus utilisées dans l'industrie de la carte à puce consistent à le soumettre à des impulsions électriques (appelées "glitch") ou l'exposer à des lumières intenses (laser, UV, flash etc...).

2.2.1.1 Attaques par glitch

Il s'agit d'insérer une impulsion d'une amplitude et d'une durée maîtrisée soit sur le signal d'horloge, soit sur la tension d'alimentation. Dans le premier cas, l'impulsion causera, sur un circuit synchrone, un échantillonnage prématuré de certaines données. Celles-ci seront erronées si les portes logiques combinatoires qui les engendrent n'ont pas fini de calculer. Dans le second cas, les transistors voient leur tension de seuil modifiée et peuvent changer d'état (de l'état passant ils passent à l'état bloqué ou vice versa).
L'injection de fautes par glitch est relativement facile à mettre en œuvre car ne nécessite pas de matériels sophistiqués ni de préparation du composant mais ne permet pas facilement de focaliser l'attaque sur une partie spécifique du circuit.

Fig. 2.1 – Générateur d'impulsion (" glitch")

2.2.1.2 Attaques lumières

Cette attaque consiste à éclairer intensément le silicium du composant à l'aide d'un laser voire d'un simple flash d'appareil photo dont la lumière est concentrée par un microscope [66]. L'énergie transportée par la source lumineuse est absorbée par les électrons, ce qui leur permet de passer de la bande interdite à la bande de conduction. Chaque nouvel électron dans la bande de conduction crée une paire électron trou et l'ensemble

de ces paires forme un courant photoélectrique. Les portes logiques ainsi que les bascules du composant étant sensibles à ce courant peuvent changer d'état. Trois paramètres principaux sont à définir pour réussir une attaque optique : la longueur d'onde, la durée de l'émission et la localisation de la zone touchée. Elle nécessite une décapsulation préalable du composant voire un amincissement en face arrière du substrat.

Fig. 2.2 – Laser pour l'injection de fautes

2.2.2 Modèles de fautes

Les fautes générées par les techniques décrites ci-dessus peuvent être soit permanentes soit transitoires [4], affecter un ou plusieurs bits (faute "simple" ou "multiple") de n'importe quel bloc logique du circuit (RAM, mémoires stockant les programmes, combinatoires, etc...). Différents effets sur ces bits sont classiquement distingués :

- La valeur du bit est inversé par rapport à la valeur qu'il aurait eu sans perturbation. On parle alors de *bit flip*.
- La valeur du bit est forcée à une valeur fixe. On parle de *set* si la valeur est collée à 1 et à 0 *reset* sinon.
- La valeur du bit est forcée à sa valeur précédente. On parle de *collage à 1* si la valeur précédente était égale à 1 et */collage à 0* sinon.
- La valeur du bit est aléatoire *random*.

2.2.3 Exemples d'attaques en fautes

Les attaques en faute, qui concernent aussi bien les algorithmes à clef publique que ceux à clef privée, sont classés en 4 catégories :

- Les attaques par analyse différentielle de fautes (DFA pour *Differential Fault Analysis*) comparent les chiffrés obtenus lors d'exécutions normales et fautées. Elles se basent sur les techniques de cryptanalyse différentielles présentées dans 1.3.2 [7] [21] [56] [26].
- Les attaques en *Safe Error* comparent la justesse des chiffrés obtenus lors d'exécutions normales et fautées [73] [9].
- Les attaques par création de collision est basée sur l'apparition d'un état intermédiaire identique lors de l'exécution [10].
- Les attaques de la machine d'état tente de court-circuiter un test conditionnel pour, par exemple, le réduire le nombre de rondes [17].

Dans cette section, les deux premières classes d'attaques en faute sont détaillées.

2.2.3.1 DFA sur DES

Dans cette section, nous allons décrire le principe de la première attaque DFA sur un crypto-système symétrique à savoir l'attaque du DES par Biham et Shamir [7].
Supposons qu'une faute affectant un bit est induite sur la partie droite du résultat temporaire au début de dernier tour ($R_{15} = L_{16}$). Nous notons respectivement C et $\tilde{C}$ la valeur du chiffré correct et du chiffré erroné. En observant la partie gauche de $IP(C)$ et $IP(\tilde{C})$, il y a seulement un seul bit qui diffère. Ce dernier correspond à la faute induite, ce qui permet d'obtenir la position du bit erroné.
En calculant $E(L_{16}) \oplus E(\tilde{L_{16}})$, nous avons la différence $\Delta_{entrées}$ entre l'entrée correcte et l'entrée erronée des boîtes de substitution dans le dernier tour. Par ailleurs, $P^{-1}(R_{16}) \oplus P^{-1}(\tilde{R_{16}})$ donne la différence $\Delta_{sorties}$ entre la sortie correcte et la sortie erronée des boîtes de substitution dans le dernier tour.
Supposons que la faute n'affecte qu'une seule S-Box, afin de trouver les 6 bits de la clef partielle en entrée de cette S-Box, on liste tous les couples (x_i, x'_i) tel que $x_i \oplus x'_i = \Delta_{entrées}$ et parcourir cette liste en éliminant les couple (x_i, x'_i) ne satisfaisant pas la relation $SBox(x_i) \oplus SBox(x'_i) = \Delta_{sortie}$.
En utilisant plusieurs chiffrés, les 6 bits de la clef du dernier tour sont retrouvés.

2.2.3.2 Attaque Safe error sur le RSA

Dans [73], Yen et Joye publient une attaque par perturbation sur l'algorithme *Square & Multiply Always* utilisé pour l'exponentiation modulaire du RSA 3. Ils remarquent que le résultat de la multiplication (noté *multiply* dans l'algorithme 3), n'est pas utilisé si le bit de l'exposant d_i est égal à 0. En perturbant une multiplication et en testant si la sortie correspondante de l'exponentiation est correcte ou pas, l'attaquant déduit la valeur du bit correspondant de l'exposant.

Lors des attaques décrites ci-dessus l'attaquant est amené à agir sur le circuit, au risque parfois de l'endommager voire de le détruire. Au contraire, les attaques par canaux cachés, qui sont décrites dans la section suivante et qui consistent seulement à "écouter" passivement le fonctionnement du circuit, n'ont pas cet inconvénient.

Algorithme 3 Implémentation *Square & Multiply Always* du RSA

Entrées: M, $d = (d_{m-1}, ..., d_0)_2, N$
Sorties: $A = M^d \bmod N$

```
A ← 1
pour 0 ≤ i ≤ m − 1 faire
    A1 ← A^2 mod N        square
    A2 ← A1 · M mod N     multiply
    si d_i = 1 alors
        A ← A2
    sinon
        A ← A1
    finsi
fin pour
Retourner A
```

2.3 Les attaques par canaux cachés

Dans les années 70, les patrons des chaînes télévisées américaines ont trouvé un bon moyen pour évaluer l'audimat. En effet, en observant les variations d'eau et d'électricité dans la ville de New York pendant les pauses publicitaires, ils étaient capables de mesurer indirectement le succès d'une émission de télévision. Ce fut la première utilisation répertoriée d'un *canal caché*. Nous expliquons dans la partie suivant comment ce principe a depuis été appliqué pour retrouver des informations stockées dans les cartes à puces.

2.3.1 Les attaques temporelles

En 1996, Kocher publie un article théorique sur l'utilisation de la mesure de temps, pour retrouver des clefs cryptographiques [34]. Il se base sur la corrélation qui peut exister entre le temps d'exécution d'une fonction et les données qu'elle manipule. Deux ans plus tard, Koeune met en application cette analyse [19]. Il démontre comment il était possible de mettre à mal une implémentation de l'algorithme *Square & Multiply* servant à l'exponentiation modulaire utilisée dans le RSA (Algorithme 4).
Dans ce type d'implémentation, le temps de calcul de chaque itération dépend de la valeur

Algorithme 4 Implémentation *Square & Multiply* du RSA

Entrées: $M, d = (e_{m-1}, ..., d_0)_2, N$
Sorties: $A = M^d \bmod N$

```
A ← 1
pour 0 ≤ i ≤ m − 1 faire
    A ← A^2 mod N         square
    si d_i = 1 alors
        A ← A · M mod N   multiply
    finsi
fin pour
Retourner A
```

de d_i. En effet, selon la valeur de d_i, une multiplication est effectuée ou non entraînant un temps de calcul supplémentaire ou non. Ainsi, en mesurant le temps d'exécution de chaque itération, l'attaquant est capable de deviner la valeur de d_i. En itérant cette opération m fois, il est capable de retrouver la clef.

2.3.2 Les attaques par analyse du courant

Le courant consommé par les circuits intégrés a été exploité initialement comme canal caché par Kocher [35] pour donner naissance à une très redoutable attaques : la DPA (pour Differential Power Analysis). Celle-ci est basée sur l'analyse des dépendances entre les données manipulés et le courant consommé, comme nous allons le montrer dans le paragraphe suivant pour les technologies CMOS (lesquelles sont actuellement les plus répandues).

2.3.2.1 Puissance consommée par les circuits CMOS

Lorsqu'un circuit est réalisé sur silicium, les portes logiques qui le constituent sont physiquement distantes les unes des autres. Les interconnexions entre les portes, réalisées par des pistes parallèles au substrat, créent des condensateurs, dits de ligne. La valeur de ces condensateurs est, en première approximation, proportionnelle à la longueur des interconnexions. De plus, chaque entrée d'une porte logique présente au fil qui lui est connecté un condensateur, dit d'entrée. La valeur de ce dernier est liée à la réalisation niveau « transistor » de la porte et au placement-routage de celle-ci. Les condensateurs de ligne et d'entrée sont des composants "parasites" qui contribuent à l'absorption de puissance du circuit. On distingue classiquement la puissance statique de la puissance dynamique. La première est due au courant "de fuite" qui traverse les transistors qui constituent la porte quand ils sont dans un état stable (bloqué ou passant) (I_{stat}). La seconde se produit quand ces transistors changent d'état. Ces commutations peuvent provoquer un courant de court-circuit (I_{cc}) entre la masse et l'alimentation. Suivant le sens de la commutation, les condensateurs parasites qui sont connectés à la sortie de la porte sont chargés ou déchargés. Si la transition a lieu de l'état bas vers l'état haut, les condensateurs parasites sont chargés par le courant (I_{cl}) en provenance de l'alimentation ; Si au contraire la transition a lieu de l'état haut vers l'état bas, la décharge des condensateurs de fait à travers la masse (et dans ce cas, aucun courant n'est absorbé). A noter que sur les technologies actuelles, I_{cl}, I_{cc}, I_{stat} représentent respectivement 70-90%, 10-30% et moins de 1% du courant total. Sur les technologies futures, la part du courant de fuite aura tendance à augmenter à cause de la diminution de l'épaisseur de l'oxyde de grille des MOS.

Le tableau 2.1 reporte les transitions et leur consommation associée pour la plus simple des portes logiques, l'inverseur. Sur cet exemple simple, une transition vers un état haut consomme plus qu'une transition vers l'état bas, la différence entre les deux étant égale à la charge du condensateur parasite en sortie de l'inverseur.

2.3.2.2 Modèles de consommation

Des modèles simplifiés, déduits des considérations précédentes sont classiquement utilisés pour réaliser des attaques par analyse de la consommation : le modèle de la distance

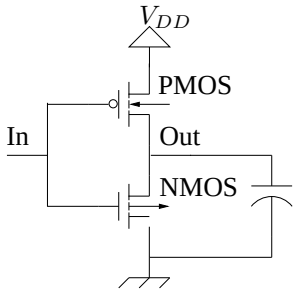

Fig. 2.3 – Porte CMOS élémentaire

Transition état courant → état suivant	Courant consommé
$0 \rightarrow 0$	I_{stat}
$0 \rightarrow 1$	$I_{cc} + I_{cl}$
$1 \rightarrow 0$	I_{cc}
$1 \rightarrow 1$	I_{stat}

Tab. 2.1 – Courant consommé en fonction des transitions

de Hamming et le modèle de poids de Hamming.

Distance de Hamming Dans ce modèle, la consommation de courant est supposée être égale au nombre de transitions des sorties. Il s'agit donc de considérer que $I_{cl} \simeq I_{cl}+I_{cc} = 1$ (c'est-à-dire que $I_{cc} \simeq 0$) et que $I_{stat} \simeq 0$. Le Tableau 2.1 devient alors le suivant :

Transition	Courant consommé	Distance de Hamming
$0 \rightarrow 0$	I_{stat}	0
$0 \rightarrow 1$	$I_{cc} + I_{cl}$	1
$1 \rightarrow 0$	I_{cc}	1
$1 \rightarrow 1$	I_{stat}	0

Tab. 2.2 – Distance de Hamming en fonction des transitions

Poids de Hamming Dans ce modèle, la consommation de courant est supposée être égale au nombre d'états hauts des sorties. Ce modèle peut se déduire des considérations exposées en 2.3.2.1 en considérant que les valeurs précédentes des sorties sont équiprobables (c'est-à-dire qu'un "1" logique a la même probabilité d'être précédée par un "1" que par un "0") et que les courants de court-circuit et de fuites sont nuls. En effet, dans ce cas, les consommations des états haut et bas sont données en moyenne par les formules suivantes :

$$moyenne(i(0 \rightarrow 1), i(1 \rightarrow 1)) = \frac{I_{cc} + I_{cl} + I_{stat}}{2} \simeq \frac{I_{cc}}{2}$$
$$moyenne(i(0 \rightarrow 0), i(1 \rightarrow 0)) = \frac{I_{stat} + I_{cc}}{2} \simeq 0$$

Le Tableau 2.1 devient alors le suivant :

Transitions	Poids de Hamming
$0 \rightarrow 0 = 1 \rightarrow 0$	0
$0 \rightarrow 1 = 1 \rightarrow 1$	1

Tab. 2.3 – Poids de Hamming en fonction des transitions

Remarques importantes Ces deux modèles, en considérant que les courant de court-circuit et de fuite des transistors sont nuls, s'affranchissent de la réalisation physique des portes logiques. Ils ne considèrent que la consommation liée à leurs sorties et à leurs variations. On ne parlera donc plus de consommation des portes mais de consommation des équipotentielles qui relient ces mêmes portes.

A noter également que la consommation d'un ensemble d'équipotentielles sera égale à la somme de la consommation de chacune de ces équipotentielles.

2.3.2.3 Mesure de courant

La chaîne d'acquisition utilisée pour acquérir les courbes de courant est illustrée par la Figure 2.4. Plusieurs étapes sont exécutées. Tout d'abord, le circuit cryptographique

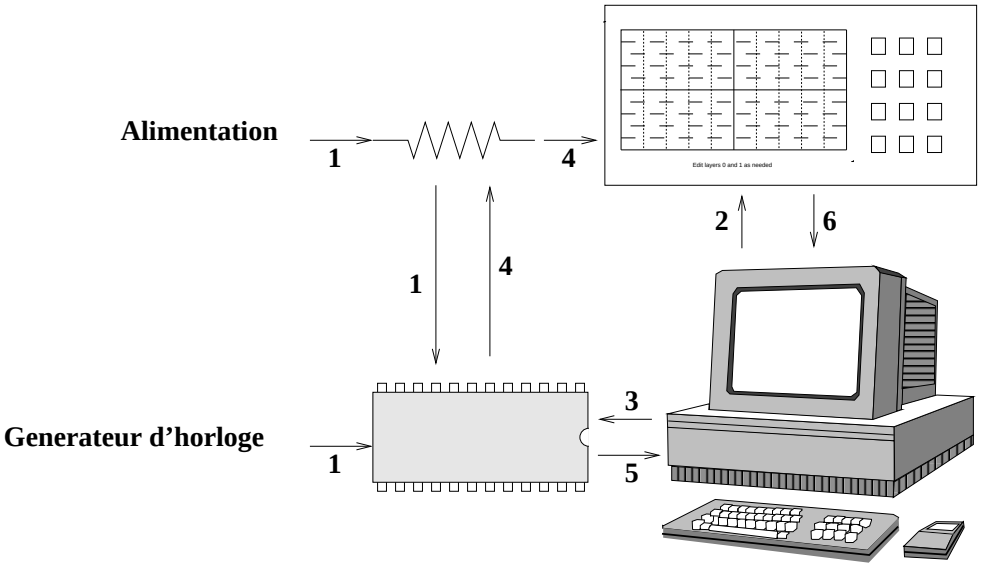

Fig. 2.4 – Schéma de mesure et d'acquisition des courbes de courant

est alimenté et le signal d'horloge est envoyé (1). Le circuit est donc opérationnel et prêt à recevoir les commandes. Ensuite, l'ordinateur configure l'oscilloscope (2) . Pendant la troisième étape, l'ordinateur envoie les commandes au circuit cryptographique, qui commence l'exécution de l'algorithme cryptographique qu'il embarque. Pendant l'exécution de cet algorithme, l'oscilloscope enregistre les courbes de courant du circuit (4). Le courant

est mesuré grâce à une faible résistance (typiquement entre 1Ω et 50Ω) insérée soit au niveau de la masse du circuit soit au niveau de sa tension d'alimentation V_{DD}. Enfin, l'ordinateur reçoit les données de sortie du composant cryptographique (5) et les courbes de l'oscilloscope(6).

La mesure de courant est une étape importante pour mener des attaques en puissance. L'étape suivante consiste à exploiter les traces de courant pour retrouver la clef secrète.

2.3.2.4 Analyse simple de consommation

Cette attaque est basée sur une simple analyse du profil de courant consommé par le circuit (d'où son nom SPA, pour "Simple Power Analysis"), d'identifier des séquences d'un algorithme ou de retrouver le poids de Hamming de certains registres (et ainsi en déduire des informations sur la clef de chiffrement). La SPA a permis d'attaquer des algorithmes cryptographiques symétriques comme l'AES [42], le RSA [52] [23], ou les courbes elliptiques. Celles implémentées grâce à l'algorithme *Double-and-add* sont particulièrement sensibles à la SPA.

Algorithme 5 Implémentation *Double-and-add* d'une courbe elliptique

Entrées: $P, d = (d_{l-1}, ..., d_0)_2$
Sorties: $Q = [d]P$

```
T ← O
pour i ← l − 1 a 0 faire
  T ← [2]T
  si d_i = 1 alors
    T ← T ⊕ P
  finsi
fin pour
retourner T
```

L'algorithme 5 montre que suivant la valeur de d_i, une addition est réalisée ou non. La Figure 2.5 représente un relevé de la consommation électrique d'un co-processeur basé sur les courbes elliptiques. Il y est clairement possible de distinguer les séquences où l'opération doublement est effectuée et celles où les opérations doublement et addition sont effectuées.

La SPA peut donc être, dans le meilleur des cas, mise en œuvre avec seule courbe de courant. Il en est tout autrement avec l'attaque présentée dans le paragraphe suivant.

2.3.2.5 Analyse différentielle du courant (DPA) et analyse du courant par corrélation (CPA)

Les attaques DPA et CPA se basent sur une corrélation entre un modèle de consommation de courant, paramétré par des bits de la clef, et une mesure de courant. Elles se déroulent en quatre étapes.

Étape 1 : Calcul de valeurs intermédiaires La première étape consiste à choisir une valeur intermédiaire de l'algorithme cryptographique. Cette valeur doit être une fonction $v(d, k)$ où d est une donnée connue par l'attaquant (d est soit le texte clair soit le texte

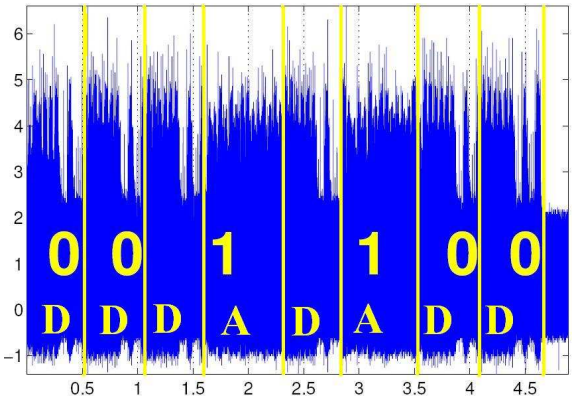

Fig. 2.5 – Courbe de courant réalisée sur une implémentation d'une courbe elliptique en $Double - and - add$

chiffré) et k un nombre restreint des bits de la clef (k est appelé *clef partielle*). Ensuite, $v(d, k)$ est calculé pour un ensemble de données D et d'hypothèses de clef K, notés respectivement $D = (d_1, d_2, ...d_{|D|})$ et $K = (k_1, k_2, ...k_{|K|})$ (avec $|X|$ désignant le cardinal de l'ensemble X). On note $v_{l,i} = v(d_l, k_i)$ la valeur intermédiaire calculée pour la donnée d_l et la clef partielle k_i.

Étape 2 : Modèle de consommation Cette étape a pour but de définir un modèle de consommation (noté $h_{l,i}$) pour les valeurs calculées dans l'étape précédente. Deux modèles de consommation sont généralement utilisés : le modèle du poids de Hamming et le modèle de la distance de Hamming.

Étape 3 : Acquisition des courbes La troisième étape consiste à mesurer la consommation de courant du composant cryptographique pendant qu'il chiffre ou déchiffre les données. Pour chaque donnée d_l, l'attaquant enregistre la trace de courant $t_{l,j}$ correspondant, avec j le pas de temps compris entre 1 et $|T|$.

Étape 4 : Calcul de corrélations La dernière étape consiste à comparer les valeurs de consommation de courant prédites pour chaque hypothèse de clef partielle aux valeurs de courant mesuré pendant la troisième phase. Le résultat de comparaison est une matrice $K \times T$ où chaque élément $r_{i,j}$ est le résultat de comparaison entre les colonnes $h_{l,i}$ et $t_{l,j}$. Deux approches différentes ont été introduites pour le calcul de comparaisons, la première a été publiée dans le papier original de Kocher et est basée sur la différence de moyennes [35]. Il s'agit de l'attaque DPA. La deuxième utilise un coefficient de corrélation [13]. Il s'agit de l'attaque CPA.

Méthode de la différence des moyennes Dans cette méthode, l'attaquant fait l'hypothèse que la consommation de courant de certaines valeurs intermédiaires est différente de la consommation de courant des autres valeurs. Ainsi, pour chaque hypothèse de

clef partielle, il calcule la différence entre la moyenne des courbes pour lesquelles $h_{l,i} = 0$ et la moyenne des courbes pour lesquelles $h_{l,i} = 1$.

$$r_{i,j} = \frac{\sum_{l=1}^{D} h_{l,i} \cdot t_{l,j}}{\sum_{l=1}^{D} h_{l,i}} - \frac{\sum_{l=1}^{D} (1 - h_{l,i}) \cdot t_{l,j}}{\sum_{l=1}^{D} (1 - h_{l,i})} \quad (2.1)$$

Attaque basée sur le coefficient de corrélation Le coefficient de corrélation est utilisé pour déterminer la relation linéaire entre les colonnes $h_{l,i}$ et $t_{l,j}$. Rappelons que le coefficient de corrélations entre deux variables est le quotient de la covariance entre ces deux variables par le produit de leurs écarts-types. Plus ce coefficient est proche de 1 ou -1, plus les variables sont fortement corrélées.

$$r_{i,j} = \frac{\sum_{l=1}^{D} (h_{l,i} - \bar{h_i}) \cdot (t_{l,j} - \bar{t_j})}{\sqrt{\sum_{l=1}^{D} (h_{l,i} - \bar{h_i})^2 \cdot \sum_{l=1}^{D} (t_{l,j} - \bar{h_j})^2}} \quad (2.2)$$

Ces deux approches partagent le même principe à savoir plus la valeur de $r_{i,j}$ est grande, plus l'estimation de consommation correspondant à l'hypothèse de clef k_i est proche de la mesure réelle. L'attaquant peut donc en déduire la clef utilisée par le composant cryptographique.

2.3.3 Les attaques électromagnétiques

L'analyse électromagnétique exploite l'information qui fuit du champ électromagnétique produit par le circuit. C'est le gouvernement américain qui a exploité les fuites électromagnétiques pendant les années 50 mais il a gardé cette observation comme secret défense. Les premiers papiers traitant de l'analyse électromagnétique pour attaquer un circuit cryptographique sont ceux de Quisquater et Samyde [57] et une équipe de Gemplus [25].

2.3.3.1 Phénomene physique

En technologie CMOS, la commutation des portes logiques est accompagnée d'une activité électrique comme nous l'avons demontré dans le paragraphe 2.3.2.1. Ce courant génère une variation dans le champ électromagnétique autour de la puce qui peut être analysé par des sondes. Selon la loi de Lentz, nous avons :

$$V = -\frac{d\phi}{dt}$$
$$\phi = \int\int \vec{B} \cdot d\vec{A}$$

Où V est la tension de sortie de la sonde, ϕ est le flux magnétique, t le temps, $\vec{B}$ le champ magnétique et $\vec{A}$ la surface pénétrée par la sonde.

2.3.3.2 Les attaques

Par analogie aux attaques par analyse de consommation, Il existe deux types d'attaques électromagnétiques : Simple ElectroMagnetic Analysis (**SEMA**) et Differential ElectroMagnetic Analysis (**DEMA**).

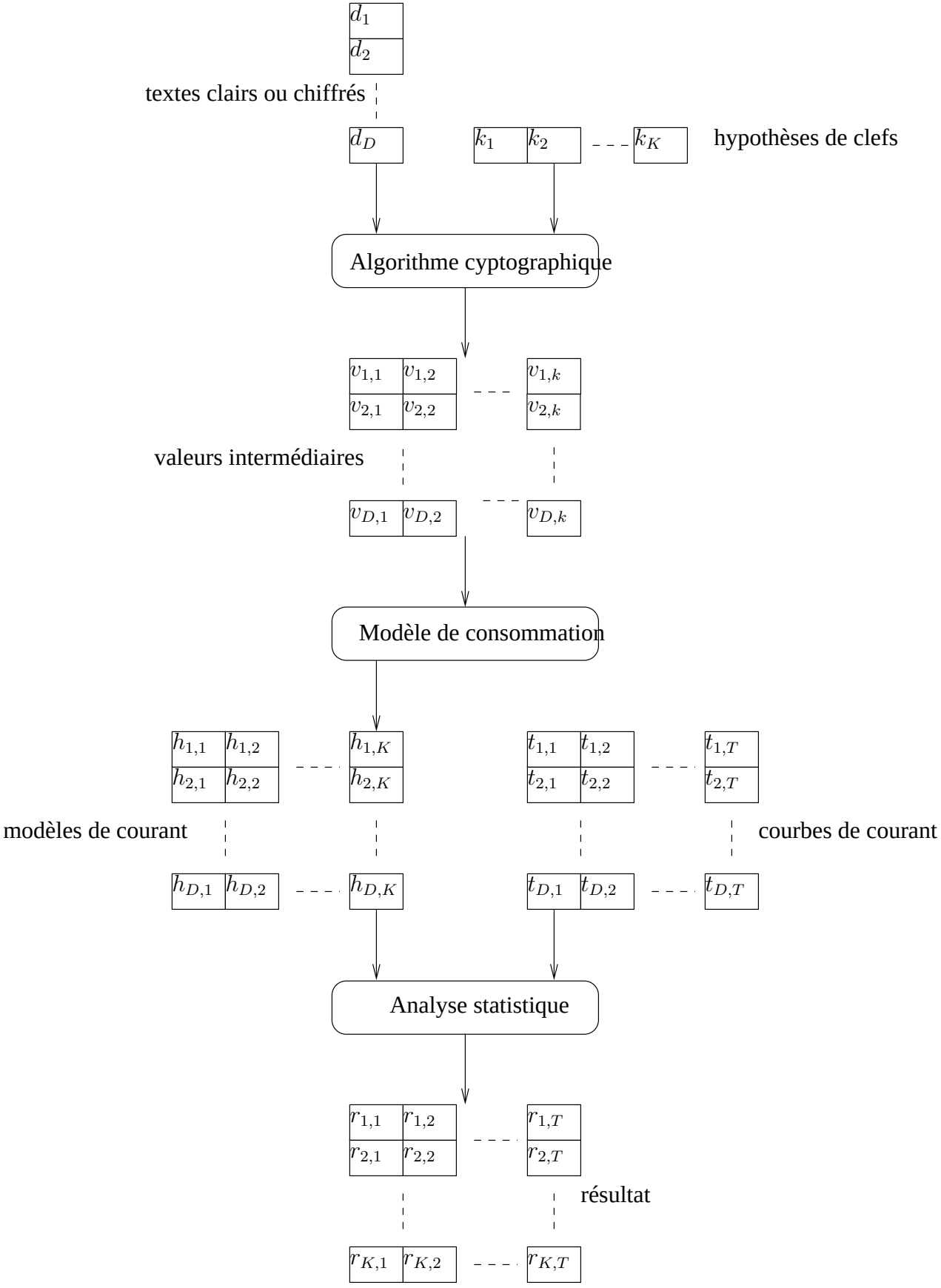

Fig. 2.6 – Déroulement de l'attaque DPA [43]

SEMA L'analyse simple du champ électromagnétique fonctionne parce que les différentes opérations d'un circuit consomment des quantités différentes de courant. Il a été démontré dans [3] que les émissions électromagnétiques de quelques instructions de la carte à puce fuient des informations sur les opérandes.

DEMA L'attaquant utilise un modèle hypothétique, pour prédire les valeurs des émissions électromagnétiques du circuit. Ces prédictions sont ensuite comparées aux mesures d'émissions électromagnétiques du circuit, en utilisant des techniques statistiques comme l'analyse de corrélation ou le test de la différence des moyennes.

2.3.4 Attaque par observation de collisions

L'attaque par observation de collisions est basée sur l'apparition de résultats intermédiaires identiques issus de chiffrement de messages différents. L'idée est de trouver une fonction h qui s'exprime en fonction de la clef k et du message m_i telle que :

$$h(k, m_0) = h(k, m_1)$$

Cette technique a été utilisée au début pour briser les fonctions de hachage. Récemment, le principe de collisions a été associé aux analyses des canaux cachés pour dévoiler des vulnérabilité dans le DES [63] et l'AES [62].

Attaque en collision sur DES Dans les boîtes de substitution du DES, chaque valeur de sortie peut être sélectionnée par 4 valeurs différentes d'entrées :

$$S_i(z) = S_i(z \oplus \delta_1) = S_i(z \oplus \delta_2) = S_i(z \oplus \delta_3), \delta_1 \neq \delta_2 \neq \delta_3 \neq 0 \qquad i \in \{1, ..., 8\}$$

Par exemple, pour la première S-Box on peut écrire :

$$\begin{aligned} S1(000000) &= S1(000000 \oplus 001001) \\ &= S1(000000 \oplus 100100) \\ &= S1(000000 \oplus 110111) = 14 \end{aligned}$$

Une table est générée pour chaque S-Box qui liste les trois différentielles δ_1, δ_2 et δ_3 qui correspondent aux 64 valeurs possibles de z. Cette table est appelée table δ.
Dans [63], une collision est crée à la sortie de 3 S-Box adjacentes, pendant le premier tour. En effet, en raison de la fonction `Expansion`, les 2 bits de poids fort et les 2 bits de poids faible de l'entrée d'une S-Box font également partie des entrées des deux S-Box voisines. Pour réaliser cette collision, des masques sont utilisés pour les registres L_0 et R_0 afin de ne faire varier que les bits nécessaires à la création d'une collision. Celle-ci est détectée au deuxième tour en observant les courbes de courant. Si une collision est détectée, l'analyse des tables δ permet de révèler les candidats possibles pour la clef.

2.3.5 Attaques par caractérisation de bruit ou "Template Attack"

L'attaque en "Template" [16] présente une approche différente de celles présentées jusqu'à maintenant. Le but de cette attaque est d'extraire une information corrélée à

une quantité secrète à partir des échantillons d'un canal caché. Mais contrairement aux attaques par analyse de consommation, l'attaque template cherche à exploiter les caractéristiques du bruit comme révélateur d'information. Toutefois, afin de caractériser le bruit, cette attaque nécessite un accès à un composant cryptographique tel que celui qu'un attaquant souhaite attaquer, afin de modifier ses paramètres comme la clef secrète.
Afin de réaliser cette attaque, l'attaquant enregistre plusieurs courbes de courants $T_1, T_2.....T_N$ pour chaque hypothèse de clef k_i . Il forme, par la suite, un "template" formé d'un couple d'éléments : le premier est la moyenne des courbes de courants M_i, le deuxième est un vecteur de bruit à N-dimensions, qui est une matrice de covariance entre tous les éléments de tous les vecteurs de bruit des différentes courbes T_i : $\sum_{N_i}$.

$$\sum_{N_i} \quad = \quad cov(N_i(P_u), N_i(P_v))$$

avec P est un point de $T - M_i$.
Ensuite, il enregistre la courbe de courant S qui correspond au composant attaqué. Pour retrouver k_i, il calcule le bruit de S comme si le composant utilisait la clef k_i

$$\mathbf{n} \quad = \quad M_i - S$$

Ensuite, il calcule la probabilité d'observer le vecteur $\mathbf{n}$ en utilisant la distribution Gaussienne multivariable

$$p_{N_i}(\mathbf{n}) \quad = \quad \frac{1}{\sqrt{(2\pi)^N |\sum_{N_i}|}} e^{(-1/2\mathbf{n}^T \sum_{N_i}^{-1} \mathbf{n})}$$

.

Si le bruit présente réellement une forme Gaussienne, alors la clef utilisée est celle qui présente la probabilité maximale.

2.3.6 Attaque par analyse différentielle de comportement

Il s'agit d'une attaque hybride qui exploite les propriétés de l'attaque en *Safe Error* et les traitements statistiques utilisés dans la DPA. Ainsi, l'attaque DBA (Differential Behavioral analysis) [60] corrèle un modèle algorithmique, paramétré par des bits de la clef, à un comportement du circuit en présence de fautes.

2.3.6.1 Hypothèses

Afin de mener une attaque DBA sur un circuit cryptographique, un attaquant doit connaître l'algorithme cryptographique embarqué dans le circuit et doit pouvoir exécuter cet algorithme plusieurs fois, avec et sans perturbation. Les perturbations présentent les caractéristiques suivantes :

- Type : la faute est de type *collage à 0*, *collage à 1*, *set* ou *reset*. L'attaquant n'est pas obligé de connaître la valeur de la faute.
- Localisation : la faute doit affecter des bits correspondant à des valeurs intermédiaires particulières.
- Focalisation : la faute doit affecter un nombre réduit de bits (typiquement inférieur à 8).

– Valeur : la valeur de la faute injectée doit être la même pour les bits visés par l'attaque.
– Répétitivité : l'attaquant doit injecter la même faute, au même moment, sur les même bits pour l'ensemble des exécutions.

Enfin, l'attaquant doit distinguer un comportement normal d'un comportement anormal du circuit attaqué. Ceci peut être détecté par le déclenchement d'un signal d'alarme ou encore l'arrêt prématuré d'un calcul.
En fonction du nombre de bits attaqués, nous distinguons deux types d'attaques DBA : la DBA mono-bit qui vise un seul bit et la DBA multi-bit qui vise plusieurs bits. A titre d'exemple, nous allons expliquer la DBA mono-bit.

2.3.6.2 Déroulement de la DBA mono-bit

Soit un circuit qui chiffre un ensemble T de textes clairs connus de l'attaquant à l'aide d'une clef K_0 inconnue mais dont l'attaquant souhaite connaître la valeur. La DBA est réalisée en quatre étapes :

1. L'attaquant choisit un ensemble R de bits manipulés par l'algorithme cryptographique. Chacun de ces bits est fonction du texte clair t_i (avec $t_i \in T$) et d'un nombre restreint de bits de la clef, k_p. L'ensemble des valeurs possibles pour k_p est noté K et la fonction qui renvoie la valeur du bit intermédiaire en fonction de k_p et de t_i est notée $r(k_p, t_i)$.
2. Pour l'ensemble des textes clairs, l'attaquant effectue deux manipulations. Dans un premier temps, il chiffre un texte clair t_i et regarde le comportement du système durant la première étape de l'algorithme cryptographique. Il note ce comportement $C_0(t_i)$. Ensuite, il chiffre le texte clair t_i en perturbant le circuit et en enregistrant le comportement $C_1(ti)$ de ce dernier durant la première étape du déroulement de l' algorithme cryptographique. Cette étape permet à l'attaquant d' associer à chaque texte clair t_i, une valeur $c(K_0, t_i)$ qui vaut 1 si $C_0(t_i) = C_1(t_i)$ et 0 sinon.
3. Pour chaque texte clair t_i, l'attaquant calcule $r(k_p, t_i)$ pour tout k_p appartenant à l'ensemble K. Pour chacune de ces sous-clefs, il scinde alors les valeurs théoriques $r(k_p, t_i)$ en deux ensembles $\{r(k_p, t_i)|c(K_0, t_i) = 0\}$ et $\{r(k_p, t_i)|c(K_0, t_i) = 1\}$. Il calcule enfin la différence des moyennes de ces deux ensembles à l'aide de l'équation suivante :

$$\Delta_{K_0}^{k_p}(T, r) = \frac{\sum_{t_i \in T}(r(k_p, t_i) \cdot c(K_0, t_i))}{\sum_{t_i \in T} c(K_0, t_i)} - \frac{\sum_{t_i \in T}(r(k_p, t_i) \cdot (1 - c(K_0, t_i)))}{\sum_{t_i \in T}(1 - c(K_0, t_i))}$$

La DBA a été appliquée avec succès sur une implémentation matérielle d'un AES synchrone [60] et d'un DES asynchrone [61].

2.3.7 Attaques en probing

L'attaque en probing proposée dans ce chapitre, s'inspire, comme pour l'attaque DBA, de l'attaque DPA. En effet, l'attaque en probing cherche à corréler un modèle, fonction d'une clef partielle, à une mesure physique. Afin de mener à bien une attaque en probing, 3 étapes sont à suivre : modélisation, expérimentations et enfin corrélation.

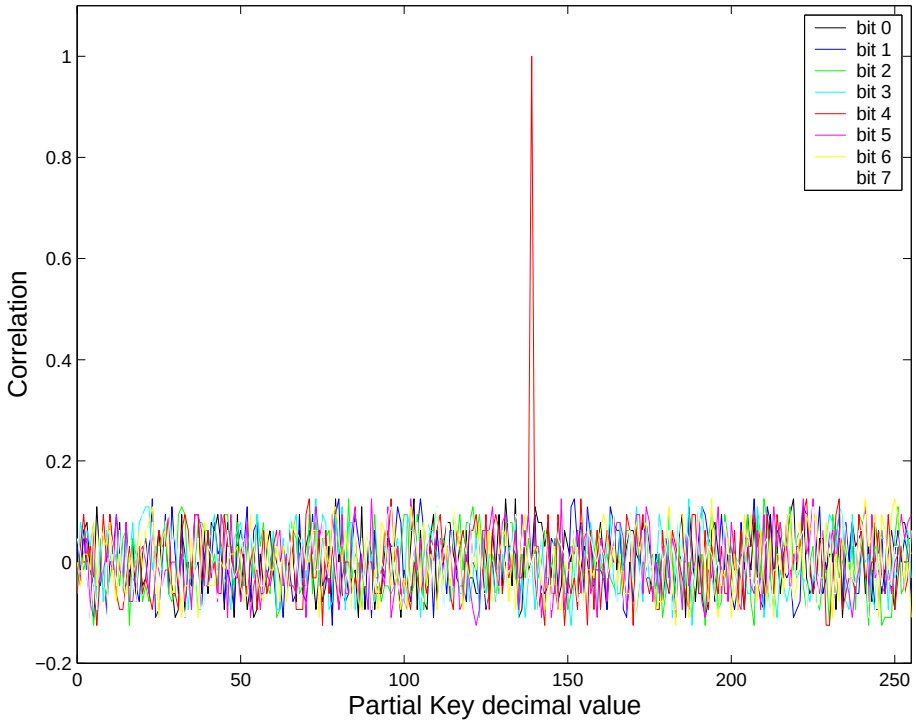

Fig. 2.7 – Courbes DBA mono-bit réalisée sur un AES-128

1. Modélisation : cette étape consiste à définir un ensemble de bits d'attaques R où chaque élément r s'exprime en fonction d'une clef partielle k_p et d'un texte clair $t_k \in T$: $r(k_p, t_k)$.
2. Expérimentations : doté de pointes, un attaquant doit récupérer les valeurs électriques de tous les signaux du circuit attaqué, pour l'ensemble des textes clairs T. Pour ce faire, une cartographie est réalisée sur le composant. Ainsi, pour chaque position (x_i, y_j), et à chaque instant de calcul $b \in B$, les signaux $s((x_i, y_j) = p, b, t_k)$ sont stockés. Une numérisation de ces signaux permet d'avoir leurs valeurs binaires.
3. Corrélation : la corrélation entre les vacteurs $r(k_p, t_k)$ et $s(p, b, t_k)$ est calculée pour l'ensemble des textes clairs T, pour chaque clef partielle k_p, chaque position p et à chaque instant b. Le maximum de corrélation permet à la fois de retrouver la clef partielle mais aussi la position et l'instant de manipulation du bit d'attaque.

2.3.8 Classe des attaques par corrélation

Parmi les attaques par canaux cachés que nous venons de décrire, certaines partagent le principe suivant : l'attaquant choisit un nombre restreint de bits de la clef (appelé clef partielle) et crée un modèle, paramétré par la valeur de cette clef et du texte clair ou chiffré. Ensuite, il enregistre des courbes de courant ou d'émissions électromagnétiques d'un signal global ou d'un signal local, ou encore des courbes de comportement en présence de fautes, du composant cryptographique pendant le chiffrement de textes clairs ou le déchiffrement de textes chiffrés. Il s'avère que le modèle qui présente un maximum de corrélation avec les mesures correspond à la clef utilisée par le composant. Les attaques

DPA, CPA, DEMA, DBA et les attaques en probing vérifient ce principe.
Dans la suite de ce manuscrit, nous allons nous intéresser uniquement à cette classe particulière d'attaque que nous nommerons "attaques par corrélation".

2.3.9 Information exploitée par l'attaquant

Les attaques par corrélation extraient une donnée secrète, la clef, grâce à une comparaison entre des mesures (de consommation instantanée, de rayonnement électromagnétique, du résultat en présence de fautes, etc..) et un modèle qui dépend de la valeur d'un résultat intermédiaire (lui-même fonction d'une hypothèse de clef) : Plus la comparaison sera probante, plus l'hypothèse de clef sera probable. Les mesures réalisées contiennent :

- Une composante "informative" : Il s'agit de la trace générée par la valeur intermédiaire qui est prédite par l'attaquant (ie., prise en compte dans le modèle). Il s'agit typiquement du signal émis par le résultat d'un calcul en sortie de S-Box.
- Une composante "parasite" : Il s'agit d'une trace qui est fonction de la valeur intermédiaire mais qui n'est pas prédite par l'attaquant (ie., non prise en compte dans le modèle). Il s'agit typiquement de la trace générée par les bits en sortie d'une S-Box du DES connexe à celle qui est attaquée. Les effets de cette composante induisent parfois ce qui est appelé dans la littérature des "pics fantômes" (ou "ghost peaks").
- Du "bruit", généré par les traces indépendantes des données pouvant être prédites par l'attaquant.

Les attaquants cherchent à maximiser le rapport signal à bruit de la composante informative (ou "signal to noise ratio", SNR). Une approche proposée à cet effet consiste à focaliser les mesures sur la zone du circuit supposée manipuler les résultats intermédiaires (bus de données, cryptoprocesseur, etc...). De telles mesures peuvent être réalisées à l'aide de sondes EM. Dans ce cas, la valeur du SNR dépend, entre autres, de la taille des sondes choisies (une faible taille réduit le bruit mais également l'amplitude du signal capté). Les résultats de calculs en présence de faute peuvent également être considérés comme une mesure locale d'un signal informatif avec un niveau de bruit qui dépend de la reproductibilité des fautes injectées. Dans le cas des attaques en safe error [73, 60], par exemple , un résultat fauté ou juste permet d'associer indirectement une valeur à l'un des bits de la clef. La technique de mesure présentant le meilleur SNR est celle du micro-sondage. Le bruit, dans ce cas, est lié à la qualité du contact réalisé et peut, de toute façon, être aisément supprimé. La contrepartie de ces 3 techniques est que la probabité de mesurer le signal informatif diminue avec la focalisation, sauf si l'attaquant procède préalablement à une rétro-ingénieurie du composant. A titre d'illustration, le signal de consommation instantanée, s'il fourni un SNR a priori faible, contient dans tous les cas un signal informatif. Au contraire, un micro-sondage possède un SNR très élevé, à condition que le signal sondé véhicule une information intéressante. Or cette probabilité, qui, dans le cas d'un sondage aveugle, est l'inverse du nombre de connexions qui existent dans un circuit, est très faible. Une autre approche pour maximiser le SNR consiste à réduire le bruit en appliquant des techniques de traitement du signal, qui vont de la simple moyenne à l'utilisation d'outils statistiques évolués comme les cumulants d'ordre élevés [37].

Cette discussion montre que l'attaquant doit réaliser un compromis entre l'amélioration du SNR et la robustesse de la mesure. Il s'avère donc que l'attaquant le plus fort est

celui qui mesure un SNR important avec une grande probabilité que le signal mesuré soit informatif. En d'autres termes, le micro-sondage sur un grand nombre de signaux ou sur un nombre restreint de fils après rétro-conception est le plus efficace.

2.4 Conclusion

Dans ce chapitre, les attaques physiques qui menacent la sécurité des composants cryptographiques ont été présentées ; les attaques par injection de fautes utilisent des impulsions sur la tension d'alimentation ou le signal d'horloge, ou encore une lumière puissante focalisée. Ces fautes sont par la suite exploitées de différentes manières afin de remonter à la clef : observation du comportement du circuit en présence de fautes, détection de collisions créées artificiellement ou encore étude de différences entre les résultats fautés et non fautés. Les attaques par canaux cachés exploitent quant à elles la dépendance entre les données manipulées par l'algorithme cryptographique et certaines grandeurs physiques comme la consommation de courant, les émissions électromagnétiques, le temps d'exécution et le bruit. Parmi ces attaques physiques, les attaques DPA, CPA, DEMA et la DBA forment la classe des attaques, appelées attaques par corrélation. Une nouvelle attaque par micro-sondage, inspirée par ces attaques par corrélation a également été proposée.

La connaissance de ces menaces est indispensable pour tout concepteur de circuit intégré souhaitant sécuriser son circuit. Cette sécurisation passe par le développement de parades ou contre-mesures mais également par la validation de ces dernière. Leur évaluation lors de la phase de conception fait l'objet du chapitre suivant.

Chapitre 3

Conception des circuits intégrés sécurisés

3.1 Introduction

La conception d'un circuit intégré consiste à rechercher un modèle vérifiant un ensemble de spécifications généralement exprimées en terme de fonction, de performance, de consommation et de coût. Dans le cas d'un circuit cryptographique, les spécifications doivent également intégrer des contraintes sécuritaires généralement exprimées en termes de "résistance à telle ou telle attaque". Or cette "résistance", si elle peut être quantifiée lors de la certification du circuit final selon des normes désignées sous le nom de "critères communs", est beaucoup plus difficile à évaluer *a priori*, c'est-à-dire lors de la conception du circuit. Toute vulnérabilité non détectée lors de cette phase entraîne alors un risque financier important pour l'entreprise ainsi qu'un surcoût significatif (temps de développement, en certification, retard de mise sur le marché).
Il s'avère donc essentiel pour les concepteurs de circuits sécurisés de disposer d'un flot de conception permettant de valider aussi tôt que possible l'efficacité de leurs contre-mesures. Ce chapitre précise cette problématique en présentant, dans un premier temps, le processus classique de conception et les difficultés rencontrées pour y intégrer des contraintes sécuritaires. Ensuite les principales contre-mesures présentées dans la littérature seront rappelées. Un état de l'art des outils évaluant leur efficacité sera proposé et les spécificités de l'approche adoptée dans cette thèse décrites.

3.2 Flot de conception des circuits intégrés

La recherche de modèle vérifiant un ensemble de spécifications est réalisée séquentiellement dans une hiérarchie d'abstractions : Dans un premier temps, le concepteur traduit la description très haut niveau (généralement formulée en langage naturel) du circuit en un langage de description matériel de haut niveau. A ce niveau d'abstraction (appelé RTL pour "Register Transfert Level"), le circuit est constitué de blocs combinatoires et d'éléments de mémorisation. Le concepteur raffine ensuite ce modèle à l'aide d'outils de synthèse logique pour obtenir une description niveau "portes" du circuit, lequel est alors considéré comme une interconnexion de portes réalisant des fonctions logiques élémentaires

(comme des ET, des OU, des XOR, etc...). En substituant ces portes par leur description niveau “transistor” (préalablement conçues dans les circuits à cellules pré-caractérisées), le concepteur obtient la description niveau “transistor” du circuit. La dernière étape consiste à positionner sur le silicium ces transistors lors de la phase de placement-routage pour obtenir le modèle physique du composant, vu alors comme un agencement de polygones aux propriétés physiques distinctes.

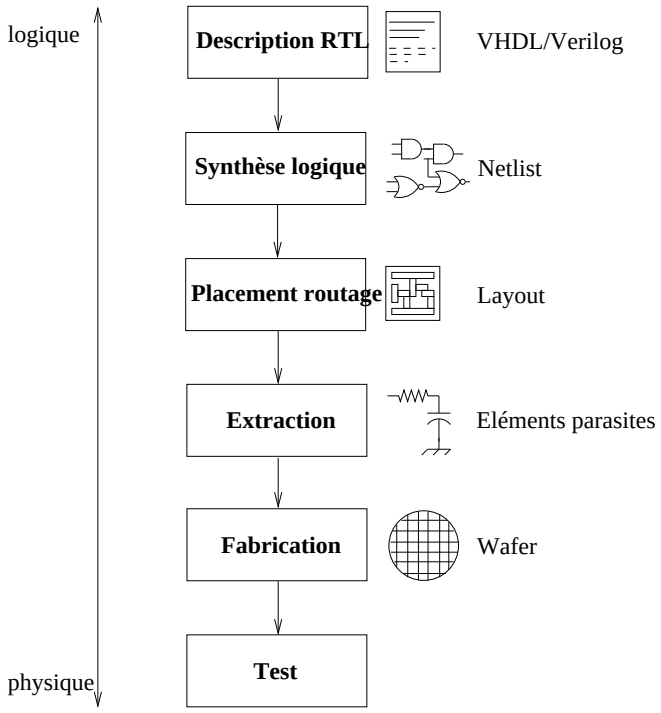

Fig. 3.1 – Flot de conception des circuits intégrés sécurisés

L’intérêt d’une telle démarche, basée sur l’utilisation de différents niveaux d’abstraction (Figure 3.2), est qu’elle permet de focaliser très rapidement la recherche vers le(s) circuit(s) vérifiant les spécifications (en élaguant dès que possible les solutions les moins prometteuses). L’inconvénient est qu’il existe autant de langages et de simulateurs que de niveaux d’abstraction. Chacune des spécifications, fatalement exprimées pour le circuit réel, doit également être estimée à chaque niveau, sachant que ces estimations sont d’autant moins précises que le modèle est abstrait et qu’elles sont d’autant difficiles à calculer que le modèle est concret.

La deuxième étape de conception consiste à fabriquer le circuit. Comme le process de fabrication peut introduire des différences entre le fonctionnement du circuit réel et son modèle, le concepteur doit vérifier pendant cette dernière étape, appelée étape de caractérisation, que le circuit vérifie bien les spécifications. Mais avant d’être commercialisé, un composant cryptographique doit en plus être évalué par un laboratoire agrée par la **DCSSI** (**D**irection **C**entrale de la **S**écurité des **S**ystèmes d’**I**nformation). Ces laboratoires, appellés **CESTI** (**C**entre **É**valuation de la **S**écurité des **T**echnologies de l’**I**nformation), réalisent l’évaluation et la certification de composants électroniques et de logiciels embar-

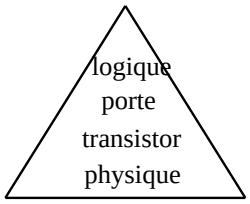

Fig. 3.2 – Niveaux d'abstraction

qués, selon des Critères Communs définis dans la norme **ISO 15408**.

Les contre-mesures peuvent être insérées à toute étape du flot de conception, avec comme objectif de n'impacter qu'aussi faiblement que possible les performances et le coût (lié à la surface) du circuit. Les principales contre-mesures proposées dans la littérature pour parer aux attaques par canaux cachés et aux attaques par injection de fautes sont présentées ci-après.

3.3 Contre-mesures

3.3.1 Hypothèses fondamentales

Les principales attaques décrites dans le chapitre précédent sont basées sur les hypothèses fondamentales suivantes :

- Hypothèse 1 : Il existe une corrélation entre la valeur d'un opérateur, d'une opérande ou d'une suite d'opérations et au moins une grandeur physique pouvant être mesurée par l'attaquant. Cette grandeur, appelée syndrome, peut être la valeur elle-même (dans le cas de micro-sondage), le temps de calcul, le courant consommé, le rayonnement électromagnétique, etc...
- Hypothèse 2 : Il existe une corrélation entre les relations statistiques entre les opérandes et au moins une grandeur physique pouvant être mesurée par l'attaquant.
- Hypothèse 3 : Il est possible de modifier la valeur d'un opérateur, d'une opérande ou d'une suite d'opérations, cette modification étant réalisée par une action sur l'environnement dans lequel le circuit est inséré.
- Hypothèse 4 : Il est possible de détecter la modification d'un opérateur, d'une opérande ou d'une suite d'opérations.
- Hypothèse 5 : Durant les calculs internes à l'algorithme de cryptographie, des sous-parties de la clef (ou *clefs partielles*) sont utilisées de façon indépendantes.
- Hypothèse 6 : L'algorithme de cryptographie est itératif et sa "force" tient au nombre élevé d'itérations réalisées (typiquement de 10 à 16).
- Hypothèse 7 : Il est possible de prédire la valeur de résultats intermédiaires à partir de tout ou partie des données d'entrée/sortie (c'est-à-dire clairs, chiffrés ou clefs).
- Hypothèse 8 : Il est possible de prédire des relations statistiques entre les résultats intermédiaires à partir de tout ou partie des données d'entrée/sortie.
- Hypothèse 9 : Il est possible de prédire les valeurs de résultats intermédiaires (via éventuellement la différence des résultats intermédiaires) à partir de la différence

entre des données d'entrée/sortie.

L'attaque SPA est basée sur l'hypothèse 1. L'attaque DPA d'ordre 1 est basée sur les hypothèses 1, 5 et 7. Les DPAs d'ordre supérieures sont basées sur les hypothèses 2, 5 et 8. Les DFAs sont basées sur les hypothèses 3, 5 et 9. Les attaques en safe-error sont basées sur les hypothèses 3, 4 (+ 5 et 7 pour la DBA). Les attaques en faute par réduction de ronde sont basées sur les hypothèses 3 et 6.
A noter que les 4 premières concernent l'implémentation matérielle du composant alors que les 5 dernières concernent les algorithmes de cryptographie.

3.3.2 Principes de contre-mesures

Les contre-mesures proposées dans la littérature consistent à invalider les hypothèses fondamentales décrites ci-dessus. A noter qu'il est impossible de rendre caduque l'hypothèse 5 car il s'agirait de manipuler la clef dans sa globalité et non par morceaux. Dans le cas de l'AES, il s'agirait d'utiliser une boîte de substitution de 128 bits par 128 bits. Implantée dans une mémoire, celle-ci devrait être constituée de $128 * 2^{128} = 2^{136}$ bits. Implantée en matériel grâce à une architecture ET/OU, elle devrait être constituée de 2^{128} portes ET à 128 entrées et 128 OU à 2^{128} entrées. Si on considère que le concepteur n'a, à sa disposition, que des portes à deux entrées, il lui faudrait $m = (128-1) * 2^{128} + 128 * (2^{128} - 1) \sim 2^{137}$ portes logiques pour l'AES. Si l'on considère une intégration de 10^{10} portes au cm^2, une boîte de substitution couvrirait la surface de $10^{27} km^2$ soit environ $2 * 10^{18}$ surfaces de la Terre (laquelle est égale à $500 * 10^6 km^2$).

Les principes fondamentaux sur lesquelles sont basées les contre-mesures sont les suivants :

- Principe 1 : Réduire la corrélation entre les données manipulées et les syndromes ("hiding" ou "dissimulation") (pour contrer l'hypothèse 1).
- Principe 2 : Rendre non prédictible la valeur des variables intermédiaires ("masking" ou "masquage") par des opérateurs connus (pour contrer l'hypothèse 7).
- Principe 3 : Rendre non prédictible l'ordonnancement des opérations faites sur les données ("secrecy") (pour contrer l'hypothèse 7, 8 et 9).
- Principe 4 : Faire en sorte que le circuit, en cas de perturbation, ne communique plus d'information exploitable par l'attaquant comme, par exemple, le résultat d'un calcul cryptographique erroné (pour contrer l'hypothèse 9).
- Principe 5 : Faire en sorte que le circuit fonctionne normalement, indépendamment de l'environnement dans lequel il est inséré, mais avec émission d'un syndrome fonction des données sensibles (pour contrer l'hypothèse 3 mais pas la 4).
- Principe 6 : Faire en sorte que le circuit fonctionne normalement, indépendamment de l'environnement dans lequel il est inséré, sans émission d'un syndrome fonction des données sensibles (pour contrer l'hypothèse 3 et 4).

Théoriquement, il suffit de rendre caduque une seule hypothèse sur laquelle est basée l'attaque pour rendre impossible cette dernière. En pratique, la sécurité du composant sera basée sur la juxtaposition de plusieurs techniques. Les principales sont détaillées ci-après.

3.3.3 Principe 1 : Réduire la corrélation entre les données manipulées et les syndromes

De nombreuses techniques ont été proposées dans le but de réduire la corrélation entre les données manipulées et les syndromes, c'est-à-dire à réduire le rapport signal à bruit mesuré par l'attaquant décrit dans le paragraphe 2.3.9. Elles sont généralement classées en trois catégories : La première consiste à réduire localement l'amplitude du syndrome (on parle d'"équilibrage"), la seconde à augmenter le bruit lors de la mesure (on parle de "randomisation") et la dernière à réduire globalement l'amplitude du syndrome (on parle de "filtrage").

3.3.3.1 Équilibrage

La notion d'équilibrage de la consommation est une notion suffisamment floue dans la littérature pour qu'il soit nécessaire de la préciser : Équilibrer une grandeur (et non équilibrer "tout court") par rapport à des données manipulées par le circuit consiste à rendre cette grandeur indépendante de ces données. On parle aussi de réduction de dissymétrie ou de signature. Diverses techniques ont été avancées en fonction de la grandeur choisie. Les principales techniques rencontrées dans la littérature sont exposées ci-après :

Équilibrer l'algorithme Il s'agit de faire en sorte que les signaux de contrôle d'un multiplexeur ou d'un démultiplexeur ne soit pas fonction d'un bit de la clef. Dans le cas d'une implémentation logicielle, il s'agit de faire en sorte que le résultat d'un test conditionnel ne soit pas fonction d'une valeur sensible. Si tel était le cas, il faut que les branches de l'algorithme ou les chemins de données soient aussi identiques que possible (même ordonnancement des opérations dans les branches, même combinatoires et mémorisation dans les chemins de données).

Équilibrer les poids de Hamming A cet effet, le codage *1-n* est généralement employé. Dans un tel codage, un rail est constitué de n fils dont un et un seul est actif, d'où un poids de Hamming constant et égal à 1 quelle que soit la donnée. Pour n=2, on parle de double rail : un bit d'information est codé à l'aide de deux fils x_t et x_f. Par convention, le "FAUX" logique est codé par ($x_t = 0$, $x_f = 1$) et le "VRAI" logique par ($x_t = 1$, $x_f = 0$). On constate sur la Figure 3.3 qu'avec ce type de codage, deux transitions logiques distinctes ($1 \rightarrow 0$ puis $0 \rightarrow 1$) génèrent chacune deux transitions électriques, l'une montante et l'autre descendante.

Équilibrer les réseaux de portes Le circuit est scindé en deux : sa partie régulière C_t et sa partie duale C_f. Si f_t désigne une fonction logique du circuit régulier ayant comme entrée x_t et comme sortie s_t, la fonction duale f_f vérifiera la propriété suivante $s_f = f_f(x_f) = not(f_t(x_t)) = not(f_t(not(x_f)))$ si $x_f = not(x_t)$. Les deux fonctions devront être réalisées à l'aide de réseaux "superposables" c'est-à-dire que les graphes par composant-connexion associés doivent être isomorphes. La partie gauche de la Figure 3.4 représente en vue "porte" la fonction logique NAND double-rail : sa partie régulière est constituée d'un NAND et sa partie duale d'un NOR. Ces parties sont superposables (l'une est le

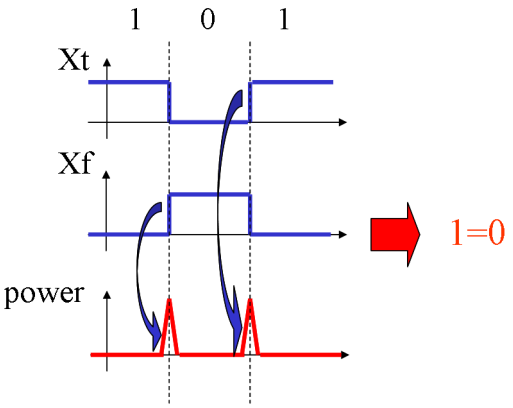

Fig. 3.3 – Codage double rail

miroir de l'autre) lorsqu'elles sont représentées aux niveaux "transistor" comme indiqué sur la partie droite de la Figure 3.4.

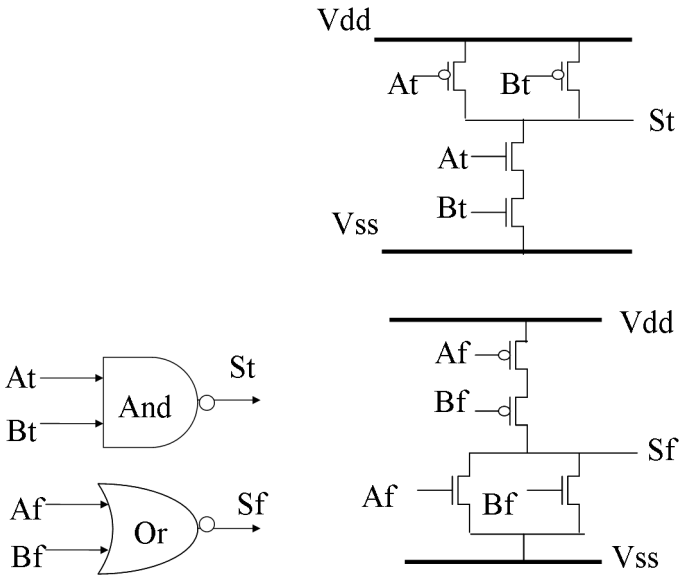

Fig. 3.4 – Porte NAND double rail (vue porte, à gauche et transistor, à droite)

Équilibrer les distances de Hamming A cet effet, une phase de calcul intermédiaire de retour vers une valeur constante est alors insérée (on parle de protocole avec réinitialisation , à "zéro" ou à "un"). Il s'agit généralement de l'envoi de la valeur $x_t = x_f = 0$. La fonction f_f doit alors vérifier en plus de la relation ci-dessus $s_f = s_t = f_f(x_f) = f_t(x_t) = 0$ si $x_f = x_t = 0$.

Équilibrer les réseaux de transistors dans les portes logiques Cette tâche étant physiquement irréalisable, des approches “au premier ordre” ont été tentées. Il s’agit par exemple de faire en sorte que le nombre de transistors qui commutent ou que le nombre de condensateurs qui se chargent ou se déchargent soient identiques quelque soient les transitions possibles sur les entrées. Cette contrainte conduit à la conception de librairie “full custom” comme Seclib [29].

Équilibrer les interconnexions entre portes logiques Il s’agit de placer les portes logiques et router les pistes qui les relient en mode différentiel (c’est-à-dire qu’une piste du réseau régulier aura la même empreinte sur le circuit que la piste duale associée). Les méthodes de fat-wire [69] et de back-end duplication [28] ont été inventées dans cet objectif. Cette dernière méthode consiste à positionner les circuits duaux de telle sorte que les pistes d’alimentation soient, modulo un léger décalage horizontal, leur axe de symétrie comme indiqué sur la partie gauche de la Figure 3.5, puis de relier les entrées/sorties de ces circuits duaux à l’aide de paires différentielles comme indiqué sur la partie droite de cette même figure.

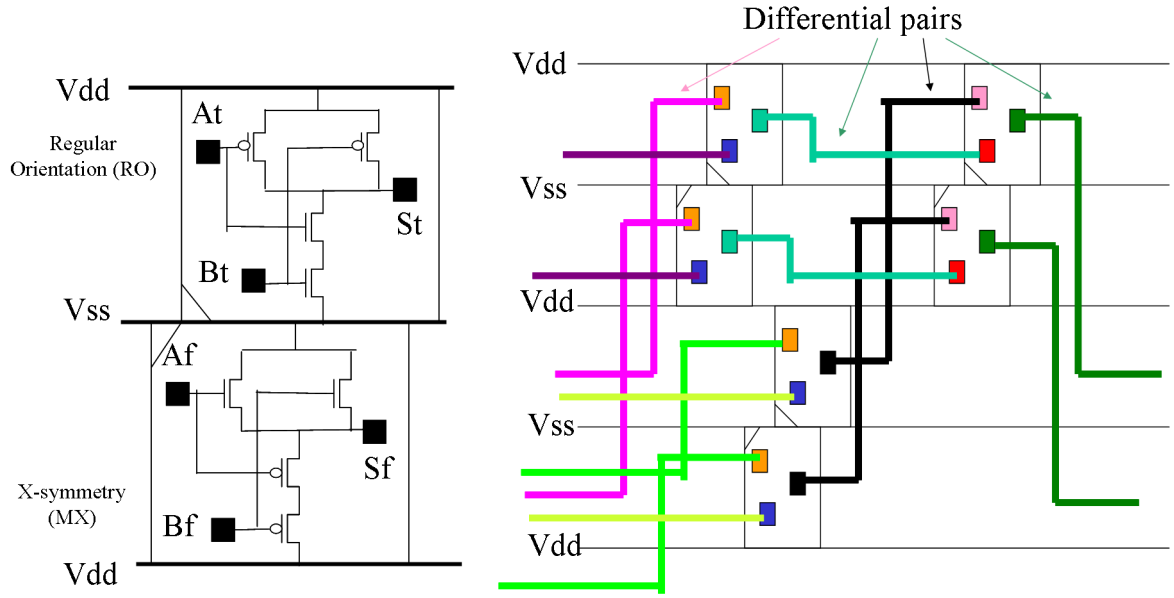

Fig. 3.5 – Routage différentiel

Annuler l’effet mémoire En fonction des états précédents, une même transition peut ne pas charger/décharger le même ensemble de condensateurs. D’où une différence de consommation pour une même transition qui peut être différente. Une des approches proposées pour supprimer l’effet mémoire consiste à décharger toutes les équipotentielles d’une porte à chaque réinitialisation. Cette approche nécessite une conception de cellules “full custom”.

Supprimer l'évaluation anticipée De nombreuses fonctions booléennes sont implantées de telle sorte que la sortie puisse être calculée avant que toutes les entrées soient stabilisées. La sortie d'un "OU" peut, par exemple, être fixée à VRAI dès qu'au moins l'une de ses entrées est elle-même à VRAI. Ces évaluations créent des différences de propagation des signaux dans le circuit même si toutes les portes logiques et rails associés sont parfaitement équilibrés. Pour éviter ce genre de phénomènes, il faut faire en sorte que la sortie soit fonction de toutes les entrées en modifiant la conception des portes logiques ou en synchronisant les données en entrée à l'aide d'arbres de Muller.

Logiques équilibrées Certaines logiques mettent en œuvre certaines techniques décrites ci-dessus. Par exemple, certaines implantations de circuits en logique asynchrone dite "QDI" [51] équilibrent naturellement les poids et les distances de Hamming. Avec une conception *ad hoc*, il est également possible d'équilibrer les réseaux de portes. La logique WDDL [70] équilibre les poids et les distances de Hamming.

3.3.3.2 Randomisation

Il s'agit soit d'ajouter du bruit sur l'instant d'exécution d'instructions sensibles (on parle alors de désynchronisation) soit sur l'amplitude. Les techniques proposées dans la littérature sont les suivantes :

Désynchronisation par ajout de jigue sur l'horloge Il s'agit de modifier aléatoirement la fréquence d'horloge du circuit. L'amplitude de la modification (ou jigue) doit être suffisamment importante pour "étaler" les calculs sensibles mais tout de même assez faible pour que les calculs soient réalisés correctement (c'est-à-dire que les temps de setup et de hold des bascules soient garantis).

Ajout d'instructions factices Il s'agit d'exécuter des opérations qui n'ont pas de fonction autre que de créer une surconsommation lors ou entre les étapes d'un calcul sensible ("dummy operations"). L'exécution des opérations peut être fixe (pour rendre plus difficile les attaques simples) ou aléatoirement distribuée à chaque calcul sensible (contre SPA et DPA). Elle peut être réalisée en parallèle (par exemple, calcul du processeur pendant un calcul cryptographique sur le coprocesseur) ou insérée temporellement (comme représenté en Figure 3.6) pour les calculs des S-Boxes dans une ronde AES. Une des réalisations consiste à mettre en fonction un générateur de nombre aléatoire dont les sorties sont fortement chargées lors d'un calcul sensible. Dans tous les cas, cet ajout coûte en consommation et s'il est réalisé temporellement, également en performance.

Redistribution aléatoire d'instructions utiles indépendantes ("suffling") Il s'agit de redistribuer aléatoirement les opérations qui sont indépendantes dans un ensemble donné d'instructions. La Figure 3.7 représente une telle redistribution des calculs de S-Boxes dans une ronde de l'AES. Cette contre-mesure n'a d'impact ni sur les performances ni sur la consommation mais son efficacité est extrêmement dépendante de la structure du code exécuté. En effet, moins il y aura d'instructions indépendantes, plus le bruit ajouté sera faible.

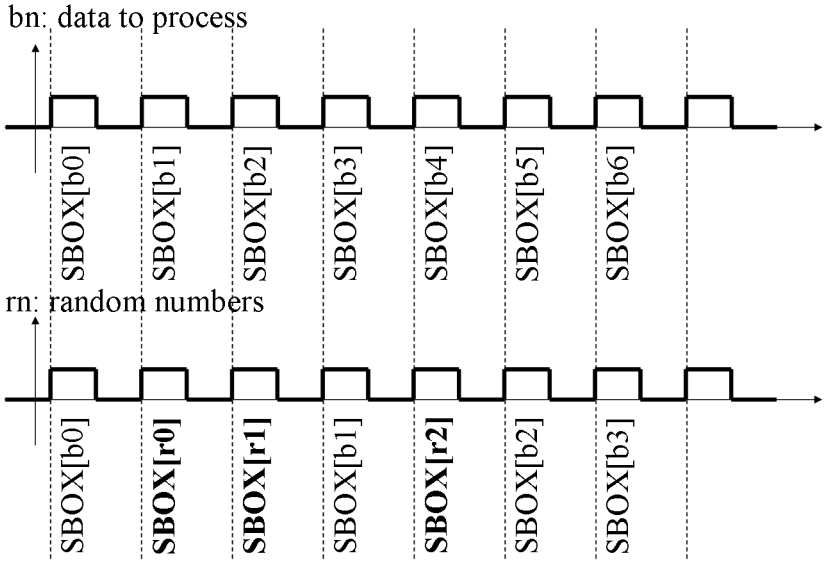

Fig. 3.6 – Insertion d'instructions factices lors d'un calcul

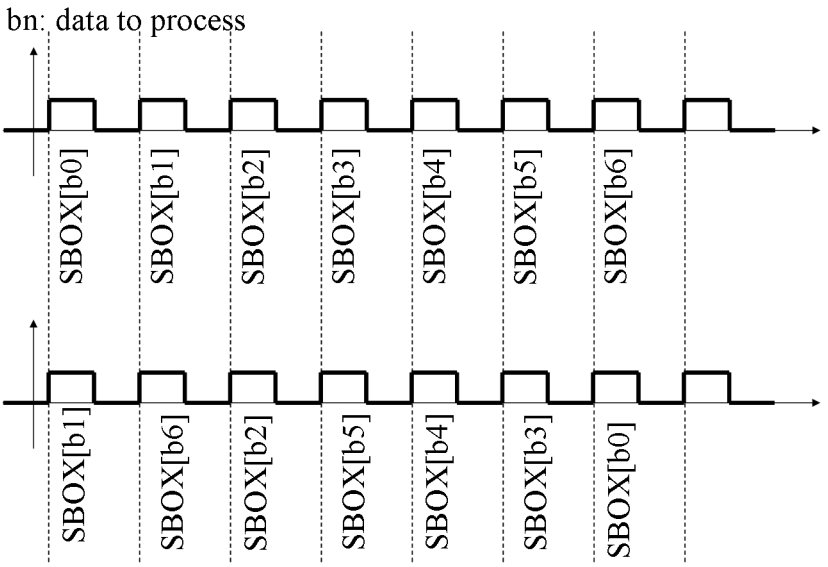

Fig. 3.7 – Redistribution aléatoire d'instructions indépendantes

Parallélisation d'opérations indépendantes Il est important de noter que ne rentre pas dans ce cadre les efforts faits sur l'équilibrage des données. En effet lors de l'équilibrage, les opérations sont faites en parallèle mais ces opérations ne sont pas indépendantes (l'une est la duale de l'autre). Dans le cas du parallélisme, c'est bien le niveau de bruit que l'on cherche à augmenter et non la valeur du signal à réduire. Par exemple, pour l'AES, soit on pipeline soit on augmente la taille des chemins de données. Il serait bien évidement

possible de faire les deux à la fois mais le surcoût deviendrait certainement prohibitif.

Modification dynamique de la tension il s'agit de faire varier aléatoirement la tension d'alimentation du circuit. L'amplitude de la variation doit être cependant relativement faible pour qu'aucune erreur de fonctionnement n'apparaisse. Les circuits asynchrones QDI sont particulièrement bien adaptés pour l'implantation d'une telle contre-mesure car ils acceptent des variations importantes de la tension d'alimentation (quasiment jusqu'à VDD/2 + 10 pourcent).

3.3.3.3 Filtrage

Il s'agit de réduire la signature à l'aide de filtres analogiques (passe bas) ou de dispositif de découplage de l'alimentation ad hoc basés sur l'utilisation de condensateurs (comme indiqué en Figure 3.8). Il a été montré que les alimentations spécifiques aux circuits téléalimentés peuvent être dimensionnées pour réduire la signature.

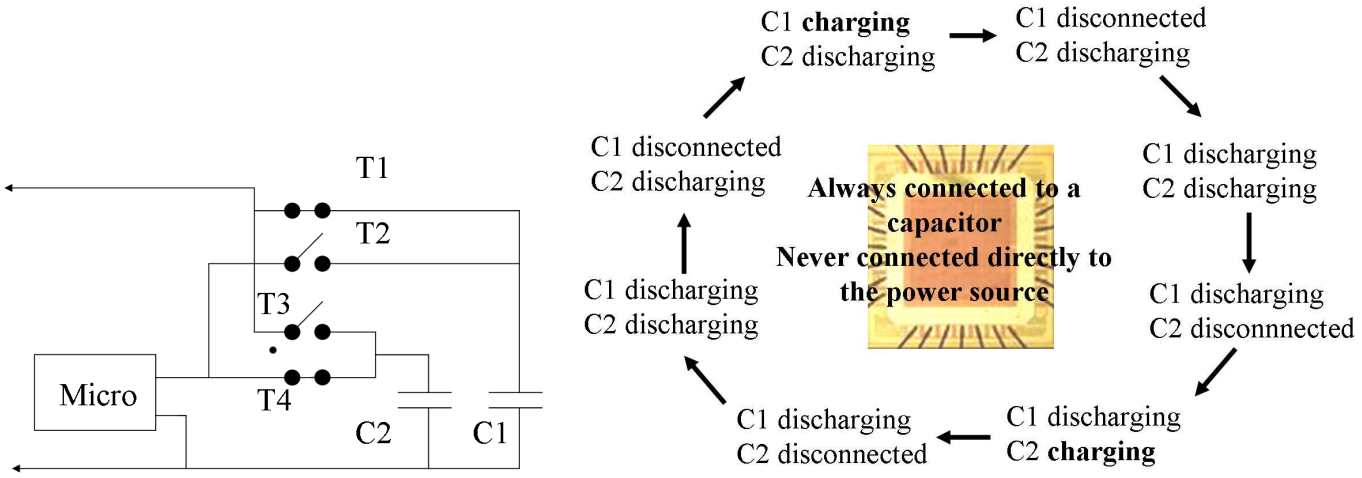

Fig. 3.8 – Découplage de l'alimentation à l'aide de condensateurs

3.3.4 Principe 2 : Rendre non prédictibles les variables intermédiaires

3.3.4.1 Masquage

Il s'agit de réaliser les opérations de cryptographie non pas directement avec les textes clairs et les clefs secrètes mais dans un premier temps avec des données issues de combinaisons entre celles-ci et un nombre aléatoire (le masque) puis de recombiner le résultat obtenu et ce nombre aléatoire pour obtenir le résultat original (c'est-à-dire non masqué).

Le principe de cette technique peut être facilement illustré par l'exemple suivant. Soit une fonction f d'une variable u. Le concepteur souhaite faire en sorte que le résultat de f ne puisse pas être prédit par l'attaquant. Il défini à cet effet une fonction f' telle que $f'(u \oplus r) = f(u) \oplus r$. $f(u)$ est dans ce cas calculée en effectuant $f'(u \oplus r) \oplus r$. Le principal

inconvénient de cette technique est qu'elle nécessite le calcul préalable de la fonction f' pour chacun des masques.

3.3.4.2 Partage du secret

Il s'agit de scinder aléatoirement les données en plusieurs "jetons", de réaliser des calculs indépendamment sur ces jetons puis de les rassembler pour reconstituer le résultat. Dans le cas d'une fonction linéaire f d'une variable u, la calcul sera, par exemple, réalisé en scindant r en deux jetons $u \oplus r$ et r, de calculer indépendamment $f(r \oplus u)$ et $f(r)$ puis d'obtenir $f(u)$ en utilisant la linéarité de la fonction f : $f(u \oplus r) \oplus f(r) = f(u) \oplus f(r) \oplus f(r) = f(u)$.

3.3.5 Principe 3 : Rendre non prédictible l'ordonnancement des opérations faites sur les données

Il s'agit de réaliser une suite d'opérations sensée ne pas être connue par l'attaquant. Ce principe est utilisé en cryptographie (on parle d'algorithme "propriétaires") mais également en développement logiciel. Cette approche est utilisée mais très controversée car elle remet en cause le principe de Kerckhoffs. Cependant elle est appliquée comme une stratégie de défense en cas d'attaque. En effet, si le circuit détecte une intrusion (à l'aide des techniques présentées dans les sections suivantes), il réalise autre chose que le standard ce qui empêche, par exemple, l'attaquant de remonter aux informations secrètes par analyse des différences. On parle généralement dans ce cas de "calculs infectieux ou pathogène". Le principe du calcul pathogène a été introduit en 2001 par S-M. Yen, S. Kim, S. Lim et S. Moon [75]. Cette contre-mesure propose de modifier l'algorithme RSA-CRT de telle sorte que si une faute modifie une composante CRT (s_p ou bien s_q), la faute contamine aussi l'autre composante. Ainsi, $\hat{S} \neq S \bmod q$ et $\hat{S} \neq S \bmod p$ ce qui rend l'attaque du *pgcd* inefficace. Deux protocoles utilisant le calcul pathogène ont été proposés dans [75] mais ils furent cassés quelques années plus tard par S-M. Yen et D. Kim dans [74]. Dans [30], M. Joye et al. présentent un AES doublé dual (c'est-à-dire 2 AES dont un effectue tous ses calculs avec des 0 à la place des 1 et inversement) dont certains blocs logiques sont partagés : une faute infecte ainsi chacun des deux chemins de manière différente et supposée inconnue par l'attaquant.

3.3.6 Principe 4 : Faire en sorte que le circuit, en cas de perturbation, ne communique plus d'information exploitable par l'attaquant

Généralement, l'approche adoptée consiste à détecter l'erreur puis à appliquer une politique de sécurité qui est censée éviter à l'attaquant de récupérer l'information sensible. Dans la suite du document, cette approche sera appelée "détection-action". Sur des circuits de cryptographie, l'information à cacher en cas d'erreur est le résultat d'un chiffré faux (pour rendre caduque l'hypothèse 9 et ainsi contrer les attaques DFAs). Dans le cas des attaques en faute sur logiciel, l'information étant potentiellement d'origine multiple, la stratégie généralement employée est le mutisme du composant.

La détection d'intrusion est basée soit sur des capteurs, soit sur de la redondance d'information, laquelle peut être temporelle ou spatiale et être d'ordre plus ou moins élevé, l'ordre considéré étant le rapport entre le nombre de bits effectivement utilisés pour

coder l'information sur le nombre de bits minimal possible pour coder cette dernière. Pour mettre en œuvre le principe 4, la redondance n'est implantée que pour détecter les erreurs (il ne s'agit pas de la corriger) : son ordre est alors compris entre 1 et 2. Plus l'ordre sera élevé, plus les capacités de détection (mais malheureusement également le coût) seront importants. Cette redondance peut prendre la forme d'un simple contrôle de parité, de signatures, de codes détecteurs ou de représentations redondantes comme le "1-n" (ou "1-hot"). La redondance d'information doit être localisée de telle sorte qu'elle détecte des erreurs lors d'un maximum d'opérations du circuit. Le choix de la localisation de la redondance est relativement naturelle dans le cas d'algorithme de cryptographie mais plus difficile sur un processeur.

3.3.6.1 Capteurs

Différents capteurs ont été implantés pour détecter des variations anormales de la tension d'alimentation, de la fréquence d'horloge, de la température et de l'éclairement du composant, ce dernier type de détection étant généralement réalisé en face avant à l'aide de photo-diodes.

3.3.6.2 Redondances spatiales

Bit de parité : Application à l'AES La plus simple forme de redondance est l'utilisation d'un bit de parité. Celui-ci permet de détecter les fautes de type SEU mais également les fautes de type MEU qui inversent un nombre impair de bits par octet. Les fautes qui inversent un nombre pair de bits sur un octet ne seront pas détectées. Deux approches basées sur l'utilisation d'un bit de parité ont été proposée pour sécuriser l'AES : il s'agit soit d'avoir un bit de parité par groupe d'octets ("état" de l'AES, constitué de 16 octets pour l'AES-128) soit d'avoir un bit de parité par octet.

Dans le premier cas, un seul bit de parité pour les 128 bits d'information ([31], [72]). Cette parité se propage dans les parties linéaires de l'algorithme :

- `ShiftRows` : aucune modification de la parité globale car cette transformation réalise un changement de place des octets dans la matrice d'état.
- `MixColumns` : chacun des octets est multiplié par les coefficients 2, 3, 1 et 1. Or comme 3 se décompose en 2 et 1, on voit apparaître chaque octet deux fois avec un coefficient 2, ce qui en annule la parité, et trois fois avec le coefficient 1, ce qui conserve donc la parité global de chaque colonne (32 bits) et donc du mot de 128 bits.
- `AddRoundKey` : la parité du résultat dépend de la parité de la clé de ronde, on doit donc effectuer un XOR du bit de parité de la donnée avec le bit de parité de la clé de ronde.

Mais le bit de parité ne se propage pas simplement dans les S-Boxes (car non linéaires). Une solution proposée dans [53] consiste à utiliser des S-Boxes ayant 8 bits en entrée et 9 en sortie, le bit supplémentaire étant le résultat du XOR entre le bit de parité de l'entrée et celui de la sortie (cette valeur est facilement calculable à partir de la description des S-Boxes). La parité du nouveau mot constitué de ces 16 bits supplémentaires (en sortie des 16 S-Boxes) est ensuite ajoutée au bit de parité des 128 bits en entrée des S-Boxes pour obtenir la parité du mot de 128 bits en sortie des S-Boxes. Cette contre-mesure est efficace pour détecter les SEU dans le chemin de donnée et le KeySchedule et le surcoût

est minimum (environ 7% en surface et 6% en temps pour le chemin de données). La difficulté d'utiliser un code dans l'AES (comme dans le DES) tient au fait qu'il y a un passage non-linéaire dans l'algorithme, les S-Boxes. Ainsi il est très difficile d'adopter un codage qui soit pertinent sans un gros surcoût.
Dans le second cas, les bits d'erreurs associés à chaque octet de l'état sont organisés en une matrice 4 bits par 4 bits [5]. Cette matrice d'erreur ne subit aucune des transformations de l'AES. Elle n'est modifiée que si une autre erreur survient. La propagation des bits de parité et de la matrice d'erreurs lors du passage dans les S-Boxes est représentée sur la figure 3.9. Comme dans le cas précédent, cette contre-mesure nécessite l'utilisation de S-Boxes avec 9 bits en sortie.

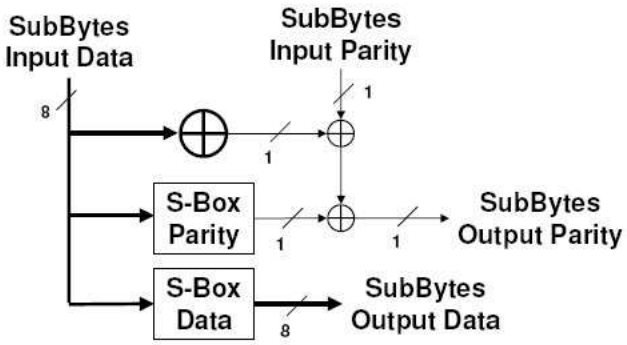

FIG. 3.9 – Schéma du passage de la parité au niveau des S-Boxes

Pour les autres opérations, il y a un surcoût par rapport à la contre-mesure précédente. En effet, si le passage dans le `ShitRows` n'induit aucune modification hormis la position, le `MixColumns` ne conserve pas la parité par octet car elle change en fonction de la multiplication dans le *xtimes()*. Un calcul est donc nécessaire pour déterminer la parité de chacun des octets. Pour l'`AddRoundKey`, il est également nécessaire de disposer du bit de parité de la clé octet par octet.

Duplication L'étude sur AES de Malkin, Standaert et Yung [41] montre que la duplication est le coût moyen nécessaire pour avoir une couverture suffisante aux MEU (Multiple Event Upset, fautes multiples). Il s'agit d'effectuer deux fois le même calcul (ou un calcul corrélé) en parallèle et de comparer à chaque étape du calcul si les deux circuits fournissent les mêmes résultats. Pour être en mesure de contourner cette contre-mesure l'attaquant doit être capable soit d'injecter simultanément deux fautes impactant simultanément les calculs redondants. Une architecture "double" d'un AES a été proposée dans [30]. L'un est le dual de l'autre (à chaque '1' logique correspond un '0' logique dans la partie duale et inversement) pour assurer un certain durcissement à l'attaque DPA.

Duplication spatiale par inversion Cette contre-mesure a été présentée par Karry et al [32]. Le principe est de déchiffrer pour vérifier. Cette vérification peut se faire au

niveau de l'algorithme (on déchiffre l'ensemble de l'algorithme, on compare le déchiffré au clair, si les deux sont identiques, on valide le résultat), à chaque ronde ou à chaque transformation, comme indiqué sur le schéma 3.10.
Le calcul de l'inverse de la transformation S-Box de l'AES est détaillé sur la figure 3.11.

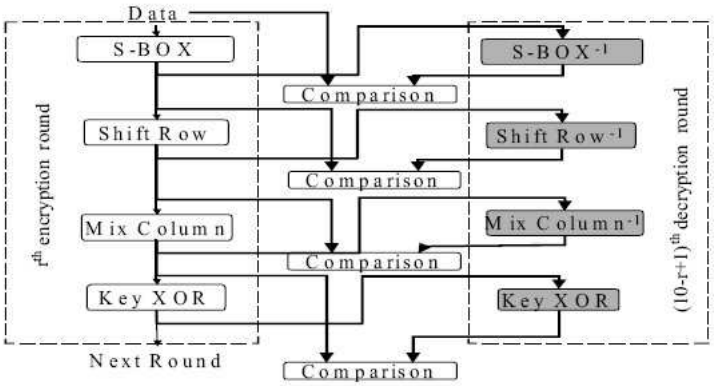

Fig. 3.10 – Schéma de la comparaison à chaque transformation

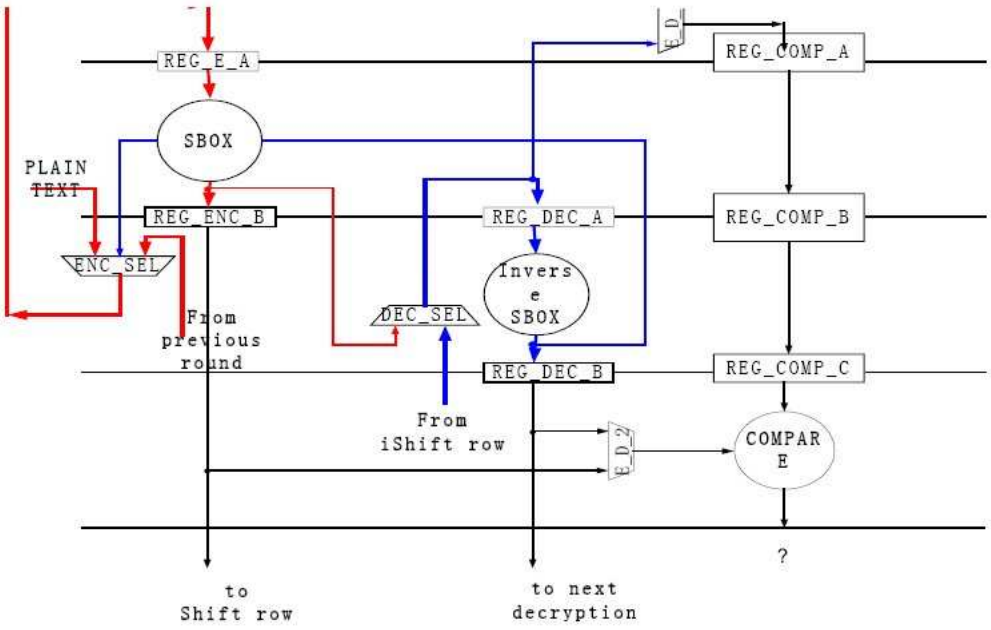

Fig. 3.11 – Schéma de la comparaison au niveau des S-Boxes

Cette contre-mesure induit un surcoût en surface très important (plus que la duplication) et augmente d'environ 50% la durée de traversée des signaux dans le chemin critique (ce qui entraîne la baisse de la fréquence maximum de fonctionnement).
En revanche, cette contre-mesure peut induire un surcoût relativement faible en surface

si on partage le maximum de matériel entre la phase de chiffrement et de déchiffrement. Dans ce cas, cette technique augmente légèrement le temps de traversée d'une ronde et surtout nécessite deux fois plus de cycles d'horloge pour réaliser un chiffrement.

3.3.6.3 Redondances temporelles

Il s'agit d'effectuer plusieurs fois le même calcul (ou un calcul corrélé) l'un après l'autre et de comparer en fin de calcul si les deux exécutions fournissent les mêmes résultats. Pour contourner cette contre-mesure l'attaquant doit réaliser deux fautes à deux instants qui correspondent aux mêmes calculs. Alors qu'on pense souvent à la redondance temporelle comme étant entière (plusieurs fois le même calcul puis comparaison des résultats), une idée peut être de penser à cette redondance temporelle comme pouvant être à granularité plus faible. Ainsi par exemple, on peut décider de ne calculer deux ou trois fois que les deux dernières rondes d'un AES (les plus sujettes aux attaques), ce qui revient à un coût temporelle de +20% seulement. De même, dans le cadre de l'exécution d'un programme, on peut décider de refaire faire seulement certaines opérations ou encore une opération parmi N.
Au lieu de faire le même calcul plusieurs fois, certaines techniques visent plutôt à faire le calcul inverse pour vérifier qu'on remonte de façon correcte à l'information initiale. Dans ce cas, une seule faute bien localisée peut permettre de contourner la contre-mesure. Une contre-mesure évidente est d'utiliser la clé publique (e, N) pour vérifier que le mécanisme de signature n'a pas été perturbé (i.e : vérifier que $S^e mod\, N = h(M)$). Malheureusement, cette contre-mesure présente deux défauts majeurs qui la rend peu utilisable en pratique. Premièrement, elle nécessite de pouvoir exécuter une seconde exponentiation modulaire. Or, cette opération peut s'avérer très coûteuse si l'exposant public e est grand même si souvent il est fait pour permettre une vérification rapide. Par ailleurs, le paramètre e, bien que public, n'est, bien souvent, pas stocké sur les périphériques cryptographiques (i.e : cartes à puce).

3.3.7 Principe 5 : Faire en sorte que le circuit donne le bon résultat, indépendamment de l'environnement dans lequel il est inséré, mais avec émission d'un syndrome fonction des données sensibles

Il s'agit de rendre le circuit tolérant aux fautes, au sens où il fournira les résultats attendus, mais avec émission d'un syndrome qui est fonction de données. Une technique triviale pour mettre en œuvre ce principe consiste dans un premier temps à réaliser deux fois le même calcul (temporellement ou spatialement) puis à comparer les résultats. Si ces derniers sont égaux, le résultat est émis ; sinon un troisième calcul et un vote majoritaire sont réalisés avant l'émission du résultat. Dans ce cas trivial, le syndrome potentiellement utilisable par l'attaquant est la troisième exécution.

Par ailleurs, cette forme de tolérance est approchée (c'est-à-dire que le principe est seulement appliqué dans certaines conditions) par les circuits asynchrones dit "QDI". En effet, ces circuits présentent les comportements suivants en présence d'une faute :

– Si la faute est transitoire et qu'elle impacte le combinatoire, les 5 cas suivants, illustrés sur la figure 3.12, peuvent se présenter : Comme dans un circuit synchrone,

la faute peut s'évanouir dans le combinatoire (cas 1) ou être absorbée par la logique (cas 2). Si ce n'est pas le cas, elle atteint des éléments de mémorisation. Dans ce cas, soit elle n'est pas mémorisée (cas 3), soit elle l'est. Dans ce cas, si la valeur injectée est identique à la valeur devant normalement transiter à l'emplacement de la faute, celle-ci crée un délai (cas 4). Dans le cas contraire, une valeur interdite ou invalide est générée (cas 5).

– Si la faute inverse la sortie d'un élément de mémorisation, trois cas se présentent : Si les entrées de l'élément de mémorisation sont toutes identiques (cas 6), la sortie s'inverse de nouveau et la faute s'apparente à une faute transitoire. Si les entrées sont différentes et si la valeur injectée est identique à la valeur devant transiter sur la sortie, la faute crée un délai (cas 7). Si la valeur injectée est différente de la valeur devant normalement transiter sur la sortie, une valeur interdite ou invalide est générée (cas 8).

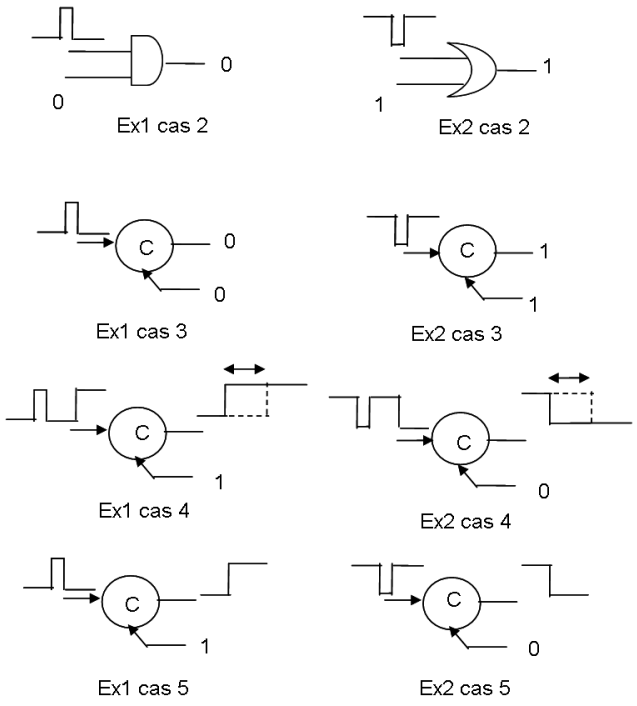

Fig. 3.12 – Illustrations des comportements d'un circuit asynchrone en présence d'une faute transitoire

Les cas 4 et 7 mettent en évidence que le circuit asynchrone peut répondre correctement, même en présence d'une erreur, mais avec un délais qui est fonction des données.

3.3.8 Principe 6 : Faire en sorte que le circuit fonctionnement normalement, indépendamment de l'environnement dans lequel il est inséré, sans émission d'un syndrome fonction des données sensibles

Seules des techniques approchées (c'est-à-dire appliquant ce principe que dans certaines conditions) ont été proposées. Il peut s'agir de mettre en place des filtres passe-bas sur l'alimentation pour absorber les glitchs sur l'alimentation, d'utiliser une horloge interne pour rendre le circuit insensible aux overclocking, d'utiliser les circuit QDI car ils sont insensibles à certaines variations de la tension d'alimentation. Des techniques basées sur la redondance d'ordre élevée (supérieure à 2) ont également été proposées. Lima et al. par exemple présentent dans [39] une technique de redondance spatiale triple avec vote majoritaire pour les FPGA basés sur de la SRAM. Le résultat du vote est utilisé pour corriger les calculs éventuellement fautés.

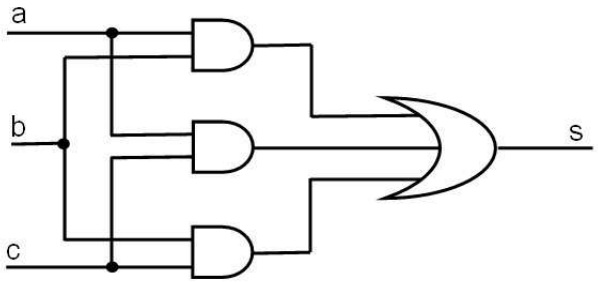

Fig. 3.13 – Schéma du vote majoritaire

Ce type de contre-mesure n'est pas appliquée dans la littérature aux algorithmes de cryptographie, essentiellement à cause de son coût (plus du facteur 3). Par contre, la triplication temporelle peut être envisagée au prix d'une division par 3 des performances et d'une triplication des registres.

3.4 Outils de conception dédiés aux circuits intégrés sécurisés

Les principales contre-mesures proposées dans la littérature ont été présentées dans la section précédente. Leur synthèse et l'estimation de leur efficacité aux différents niveaux d'abstraction ne peuvent que rarement être réalisés dans les flots de conception de circuits intégrés classiques. A cet effet, de nombreux outils dédiés ont donc été développés (se basant cependant très souvent sur des outils commerciaux).
Les outils de synthèse proposés ont essentiellement été développés pour implanter des contre-mesures très spécifiques telles que l'usage de double rail synchrone ou asynchrone (TAST du TIMA avec option d'optimisation particulières). Ils ne seront donc pas traités dans ce manuscrit.
L'estimation de la résistance d'un circuit à une attaque est basée sur une analyse soit fonctionnelle soit structurelle du circuit. Dans le premier cas, il s'agit de simuler un syndrome

particulier (courant, comportement en présence de faute, etc...) puis d'estimer l'efficacité de l'attaque considérée sur les résultats de simulation. Le niveau de résistance obtenu dépendra alors de la qualité de la simulation du syndrome mais également de la pertinence de l'attaque mise en œuvre. Dans le second cas, il va s'agir de comparer des grandeurs physiques sur des sous-circuits particuliers comme, par exemple, la valeur des condensateurs parasites sur des chemins de données duaux. Les résultats dépendront alors du choix des grandeurs physiques prises en compte et de leur méthode de comparaison.

3.4.1 Analyse fonctionnelle : effets de bord

Les simulations envisagées dans la littérature concernent essentiellement la consommation de courant instantané et le comportement du circuit en présence de faute. Des études plus prospectives adressent la simulation du rayonnement électromagnétique. Dans tous les cas, les simulations sont réalisées à différents niveaux d'abstraction, sachant que plus le niveau est élevé, plus la simulation est rapide (et permet donc l'analyse de circuit de taille importante) mais moindre est la précision.
Pour les descriptions algorithmiques, RTL et "porte", les simulations sont basées sur la propagation d'évènements discrets. On parle de simulation "logique" (ou "event-driven") ; les simulateurs utilisés sont des classes SystemC pour les descriptions algorithmiques ou des outils tels que ModelSim pour les descriptions RTL et "porte". Pour les descriptions niveau transistor, les simulateurs électriques (de type "spice" comme Eldo) intègrent les équations différentielles qui gouvernent le fonctionnement du circuit. Il existe enfin des simulateurs "mixtes" (comme NanoSim) qui, dans un premier temps, propagent les événements dans le circuit et qui, pour chaque événement propagé, réalisent ensuite des simulations électriques. Cette approche permet d'analyser des circuits de grandes tailles avec des temps de simulation et une précision raisonnables.

3.4.1.1 Simulation puissance et rayonnement électromagnétique

Pour les descriptions algorithmiques, RTL et "porte", la consommation de courant instantané est estimée à partir de résultats de simulations logiques. A cet effet, des outils commerciaux ou ad-hoc calculent, à chaque pas de temps, le poids de Hamming (HW), la distance de Hamming (HD) ou encore le nombre de transitions montantes (HDZO pour transition de 0 à 1) ou descendantes (HDZO pour transition de 1 à 0) des données manipulées. Pour les descriptions niveaux "porte", ces estimations, basées sur des simulations purement logiques et donc assez imprécises, peuvent être affinées grâce à des données de consommation précalculées à partir des descriptions de plus bas niveau et propres à chaque type de portes logiques mais aussi à la prise en compte de condensateur parasites des lignes d'interconnections. Ces estimations dites "tabulées" permettent d'obtenir le courant instantané avec une précision de l'ordre de 10 à 15% (pour un outil comme PrimePower ou NanoSim). Au niveau transistor, le simulateur électrique permet d'obtenir directement la valeur du courant consommé avec une précision très élevée (de l'ordre de 5%).
Le champ rayonné par un composant intégré vérifie les équations de Maxwell. Des outils les intègrent pour tout point de l'espace mais nécessitent des moyens de calculs beaucoup trop importants pour être mis en œuvre dans le cas du rayonnement d'un système aussi complexe qu'un circuit intégré. Des solutions plus simples mais approchées ont été envisagées. La première approche, proposée par Li et *al.*, consiste à diviser le système,

constitué d'une puce et d'un PCB, en deux parties. Un modèle électrique (RLC) du PCB est dans un premier temps extrait. Une simulation du courant consommé par l'ensemble est ensuite réalisée à l'aide d'un simulateur mixte. Le courant obtenu est considéré comme une image du champ rayonné par le circuit. Les auteurs ont, pour tenter de comparer les résultats obtenus à des résultats expérimentaux, modéliser l'effet "coupe-bas" de la sonde électromagnétique. Cette méthodologie a été appliquée sur un microcontrôleur pendant l'exécution de l'opération XOR. Deux tests sont réalisés avec des opérandes présentant des poids de Hamming différents. La courbe différentielle correspondant à la différence d'amplitude des émissions électromagnétiques des deux opérations XOR présente un pic ce qui montrent une relation entre le champ simulé et les données. Cette approche, si elle permet d'avoir une estimation de l'amplitude du champ rayonné (et ce, à une distance importante), ne prend pas en compte la localisation ni des sources ni de la position de la sonde.
Des simulations du champ rayonné en tenant compte de la géométrie du composant sont actuellement en cours de réalisation au LIRMM. L'approche consiste à simuler le courant qui transite dans les alimentations du composant. Chacune est scindée en dipôles élémentaires dont le rayonnement est prédit par la loi de Biot et Savard. Le champ global en tout point de l'espace est calculé en intégrant l'effet de ces dipôles au point considéré.

3.4.1.2 Estimation de la résistance

Le concepteur estime la résistance d'un circuit à une attaque à partir des résultats des simulations décrites précédemment. La première approche est basée sur l'hypothèse que l'attaquant peut réaliser des attaques par dictionnaire car il connaît et maîtrise l'ensemble des données manipulées par le circuit. Dans ces conditions, le concepteur réalise une simulation exhaustive de tous les états possibles du circuit (comme dans la phase d'apprentissage de l'attaque par dictionnaire) et recherche ensuite toutes les dépendances entre les données et les syndromes. La sécurité sera estimée par l'existence ou non de ces dépendances.
Une simplification, nécessaire pour analyser la résistance de circuits où la simulation exhaustive des syndromes pour tous les états est impossible, consiste à ne simuler le syndrome émis que pour deux valeurs particulières des données (typiquement $00\ldots00$ et $FF\ldots FF$) et à calculer la différence entre ces syndromes (on parle d'analyse de courbes différentielles).
La seconde approche, très répandue, consiste à considérer que l'attaquant ne peut réaliser qu'une attaque par corrélation (il n'a pas la maîtrise des clefs manipulées). Le concepteur réalise alors des simulations des syndromes pour différentes valeurs de textes clairs mais à clef constante. Il réalise ensuite une attaque par corrélation sur les résultats de simulation. L'opérateur qui est utilisé pour corréler les modèles aux simulations est soit la différence de moyenne, soit le coefficient de Pearson. L'information mutuelle a été récemment utilisée car elle permet de détecter des corrélations autres que linéaires entre des résultats de simulations et des modèles de consommation (ou de rayonnement électromagnétiques).
Une métrique classiquement utilisée pour estimer la résistance du circuit est le nombre de textes clairs nécessaires pour récupérer tout ou partie de la clef.

3.4.1.3 Résumé des travaux

Les approches décrites ci-dessus ont été appliquées sur des modèles aussi variées que des portes logiques, des blocs cryptographiques, voire des SOC et sont reportées dans le Tableau 3.1.

Ref	Circuit	Niveau	Simulation courant	Information exploitée
[40]	Portes logiques	Transistor	Electrique	Information mutuelle
[44]	Portes logiques masqués	Transistor	Electrique	Courbe Différentielle
[70]	AES en WDDL	Transistor	Electrique	Courbes DPA
[14]	AES	Transistor	Nanosim	Courbes DPA
[24]	ALU	Portes	PrimePower	Courbe différentielle
[15]	AES	RTL	HW	Courbes DPA
[54]	AES	RTL	HD	Courbes DPA
[47]	SOC	Système	HD	Courbes DPA

Tab. 3.1 – Etude de la robustesse des circuits face aux attaques en puissance

3.4.2 Analyse fonctionnelle : attaques en faute

Pour estimer la résistance des circuits face aux attaques optiques, Li et Moore proposent une procédure permettant de simuler l'effet du laser sur une netlist écrite en Verilog [38]. Ils commencent par extraire de la netlist les modules sur lesquelles la faute sera injectée. Ces modules sont ensuite remplacés par la description niveau transistor. Certaines cellules de ces modules vont ensuite être modifiées pour émuler les impulsions de courant créées par le laser. Un co-simulateur HDL/SPICE est enfin utilisé pour simuler la netlist Verilog avec les modules décrits niveau transistor. Ceci permet d'avoir à la fois la vitesse d'une simulation niveau portes et la précision d'une simulation niveau transistor. Une série d'expériences a été réalisée sur un processeur synchrone. L'ALU et le décodeur d'un système représentatif d'une carte à puce étaient visés. Ces comportements simulés ont été validés par les résultats expérimentaux.

Monnet et Renaudin se sont intéressés aux circuits asynchrones et tout particulièrement les circuits quasi insensibles aux délais (QDI), afin d'évaluer leur sensibilité aux attaques en fautes [50]. Ils ont défini le critère de sensibilité d'une porte de Muller suivant : une porte de Muller à N entrées est dite M-sensible à 0 (resp. 1) si et seulement si M de ces entrées ainsi que la sortie sont égales à 0 (resp. 1). Par conséquent, si M fautes sont injectées sur ces M entrées alors la porte génère une transition montante (resp. descendante). La sensibilité d'une porte de Muller est définie par une métrique qui mesure le temps pendant lequel la porte est dans un état sensible en respect au temps total pendant lequel la porte est utilisée. En considérant l'état d'un circuit comme l'état de toutes les portes de Muller, la sensibilité d'un circuit est définie comme la moyenne de temps passé dans un état sensible par le temps total de l'utilisation des portes. Ce critère a été appliqué sur une netlist décrivant un sous module du DES (les 8 S-Box et le XOR) en utilisant un programme développé en langage C et est interfacé au simulateur grâce à une librairie

logicielle (PLI). Les résultats montrent que le module est sensible à une faute de type transitoire seulement 27% du temps.

Faurax [22] a proposé un outil d'analyse de sécurité des circuits par injection de fautes en simulation, qu'il a baptisé **PAFI** (Prototype of Another Fault Injector). Son outil permet d'injecter des fautes transitoires (pendant un cycle d'horloge), par ajout de délai. Les fautes générés sont de type collage ou inversion. PAFI permet également de réaliser une étude statistique du circuit étudié afin de déterminer les endroits stratégiques d'injection de fautes. L'outil a été testé sur une version non sécurisée de l'AES.

3.4.3 Analyse structurelle

L'analyse structurelle est réalisée à des niveaux d'abstraction bas (typiquement au niveau porte avec rétro-annotation) et concerne à notre connaissance qu'exclusivement les circuits en double rail. Les métriques d'évaluation sont basées sur l'étude des dissymétries des condensateurs de charges et parasites, des temps de transition et d'arrivée des signaux. A titre d'illustration, considérons la modélisation d'une cellule double rail donnée par la figure 3.14 [58], C_1 et C_2 désignant les condensateurs de charge et d'interconnexion parasites.

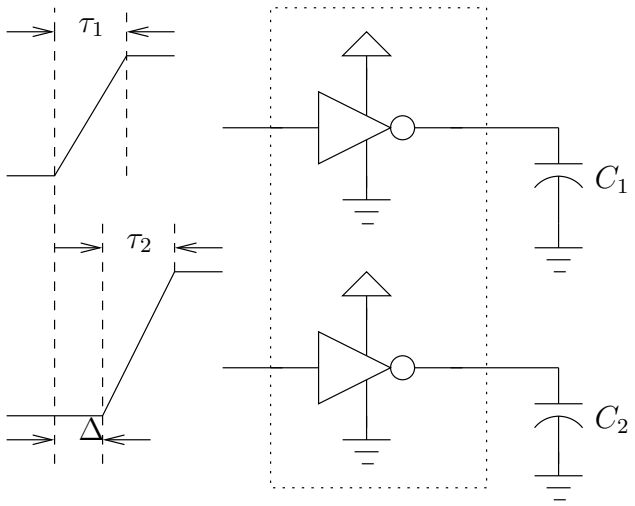

Fig. 3.14 – Cellule double rail

3.4.3.1 Déséquilibre des capacités de charges

Bouesse et al [12] ont défini le critère suivant pour estimer la résistance d'un double rail à la DPA :

$$d = \frac{|C_1 - C_2|}{min(C_1, C_2)}$$

Plus la valeur de d est petite, plus le rail (et le circuit) est résistant à la DPA. Ce critère a été appliqué sur deux netlists de l'AES placé routé : une netlist plate et une hiérarchique.

Les résultats montrent que le placement routage plat augmente considérablement la dissymétrie. Les auteurs recommandent donc un placement routage hiérarchique qui permet de maîtriser la longueur des pistes et la dispersion des modules.

Guilley a évalué la robustesse de la méthode de duplication [28]. Son critère de sécurité est basé sur le rapport des capacités de charge.

$$d = \frac{C_1}{C_2}$$

Plus d est proche de 1, plus le circuit est résistant à la DPA. En appliquant ce critère sur les 2211 paires de noeuds d'un DES avec duplication, il s'avère que 80% des noeuds présentent un rapport de capacités très proche de 1.

Une autre étude a été menée dans [58] où les auteurs définissent une valeur critique pour le rapport des capacités de charge au-delà de laquelle l'amplitude du courant différentiel en courant devient supérieur au plus petit déséquilibre en courant qui peut être détecté (I_{TH}), et donc peut être capturé par une attaque DPA.

$$\frac{C_1}{C_2} = max\{\frac{1}{R_i}\frac{\frac{V_{DD}}{V_{DSAT}}-1}{1-\beta}+1; (\frac{V_{DSAT}}{\beta.V_{DD}}(1-\frac{1}{R_i}))^{-1}\}$$

Avec $R_i = \frac{I_{max}}{I_{TH}}$, V_{DD}, V_T et V_{DSAT} sont respectivement la tension d'alimentation, la tension seuil et la tension de saturation du transistor. β est le rapport entre le courant que fournissent les inverseurs de la figure 3.14 lorsque $V_{DS} = V_{DD}$ et $V_{DS} = V_{DSAT}$.

3.4.3.2 Déséquilibre des temps de transition

En procédant comme pour les déséquilibres de charges, les auteurs de [58] ont défini une métrique de robustesse des cellules double rail au déséquilibre des temps de transition :

$$\frac{\tau_1}{\tau_2} = 1 - \frac{V_{DD}-V_T}{V_{DD}}\frac{1}{R_i} \text{ si } I_{max} > I_{TH}$$

I_{MAX} est le courant maximal que peuvent fournir les inverseurs.

3.4.3.3 Déséquilibre des temps d'arrivée des signaux

Un troisième déséquilibre étudié dans [58] est celui des temps d'arrivée des signaux :

$$|\frac{\Delta}{\tau}| = \frac{V_{DD}-V_T}{V_{DD}}\frac{1}{R_i} \text{ si } I_{max} > I_{TH}$$

3.5 Approche proposée

Le travail de cette thèse se situe dans la lignée des outils décrits précédemment, qui estiment l'efficacité des contre-mesures visant à réduire la corrélation entre les syndromes et les valeurs intermédiaires prédites par l'attaquant (principe 1). A noter que ces outils ont paradoxalement été également utilisés pour tester les contre-mesures visant à rendre non prédictibles ces valeurs intermédiaires (principe 2). L'explication de ce paradoxe vient

du fait que certains masquages laissent apparaître des données manipulées en clair (ie. non masquée) et que ces fuites sont détectables par les outils qui vérifient les contre-mesures implantant le principe 1.
La spécificité de ce travail de thèse est décrite dans les paragraphes suivants.

3.5.1 Recherche heuristique

La conception consiste à rechercher dans une hiérarchie d'abstractions un modèle de circuit qui vérifie un ensemble de contraintes sécuritaires données. Pour orienter sa recherche (dite "heuristique" car non exhaustive) vers ceux qui minimisent la valeur du quotient entre la composante informative et le bruit (c'est-à-dire qui mettent en œuvre le principe 1), le concepteur peut appliquer trois stratégies :

- Stratégie 1 : Réduire le rapport composante informative du signal à bruit.
- Stratégie 2 : Réduire la composante informative du signal.
- Stratégie 3 : Augmenter le bruit.

La dernière stratégie n'est pas considérée dans ce manuscrit car elle consiste, par définition, à rajouter des blocs qui n'ont aucun lien logique avec le traitement des données sensibles. Les techniques mises en œuvre à cet effet ont été décrites dans les paragraphes 3.3.3.2. Pour comparer les deux premières stratégies de recherche, l'évolution du signal et du bruit dans la hiérarchie d'abstraction va être analysée. Par analogie généalogique, on parlera de "génération" pour l'ensemble des circuits possibles à un niveau d'abstraction donné, de fils, le résultat d'une synthèse à partir d'un circuit de la génération précédente (le "père"), de frères ou de cousins les circuits d'une même génération issus du même père ou de pères différents, etc...

3.5.1.1 Evolution du signal

La composante informative du signal augmente lorsque l'on descend dans les niveaux d'abstraction car les descriptions des phénomènes physiques y sont plus précises. Considérons, par exemple, une porte `XOR` en double rail décrite au niveau RTL (ci-dessous) et au niveau porte (représentée sur la Figure 3.15) .

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
use ieee.std_logic_unsigned.all;

ENTITY dl_xor IS
    PORT    (
a    : IN  std_logic_vector(0 to 1);
       b    : IN  std_logic_vector(0 to 1);
       s    : OUT  std_logic_vector(0 to 1) );
end dl_xor;

ARCHITECTURE arch_my_dl_xor OF dl_xor IS
  signal ai : std_logic_vector(0 to 1) ;  -- signaux d'entree ( Af=ai(0) et At=ai(1) )
  signal bi : std_logic_vector(0 to 1) ;  -- signaux d'entree ( Bf=bi(0) et Bt=bi(1) )
  signal si : std_logic ;  -- signaux de sortie ( Sf=si(0) et St=si(1) )
```

```
begin
   ai <= a;
   bi <= b;
   si(0)<= ( ai(0) and bi(0) )  or ( ai(1) and bi(1) ) ;
   si(1)<= ( ai(0) and bi(1) )  or ( ai(1) and bi(0) ) ;
   s<=si;
end arch_my_dl_xor;
```

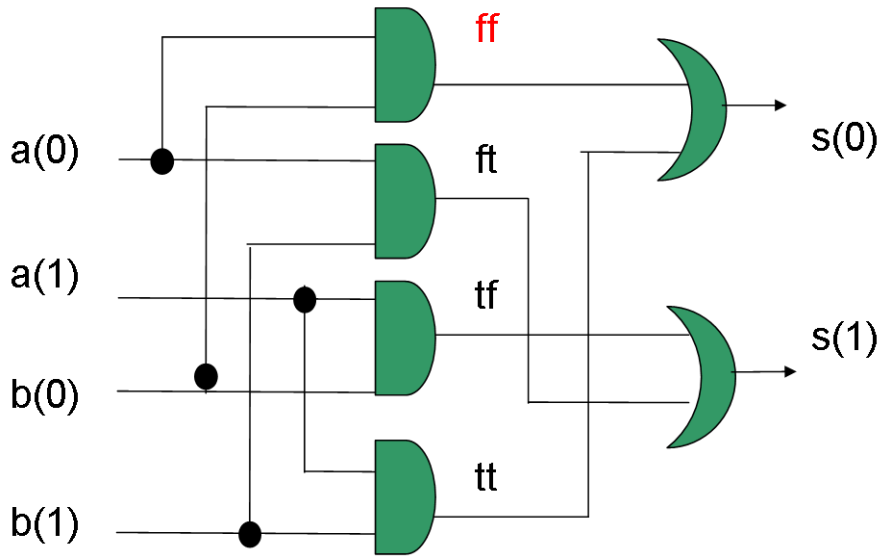

Fig. 3.15 – Porte `XOR` double rail (vue "porte")

Au niveau RTL, on suppose généralement que le transfert des données est instantané dans les fonctions logiques et que la consommation est fonction uniquement des transitions des signaux déclarés dans le modèle (hypothèse "distance de Hamming" décrite au 2.3.2.2). Les chronogrammes de la Figure 3.16 représentent les transitions engendrées pour toutes les valeurs possibles des données ainsi que la puissance associée. On constate que celle-ci est indépendante des données (3 transitions quelle que soit la donnée), donc la composante informative du signal est nulle.
Au niveau "porte", si l'on prend les mêmes hypothèses qu'au niveau RTL (transfert instantané des données et consommation "distance de Hamming"), le signal informatif reste nul (4 transitions quelle que soit la donnée, la transition supplémentaire étant générée par l'un des fils en sortie des portes "and").
Au niveau "porte" après la phase de placement-routage (c'est-à-dire au niveau d'abstraction inférieur), les temps de traversée des portes et de transfert entre ces portes sont non nulles. Si l'on suppose, par exemple, que le temps de traversée sur le fil "ff" est non nul (et noté Δ) mais que, par souci de simplification, les autres temps restent nuls, on obtient les chronogrammes de la Figure 3.17. Il s'avère que dans ce cas, les données $a(0) = 1; a(1) = 0; b(0) = 1; b(1) = 0$) en entrée engendrent 2 fois 2 transitions séparées temporellement de Δ (en rouge) alors que les autres données engendrent 4 transitions

simultanées. Dans ce cas, le signal informatif est non nul.

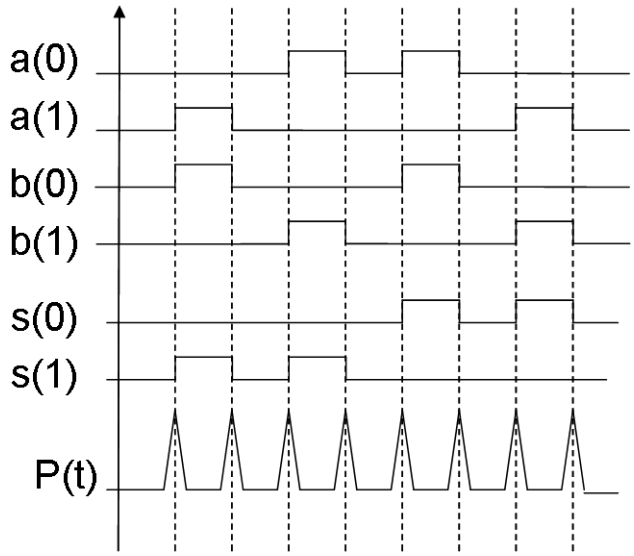

Fig. 3.16 – Profil de courant d'une porte `XOR` en double rail (niveau RTL)

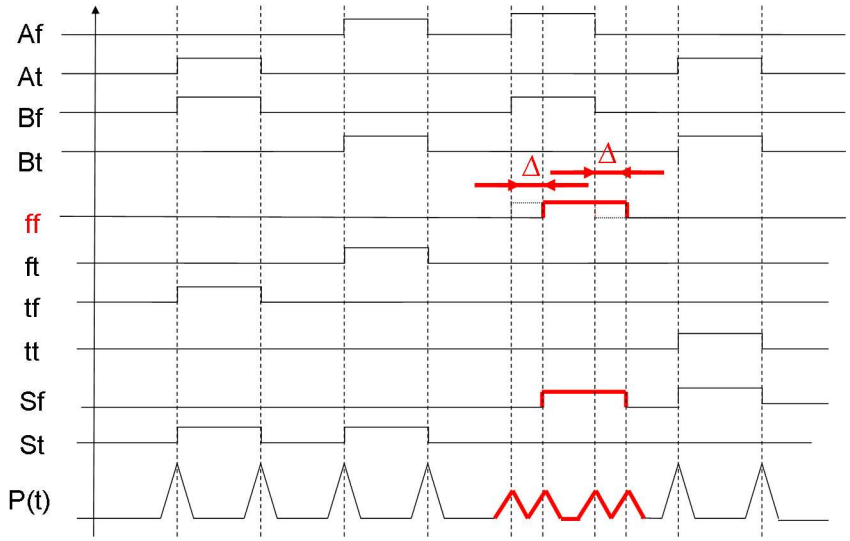

Fig. 3.17 – Profil de courant d'une porte `XOR` en double rail (niveau porte + prise en compte des temps de traversée dans les portes et les interconnexions)

3.5.1.2 Évolution du bruit

Il n'existe pas de relation similaire concernant le bruit. En effet, d'une part, le bruit peut augmenter lorsque l'on descend dans les niveaux d'abstraction car le nombre d'événements

qui se produisent simultanément est susceptible d'augmenter. D'autre part, le bruit peut diminuer car des phénomènes tels que les délais de propagation dans les portes logiques peuvent décaler l'instant d'apparition du bruit avec celui du signal informatif. Ces deux cas sont illustrés par la Figure 3.18. A gauche est représentée la consommation d'un circuit décrit au niveau RTL pour différentes valeurs de données. Les barres d'erreurs symbolisent le niveau de bruit (qui lui est indépendant des données). A droite est représenté la consommation du même circuit décrit au niveau porte. Sur la trace du haut, le niveau de bruit a augmenté par rapport au niveau RTL ; sur la trace du bas, le niveau de bruit a également augmenté mais comme le bruit est décalé par rapport au signal informatif, il ne le "cache" plus. Pour l'attaquant, le bruit apparaît comme plus faible.

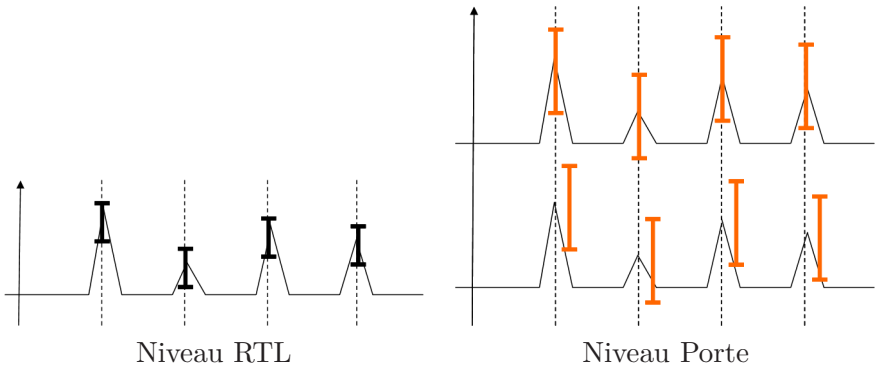

Fig. 3.18 – Évolution du bruit dans les niveaux d'abstraction

3.5.1.3 Exemple

Les considérations précédentes sont illustrées par des simulations de deux modèles d'un AES synchrone dual. Le premier modèle est décrit au niveau "portes" ; le second est le modèle obtenu à partir du premier après la phase de placement-routage. La puissance consommée est considérée être égale dans les deux cas à la distance de Hamming des données manipulées. Pour la simulation du second modèle, les rétro-annotations temporelles issues de la phase de placement-routage ont été prises en compte (usage du fichier sdf).
La figure 3.19 montre le résultat de simulation de consommation pour différentes valeurs de textes clairs. On constate que le nombre de transitions à chaque période d'horloge est sensiblement le même.
En tenant compte des délais des portes, une dispersion apparaît, laissant distinguer certaines valeurs (voir figure 3.19). Ces valeurs particulières peuvent créer des pics DPA ou être utilisées pour monter une attaque par dictionnaire.

Sur cet exemple simple, le rapport signal à bruit a augmenté suite à la prise en compte d'une information supplémentaire (les rétro-annotations temporelles).

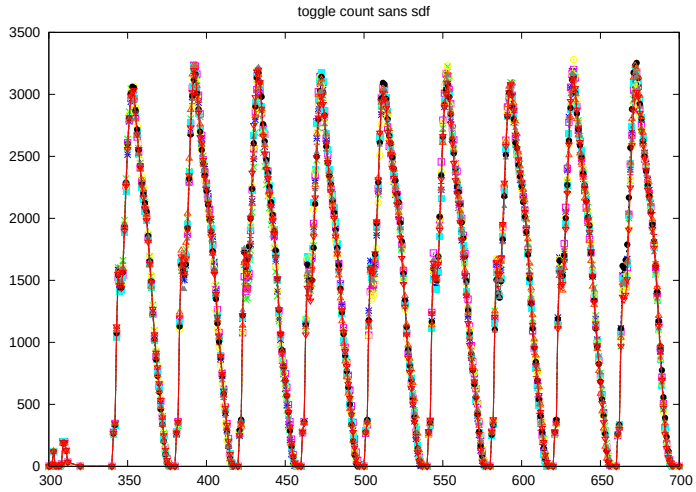

Fig. 3.19 – Simulation de consommation sur un AES dual sans le fichier SDF

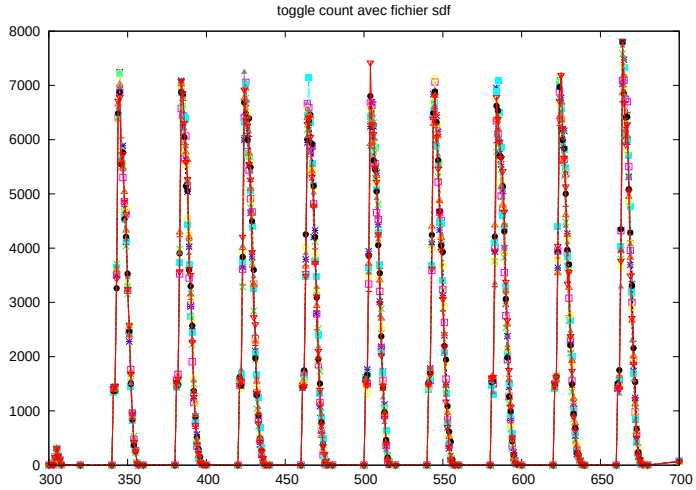

Fig. 3.20 – Simulation de consommation sur un AES dual avec le fichier SDF

3.5.2 Stratégie 1

3.5.2.1 Description

La stratégie de recherche heuristique 1 consiste à utiliser le rapport composante informative sur bruit (SNR) comme guide dans la hiérarchie d'abstractions. Elle est généralement utilisée couplée aux méthodes fonctionnelles décrites aux sections 3.4.1. Dans ce cas, le concepteur, à un niveau d'abstraction donné, choisi le modèle qui présente le plus mauvais SNR (ie, le meilleur niveau de sécurité). Il dérive ensuite de ce modèle un ou plusieurs modèle fils à l'aide des outils de synthèse. Comme observé dans les paragraphes précédents, un modèle fils peut avoir un rapport signal à bruit plus élevé, identique ou

plus faible (et donc un niveau de sécurité respectivement plus faible, identique ou plus élevé) que celui de son père. Les trois cas possibles sont représentés sur la figure 3.21 qui symbolise cette recherche : Les lignes pleines représentent les choix de modèles ; Les lignes en pointillés représentent les pistes non explorées. De ces modèles fils, il choisi de nouveau celui qui présente le SNR le plus bas. Il procède ainsi pour atteindre le niveau d'abstraction le plus bas.

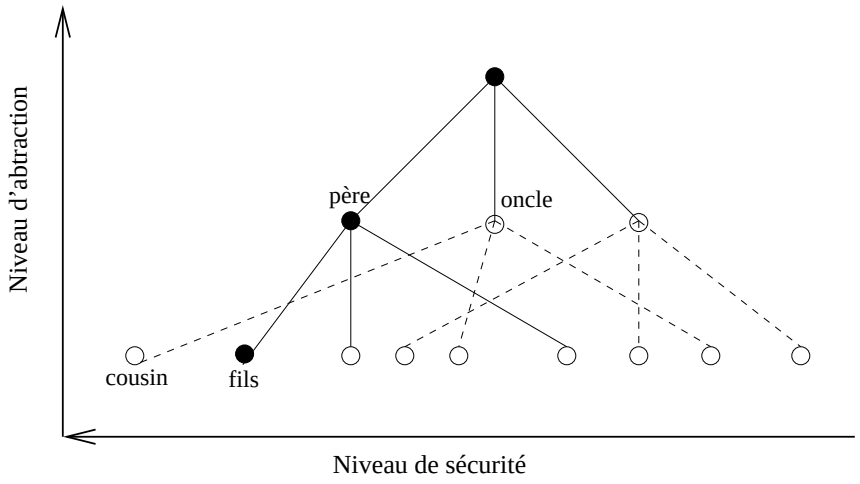

Fig. 3.21 – Recherche de modèles basée sur la réduction du SNR

3.5.2.2 Avantages

Comme expliqué dans le paragraphe 2.3.9, la DPA mesure un rapport signal-informatif sur bruit. La stratégie 1, qui vise justement à réduire ce rapport, est donc la plus naturelle pour contrer ce type d'attaque. La stratégie 1 est également relativement simple à mettre en œuvre puisqu'elle ne nécessite que l'adjonction d'une phase de simulation de puissance consommée (des outils existent pour tous les niveaux d'abstraction) et d'une phase d'attaque sur ces résultats de simulation (pour l'estimation du SNR). Enfin, les estimations peuvent être comparées aux résultats obtenus sur le circuit réel.

3.5.2.3 Inconvénients

Lorsque le concepteur souhaite augmenter le SNR du circuit en mettant en place des contre-mesures qui augmentent le bruit et qui diminuent le signal, la stratégie 1 est utilisable pour comparer diverses solutions à un même niveau d'abstraction mais est plus délicate à mettre en œuvre lors de la descente dans les niveaux d'abstraction. En effet, comme expliqué dans le paragraphe 3.5.1, l'évolution du rapport SNR entre ces niveaux d'abstraction est indéterminée. Autrement dit, il n'existe pas de relation entre le SNR d'un modèle à un niveau n et les modèles fils dérivés de celui-ci. Il peut donc arriver, comme représenté sur la Figure 3.21, qu'une solution "cousine" soit plus sécurisée que la solution retenue . Une telle configuration remet en cause l'efficacité de la recherche heuristique qui, par définition, doit focaliser l'attention du concepteur sur les meilleures solutions.

Cependant, dans la plupart des cas, la stratégie 1 est utilisée pour l'évaluation de contre-mesures visant à réduire le signal informatif (et non à augmenter simultanément le niveau de bruit). Dans ce cas, elle est adapté ni, comme dans le cas précédent, à la descente dans les niveaux d'abstraction ni à la comparaison de solutions à un même niveau d'abstraction. En effet, de mauvaises contre-mesures (d'équilibrage par exemple) peuvent apparaître comme efficaces alors qu'en fait c'est le bruit élevé d'une autre partie du circuit qui réduit le SNR. Inversement, une bonne solution peut être pénalisée par le faible bruit qu'elle engendre.
De plus, adopter la stratégie 1, en axant la conception sur la réduction d'un SNR, suppose implicitement que le concepteur ne cherche à contrer que les attaques qui extraient l'information d'un signal bruité (comme c'est le cas pour la DPA). Or, comme expliqué dans la section 2.3.9, l'attaquant cherche, avant l'extraction d'information en elle-même, à réduire la valeur de ce bruit grâce à des méthodes de traitement du signal mais également grâce à de nouvelles techniques de mesures (comme, par exemple, celles qui consistent à utiliser des sondes électromagnétiques extrêmement fines et précises ou pratiquer du micro-sondage). L'idéal pour lui étant de s'affranchir totalement du bruit. La stratégie 1 est alors inadaptée pour trouver les solutions optimales à ces types d'attaques (ie, sans bruit).
Par ailleurs, dans cette approche, le SNR est généralement estimé à partir des résultats d'une attaque DPA (réalisée sur les résultats de simulation). Or, cette attaque nécessite le choix d'un grand nombre de paramètres (nombre de bits d'attaque, modèle choisi pour prédire la consommation ou le rayonnement électromagnétique, choix des points d'intérêt, choix des textes clairs, etc...) et les résultats sont souvent soumis à interprétation. Ce lien très tenu entre attaque et conception peut mettre à mal l'évaluation objective de l'efficacité d'une contre-mesure dans la mesure ou un mauvais concepteur "mauvais attaquant" peut trouver une solution avec un SNR identique à celui obtenu par la solution d'un bon concepteur "bon attaquant".

3.5.3 Stratégie 2

3.5.3.1 Description

La stratégie 2 consiste à n'utiliser que la composante informative pour guider la recherche dans la hiérarchie d'abstractions. Elle est généralement couplée aux méthodes structurelles décrites aux sections 3.4.3. Dans ce cas, le concepteur, à un niveau d'abstraction donné, choisi le modèle qui présente le plus faible signal (ie, le meilleur niveau de sécurité). Il dérive ensuite de ce modèle un ou plusieurs modèle fils à l'aide des outils de synthèse. Comme observé dans les paragraphes précédents, un modèle fils présente un niveau de signal plus élevé que celui de son père. Cette recherche est symbolisée par la Figure 3.22. De ces modèles fils, il choisi de nouveau celui qui présente le signal informatif le plus faible. Il procède ainsi pour atteindre le niveau d'abstraction le plus bas.

3.5.4 Avantages

Lorsque le concepteur souhaite mettre en œuvre des contre-mesures qui diminuent le signal informatif, la stratégie 2 est utilisable pour comparer différentes solutions à un même niveau d'abstraction. Elle reste cependant délicate à mettre en œuvre lors de la descente

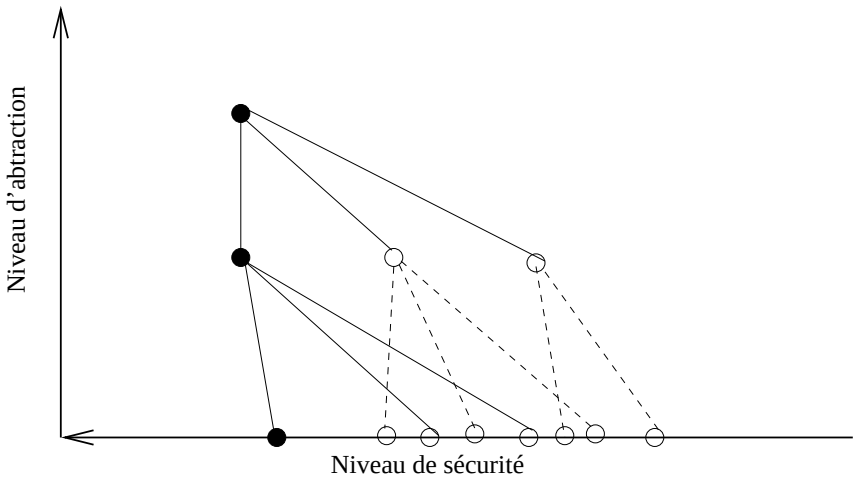

Fig. 3.22 – Recherche de modèles basée sur la réduction du signal informatif

dans les niveaux d'abstraction. Elle l'est cependant moins que la stratégie 1 car la configuration représentée sur la Figure 3.22 où un cousin de la solution retenue présente un niveau de sécurité plus élevé qu'un de ses frères est peu probable. En effet, ce croisement est dû à une forte augmentation du signal informatif lors de la synthèse, ce qui dénoterait d'une non maîtrise par le concepteur de ses contraintes sécuritaires.
La stratégie 2 axe la conception sur la réduction du signal informatif. Elle est en ce sens plus générique que la stratégie 1 car elle vise à contrer toutes les attaques par corrélation (DPA, CPA, EM, DBA, microprobing, etc ...).
Dans cette approche et contrairement à la stratégie 1, le signal n'est pas considéré comme une somme de contributions élémentaires (telle que la consommation globale) mais comme un ensemble de contributions élémentaires (telles que les dissymétries sur des double rail). Celles-ci sont plus faciles à traiter pour estimer la quantité d'information qu'elles contiennent. Les paramètres liés aux attaques sont moins nombreux et les résultats moins sujet à interprétation.

3.5.5 Inconvénients

La stratégie 2 n'est pas applicable pour développer des contre-mesures qui visent à augmenter le niveau de bruit.
Cette stratégie est contre-intuitive pour les concepteurs habitués à réaliser des attaques de type DPA sur les résultats de simulation. En particulier, sa mise en œuvre avec des critères structuraux est assez abstraite au sens où le lien entre la mesure du signal informatif (comme une différence de condensateurs parasites) et les mesures réelles n'a pas été établi de façon formelle.
Comme la stratégie 2 traite les signaux de façon indépendante, le volume de données à traiter augmentent avec la taille des circuits.

3.5.6 Spécificités du travail de thèse

Le travail proposé dans cette thèse consiste à appliquer la stratégie 2, non pas comme cela a déjà été proposé, en estimant la force du signal informatif avec des données structurelles mais avec des données fonctionnelles. Il s'agit concrètement de réaliser des attaques en corrélation non pas sur un signal global comme c'est le cas pour la stratégie 1 mais sur un ensemble de signaux élémentaires, pris un à un. Cet ensemble sera, par exemple, constitué de toutes les équipotentielles définies dans le modèle du composant. Le nombre d'équipotentielles qui délivrent effectivement une information sur les données sensibles sera alors considéré comme un critère de sécurité.
L'approche proposée :

- Représente un compromis entre l'application actuelle de la stratégie 2, qui est très abstraite, et celle de la stratégie 1 qui est très proche des attaques réalisées sur le composant.
- Est applicable à tout type de circuit et est donc plus générale que les applications actuelles de la stratégie 2 qui n'ont concernées qu'une classe particulière de circuit (les circuits double rail).
- Place le concepteur dans le pire cas, celui où l'attaquant s'affranchit de l'effet du bruit mais également des signaux parasites. Le concepteur anticipe les attaques futures.
- rend les résultats moins sensibles au talent d'attaquant du concepteur. En effet, comme les signaux pris en considération sont très simples, les différents paramètres d'attaque ont moins d'influence.
- Couvre toutes les attaques en corrélation. Elle est donc plus générique que la stratégie 1.

Elle sera appliquée dans la suite de ce manuscrit à des circuits décrits à des niveaux d'abstraction élevés (RTL, porte, porte avec rétro-annotation temporelle) mais pourrait être appliquée aux niveaux inférieurs.

3.6 Conclusion

Ce chapitre nous a montré que la conception des circuits intégrés sécurisés est un challenge pour les concepteurs. Les circuits intégrés sécurisés étant avant tout des composants micro-électroniques, ils suivent naturellement les étapes du flot de conception de ces composants. Avec l'apparition des attaques physiques qui exploitent les faiblesses de l'implémentation matérielle des algorithmes cryptographiques, diverses parades ont été proposées afin de protéger les circuits intégrés de ces attaques. Ces contre-mesures utilisent des techniques différentes. Une des techniques est basée sur une réduction de la corrélation entre les données manipulées par le crypto-système et les canaux auxiliaires (courant, émissions électromagnétiques). Des méthodes de masquage et de duplication sont également employées afin de rendre non prédictible les variables intermédiaires. Enfin, des éléments pour détecter et/ou corriger les erreurs sont utilisés pour se prémunir des attaques par injection de fautes.
Pour valider ces contre-mesures, les circuits intégrés sont soumis à des tests de sécurité. Traditionnellement, le test de sécurité est réalisé une fois que le circuit est fabriqué, ce qui présente un coût assez important dans le cas où le circuit échoue le test. Suite à ces pertes matérielles, les acteurs du domaine ont commencé à réfléchir à des moyens qui permettent

d'évaluer la sécurité de leurs composants aussi tôt que possible dans le flot de conception. Deux grands approches ont été proposées pour estimer le niveau de sécurité des modèles décrits à tous les niveaux d'abstraction : la première, appelée fonctionnelle, consiste à réaliser les attaques sur des résultats de simulations d'un syndrome global (comme le rayonnement électromagnétique ou la puissance consommée), la difficulté de cette attaque donnant un indice de la robustesse des contre-mesures. L'autre, appelée structurelle, consiste à mesurer les déséquilibres de consommation qui peuvent apparaître entre deux équipotentielles formant un double rail.

Les avantages et les inconvénients de ces deux approches ont été comparées. Celle proposée dans le cadre de cette thèse consiste à réaliser des attaques en corrélation sur les signaux internes du circuit pris un à un. Cette approche réunis les avantages de chacune des deux approches, au prix d'une augmentation du volume de données à traiter. Elle est mise en œuvre dans les chapitres suivants pour des circuits décrits à différents niveaux d'abstraction.

Chapitre 4

Méthodologie de conception pour l'évaluation des CIs

4.1 Introduction

Dans le chapitre précédent, nous avons introduit notre approche pour estimer la résistance des circuits cryptographiques face aux attaques par corrélation, basée sur l'exploitation du signal informatif de ces circuits. Cette approche consiste à réaliser des attaques par corrélation sur des signaux élémentaires pris un à un. Les résultats de ces attaques permettront de définir des métriques de sécurité.
Ce chapitre traite de la problématique d'analyse de corrélations dans les algorithmes cryptographiques : un exemple d'analyse de corrélations dans des modèles algorithmiques du DES et de l'AES sera présenté dans un premier temps. Les différents types d'analyses de corrélation dans des modèles plus complexes seront présentés et les métriques de sécurité associées seront discutées. Notre analyse de corrélations est décrite et une simplification de cette analyse est proposée et est justifiée afin d'être intégrée dans le flot de conception.

4.2 Principe de corrélations dans les crypto-systèmes

4.2.1 Principe de Shannon

Shannon a annoncé deux principes généraux pour assurer la sécurité d'un crypto-système : la confusion et la diffusion [65] :

- La confusion doit cacher les structures algébriques et statistiques.
- La diffusion doit permettre à chaque bit de texte clair d'avoir une influence sur une grande partie du texte chiffré. Ce qui signifie que la modification d'un bit d'un bloc d'entrée doit entraîner la modification de nombreux bits du bloc de sortie correspondant.

Partant du principe de diffusion, nous savons qu'il existe des corrélations entre les bits d'un algorithme cryptographique.

Les corrélations existent aussi dans les circuits intégrés numériques. Prenons l'exemple d'une porte `XOR` dont les 2 entrées A et B et la sortie S. Si A est variable et B est fixe,

nous avons :
- Si $B = 0 : S = A$
- Si $B = 1 : S = -A$

Par conséquent, si le bit B est fixe, S est corrélé au bit A.

4.2.2 Corrélations dans les algorithmes cryptographiques

4.2.2.1 Cas du DES

La structure de Feistel utilisée dans le DES met en évidence quelques corrélations. En effet, nous avons :

$$R_1 \quad = \quad L_0 \oplus f(R_0, K_1)$$

Supposons que L_0 est constant pour un ensemble d'exécutions de l'algorithme et considérons, par exemple, le premier bit en sortie de la première S-Box. Après la fonction `Permutation`, ce bit va se retrouver à la position 9 du registre R_1 :

$$R_1[9] \quad = \quad L_0[9] \oplus f(R_0, K_1)[9]$$

$R_1[9]$ est donc corrélé au premier bit en sortie de la première S-Box pour un ensemble de textes clairs dont la partie gauche est constante.
L'algorithme DES vérifie aussi la relation suivante :

$$L_2 \quad = \quad R_1$$

Les bits $L_2[9]$ et $R_1[9]$ sont donc corrélés.
Par conséquent, pour une attaque par corrélation réalisée sur le premier bit en sortie de la première S-Box de l'algorithme DES et pour un ensemble de textes clairs dont la partie gauche est constante, des pics peuvent apparaître quand les bits $L_2[9]$ et $R_1[9]$ sont manipulés, donnant ainsi des informations sur la clef de chiffrement. Ce résultat est valable aussi pour les autres bits en sortie des S-Boxes.

4.2.2.2 Cas de l'AES

Les transformations linéaires utilisées dans le chiffrement et le déchiffrement de l'AES mettent en évidence des corrélations. En effet, la transformation `AddRoundKey`, qui est un `XOR` entre la clef de ronde et le texte clair, a sa sortie corrélé au texte clair. La fonction `ShiftRows` est un décalage de bits, par conséquent les entrées et sorties de cette fonction sont corrélées. Enfin, la transformation `MixColumns` vérifie les équations suivantes :

$$S'_{0,c} \quad = \quad (02 \bullet S_{0,c}) \oplus (03 \bullet S_{1,c}) \oplus S_{2,c} \oplus S_{3,c} \tag{4.1}$$

$$S'_{1,c} \quad = \quad S_{0,c} \oplus (02 \bullet S_{1,c}) \oplus (03 \bullet S_{2,c}) \oplus S_{3,c} \tag{4.2}$$

$$S'_{2,c} \quad = \quad S_{0,c} \oplus S_{1,c} \oplus (02 \bullet S_{2,c}) \oplus (03 \bullet S_{3,c}) \tag{4.3}$$

$$S'_{3,c} \quad = \quad (03 \bullet S_{0,c}) \oplus S_{1,c} \oplus S_{2,c} \oplus (02 \bullet S_{3,c}) \tag{4.4}$$

En considérant un ensemble de textes clairs dont le premier octet varie et les 15 autres sont constants, d'après les équations 5.12 et 5.12, $S'_{1,c}$ et $S'_{2,c}$ sont corrélés à $S_{0,c}$.

Par conséquent, pour une attaque par corrélation réalisée sur un bit en sortie d'une S-Box de l'AES et pour un ensemble de textes clairs dont le premier octet varie et les autres sont constants, des pics peuvent apparaître pendant le calcul des transformations `ShiftRows` et `MixColumns`.
Pour illustrer ces propos, nous avons décidé de réaliser des attaques DPA sur un modèle algorithmique de l'AES. Ce modèle correspond aux valeurs intermédiaires et aux instants d'occurrences définis dans l'annexe C1 du standard [68]. Sept registres sont définis : *input*, *k_sch*, *start*, *s_box*, *s_row*, *m_col* et *output*. Ces registres sont reportés sur la Figure 4.1.

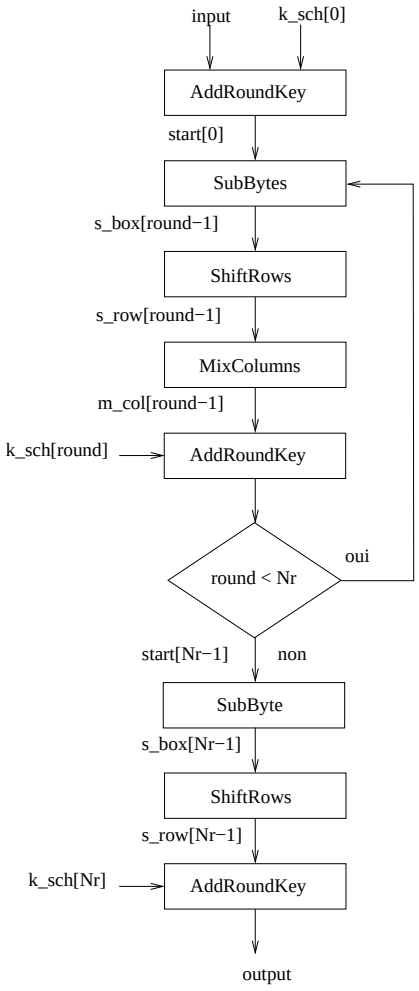

Fig. 4.1 – Définition des registres dans l'AES

Nous avons également les pas de temps suivants :
– Pour la première ronde : $t_{input} = 0$, $t_{k\_sch} = 1$
– Pour la ronde $1 < round < 10$:
  – $t_{start} = 5 * (round - 1) + 2$

- $t_{s\_box} = 5 * (round - 1) + 3$
- $t_{s\_row} = 5 * (round - 1) + 4$
- $t_{m\_col} = 5 * (round - 1) + 5$
- $t_{k\_sch} = 5 * (round - 1) + 6$

– Pour la dernière ronde : $t_{start} = 47$, $t_{s\_box} = 48$, $t_{s\_row} = 49$, $t_{k\_sch} = 50$ et $t_{output} = 51$.

Le modèle de consommation choisi est le poids de Hamming du premier bit du registre $s\_box[0]$ et les traces de courant simulées correspondent à la somme des poids de Hamming des 7 registres définis ci dessus.

La Figure 4.2 représente les courbes DPA calculées pendant les 2 premières rondes pour la clef `2B 7E 15 16 28 AE D2 A6 AB F7 15 88 09 CF 4F 3C`. Les textes clairs sont choisis de manière à ne faire varier que le premier octet. Une et une seule courbe ressort parmi les 256 courbes DPA. Cette courbe correspond à la clef $43_d = 2B_h$. On peut constater la présence de deux pics sur cette courbe qui correspondent aux calculs de s_box et s_row.

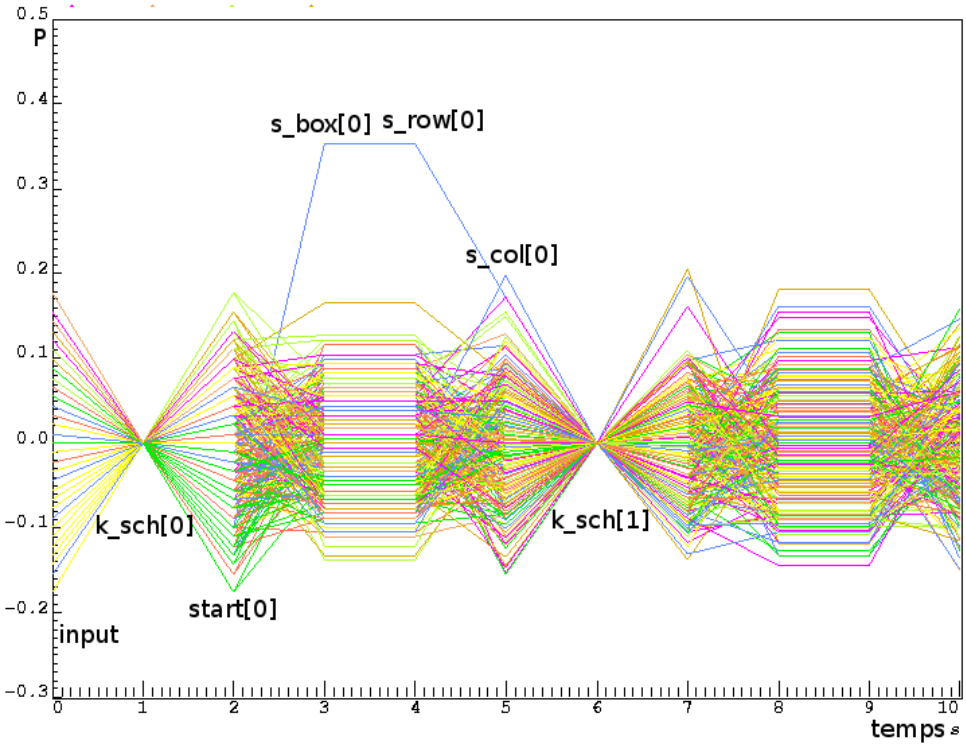

Fig. 4.2 – Courbes DPA sur le premier bit de $s\_box[0]$

Il a été démontré, ci-dessus, qu'il existe des bits corrélés aux bits d'attaques. Si la recherche de ces bits peut se faire *à la main* dans un modèle algorithmique, il en est tout autrement pour des modèles plus précis comme ceux décrits au niveau RTL ou au niveau portes. En effet, plus on descend dans la hiérarchie, plus les modèles deviennent complexes et donc la recherche de corrélations est quasi impossible si on n'est pas doté d'outils spécifiques.

C'est dans ce contexte que nous allons proposer un outil d'analyse de corrélations qui permet de détecter les bits corrélés aux bits d'attaques, qui peuvent être prédits par l'attaquant, mais aussi de déterminer de nouveaux bits d'attaques non connus dans la littérature.

4.3 Analyse de corrélations

Dans un premier temps, un formalisme décrivant de manière unique les différentes attaques par corrélation (DPA multi et monobits, CPA, DBA, micro-sondage, etc..) est proposé. Il souligne l'existence d'un espace (dont la taille varie en fonction des attaques) dans lequel des corrélations entre la clef qui est utilisée par le circuit (ou son modèle) et celles qui peuvent être prédites par l'attaquant sont calculées. Nous montrons que la valeur de ces corrélations, en absolue ou relativement aux autres, ont servi dans la littérature comme indicateurs du niveau de sécurité d'un circuit. Nous reformulerons ces critères dans le formalisme proposé et proposerons des indicateurs à appliquer spécifiquement aux modèles manipulés dans le flot de conception.

4.3.1 Formalisme

Les attaques par corrélation consistent à corréler, pour différents textes clairs, des mesures réalisées sur un composant utilisant une clef inconnue avec un modèle d'émission de syndrome (puissance, EM, résultat en présence de faute, etc...) paramétrés par des valeurs de clefs partielles. La corrélation maximale est obtenue pour la clef partielle correspondant à la clef effectivement utilisée pour réaliser les calculs. Cette recherche de corrélation se fait dans un espace qui est formellement indépendant du contexte de l'attaque ou de la simulation de celle-ci. La terminologie associée au formalisme proposé est décrite ci-dessous.

4.3.1.1 Circuit et modèle

M désigne le circuit dont on veut estimer la sécurité. Il s'agit soit un circuit "réel" soit un modèle de ce circuit. Dans ce cas, M est typiquement la description RTL ou porte (avec ou sans prise en compte des fichiers de rétro-annotation temporelle). Le modèle prédit par l'attaquant sera quant à lui noté M'. Dans le cadre des attaques DPA proposées dans la littérature, M' est la description algorithmique définie dans les standards mais dans le cadre de l'évaluation a priori, M' peut être plus précis (c'est-à-dire que l'attaquant possède plus d'information sur le circuit ou qu'il réalise des hypothèses sur ce dernier). Il pourra donc également s'agir d'une description RTL ou porte.

4.3.1.2 Signaux

Les circuits réels ou leurs modèles modifient au cours du temps, des données appelées "variables" ou "signaux" respectivement dans la terminologie logicielle ou matérielle (ces termes seront par la suite utilisés indifféremment). Pour les premiers, les signaux sont constitués de l'ensemble des équipotentielles internes (accessibles uniquement par micro-sondage ou par microscopie à contraste de potentiel). Pour les seconds, la nature des signaux dépendent du niveau d'abstraction du modèle du circuit considéré. Il s'agit, par

exemple, des entrées/sorties des registres pour les modèles RTL, et des entrées/sorties des portes logiques (dont celles de mémorisation) pour les modèles "portes". Dans les deux cas, les signaux étant fonction de M, du texte clair t_i, de la clef k et du temps δ, ils sont notés $r^M(t_i, k, \delta)$.

4.3.1.3 Syndromes

Les syndromes désignent les phénomènes physiques engendrés par l'évolution des signaux (tels que définis précédemment) et mesurés (ou prédits) par l'attaquant. En d'autres termes, les syndromes sont des images des signaux récoltées (ou prédites) par l'attaquant ; Ces images dépendent donc grandement à la fois du circuit analysé mais également des mesures effectuées. Par exemple, si l'attaque est réalisée par un micro-sondage, l'image est très proche du signal et peut donc être assimilée à la transformation "identité". Les syndromes sont aussi calculés sur des modèles. Les plus populaires sont :

- le poids ou la distance de Hamming dans le cas la mesure de puissance ($S = \{HD; HW\}$)
- la distance de Hamming dans le cas de la mesure de rayonnement électromagnétique ($S = \{HD\}$)

A noter que le poids de Hamming d'un signal binaire pris séparément est égal au signal lui même. Le syndrome associé est donc l'identité. Enfin, il est donc important de noter que le terme "signal" utilisé dans le chapitre précédent correspond au syndrome émis par l'ensemble des "signaux" définis dans le paragraphe ci-dessus.

4.3.1.4 Résumé

A l'aide des définitions ci-dessous, les analyse de corrélations du circuit ou du modèle M par rapport au modèle M' mettent en jeu les paramètres suivants :

- T l'ensemble des textes clairs t_i.
- K l'ensemble des clefs de chiffrements k_q utilisées par M pour chiffrer t_i.
- K' l'ensemble des clefs de chiffrements k'_p utilisées par M' pour chiffrer t_i.
- R^M l'ensemble des signaux r^M du modèle M ou des mesures de C.
- $R^{M'}$ l'ensemble des signaux $r^{M'}$ du circuit ou du modèle M'.
- S syndromes émis par les signaux du modèle M ou des mesures de C.
- S' syndromes émis par les signaux du modèle M'.
- ζ l'ensemble des instants de calcul δ des signaux de R^M.
- ζ' l'ensemble des instants de calcul δ' des signaux de $R^{M'}$.
- $r^M(t, k, \delta)$ valeur du signal r^M pour un texte clair $t_i \in T$, une clef de chiffrement $k_q \in K$ à un instant de calcul $\delta \in \zeta$.
- $r^{M'}(t, k', \delta')$ valeur du signal $r^{M'}$ pour un texte clair $t_i \in T$, une clef de chiffrement $k'_p \in K'$ à un instant de calcul $\delta' \in \zeta'$.
- $OCorr$: opérateur qui calcule la corrélation entre les vecteurs R^M et $R^{M'}$. De type, coefficient de Pearson (noté CP), différence des moyennes noté (DM) ou information mutuelle noté (IM).

4.3.2 Espace de recherche

L'algorithme 6 permet de calculer les corrélations maximales entre la clef utilisée par le circuit (ou son modèle) et celles qui peuvent être prédites par l'attaquant, pour toutes les valeurs de paramètres décrits ci-dessus.

Nous montrons dans les paragraphes suivants, sur quelques exemples, que les attaques

Algorithme 6 Recherche exhaustive (ie. pour toutes les valeurs de paramètres possibles) des corrélations entre la clef utilisée par le circuit (ou son modèle) et celles qui peuvent être prédites par l'attaquant

Entrées: Un ensemble de textes clairs T
Entrées: Un ensemble de pas de calcul $\delta \in \zeta$
Entrées: Un ensemble de clefs K
Entrées: Un ensemble de signaux R^M définis dans M
Entrées: Un syndrome S pour les émissions des signaux r^M
Entrées: Un ensemble de pas de calcul $\delta' \in \zeta'$
Entrées: Un ensemble de clefs K'
Entrées: Un signal $r^{M'}$ définis dans M'
Entrées: Un syndrome S' pour les émissions des signaux $r^{M'}$
Entrées: Un opérateur de corrélation $OCorr$
Sorties: Une matrice $Corr_{T;S;S'}$ de dimension 6 (6D)

{ Simulation de tous les syndromes émis par les signaux de M, pour tous les pas de temps et pour toutes les clefs}
pour $t \in T$, $r^M \in R^M$, $k \in K$,$\delta \in \zeta$ **faire**
  Calculer $S(r^M(t,k,\delta))$
fin pour
{ Simulation des syndromes émis par les signaux de M', pour tous les pas de temps et pour toutes les clefs}
pour $t \in T$, $r^{M'} \in R^{M'}$, $k' \in K'$,$\delta' \in \zeta'$ **faire**
  Calculer $S(r^{M'}(t,k',\delta'))$
fin pour
{Calculs des corrélations pour toutes les valeurs de paramètres}
pour $k \in K$ **faire**
  pour $k' \in K'$ **faire**
    pour $\delta \in \zeta$, $r^M \in R^M$, $k' \in K'$,$\delta' \in \zeta'$, $r^{M'} \in R^{M'}$ **faire**
      Calculer $Corr_{T;S;S'}(r^M,k,\delta,r^{M'},k',\delta') = OCorr_{t\in T}(S(r^M(t,k,\delta)),S'(r^{M'}(t,k',\delta')))$
    fin pour
  fin pour
fin pour
Retourner $Corr_{T;S;S'}(r^M,k,\delta,r^{M'},k',\delta')$

par corrélation se réduisent toutes à l'extraction d'information dans la matrice $Corr_{T;S;S'}$ (avec évidement des valeurs de paramètres différents). Calculer celle-ci, si $|X|$ désigne le nombre d'éléments de l'ensemble X et si l'opérateur de corrélation réalise pour chaque calcul $a * |T|$ opérations, avec a une constante (ce qui est le cas pour les opérateurs

classiques), nécessite, à une constante multiplicative près, $|Op|$ opérations avec :

$$|Op| \quad = \quad |T| * |K| * |K'| * |\zeta| * |\zeta'| * |R^M| * |R^{M'}|$$

Nous montrons également que la sécurité est généralement estimée à partir des éléments de cette matrice.

4.3.3 Exemple d'attaques

Dans le cadre des attaques, l'ensemble K est la clef qui est inconnu donc K est réduit à un singleton $\{k_0\}$.

4.3.3.1 Exemple : CPA power et EM

Conformément aux notations précédentes, les paramètres choisis pour une attaque par corrélation de type CPA sur la S-Box N sont typiquement les suivants :

- T : Ensemble des textes qui parcourent l'entrée de la S-Box N ($|T| = 256$ pour l'AES).
- R^M : Courant consommé par le circuit, noté $r_0 \in \mathcal{R}$, ($|R^M| = 1$).
- K : Clef utilisée par le composant pour chiffrer les textes T, noté k_0 ($|K| = 1$).
- S : Fonction "Identité" si aucun traitement n'est fait sur le signal r_0($|S| = 1$).
- ζ : Instants d'échantillonnage des courbes de courant (typiquement $|\zeta| = 10^3$).
- K' : Ensemble des hypothèses de clefs partielles qui parcourent l'entrée de la S-Box N ($|K'| = 256$).
- $R^{M'}$: Ensemble des bits en sorties de la S-Box N ($|R^{M'}| = 8$).
- S' : Poids de Hamming du bit en sortie de la S-Box, noté HW ($|S'| = 1$).
- ζ' : Instant de calcul de la sortie des S-Box, noté δ'_0 ($|\zeta'| = 1$).
- $Corr$: La fonction de corrélation est le coefficient de Pearson.

Pour un bit d'attaque donné, l'attaquant applique alors l'algorithme 6 avec ces différents paramètres, puis recherche la clef et le temps tels que la fonction $Corr_{T,Id,HW,CP}(r_0, k_0, \delta, SB_N(j), k', \delta'_0)$ soit maximale dans un espace à analyser de $2^8 * 2^8 * 2^3 * 10^3 = 5,2 * 10^8$ éléments. Généralement, cette recherche se fait visuellement en scindant cette fonction en sous-fonctions paramétrées par la clef partielle k'. L'attaquant obtient alors $|K|$ fonctions $Corr^{k'}_{T,Id,HW,CP}(r_0, k_0, \delta, SB_N(j), \delta'_0)$ qui ne dépendent que de δ (les autres valeurs étant constantes). Il trace alors les points $\{\delta; Corr^{k'}_{T,Id,HW,CP}(r_0, k_0, \delta, SB_N(j), \delta'_0)\}$ sur un même graphe pour toutes les clefs partielles et visuellement retrouve k et δ qui maximisent cette corrélation.
A noter que si l'attaque est réalisée sur p bits, l'ensemble des combinaisons à p éléments parmi 8 éléments devra être considéré et le poids de Hamming pour ces bits devra être calculé, l'algorithme de recherche de maximum de corrélation restant strictement identique. Il en est de même, pour les attaques de type "électromagnétique". Dans ce cas, l'ensemble des signaux R^M à prendre en considération sont ceux qui sont mesurés à chaque position (X, Y) de la sonde ($|R^M| = 100 * 100$ typiquement). Ce cardinal peut encore augmenter si différentes sondes sont utilisées (mesurant un champ électrique ou magnétique avec orientation de la sonde verticale ou horizontale etc ...).

4.3.3.2 DBA et micro-sondage

Conformément aux notations précédentes, les paramètres choisis pour une attaque par corrélation de type DBA (dans le cas où l'injection de la faute se fait avec un laser dont l'impulsion a lieu à un instant donné et à un endroit donné) sur la S-Box N sont typiquement les suivants :

- T : Ensemble des textes qui parcourent l'entrée de la S-Box N ($|T| = 256$).
- R^M : Résultat chiffré faux ou non, noté $r_0 \in \{0, 1\}$, ($|R^M| = 1$).
- K : Clef utilisée par le composant pour chiffrer les textes T, noté k_0 ($|K| = 1$).
- S : Fonction "Identité" car aucun traitement n'est fait sur le signal r_0 ($|S| = 1$).
- ζ : Instant de récupération du chiffré , noté δ_0 ($|\zeta| = 1$).
- K' : Ensemble des textes qui parcourent l'entrée de la S-Box N ($|K'| = 256$).
- $R^{M'}$: Ensemble des bits en sorties de la S-Box N, noté $SB_N(j)$ ($|R^{M'}| = 8$).
- S' : Poids de Hamming du bit en sortie de la S-Box, noté HW ($|S'| = 1$).
- ζ' : Instant de calcul de la sortie des S-Box, noté δ'_0 ($|\zeta'| = 1$).
- $Corr$: La fonction de corrélation est le coefficient de Pearson, noté CP.

Pour un instant et une position donnée du laser, l'attaquant applique l'algorithme 6 avec ces différents paramètres, puis recherche la clef telle que la fonction $Corr_{T,Id,HW,CP}(r_0, k_0, \delta_0, SB_N(j), k', \delta_0)$ soit maximale dans un espace à analyser de $2^8 * 2^8 * 2^3 = 5.2 * 10^5$ éléments. Dans l'article [60], cette recherche se fait visuellement en scindant cette fonction en sous-fonctions paramétrées par un bit $SB_N(j)$ de $R^{M'}$. L'attaquant obtient alors $|R^{M'}|$ fonctions $Corr^{SB_N(j)}_{T,Id,HW,CP}(r_0, k_0, \delta_0, k', \delta'_0)$ qui ne dépendent que de k' (les autres valeurs étant constantes). Il trace alors les points $\{k'; Corr^{SB_N(j)}_{T,Id,HW,CP}(r_0, k_0, \delta_0, k', \delta'_0)\}$ sur un graphe pour tous les bits en sorties de la S-Box et visuellement retrouve k et $SB_N(j)$ qui maximisent cette corrélation.
A noter que l'ensemble ζ peut être utilisé pour prendre en considération différents instants d'impulsion du laser. De même, si ce laser bouge par rapport au circuit, l'ensemble des signaux R^M seront ceux mesurés à chaque position (X, Y) du laser ($|R^M| = 100 * 100$ typiquement).
Dans le cas du microsondage, les signaux R^M dépendent du nombre de sondages réalisés et leur valeur évolue au cours du temps ($|\zeta| = 10^4$ typiquement).

4.3.3.3 Résumé

Dans le Tableau ci-dessous est indiqué la complexité calculée pour attaquer une S-Box d'un AES pour les différentes attaques :

- T : Ensemble des textes clairs. Soit l'attaquant utilisent des textes aléatoires (typiquement de 10^2 à 10^7) soit il utilise ceux qui parcourent l'entrée de la S-Box considérée soit 256 pour l'AES. Donc $|T|$ varie de 256 à une dizaine de millions.
- R^M : Ensemble des signaux analysés par l'attaquant. Égal à 1 dans le cas de la DPA et jusqu'à une centaine voire une dizaine de millier pour les attaques EM. Pour la DBA, le nombre de signaux dépend du degré de liberté qu'a l'attaquant sur l'injection de faute (spatiale et temporelle). Si la génération de faute n'est réalisée qu'avec un signal global (modification de la tension d'alimentation ou de la fréquence d'horloge), seul le degré de liberté associé au temps est maîtrisé. Pour les attaques en probing, en pratique moins d'une dizaine de signaux peuvent être

récupérés. Mais prospectivement, on peut supposer qu'avec des méthodes avancées (EM ultra précise ou microscope à contraste de potentiel, AFM, etc...), l'attaquant puisse mesurer l'ensemble des signaux internes du circuit qui sont de l'ordre de 10^5 pour un cryptoproceseur à clef privée.
- K : L'ensemble des clefs utilisées par le composant pour chiffrer les textes $|T|$. Il est constitué d'un unique élément pour les attaques.
- S : Dans le cas des attaques, il n'y a pas de transformation qui modifient les signaux.
- ζ : Instants d'échantillonnage, typiquement de $|\zeta| = 10^2$ à $|\zeta| = 10^4$.
- K' : Ensemble des textes qui parcourent l'entrée de la S-Box N ($|K'| = 256$).
- $R^{M'}$: Ensemble des bits en sorties de la S-Box N ($|R^{M'}| = 8$).
- S' : Les fonctions utilisées pour prédire les signaux sont en pratique les éléments $\{HW, HD\}$. Certaines fonctions prenant en compte le poids de Hamming de l'ensemble des combinaisons à au plus 8 éléments font que le cardinal de S' peut attendre 255. Dans le cas du probing, le multi-bit n'est pas à prendre en considération, seul la fonction HW est utilisée.
- ζ' : L'instant de calcul des bits d'attaque est généralement unique ($|\zeta'| = 1$).
- $Corr$: La fonction de corrélation est le coefficient de Pearson, la différence des moyennes ou l'information mutuelle.

Analyse	$\|T\|$	$\|R^M\|$	$\|K\|$	S	$\|\zeta\|$	$\|K'\|$	$\|R^{M'}\|$	$\|S'\|$	$\|\zeta'\|$	$\|OCorr\|$	Op
DPA	2^8 - 10^7	1	1	1	10^2 - 10^4	2^8	2^3	1 - 2^8	1	3	$2*10^7$ - $2*10^{17}$
EM	2^8 - 10^7	10-10^4	1	1	10^2 - 10^4	2^8	2^3	1 - 2^8	1	3	$2*10^8$ - $2*10^{21}$
DBA (laser)	2^8 - 10^7	1-10^4	1	1	1 -10^4	2^8	2^3	1 - 2^8	1	3	$2*10^5$ - $2*10^{21}$
DBA (Clk,Vcc)	2^8 - 10^7	1	1	1	1 -10^4	2^8	2^3	1 - 2^8	1	3	$2*10^5$ - $2*10^{17}$
Probing	2^8 - 10^7	1-10^5	1	1	10^2 -10^4	2^8	2^3	1	1	3	$2*10^7$ - $5*10^{19}$
EAP	2^8 - 10^7	10^5	2^8	2	10^4	2^8	2^3-10^5	2	1-10^4	3	$2*10^{18}$ - $8*10^{30}$
EAP (opt)	2^8	10^5	1-3	2	1-10^4	2^8	2^3-10^5	1	1-10^4	1	10^{11} - $4*10^{23}$

Tab. 4.1 – Complexité d'attaques d'une S-Box de l'AES pour les attaques DPA, EM, DBA, probing et estimation a priori

4.3.3.4 Estimation de la sécurité

L'estimation de la sécurité pour les attaques par corrélation est généralement réalisée en appliquant l'algorithme 6 pour différents ensembles de textes ayant un cardinal variable (typiquemement de 100 à 10000). Pour l'attaque CPA par exemple, sur un même graphe est reporté, pour un instant δ_0 et pour une clef inconnue k_0, l'ensemble des courbes $\{|T|; Corr^{k'}_{T,Id,HW,CP}(r_0, k_0, \delta_0, SB_N(j), \delta'_0)\}$ (ensemble paramétré par k') avec $|T|$ variable. La sécurité est ensuite estimée comme le nombre minimal de textes pour lesquels la courbe $\{|T|; Corr^{k_0}_{T,Id,HW,CP}(r_0, k_0, \delta_0, SB_N(j), \delta'_0)\}$ ressort distinctement. On parle alors parfois de "nombre de courbes nécessaires à la découverte de la clef" ou "Measurments To Disclose" (MDT).
Ce type d'approche n'estime donc la sécurité que pour un ensemble réduit de paramètres. Cette approche, si elle peut être appliquée dans le cadre des attaques (où ce jeu de paramètres peut effectivement être efficace dans 90% des cas), n'est pas applicable dans le cadre de l'estimation a priori. En effet, ce genre d'analyse concerne principalement des circuits implantant des contre-mesures nouvelles qui constituent en grande partie les 10% des cas restants. Par exemple, le syndrome "HW" calculé sur plusieurs bits, s'il donne de

bons résultats sur des circuits standards, n'a pas d'efficacité réelle pour les circuits double rail (où chacun des bits manipulés a une consommation très différente des autres, en signe comme en module).

Il s'agit donc de considérer un sous-ensemble aussi important que possible de paramètres. Cette méthode, en augmentant la taille de l'espace de recherche, garanti une estimation plus précise de la sécurité mais les calculs associés et l'exploitation des résultats sont plus longs. Il faut également s'assurer de l'existence du maximum de corrélation dans ce sous-ensemble de paramètres. Cette discussion est abordée dans le paragraphe suivant.

4.3.4 Position du problème

Dans le cadre de l'estimation a priori de la sécurité ("EAP"), la discussion du chapitre précédent a montré qu'il était pertinent de supposer que l'attaquant est capable de mesurer les signaux au cours du temps indépendamment les uns des autres. D'où R^M de l'ordre de 10^5 et $|\zeta|$ de l'ordre de 10^4. On souhaite que les outils proposés puissent être utilisés, non seulement dans le cas classique où l'attaquant ne dispose que d'une description algorithmique du circuit, mais également dans le cas où l'attaquant est capable de prédire l'ensemble de ces équipotentielles et leur variation temporelle ($M = M'$). Ces outils pourront donc être utilisés en fonction de la force estimée de l'attaquant. Ceci explique les chiffres de la ligne "EAP" du Tableau 4.1. La complexité de l'EAP (en terme d'opérations à calculer) est au moins supérieure aux attaques les plus complexes (micro-sondage sur l'ensemble des équipotentielles du circuit). Pour être réalisable en pratique, l'espace de recherche doit être réduit. Les paragraphes suivants expliquent les hypothèses réalisées à cet effet.

4.3.4.1 Textes clairs, T

Nous considérons par la suite que l'attaquant est susceptible de pouvoir choisir les textes clairs et, en particulier, ceux qui maximisent la corrélation entre les calculs internes des algorithmes de cryptographies. Ceux-ci appliquent très largement l'opérateur "ou exlusif" (`XOR`) entre deux bits (pour l'AES dans les fonctions `AddRoundKey` et `MixColumns`). Or, comme expliqué dans le paragraphe 4.2.1, les entrées et les sorties de ces fonctions sont corrélées (ie, la sortie est identique ou opposée à l'entrée) lorsque l'une des deux entrées est constante. Par conséquent, l'ensemble des textes choisis pour attaquer une S-Box sera l'ensemble des textes qui parcourent ses entrées, les autres bits restant constants (choisis tous identiques et arbitrairement nuls). Dans ces conditions, $|T| = 256$.

4.3.4.2 Clefs K

Nous considérons que les corrélations entre les signaux mesurés par l'attaquant et les prédictions de ce dernier sont indépendantes de la clef. Cette hypothèse, qui semble assez forte, est implicitement admise dans le cas des attaques. En effet, il est communément admis que celles-ci fonctionnent indépendamment de la clef utilisée, même si parfois les résultats obtenus peuvent légèrement différer. Pour s'assurer de la validité de cette hypothèse, les analyses sont par défaut réalisés avec 2 ou 3 clefs distinctes. D'où $|K| = 2$ ou $|K| = 3$.

4.3.4.3 Syndromes S

Comme expliqué dans la section 3.5, chaque signal est supposé émettre un syndrome indépendamment des autres signaux. Le syndrome pris en considération dépend des moyens de l'attaquant. En effet, celui-ci mesure une grandeur sensiblement égale :

- à un poids ou une distance de Hamming dans le cas la mesure de puissance ($S = \{HD; HW\}$)
- à une distance de Hamming dans le cas de la mesure de rayonnement électromagnétique ($S = \{HD\}$)
- à un poids Hamming dans le cas du micro-sondage (directe ou indirecte par injection de faute) ($S = \{HW\}$)

Pour prendre en compte ces différents type d'attaque, les syndromes distance et poids de Hamming sont considérés par la suite. On a donc $S = \{HD; HW\}$

4.3.4.4 Syndromes S'

Nous supposerons que l'attaquant ne prédit que les poids de Hamming des signaux $R^{M'}$ (la prédiction de la distance nécessitant la connaissance de ces derniers à deux instants distincts), soit $S' = \{HW\}$. A noter que la prise en compte d'autres syndromes pour S et S' (telles que les transitions montantes ou descendantes) ne nécessite aucune modification de la méthode proposée dans le cadre de cette thèse. Elle ne fait qu'augmenter la taille de l'espace de recherche.

4.3.4.5 Bases de temps

Les instants de calculs des valeurs de r^M et de $r^{M'}$ dépendent du niveau d'abstraction considéré. Pour le niveau algorithmique, un instant de calcul (dont l'indice est choisi arbitrairement) correspond à la modification d'une des variables de l'algorithme. Un exemple de choix arbitraire de la base de temps a été proposé pour l'AES dans le paragraphe 4.2.2.2. On a, dans ce cas, $|\zeta|$ ou $|\zeta'|$ de l'ordre de l'unité (si l'on ne considère qu'une ronde) à la dizaine d'éléments. Au niveau RTL, les signaux sont stables avant chaque front (montant ou descendant suivant le style de logique utilisée) d'horloge. Enfin, au niveau "portes", deux hypothèses sont envisagées : La première consiste à n'échantillonner que les valeurs stabilisées des signaux. Dans ce cas, la base de temps correspond aux instants qui précèdent les fronts montants d'horloge (comme c'est le cas pour le niveau RTL). On a dans ce cas $|\zeta|$ ou $|\zeta'|$ de l'ordre de la centaine d'éléments. La seconde consiste à enregistrer la trace complète du signal. Dans ce cas, la base de temps est celle du simulateur utilisé et est paramétrable par le concepteur.

4.3.4.6 Opérateur de corrélation, $OCorr$

Dans le cas particulier où un signal unique est corrélé à une valeur binaire, les trois opérateurs de corrélation (différence des moyennes, MIA et Pearson) donnent des résultats très proches. Nous n'en retiendrons donc qu'un : le coefficient du Pearson car il est le plus utilisé. Pour deux vecteurs a et b de n bits notés respectivement $a(i)$ et $b(i)$, le coefficient de Pearson est calculés grâce aux formules suivantes.

La moyenne d'un vecteur v est :

$$\mu_v = \frac{1}{n} * \sum_{i=1}^{n} v(i)$$

Son écart-type est :

$$\sigma_v = \sqrt{\sum_{i=1}^{n} (v(i) - \mu_v)^2}$$

Le coefficient de Pearson entre deux vecteurs a et b est donnée par la formule suivante :

$$\Delta(a,b) = \frac{\sum_{i=1}^{n} (a(i) - \mu_a) \times (b(i) - \mu_b)}{\sigma_a \times \sigma_b}$$

4.4 Réduction de l'analyse et définitions associées

La complexité des calculs nécessaires à l'EAP avec les hypothèses décrites ci-dessous est reportée dans la dernière ligne du Tableau 4.1. Cette valeur restant encore élevée dans le cas général ($4 * 10^{23}$), la recherche des corrélations ne peut pas se faire par une méthode de type "force brute" (ie. en calculant toutes les corrélations pour tous les paramètres). Pour illustrer l'approche simplificatrice proposée, sur la Figure 4.3 sont représentés les résultats des calculs de $Corr_{T;S;S'}(r^M, k_0, \delta_0, r^{M'}, k', \delta_0)$ pour une clef k_0 de M et pour des pas de temps prédits δ'_0 et simulés δ_0 donnés, dans les trois dimensions suivantes : $r^M \in R^M$, $r^{M'} \in r^{M'}$ et $k' \in K'$.

4.4.1 Bits corrélés

Dans cet espace, soit le plan (délimité en rouge) $\phi = (r^M \in R^M, r^{M'} \in R^{M'}, k' = k_0)$, qui est l'ensemble des valeurs de corrélation entre les bits de R^M et de $R^{M'}$ pour la bonne clef et soit un point quelconque $\gamma = (r_0, r'_0, k_0)$ de ce plan. Si la valeur de corrélation en ce point est égale à 1, le concepteur peut en déduire que r'_0 "prédit" parfaitement la valeur de r_0. En d'autres termes, ces deux bits sont parfaitement "corrélés". Plus formellement, s'il existe des étapes de calcul δ_0 et δ'_0, un ensemble de textes clairs T et des syndromes S et S' tels que $Correl_{T,S,S'}(r^M, k_0, \delta_0, r^{M'}, k' = k_0, \delta'_0)$ est égale à 1 (resp. -1) pour tout $k_0 \in K$, alors les bits r^M et $r^{M'}$ sont dits *parfaitement corrélés* (resp. *inversement corrélés*). Si la corrélation est supérieure à un seuil (typiquement 0,5), on parlera de bits *partiellement corrélés*. Par abus de langage, les bits parfaitement corrélés ou inversement corrélés sont appelés bits corrélés.

Exemple 1 Par exemple, tous les bits cités dans le paragraphe 4.2.2.1 (ie., L_2, R_1) pour le DES sont corrélés à un bit en sortie des S-Boxes.

Exemple 2 Soit M la description algorithmique de l'AES décrite dans le paragraphe 4.2.2.2. Si on considère que M' est identique à M, que R^M est l'ensemble des bits du registre d'entrée $Input$ ($r^M = Input(i)$ désignant le $i^{ième}$ bit de ce registre), si $R^{M'}$ est l'ensemble des bits du registre de début de ronde $Start$ ($r^{M'} = Start(i)$ désignant le $i^{ième}$

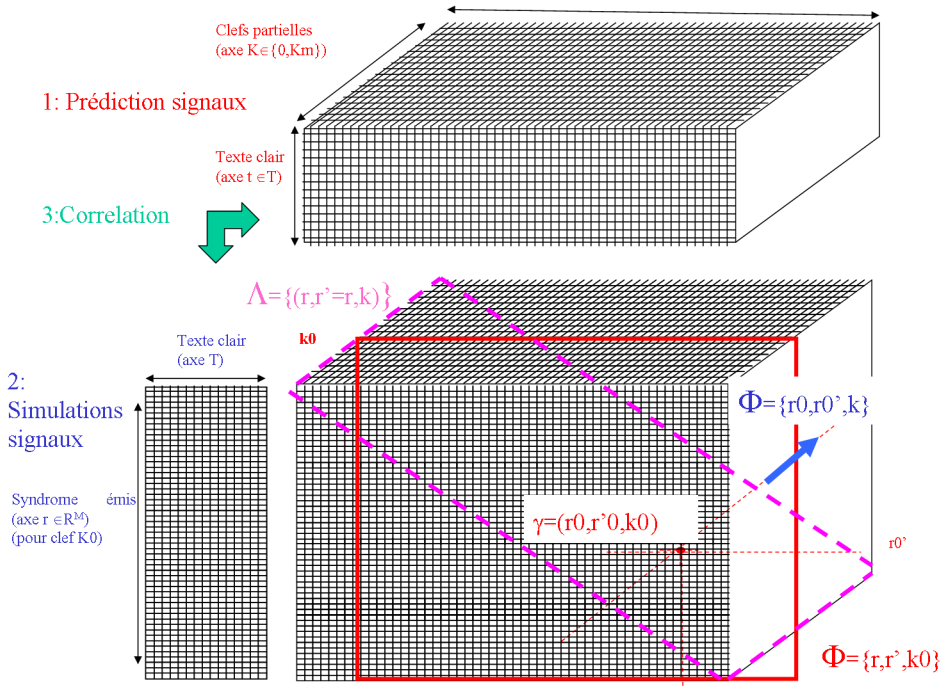

FIG. 4.3 – Calcul des signaux de R^M pour un ensemble de clefs K, un ensemble de textes T et à un instant δ

bit de ce registre) et que $\delta_0 = \delta'_0$ est l'instant de calcul de la première ronde alors, quelque soit l'ensemble de textes clairs T, les syndromes tels que $S = S'$ et la valeur de l'indice i, la valeur $Correl_{T,S,S'}(Input(i), k, \delta_0, Start(i), k' = k, \delta_0)$ est égale à 1 si $k\_sch(i) = 0$ et est égale à -1 si $k\_sch(i) = 1$. Par conséquent, les bits $Input(i)$ et $Start(i)$ sont parfaitement ou inversement corrélés.

4.4.2 Bits d'attaque, bits sensibles

Si l'on considère maintenant le vecteur en bleu $\Theta = (r^M = r_0, r^{M'} = r'_0, k' \in K')$ qui est l'ensemble des corrélations obtenues pour les sous-clefs partielles avec r_0 le signal mesuré et r'_0 le bit prédit. Ce vecteur correspondrait aux valeurs de corrélation obtenues pour les différentes clefs partielles en un point d'une courbe "DPA". Si r_0 et r'_0 sont parfaitement corrélés et que pour toute valeur de $k' \neq k_0$ la corrélation est inférieure à 1, on parle généralement de pic permettant à l'attaquant de déduire des informations sur la clef. Dans ce cas, le bit r_0 est appelé "bit sensible" et r'_0 est appelé "bit d'attaque". Plus formellement, s'il existe des étapes de calcul δ_0 et δ'_0, un ensemble de textes clairs T, des syndromes S et S', une clef k_0 et un bit r^M tels que $|Correl_{T,S,S'}(r^M, k_0, \delta_0, r^{M'}, k_0, \delta'_0)| > |Correl_{T,S,S'}(r^M, k_0, \delta_0, r^{M'}, k', \delta'_0)|$ pour tout $k' \neq k_0 \in K'$, alors le bit $r^{M'}$ est dit *bit d'attaque* du modèle M' envers le modèle M et r^M est dit *bit sensible* du modèle M sachant M'. Cette définition peut s'appliquer également au cas où les bits r_0 et r'_0 sont partiellement corrélés.

Exemple Soit M la description algorithmique de l'AES décrite dans le paragraphe 4.2.2.2. Si on considère que M' est identique à M, que $R^M = R^{M'}$ est l'ensemble des bits en sortie de la S-Box N ($r^M = SBox_N(i)$ désignant le $i^{ième}$ bit de ce registre) et que $\delta_0 = \delta'_0$ est l'instant de calcul de la première ronde alors, pour l'ensemble des textes clairs T qui parcourent cette S-Box, pour les syndromes $S = S' = HW$ et pour tout indice i, on a $Correl_{T,S,S'}(SBox_N(i), k_0, \delta_0, SBox_N(i), k, \delta_0) < 1$ pour tout $k \neq k_0$. Par conséquent, les bits $SBox_N(i)$ sont des bits d'attaque. Ces bits d'attaque sont les plus largement cités dans la littérature [18], [48],[54],[25]. Par contre, si $r^M = Start(i)$, on a $Correl_{T,S,S'}(Start(i), k_0, \delta_0, Start(i), k, \delta_0) = \pm 1$ pour tout k et tout i. Les bits $Start(i)$ ne sont donc pas des bits d'attaque : leur prédiction ne réduit pas l'ensemble des clefs de chiffrement possibles donc ne fournit pas d'information sur celle qui est effectivement utilisée.

4.4.3 Recherche des bits sensibles

Nous considérons par la suite que le nombre de bits sensibles associés à un ensemble connu de bits d'attaque (typiquement les sorties des S-Boxes) est un indicateur du niveau de sécurité du circuit considéré. Moins il y en aura, plus le circuit sera sûr. Pour rechercher ces bits sensibles à des bits d'attaque connus, l'approche proposée (approchée car elle évite de calculer la matrice complète) est la suivante :

- Calculer les corrélations dans le plan $\phi = (r^M \in R^M, r^{M'} \in R^{M'}, k' = k_0)$ avec M' un modèle facilement accessible pour l'attaquant (typiquement une description algorithmique d'un algorithme standard de cryptographie). Dans ce modèle, des bits d'attaque sont connus (comme ceux en sortie des S-Boxes).
- Détecter dans ce plan les bits parfaitement ou partiellement corrélés avec ces bits d'attaque. Ces points seront par la suite appelés bits *potentiellement sensibles*.
- Optionnellement, pour chacun de ces bits potentiellement sensibles, calculer le vecteur $\Theta = (r^M = r_0, r^{M'} = r'_0, k' \in K')$ (avec r_0 le bit partiellement ou parfaitement corrélé au bit d'attaque r'_0). Si la corrélation en γ est supérieure à tous les autres éléments du vecteur Θ, alors r_0 est un réellement bit sensible.

En pratique et conformément à l'hypothèse justifiée dans le paragraphe 4.3.4.2, l'ensemble de ces calculs sera effectué, non pas pour toutes les clefs k_0 possibles mais uniquement deux ou trois d'entre-elles.
Cette recherche approchée est justifiée par le fait que si, avec la bonne clef, la corrélation est faible, il y a très peu de chance qu'elle soit la plus grande de celle obtenue avec les mauvaises clefs. Il y a donc peu de chances pour que cette recherche ne permette pas de déterminer l'ensemble des bits sensibles de M.

4.4.4 Recherche des bits d'attaque

Nous considérons également que définir la sécurité d'un circuit par rapport aux seuls bits d'attaques décrits dans la littérature est insuffisant. Il convient, en plus, de lister les bits d'attaque que l'attaquant est susceptible d'exploiter s'il possède une description plus précise que celle d'un standard (en se basant, par exemple, sur les architectures matérielles décrites dans des articles académiques ou sur de quelconques fuites d'information). Pour lister ces bits d'attaque, nous plaçons le concepteur dans le pire des cas, à savoir

celui où le modèle M' est le modèle du circuit lui-même ($M = M'$). Pour rechercher ces bits d'attaque, le calcul de la matrice complète ne peut pas être évitée. La recherche sera cependant réalisée uniquement dans un sous-ensemble des bits de M et conformément à l'hypothèse justifiée dans le paragraphe 4.3.4.2, le calcul sera effectué, non pas pour toutes les clefs k_0 possibles, mais uniquement deux ou trois d'entre-elles.

4.5 Flot de conception sécurisé

Cette partie décrit les outils de CAO mis en œuvre pour détecter les bits sensibles et les bits d'attaques tels que définis dans la partie précédente. Il s'agit :

1. D'automatiser les chiffrements multiples des modèles M et M'. Cette automatisation a été réalisée grâce à des scripts TCL.
2. De récupérer de ces simulations les valeurs numériques des signaux r^M et $r^{M'}$ via des formats de fichier textuels puis de calculer les syndromes associés.
3. De calculer la matrice de corrélation $Corr_{T,S,S',CP}(r^M, k, \delta, r^{M'}, k', \delta')$ ou seulement certains de ces éléments (comme le plan ϕ).

4.5.1 Automatisation des simulations

Un fichier écrit en langage TCL (*Tool Command Langage*) commande le simulateur Modelsim® pour qu'il réalise toutes les simulations nécessaires aux calculs des bits sensibles et d'attaque. Ce fichier, d'extension ".do", précise les entrées des simulations (T, K et K'), les actions à réaliser (durée et pas de temps de la simulation), ainsi que le type de sorties choisies. Les commandes associées sont les suivantes :

1. Initialisation des vecteurs de test T, K et K' (commande `force`)
2. Ajout des signaux r^M et $r^{M'}$ dans une liste (commande `add list`)
3. Détermination de la durée de simulation (commande `run 100ns`)
4. Écriture des valeurs numériques des signaux $r^M(t, k, \delta)$ et $r^{M'}(t, k', \delta')$ dans une liste (commande `write list`).
5. Écriture des noms des signaux $r^M(t, k, \delta)$ et $r^{M'}(t, k', \delta')$ dans un fichier texte (commande `write format list`)

Les simulations sont menées sur les descriptions RTL ou portes. Au niveau portes, le fichier SDF (*Standard Delay Format*) qui contient les délais de propagation des noeuds et des portes logiques peut être inclu ou non suivant le niveau de précision souhaité. Dans le cas où les simulations tiennent compte du fichier SDF, les bases de temps ζ ou ζ' correspondent à la résolution imposée au simulateur. A noter que pour avoir une simulation précise qui tient compte de tous les événements qui peuvent se produire (notamment les phénomènes de glitch), la résolution imposée au simulateur doit être inférieure aux délais de propagation des portes logiques.

4.5.2 Récupération des valeurs numériques et calcul des syndromes associés

Les fichiers textes qui contiennent les listes, obtenus à la fin de la simulation présentent la forme suivante :

```
     v0     v1                    v2                          vk
p0   0   10111100    110101011101001100011101 01110001       0
p1   1   01101001    100111110001001001110100 01011001       1
p2   1   10110000    111101001101111110000111 11110000       1



pn   0   00011110    000101000000000011011100 010000011      0
```

où `pi` est le pas de temps et `vi` le nom du signal considéré.

Les valeurs des syndromes associés à chacun des signaux sont extraits à l'aide d'un script écrit en langage Perl.

4.5.3 Calcul de la matrice de corrélations

L'algorithme défini dans 6 qui permet de calculer la matrice de corrélation a été écrit en langage Ruby en intégrant une GSL [1]. Celle-ci gère les matrices et les calculs numériques comme le calcul de coefficient de Pearson. Les signaux potentiellement sensibles sont récupérés dans un fichier. Pour les modèles de circuits décrits de manière hiérarchique, des doublons peuvent apparaitre, il s'agit de signaux distincts mais qui sont reliés uniquement par un fil ; ce sont souvent les sorties de certains blocs qui forment les entrées d'autres blocs. Ces doublons sont supprimés par post-traitement.

4.6 Exemple d'application de l'analyse de corrélations

Cette partie compare la méthode d'estimation de la sécurité classique (c'est-à-dire telle que présentée dans le paragraphe 4.3.3.4) à celle proposée dans le cadre de cette thèse sur une représentation "porte" avec rétro-annotation d'un AES.
Les paramètres suivants sont utilisés pour cette comparaison :

- M : Description niveau portes de l'AES.
- M' : Description algorithmique de l'AES.
- T : Ensemble des textes clairs qui parcourent l'entrée de la première S-Box soit 256 textes clairs.
- K : Clef de chiffrement k_0 qui est égale à `2B 7E 15 16 28 AE D2 A6 AB F7 15 88 09 CF 4F 3C`.
- K' : Ensemble des 256 hypothèses de clefs partielles qui parcourent l'entrée de la première S-Box.
- $R^{M'}$: Le $4^{éme}$ bit en sortie de la première S-Box ($SB_0[4]$).
- S' : Poids de Hamming du bit $SB_0[4]$.

[1] GNU Scientific Library

– ζ' : Fin de la première ronde, noté δ'_0
– ζ : Ensemble des pas de temps ($\times 100ps$) entre la ronde 0 et la ronde 3.
– Pour la méthode classique, R^M est la distance de Hamming des signaux du chemin de données et S est la fonction identité. Pour notre méthode R^M est l'ensemble des bits du chemin de données ($|R^M| = 22932$) et S est la fonction poids de Hamming.
– La fonction de corrélation est le coefficient de Pearson.

Les S-Boxes implémentées dans l'AES utilisent l'architecture proposé par Wolkerstorfer et al. dans [71].

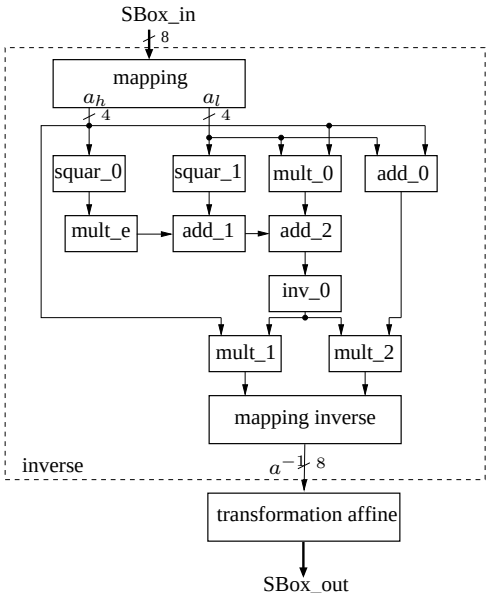

Fig. 4.4 – Architecture de l'implémentation de la S-Box

SB_0	SB_1	SB_2	SB_3
SB_4	SB_5	SB_6	SB_7
SB_8	SB_9	SB_{10}	SB_{11}
SB_{12}	SB_{13}	SB_{14}	SB_{15}

Fig. 4.5 – Numérotation des 16 S-Box de l'AES

4.6.1 Résultat de la méthode classique

Les fonctions de corrélation $Corr_{T,Id,HW,CP}(r^M, k_0, \delta, SB_0[4], k', \delta'_0)$ sont calculées pour les différentes hypothèses de clefs partielles et à tous les pas de temps. L'ensemble des points $(\delta, Corr_{T,Id,HW,CP}(r^M, k_0, \delta, SB_0[4], k', \delta'_0))$ sont reportés sur les Figures 4.6 et 4.7. Parmi les 256 courbes qui figurent sur ces graphes, la courbe bleue qui correspond à la

clef $2B_h$ présente le maximum de corrélations et ce, aux instants $301,9ns$, $302ns$, $302,1ns$, $302,4ns$, $302,5ns$, $302,6ns$, $340,7ns$, $341,1ns$, $341,2ns$, $341,3ns$, $341,7ns$ et $341,8ns$. Ces maxima montrent qu'il existe des fuites d'information durant les deux premières rondes du chiffrement de l'algorithme.

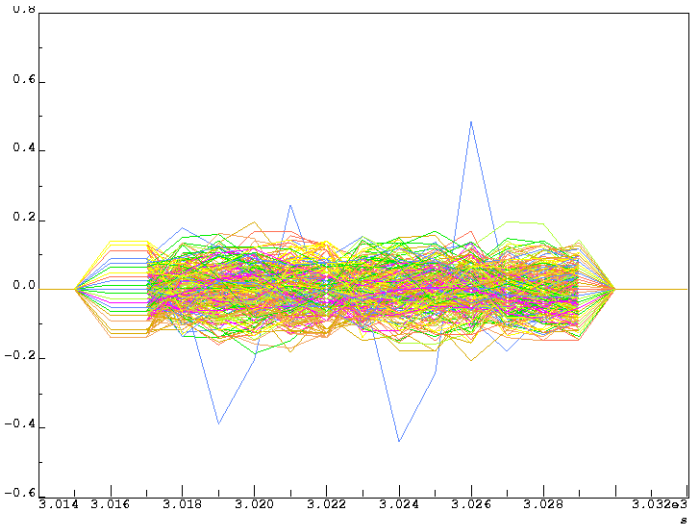

Fig. 4.6 – Courbes CPA de l'AES pendant la première ronde

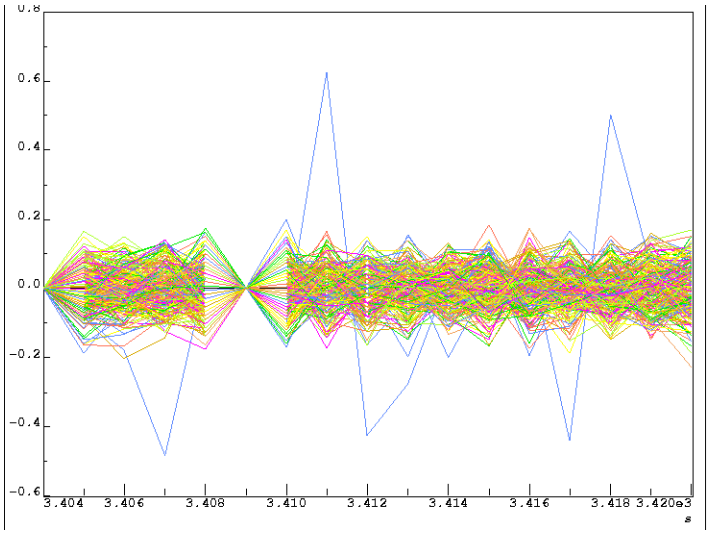

Fig. 4.7 – Courbes CPA de l'AES pendant la deuxième ronde

4.6.2 Résultat de notre méthode

Pour illustrer l'intérêt de notre approche, la recherche de bits sensibles est réalisée dans les mêmes conditions.
Dans un premier temps, les corrélations $Corr_{T,HD,HW,CP}(r^M, k_0, \delta, SB_0[4], k_0, \delta'_0)$ du plan *Phi* sont calculées pendant les trois premières rondes avec une résolution égale à $100ps$. La répartition du nombre des bits de R^M en fonction de leurs valeur de corrélation avec le bit d'attaque $SB_0[4]$ est donnée par la Figure 4.8. Il s'avère que 15 de ces bits présentent

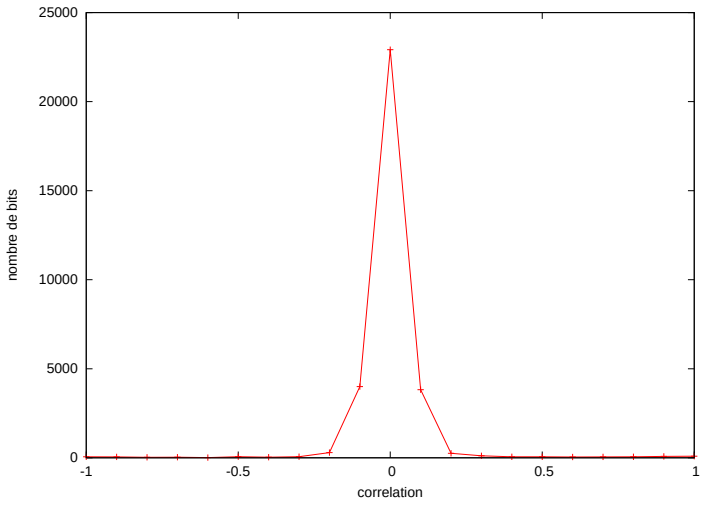

Fig. 4.8 – Répartition des bits du chemin de données en fonction de leurs corrélations

des corrélations égales à ± 1 (ils sont donc parfaitement corrélés à $SB_0[4]$) et que la valeur absolue de 10 autres bits est supérieure à 0.5 tout en restant inférieure à 1 (ils sont donc partiellement corrélés). Comme $SB_0[4]$ est un bit d'attaque, ces 25 bits sont potentiellement sensibles.
Dans un second temps, les corrélations $Corr_{T,HD,HW,CP}(r^M, k_0, \delta, SB_0[4], k', \delta'_0)$ des vecteurs Θ sont calculées pour tous ces bits aux instants où ils présentent le maximum de corrélations. La Figure 4.9 représente, par exemple, les valeurs de corrélations obtenues entre le bit $SB_4/inverse/mapping/n2$ (un des 10 bits partiellement corrélés) et le bit d'attaque $SB_0[4]$ pour les différentes hypothèses de clefs partielles à l'instant $341,2ns($ où la corrélation entre $SB_4/inverse/mapping/n2$ et le bit d'attaque est maximale). Sur cet exemple, la clef réellement utilisée pour de chiffrement $2B_h = 43_d$ présente la valeur maximale de corrélation ; le bit $SB_4/inverse/mapping/n2$ est donc sensible. Il en est de même pour les 25 autres bits déclarés potentiellement sensibles.

4.6.3 Comparaison entre les approches

L'approche proposée dans le cadre de cette thèse permet de déterminer une liste des 25 bits sensibles. Une relation entre ces derniers et les fuites visibles sur les courbes CPA peut être établie en associant, comme indiqué dans le Tableau 4.2, les instants d'apparition des pics CPA aux instants de corrélation des bits sensibles avec le bit d'attaque. On constate

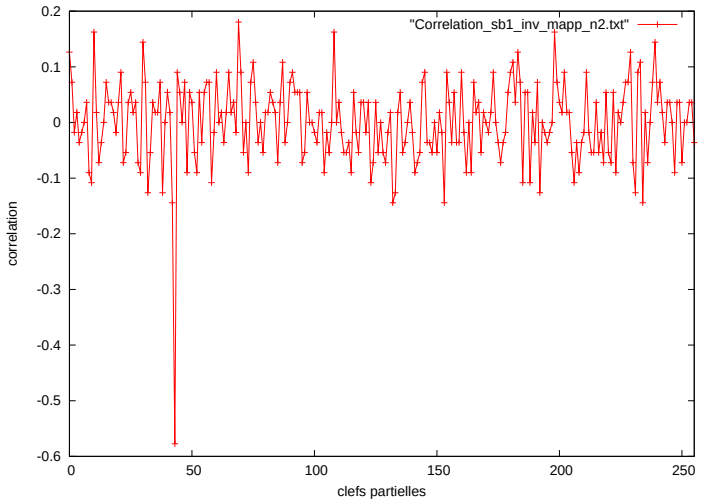

Fig. 4.9 – Valeurs de corrélation pour toutes les hypothèses de clefs partielles pour le bit $SB_4/inverse/mapping/n2$

sur ce tableau que :

- Tous les pics CPA sont associés à au moins un bit sensible. Cette première constation montre que l'approche proposée explique au moins tous les résultats de l'approche classique. Notre approche, est dans ce sens, **juste**.
- Certains pics CPA peuvent être associés à plusieurs bits sensibles. Cette constation montre que des calculs peuvent être simultanés (même avec une résolution de simulation minimale) et que donc, il est délicat pour le concepteur de déterminer précisément (c'est-à-dire au bit près) les fuites d'information à partir de l'analyse de courbes CPA seules. Notre approche est dans ce sens plus **précise**.
- Certains bits sensibles (4 sur 25) ne sont pas visibles sur les courbes CPA. Cette constatation montre que les courbes CPA ne permettent pas de lister toutes les fuites d'information. Notre approche, est dans ce sens, plus **complète**.

Tab. 4.2 – Correspondance entre les pics CPA et les bits sensibles

temps ($\times 100ps$)	pics DPA	Bit(s) sensible(s) ($Corr = \pm 1$)	Bit(s) sensible(s) ($Corr \neq \pm 1$)
3019	×	$Mux/n223$	
		Mux/191	
3020	×	$mux\_o[92]$	
		mux_o[60]	
3021	×	$AddRoundKey/n253$	
		mux_o[125]	
3024	×	$AddRoundKey/n222$	
		AddRoundKey/n190	
3025	×	$AddRoundKey/n253$	
		byte_o[92]	
		state_o[60]	
3026	×	$state\_o[125]$	
3407	×	$byte\_i[125]$	
		byte_i[60]	
		byte_i[92]	
3411	×		$SB_8/inverse/mapping/n2$
3412	×		$SB_4/inverse/mapping/n2$
3413	×		$SB_8/inverse/mapping/n2$
3417	×		$SB_8/inverse/squar\_1/al[3]$
			$SB_8/inverse/mult\_0/n1$
3418	×		$SB_4/inverse/squar\_1/al[3]$
			$SB_4/inverse/mult\_0/n1$
3421			$SB_4/inverse/add\_1/add\_in2[3]$
			$SB_8/inverse/add\_1/add\_in2[3]$
3425			$SB_4/inverse/add\_2/add\_in1[3]$
			$SB_8/inverse/add\_2/add\_in1[3]$

Pour expliquer ce dernier point, des attaques CPA sont réalisées sur l'estimation du courant du seul bloc "add_2" de la cinquième S-Box, lequel contient l'un des bits sensibles ($SB_4/inverse/add\_2/add\_in1[3]$) qui n'est pas associé à un pic CPA. L'ensemble des points $(\delta, Corr_{T,Id,HW,CP}(r^M, k_0, \delta, SB_0[4], k', \delta'_0))$ calculés pour tous les bits r^M de ce bloc "add_2" sont reportés sur la Figure 4.10. La courbe CPA qui correspond à la clef k_0 présente un pic à l'instant $342,5ns$ (lequel correspond à la manipulation du bit sensible). Ceci prouve que ce dernier contribue bien à la signature CPA. Il fuit donc de l'information quand celle-ci est extraite d'un signal où le bruit est réduit (comme pourrait le faire une analyse EM). Ce résultat est vérifié pour les 3 autres bits sensibles.

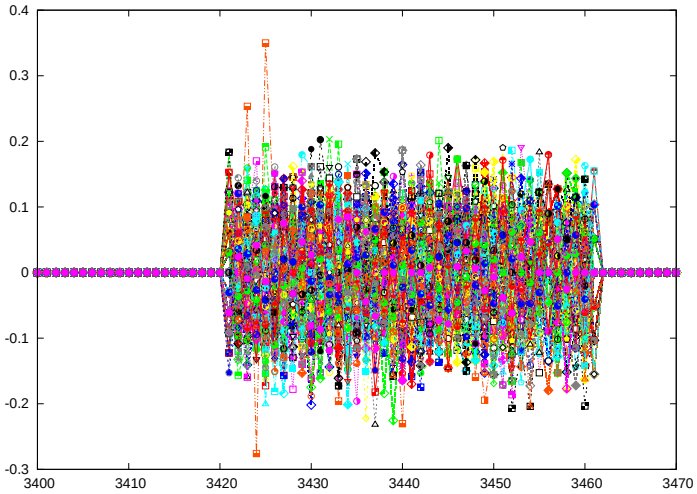

Fig. 4.10 – Courbes CPA réalisées sur SB_4

4.7 Conclusion

Dans ce chapitre, un formalisme décrivant les attaques par corrélation a été proposé. Les valeurs de corrélations issues de ce formalisme ont permis de définir des critères de sécurité. Dans la mèthode classique qui consiste à réaliser des attaques de type CPA pour évaluer la résistance d'un circuit, la sécurité est estimée comme le nombre minimal de courbes nécessaires à la découverte de la clef. Dans le cadre de l'estimation a priori de la sécurité, où chaque signal est considéré séparémment, notre critère de sécurité est basé sur le nombre de bits sensibles lesquelles sont des bits corrélés aux bits d'attaque connus mais aussi sur de nouveaux bits d'attaque non connus dans la littérature.
Notre approche a été illustrée sur une description de l'AES au niveau portes avec rétro-annotation. Les résultats de corrélations obtenus ont permis de déceler les signaux qui sont à l'origine des fuites dans l'AES. En outre, la comparaison de ces résultats avec ceux issus des attaques CPA a mis en avant les avantages de notre approche dans la mesure où elle est plus précise et plus complète.

Dans le chapitre suivant, deux applications de notre outil seront présentés : la première

permet de comparer trois implémentations différentes de l'AES et la seconde sera une évaluation de deux contre-mesures. Nous allons également présenter de nouveaux bits d'attaques dans deux implémentations différentes de la S-Box.

Chapitre 5

Évaluation d'un crypto-processeur à clef secrète : l'AES

5.1 Introduction

Dans le chapitre précédent, un flot de conception pour l'évaluation de la résistance des circuits intégrés sécurisés face aux attaques par corrélation a été proposé. Le critère de sécurité utilisé dans ce flot est basé sur le nombre de bits sensibles, et sur des nouveaux bits d'attaque. Afin de montrer l'intérêt de la recherche des bits sensibles, plusieurs modèles de l'algorithme AES avec et sans contre-mesures décrits au niveau RTL et portes sont étudiés. Par ailleurs, une recherche de nouveaux bits d'attaque est effectué dans deux implémentations différentes des boîtes de substitution.

5.2 Architectures des crypto-processeurs

5.2.1 Architecture globale

L'architecture globale de l'AES (Figure 5.1) a été définie à partir des spécifications du standard et de l'environnement de fonctionnement du circuit. Il s'agit d'un AES-128 qui permet le chiffrement de données de 128 bits avec une clef de taille 128 bits également.
Le circuit est constitué de deux blocs : une interface 32 bits et le coeur de l'AES.

5.2.1.1 Interface

L'interface 32 bits permet la communication entre le coeur de l'AES et la carte de test. Pour cela, elle utilise le protocole APB (*Advanced Peripheral Bus*). Ce protocole transforme les données qui arrivent par blocs de 32 bits en un mot de 128 bits. L'adresse est codée sur 4 bits.
Les modes de lecture et écriture implémentés ici supportent les états d'attente (*wait state*) ce qui permet de charger de nouvelles données pendant un chiffrement ou de suspendre les transactions durant une expérimentation.

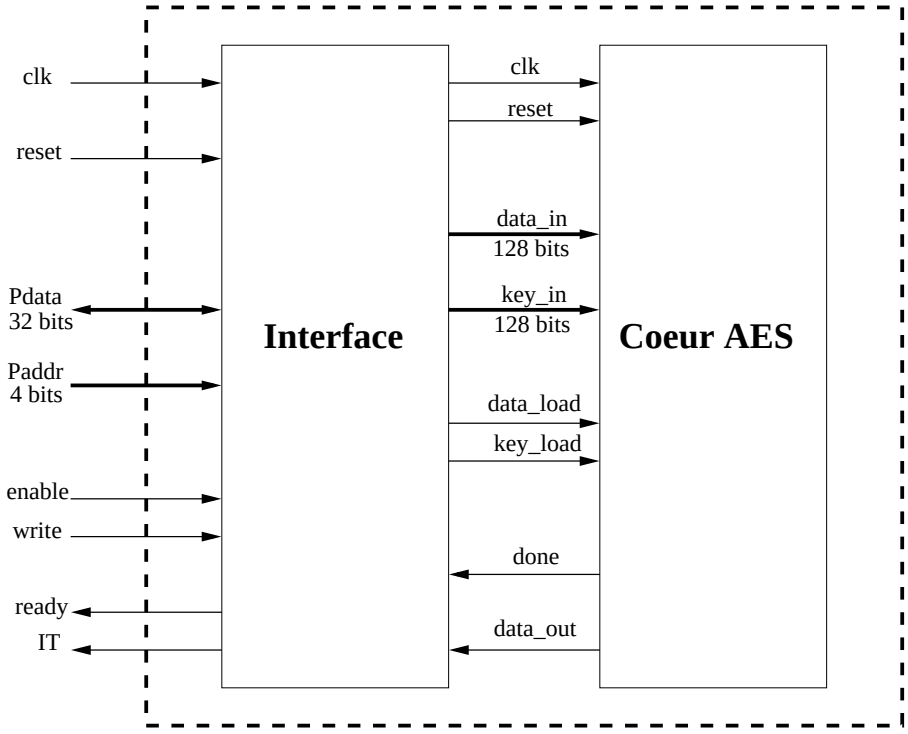

Fig. 5.1 – Architecture globale du crypto-processeur

5.2.1.2 Le coeur de l'AES

Le coeur de l'AES est constitué de deux blocs : un bloc de chiffrement des données et un bloc de génération de clefs partielles (Figure 5.2).
Le bloc de chiffrement est constitué de 5 composants :

1. Mux : ce bloc de permet de transformer une donnée de 128 bits en type "état" (state) tel que défini dans le standard. Le signal mux_sel_i permet de choisir entre la donnée d'entrée (pour la première ronde) et la donnée de ronde (pour les autres rondes).
2. AddRoundKey (ARK) : réalise le XOR entre l'état et la clef de ronde. Le signal add_sel_i permet d'envoyer le résultat de sortie à la sortie data_o pendant la dernière ronde.
3. SubBytes (SB) : les données en entrée de cette transformation sont mémorisées grâce à des bascules Q avant de passer dans les boîtes de substitution S-Box.
4. ShiftRows (SR) : ce bloc effectue un croisement de fils.
5. MixColumns (MC) : ce bloc réalise la transformation `MixColumns` du standard. Le signal mix_sel_i permet d'éviter cette transformation à la dernière ronde.

Le bloc de génération de clefs de ronde comporte 6 composants :

1. Key map : transforme la clef de 128 bits en 4 colonnes de 32 bits. Chaque colonne passe par la suite par un buffer 32 bits.

2. SubBytes : il s'agit d'une substitution de la dernière colonne de la clef en utilisant 4 S-Box.
3. Rot_Word : il s'agit d'une rotation des octets de la colonne.
4. Add_Rcon : ce bloc est un XOR entre la colonne courante et une colonne qui dépend du numéro de ronde.
5. Add Columns : il s'agit d'un XOR entre les colonnes.
6. Output_Key : permet de constituer la clef de ronde sur 128 bits.

Les deux blocs de chiffrement et de génération de clefs de ronde sont cadencés par deux machines d'état différentes non représentées sur la Figure 5.2.

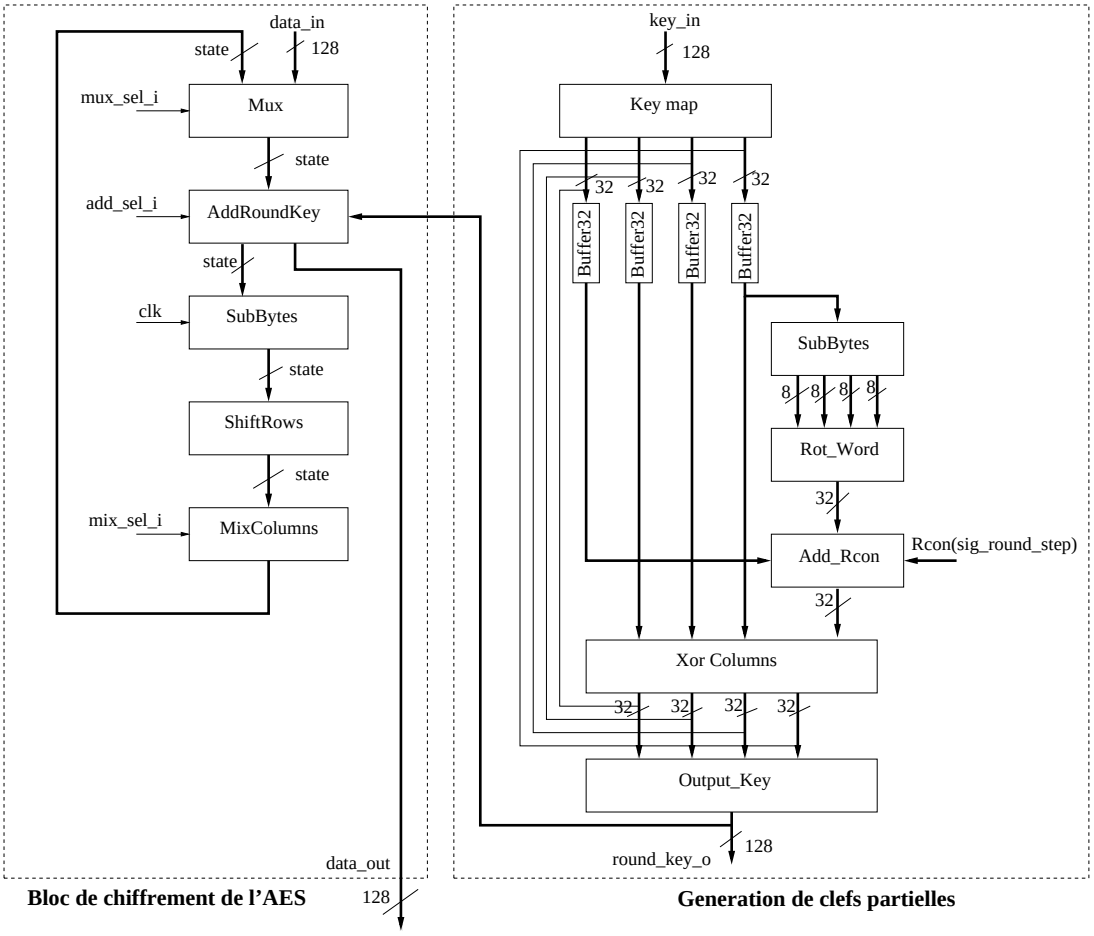

Fig. 5.2 – Architecture du bloc de chiffrement et de l'expansion de clef

5.2.2 Versions sans contre-mesure

Les fonctions décrites dans le paragraphe précédent peuvent être implantées de différentes façons. Cette section décrit les solutions qui ont été retenues.

5.2.2.1 Bloc SubBytes

L'architecture choisie pour les boîtes de substitution S-Box a été proposée dans [71]. Elle utilise des opérations dans le champ de Galois $GF(2^4)$. Afin de comprendre la structure de cette S-Box, rappelons que la S-Box est construite à partir de deux transformations :

1. Inversion dans $GF(2^8)$
2. Transformation affine dans $GF(2)$:

$$b_i^{'} = b_i \oplus b_{(i+4)mod8} \oplus b_{(i+5)mod8} \oplus b_{(i+6)mod8} \oplus b_{(i+7)mod8} \oplus c_i$$

avec b_i le $i^{éme}$ bit de l'octet b et c vaut 63.

Sachant que le champ de Galois $GF(2^8)$ peut être vu comme une extension du champ de Galois $GF(2^4)$, un élément $a \in GF(2^8)$ est représenté comme un polynôme linéaire avec des coefficients dans $GF(2^4)$

$$a \cong a_h x + a_l, \qquad a \in GF(2^8), \qquad a_h, a_l \in GF(2^4)$$

L'inversion d'un polynôme dans $GF(2^4)$ est donnée par l'équation suivante :

$$\begin{aligned}
(a_h x + a_l)^{-1} &= a_h^{'} x + a_l^{'} \\
a_h^{'} &= a_h * d' \\
a_l^{'} &= (a_h + a_l) * d' \\
d &= (a_h^2 * e) + (a_h * a_l) + a_l^2 \\
d' &= d^{-1}
\end{aligned}$$

L'architecture résultante est donnée par la Figure 5.3. Elle comporte 6 multiplications, 3 additions et une inversion dans $GF(2^4)$. La transformation map permet de passer dans $GF(2^4)$. Inversement map^{-1} permet de revenir dans $GF(2^8)$.

5.2.2.2 Bloc MixColumns

Les équations qui gouvernent la fonction `MixColumns`, pour la colonne c, sont les suivantes :

$$S_{0,c}^{'} = (02 \bullet S_{0,c}) \oplus (03 \bullet S_{1,c}) \oplus S_{2,c} \oplus S_{3,c} \quad (5.1)$$

$$S_{1,c}^{'} = S_{0,c} \oplus (02 \bullet S_{1,c}) \oplus (03 \bullet S_{2,c}) \oplus S_{3,c} \quad (5.2)$$

$$S_{2,c}^{'} = S_{0,c} \oplus S_{1,c} \oplus (02 \bullet S_{2,c}) \oplus (03 \bullet S_{3,c}) \quad (5.3)$$

$$S_{3,c}^{'} = (03 \bullet S_{0,c}) \oplus S_{1,c} \oplus S_{2,c} \oplus (02 \bullet S_{3,c}) \quad (5.4)$$

Trois architectures différentes sont envisagées par la suite pour implémenter cette fonction. Les deux premières, notées **V1** et **V2**, sont basées sur le fait que $03 = 01 \oplus 02$. Les équations précédentes, qui mettent en œuvre des multiplications par 02 et 03, peuvent

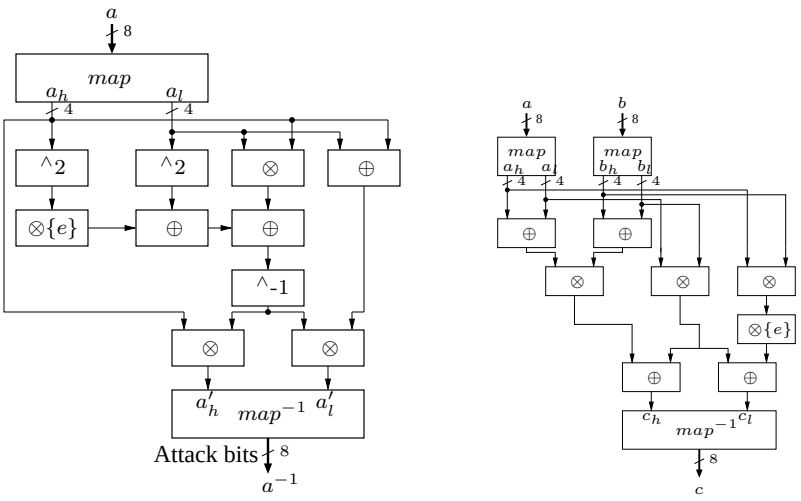

FIG. 5.3 – S-Box (gauche) et MixColumns (droite) utilisant les opérations dans $GF(2^4)$

alors se réécrire avec seulement des multiplications par 02, conformément au système d'équations suivant :

$$S'_{0,c} = 02 \bullet (S_{0,c} \oplus S_{1,c}) \oplus S_{1,c} \oplus S_{2,c} \oplus S_{3,c} \quad (5.5)$$

$$S'_{1,c} = S_{0,c} \oplus 02 \bullet (S_{1,c} \oplus S_{2,c}) \oplus S_{2,c} \oplus S_{3,c} \quad (5.6)$$

$$S'_{2,c} = S_{0,c} \oplus S_{1,c} \oplus 02 \bullet (S_{2,c} \oplus S_{3,c}) \oplus S_{3,c} \quad (5.7)$$

$$S'_{3,c} = S_{0,c} \oplus S_{1,c} \oplus S_{2,c} \oplus 02 \bullet (S_{3,c} \bullet S_{0,c}) \quad (5.8)$$

Il existe ensuite deux façons d'implémenter la fonction "multiplication par 02", la première est la traduction directe de la fonction `xtime()` telle que définie dans le standard. Elle est implantée dans la version **V1** à l'aide d'un multiplexeur.

$$02 \bullet a = \begin{cases} a_6 a_5 a_4 \bar{a_3} \bar{a_2} a_1 \bar{a_0} a_7 & \text{si } a_7 = 1 \\ a_6 a_5 a_4 a_3 a_2 a_1 a_0 a_7 & \text{sinon} \end{cases} \quad (5.9)$$

La seconde permet d'éviter le test sur le bit a_7. Elle est implantée dans la version **V2** à l'aide d'un bloc de décalage (pour le calcul de l'opérande de gauche de l'équation 5.10), d'un bloc qui réalise la concaténation des "0" logiques avec 4 recopies du bit a_7 (pour l'opérande de droite), et d'un XOR bit à bit.

$$02 \bullet a = a_6 a_5 a_4 a_3 a_2 a_1 a_0 0 \oplus 000 a_7 a_7 0 a_7 a_7 \quad (5.10)$$

La dernière architecture **V3** implante directement le système d'équations qui gouverne `MixColumns`, l'opération générique de multiplication étant effectué pour gagner en surface dans le champ de Galois $GF(2^4)$. Les opérations `xor` sont quant à elles réalisées dans $GF(2^8)$. L'implémentation d'une multiplication entre deux éléments de $GF(2^8)$, dans $GF(2^4)$ est donnée par la partie droite de la Figure 5.3.

5.2.3 Comparaison des versions sans contre-mesure

Pour résumer, nous disposons de 3 implémentations pour le crypto-processeur AES qui dépendent de l'architecture du bloc `MixColumns` :

1. **V1** correspond à l'architecture du MixColumns qui utilise la fonction `xtime()`.
2. **V2** correspond l'architecture du MixColumns qui n'effectue pas le test sur a_7.
3. **V3** implémente la multiplication générique dans le corps $GF(2^4)$.

Ces trois implémentations ont été décrites au niveau RTL et ont été synthétisées en utilisant la bibliothèque de cellules standards AMS en $0.35\mu m$. Deux options de synthèse ont été utilisées : *hierarchical* et *flatten*. La première conserve la structure hiérarchique du circuit définie dans le RTL, la seconde la supprime si besoin. La surface, le nombre de portes et la vitesse des 3 versions de l'AES pour les deux options de synthèse différentes sont données par la Table 5.1.

Architecture	Option Synthèse	Surface(mm^2)	Nombre de portes	Vitesse(MHZ)
V1	*flatten*	0.806	6161	63
	hierarchical	0.875	6951	59
V2	*flatten*	0.805	6097	62
	hierarchical	0.865	6783	59
V3	*flatten*	0.925	8670	50
	hierarchical	1.1	9058	48

Tab. 5.1 – Caractéristiques des différentes versions de l'AES après synthèse (technologie AMS $0.35\mu m$)

5.2.4 Versions avec contre-mesures

Deux types de contre-mesures aux attaques par analyse de la consommation sont analysées. La première consiste à équilibrer les chemins de données grâce à une codage dual rail (principe 1), la seconde à rendre non prédictible les valeurs intermédiaires par masquage des données (principe 2).

5.2.4.1 Dual-rail

Cette architecture a été proposée dans [30] pour contrer les attaques en fautes de type DFA. Il s'agit d'implémenter deux AES en parallèle. A chaque calcul, le résultat des deux AES sont comparés et si une différence apparaît, un mécanisme d'offuscation rend les attaques de type DFA inopérantes. Pour rendre moins sensible ce circuit aux attaques en consommation, l'un est des AES est le dual de l'autre. Ainsi, à chaque '1' logique correspond un '0' logique dans la partie duale et inversement. La symétrie des chemins de données et donc la sécurité s'en trouvent théoriquement renforcées.
Pour cette contre mesure, l'architecture **V2** est choisie pour implémenter la transformation `MixColumns`.

5.2.4.2 Masquage

Le principe du masquage additif déjà présenté dans [27] est utilisé pour implémenter cette contre-mesure. L'intégration du masque dans l'AES est donnée par la Figure 5.4 :

- Le masque est ajouté à la clef de chiffrement pendant la première ronde.
- Passage de la donnée masquée dans les S-Box modifiées.
- Sortie des S-Box avec le même masque qu'en entrée.
- Passage parallèle des données masquées et du masque dans les blocs ShiftRows et MixColumns.
- Ajout du masque passé par les transformations `ShiftRows` et `MixColumns` et du masque initial à la clef de ronde.
- Démasquage lors de la sortie.

En raison de la non linéarité de la fonction `SubBytes`, une modification de la S-Box est nécessaire afin de propager le masque. La modification de la transformation `SubBytes` se déroule en deux étapes : une modification de la fonction inversion et une modification de la transformation affine.
La modification de la fonction inversion a été proposée dans [55]. L'idée est de retrouver les mêmes équations définies dans la section 5.2.2.1 pour l'inversion dans $GF(2^4)$:

$$\begin{aligned}((a_h+m)x+(a_l+m_l))^{-1} &= (a_h^{'}+m_h^{'})x+(a_l^{'}+m_l^{'})\\ a_h^{'}+m_h^{'} &= a_h*d'+m_h^{'}\\ a_l^{'}+m_l^{'} &= (a_h+a_l)*d'+m_l^{'}\\ d+m_d &= (a_h^2*e)+(a_h*a_l)+a_l^2+m_d\\ d'+m_d^{'} &= d^{-1}+m_d^{'}\end{aligned}$$

Pour obtenir ces équations, des fonctions sont écrites afin d'annuler l'effet du masque. Pour la transformation affine, le masque ajoute pour le bit i la valeur $m_i \oplus m_{(i+4)mod8} \oplus m_{(i+5)mod8} \oplus m_{(i+6)mod8} \oplus m_{(i+7)mod8}$. Il faut donc ajouter $m_{(i+4)mod8} \oplus m_{(i+5)mod8} \oplus m_{(i+6)mod8} \oplus m_{(i+7)mod8}$ au résultat de la transformation affine pour avoir en sortie $b_i^{'} \oplus m_i$.

5.2.4.3 Comparaison des versions avec contre-mesure

Les performances des deux contre-mesures "dual" et "masquage" après synthèse logique, sont données dans le Tableau 5.2.

Contre-mesure	**Surface(mm^2)**	**Nombre de portes**	**Vitesse(MHZ)**
DUAL	1.52	12685	50
Masquage	1.55	11211	33

Tab. 5.2 – Performances des contre-mesures de l'AES après synthèse (technologie AMS $0.35\mu m$)

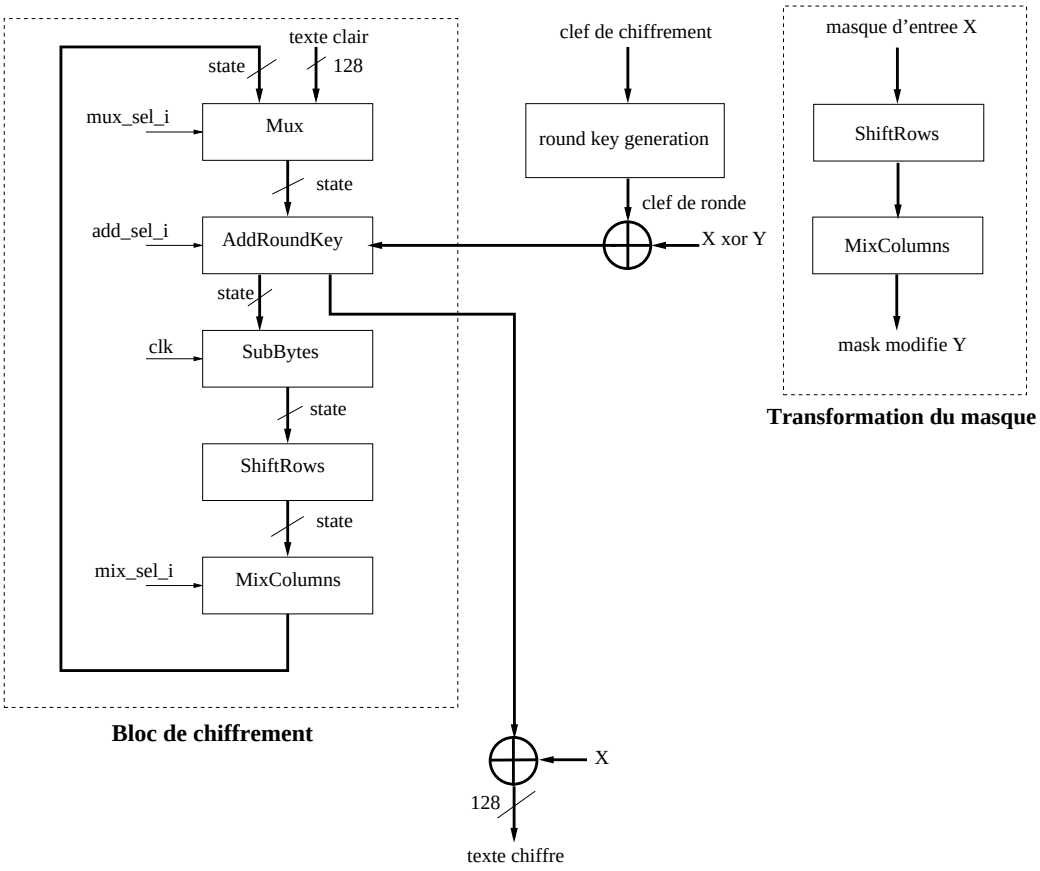

Fig. 5.4 – Architecture de l'AES masqué

5.3 Évaluation sécuritaire des différents modèles

5.3.1 Paramètres de l'étude

Pour illustrer l'intérêt de la méthode décrite dans le chapitre précédent lors du choix par le concepteur d'une architecture, nous l'appliquons dans cette partie aux modèles de l'AES décrits dans la section précédente. Pour que la comparaison entre ces modèles soit non biaisée, les paramètres relatifs aux attaques sont communs. Ceux choisis sont les suivants :

- M' : Description algorithmique de l'AES.
- T : Ensemble des 256 textes clairs qui varient par leur premier octet (de $0x00$ à $0xFF$). Les 15 autres octets sont constants. Pour des raisons de simplicité, ces 15 octets sont fixés à $0x0$. Ce choix de textes clairs permet d'avoir un maximum de corrélations ce qui permet de positionner le concepteur dans le pire des cas.
- K : Clef de chiffrement k_0 qui est égale à `2B 7E 15 16 28 AE D2 A6 AB F7 15 88 09 CF 4F 3C`.
- K' : Ensemble des 256 hypothèses de clefs partielles qui parcourent l'entrée de la première S-Box.

– $R^{M'}$: Ensemble des bits en sortie de la première S-Box ($|R^M| = 8$). Les bits de l'ensemble $R^{M'}$ sont notés $SB_0[i]$, avec $i \in [0, \ldots, 7]$ et $SB_0[7]$ étant le bit de poids fort (conformément au standard).
– S' : Poids de Hamming des signaux $SB_0[i], i \in [0, \ldots, 7]$, c'est-à-dire valeur des signaux eux-même.
– ζ' : Instant de calcul des sorties des S-Box pendant la première ronde, noté δ'_0.

Les paramètres relatifs aux modèles étudiés seront précisés pour chacune des expérimentations (description M, syndrome émis S, instants de calculs ζ et bits de R^M).

5.3.2 Comparaison des versions sans contre-mesure

5.3.2.1 Niveau RTL

Pour les modèles de l'AES décrits au niveau RTL, l'ensemble R^M est constitué des bits du bloc de chiffrement des données, lequel est formé de 7439 bits pour les versions **V1** et **V2** et de 14095 bits pour l'AES **V3**. Le syndrome S émis par ces bits est le poids de Hamming et l'ensemble ζ est constitué des instants d'échantillonnage réalisés en fin de première, seconde et troisième ronde ($zeta = \delta_0, \delta_1, \delta_2$).

Première ronde Les corrélations $Corr_{T,HW,HW,CP}(r^M, k_0, \delta_0, SB_0[i], k_0, \delta'_0)$ en fin de première ronde ont été calculées pour $i \in [0, \ldots, 7]$ et pour les 3 implémentations de l'AES sans contre-mesures. Le nombre de bits corrélés aux bits d'attaques $SB_0[i]$ sont reportés dans le Tableau 5.3. Ce nombre, qui varie de 13 à 22, dépend à la fois de l'architecture de la transformation `MixColumns` (laquelle différencie les 3 versions de l'AES) et du bit d'attaque choisi. Ces dépendances sont expliquées ci-dessous pour les versions **V1** et **V2**.

AES	**Numéro du bit d'attaque**								**Total**
	$SB_0[0]$	$SB_0[1]$	$SB_0[2]$	$SB_0[3]$	$SB_0[4]$	$SB_0[5]$	$SB_0[6]$	$SB_0[7]$	
V1	13	18	13	13	18	18	18	18	129
V2	14	19	14	14	19	19	19	22	140
V3	21	22	15	15	22	22	22	22	161

Tab. 5.3 – Nombre de bits sensibles pour les 3 versions de l'AES au niveau RTL k_0

Tout d'abord, conformément à l'architecture présentée en Figure 5.2 et simplifiée en Figure 5.7, l'octet constitué des 8 bits d'attaque $SB_0[i]$ traverse, sans être modifié, le ShiftRows. Chaque bit en sortie de ShiftRows est donc identique (donc corrélé) à un bit d'attaque. L'octet atteint ensuite l'entrée notée $S_{0,0}$ du bloc MixColumns. Les autres octets en entrée de ce bloc sont notés $S_{1,0}$, $S_{2,0}$ et $S_{3,0}$.
La corrélation entre $S_{0,0}$ et les bits internes au bloc MixColumns dépend en partie de l'architecture de ce dernier. Les relations suivantes sont vraies pour les versions **V1** et **V2** (représentées respectivement sur les Figure 5.5 et 5.6) : Comme $S_{1,0}$, $S_{2,0}$ et $S_{3,0}$ sont constants pour l'ensemble des réalisations (les 15 derniers octets des textes clairs étant choisis constants), tout bit i en sortie de $S_{0,0} \oplus S_{1,0}$ est parfaitement corrélé au bit i de $S_{0,0}$. Il en est de même pour tout bit i en sortie de $S_{0,0} \oplus S_{3,0}$, $S_{0,0} \oplus S_{1,0} \oplus S_{2,0}$,

$S_{0,0} \oplus S_{2,0} \oplus S_{3,0}$ et $S_{0,0} \oplus S_{1,0} \oplus S_{3,0}$. Comme $S_{1,0}$, $S_{2,0}$ et $S_{3,0}$ sont constants, les valeurs $02 \bullet (S_{1,0} \oplus S_{2,0})$ et $02 \bullet (S_{2,0} \oplus S_{3,0})$ le sont aussi. Par conséquent, tout bit i des sorties $S'_{1,0}$ et $S'_{2,0}$ est corrélé au bit i de $S_{0,0}$. Les 8 octets h tels que pour tout $i \in \{0 \dots 7\}$, $h[i]$ est parfaitement corrélé au bit i de $S_{0,0}$ sont reportés en rouge sur les Figures 5.5 et 5.6 représentant la fonction `MixColumns`.
La fonction `xtime` est implantée sur la version **V1** grâce à un multiplexeur qui sélectionne la sortie en fonction du dernier bit de $S_{0,0}$, noté $S_{0,0}[7]$ conformément à l'équation 5.9. Comme 5 bits en entrée de `xtime` sont recopiés en sortie, les 5 bits d'indice 7, 6, 5, 2 et 0 de l'octet $02 \bullet S_{0,0}$ en sortie de la fonction `xtime`, sont corrélés respectivement aux bits 6, 5, 4, 1 et 7 de $S_{0,0}$ (en rose sur le Figure 5.5).
La fonction `xtime` est implantée sur la version **V2** grâce à un bloc de décalage (pour le calcul de l'opérande de gauche de l'équation 5.10), d'un bloc qui réalise la concaténation des bits '0' avec 4 recopies du bit $S_{0,0}[7]$ (pour l'opérande de droite), et d'un XOR bit à bit. Par conséquent, tout bit i en sortie du bloc de décalage est corrélé à bit i de $S_{0,0}$ (en rouge sur la Figure 5.6) et 4 bits en sortie de la concaténation sont corrélés au bit $S_{0,0}[7]$ (en bleu sur la Figure 5.6). Comme pour la version **V1**, les bits 7, 6, 5, 2 et 0 de l'octet $02 \bullet S_{0,0}$ (en sortie du XOR) sont corrélés respectivement aux bits 6, 5, 4, 1 et 7 de $S_{0,0}$ (en rose sur la Figure 5.6).
En sortie du `MixColumns` et pour les 2 versions, les 2 octets $S'_{1,0}$ et $S'_{2,0}$ sont corrélés à $S_{0,0}$ et l'octet $S'_{0,0}$ présente 5 bits sur 8 qui sont corrélés à des bits de $S_{0,0}$ (car $S_{2,0} \oplus S_{3,0}$ est constant pour toutes les réalisations). Ces corrélations se propagent en aval à travers le bloc Mux et AddRoundKey.

Ces considérations sont résumées dans les Tableaux 5.4 où sont répartis les bits corrélés en première ronde pour les deux premières versions de l'AES. On constate que les résultats obtenus par cette analyse "manuelle" sont identiques à ceux obtenus grâce à notre outil.

AES	**Numéro du bit d'attaque**							
	$SB_0[0]$	$SB_0[1]$	$SB_0[2]$	$SB_0[3]$	$SB_0[4]$	$SB_0[5]$	$SB_0[6]$	$SB_0[7]$
Sortie SB	1	1	1	1	1	1	1	1
Sortie SR	1	1	1	1	1	1	1	1
Dans MC (V1)	5	6	5	5	6	6	6	6
Dans MC (V2)	5	7	5	5	7	7	7	10
Sortie MC	2	3	2	2	3	3	3	3
Sortie Mux	2	3	2	2	3	3	3	3
Sortie ARK	2	3	2	2	3	3	3	3
Total V1	13	18	13	13	18	18	18	18
Total V2	14	19	14	14	19	19	19	22

Tab. 5.4 – Répartition des bits corrélés aux bits de $S_{0,0}$ lors de la première ronde pour l'AES

A noter qu'une telle analyse "manuelle" serait beaucoup plus complexe à réaliser sur la version **V3**. Cependant, pour cette version comme pour les deux autres, le nombre de bit corrélés dépend de l'indice du bit d'attaque considéré. La Figure 5.7 réprésente la propagation des corrélations communes aux trois versions de l'AES. Sur cette Figure, les 8 bits i des octets en rouge sont corrélés aux bits d'attaque $SB_0[i]$. Les 5 bits d'indice 7, 6,

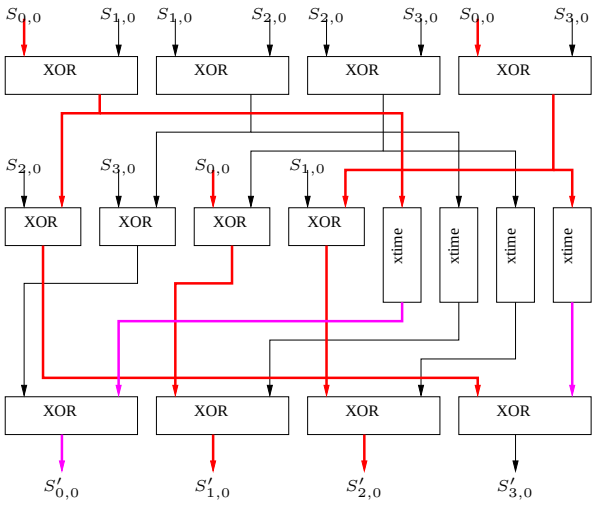

Fig. 5.5 – Architecture du MixColumns de l'AES **V1** : les octets corrélés correspondent aux flèches en gras

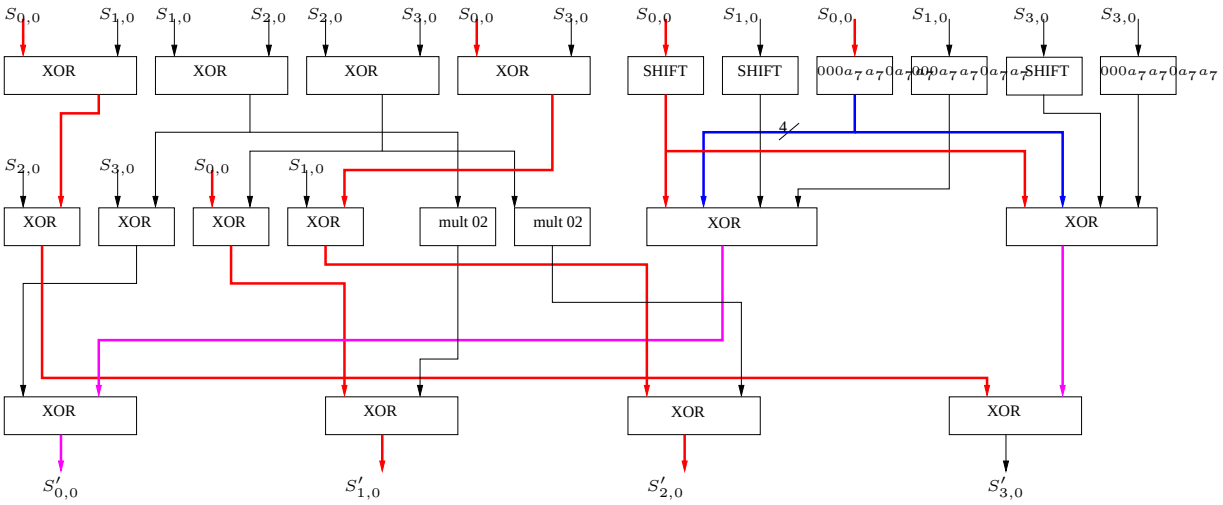

Fig. 5.6 – Architecture du MixColumns de l'AES **V2** : les octets corrélés correspondent aux flèches en gras

5, 2 et 0 des octets en rose sont corrélés respectivement aux bits d'attaque $SB_0[6]$, $SB_0[5]$, $SB_0[4]$, $SB_0[1]$ et $SB_0[7]$.
Nous avons, en particulier, 8 bits des octets $S_{1,0}$ et $S_{2,0}$ et 5 bits de $S_{0,0}$ en sortie de AddRoundKey de la première ronde qui sont corréles à des bits en sortie des S-Boxes.

Afin de s'assurer que le nombre de bits potentiellement sensibles ne dépend pas de la clef de chiffrement choisie, une étude identique a été réalisée mais avec une clef de chiffrement $k_1 \neq k_0$ (`57 49 53 A2 EC 63 7D AA 9E 0F 14 16 28 0F E5 76`). Les résultats obtenus s'avèrent être strictement les mêmes.

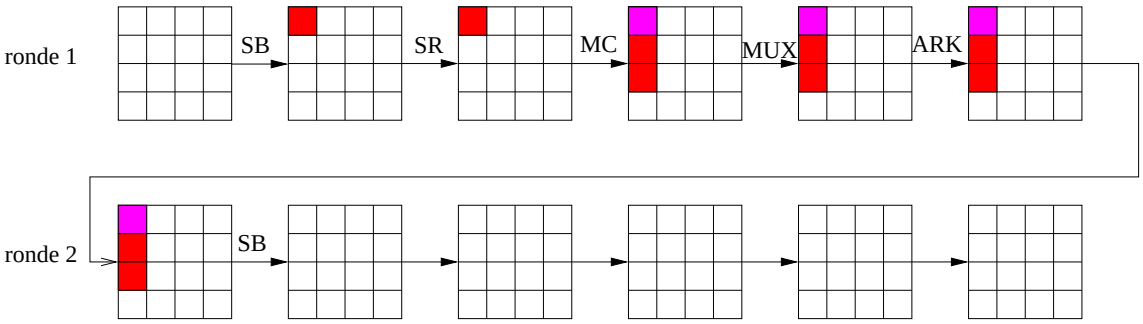

Fig. 5.7 – Répartition des bits corrélés pendant les deux premières rondes du chiffrement de l'AES

Deuxième ronde Le calcul des corrélations en fin de seconde ronde $Corr_{T,HW,HW,CP}(r^M, k_0, \delta_1, SB_0[i], k_0, \delta'_0)$ a été réalisé pour $i \in [0, \ldots, 7]$ et pour les 3 implémentations de l'AES sans contre-mesures. Les bits corrélés aux bits d'attaques $SB_0[i]$ ainsi que leur nombre sont reportés dans le Tableau 5.5. On constate que ce nombre, qui varie de 2 à 5, ne dépend pas de l'architecture choisie mais du numéro du bit d'attaque.

AES	**Numéro du bit d'attaque**			
	$SB_0[0]$	$SB_0[1]$	$SB_0[2]$	$SB_0[3]$
Entrée SB R2	SB4/S-Box_in[0] SB8/S-Box_in[0]	SB0/S-Box_in[2] SB4/S-Box_in[1] SB8/S-Box_in[1]	SB4/S-Box_in[2] SB8/S-Box_in[2]	SB4/S-Box_in[3] SB8/S-Box_in[3]
Dans SB R2	SB4/mult_in2[0] SB8/mult_in2[0]	SB4/add_in1[2] SB8/add_in1[2]		
Total	4	5	2	2
	Numéro du bit d'attaque			
	$SB_0[4]$	$SB_0[5]$	$SB_0[6]$	$SB_0[7]$
Entrée SB R2	SB0/S-Box_in[5] SB4/S-Box_in[4] SB8/S-Box_in[4]	SB0/S-Box_in[6] SB4/S-Box_in[5] SB8/S-Box_in[5]	SB0/S-Box_in[7] SB4/S-Box_in[6] SB8/S-Box_in[6]	SB0/S-Box_in[0] SB4/S-Box_in[7] SB8/S-Box_in[7]
Dans SB R2	SB4/add_out1[2] SB8/add_out1[2]			SB0/mult_in2[0]
Total	5	3	3	4

Tab. 5.5 – Nombre de bits sensibles manipulés pendant la deuxième ronde pour les 3 architectures de l'AES

D'après l'analyse faite dans le paragraphe précédent pour la première ronde, 5 des bits de $S_{0,0}$ et les 8 bits des octets $S_{1,0}$ et $S_{2,0}$ en sortie de AddRoundKey sont corréles à des bits d'attaque. Comme ces 3 octets sont recopiés respectivement en entrées des S-Boxes 0, 4 et 8 en début de seconde ronde, ces entrées sont également corrélés aux bits d'attaques. Ceci explique les premières lignes du Tableau 5.5. Les autres lignes concernent les corrélations apparaissant à l'intérieur même des S-Boxes en seconde ronde. Ces corrélations

s'expliquent par une analyse des équations qui régissent les S-Boxes implémentées dans $GF(2^4)$. Par exemple, le bit $add\_in1[2]$ constitue un bit en sortie du bloc "mult_e" qui est une multiplication par e :

$$add\_in1[2] = mult\_e\_out[2] \quad = \quad mult\_e\_in[0] \oplus mult\_e\_in[1] \oplus mult\_e\_in[2]$$

Les entrées du bloc "mult_e"forment la sortie du bloc "squar_0" qui vérifie les équations suivantes :

$$\begin{aligned} mult\_e\_out[0] = squar\_out[0] &= squar\_in[0] \oplus squar\_in[2] \\ mult\_e\_out[1] = squar\_out[1] &= squar\_in[2] \\ mult\_e\_out[2] = squar\_out[2] &= squar\_in[1] \oplus squar\_in[3] \end{aligned}$$

Par conséquent :

$$add\_in1[2] \quad = \quad squar\_in[0] \oplus squar\_in[1] \oplus squar\_in[3]$$

Or, selon la fonction *map*, nous avons les équations suivantes :

$$\begin{aligned} squar\_in[0] &= SBox\_in[4] \oplus SBox\_in[5] \oplus SBox\_in[6] \\ squar\_in[1] &= SBox\_in[1] \oplus SBox\_in[4] \oplus SBox\_in[6] \oplus SBox\_in[7] \\ squar\_in[3] &= SBox\_in[5] \oplus SBox\_in[7] \end{aligned}$$

Il en résulte que $add\_in1[2] = SBox\_in[1]$. On peut montrer de la même manière que $add\_out1[2] = SBox\_in[4]$ et que $mult\_in2[0] = SBox\_in[0]$, ces relations étant vraies pour toutes les S-Boxes et en particulier les 0, 4 et 8.

Les résultats obtenus montrent que les corrélations en seconde ronde n'apparaissent qu'au tout début des calculs dans les S-Boxes et, qu'en particulier, elles n'affectent pas de bits dans le bloc `Mixcolumns`, lequel différencie les trois versions de l'AES. Ce qui explique pourquoi les résultats sont identiques pour **V1**, **V2** et **V3**.

Troisième ronde Le calcul des corrélations en fin de troisième ronde a été réalisée pour les 8 bits d'attaque et pour les 3 implémentations de l'AES sans contre-mesure. Les valeurs de corrélation obtenues varient entre -0.2 et 0.2 (celles obtenues pour le bit d'attaque $SB_0[0]$ et la version **V1** sont, par exemple, reportées en Figure 5.9). Nous considérons, vu ces faibles valeurs qu'il n'y a pas de bits corrélés aux bits d'attaques pendant la troisième ronde.

Conclusion sur la sécurité au niveau RTL Les bits potentiellement sensibles ont été déterminés pendant les trois premières rondes du chiffrement de l'AES. Les corrélations obtenues pendant la troisième ronde étant faibles, il y a donc peu de chances qu'il ait des corrélations importantes pendant les rondes suivantes. Nous pouvons donc conclure que le nombre total de bits potentiellement sensibles est la somme des bits potentiellement sensibles manipulés pendant les deux premières rondes. Leur nombre total pour les 3 architectures de l'AES est donné par la Table 5.6.

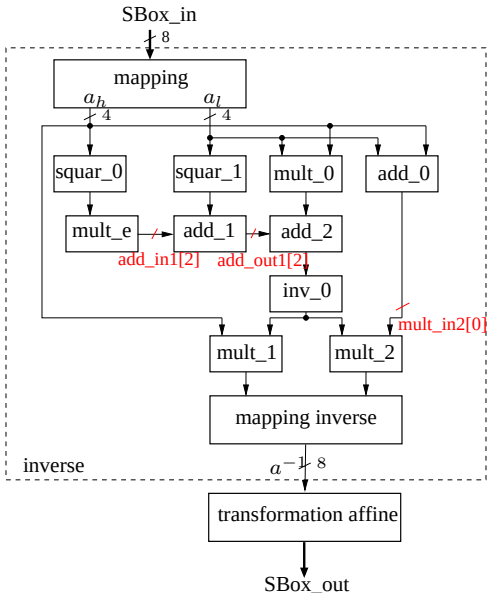

Fig. 5.8 – Architecture de l'implémentation de la S-Box

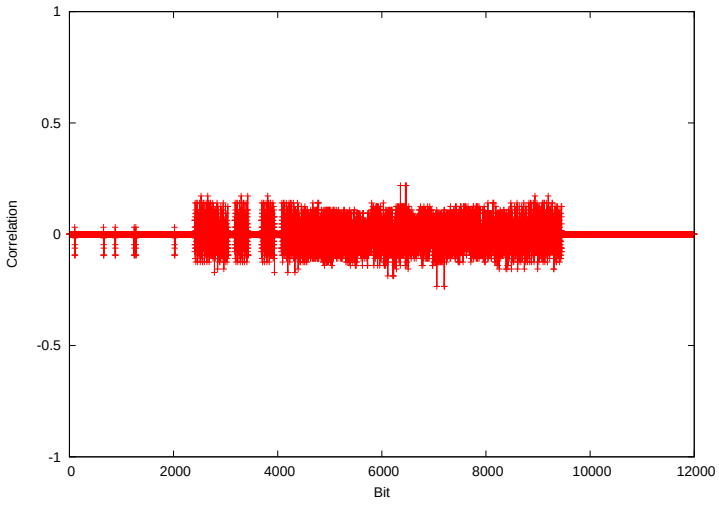

Fig. 5.9 – Courbes $r^M, Corr_{T,HW,HW,CP}(r^M, k_0, \delta_2, SB_0[0], k_0, \delta'_0)$ de l'AES **V1** au niveau RTL à la fin de la troisième ronde

5.3.2.2 Niveau portes

Afin d'estimer l'impact de la phase de synthèse sur la sécurité de l'AES, une analyse de corrélations sur les 3 implantations de l'AES après synthèse, est réalisée. Pour ces modèles, l'ensemble R^M est constitué des bits du bloc de chiffrement des données et le syndrome S émis par les bits de R^M est le poids de Hamming. Pour chaque AES, deux netlists sont étudiées : une hiérarchique et une à plat. En raison de la lenteur des simulations

AES	Nbr total de bits sensibles
V1	157
V2	168
V3	189

Tab. 5.6 – Nombre total de bits sensibles pour les différentes implémentations de l'AES au niveau RTL

numériques post-synthèse et du grand nombre de simulations nécessaires à cette étude, le fichier SDF n'a pas été pris en compte lors des simulations numériques. Par conséquent, les valeurs logiques des signaux du chemin de données sont échantillonnées lorsque ces dernières sont stables (comme au niveau RTL et avec les mêmes notations).
Le calcul des corrélations en fin de première ronde $Corr_{T,HW,HW,CP}(r^M, k_0, \delta_0, SB_0[i], k_0, \delta'_0)$ a été réalisé pour $i \in [0, \ldots, 7]$ et pour les 3 implémentations "portes" de l'AES sans contre-mesure. Le nombre de bits corrélés aux bits d'attaques $SB_0[i]$ est reporté dans le Tableau 5.7.

On constate tout d'abord que la synthèse à plat génère beaucoup moins de corrélations que la synthèse hiérarchique (moyenne de 6 bits corrélés par bit d'attaque contre 18 au niveau RTL). Cette baisse peut s'expliquer par le fait que dans le premier cas, l'outil de synthèse, ayant la liberté de modifier la structure du circuit, peut procéder à des regroupements de calculs et les faire réaliser avec des portes logiques complexes. Si l'on considère, par exemple, le bloc `MixColumns` de l'AES **V2** représenté sur la Figure 5.6, le XOR entre $S_{0,0}$, $S_{3,0}$ et $S_{1,0}$ est réalisé avec 1 porte XOR à deux entrées qui réalise $(x = S_{0,0} \oplus S_{3,0})$ et avec une autre porte XOR à deux entrées qui réalise $x \oplus S_{1,0}$. Le calcul peut être pareillement effectué à l'aide d'une porte XOR à trois entrées, faisant disparaitre la variable x qui dans notre cas est corrélée à un bit d'attaque.

On retrouve également pour les modèles de portes "hiérarchiques", la même tendance qu'au niveau RTL, à savoir que des bits potentiellement sensibles dépendent du numéro du bit d'attaque ; les bits d'attaques $SB_0[0]$, $SB_0[2]$ et $SB_0[3]$ présentent en moyenne moins de bits sensibles que les autres bits d'attaque. En outre, les versions "portes" hiérarchiques présentent un nombre de bits potentiellement sensibles proche de celui obtenu au niveau RTL.

AES	Option Synthèse	Numéro du bit d'attaque								Total
		$SB_0[0]$	$SB_0[1]$	$SB_0[2]$	$SB_0[3]$	$SB_0[4]$	$SB_0[5]$	$SB_0[6]$	$SB_0[7]$	
V1	*flatten*	5	5	4	4	6	6	8	5	43
	hierarchical	16	21	16	16	20	21	22	21	153
V2	*flatten*	5	6	8	6	6	6	6	6	49
	hierarchical	16	22	16	16	22	22	22	23	159
V3	*flatten*	7	5	4	3	6	5	7	7	44
	hierarchical	23	25	18	16	24	24	24	24	178

Tab. 5.7 – Nombre de bits sensibles pour les différentes versions de l'AES au niveau portes

5.3.3 Evaluation de la contre-mesure "dual" face aux attaques CPA

La logique dual rail consiste à rendre le poids de Hamming des données manipulées indépendant des données en équilibrant tout '1' logique et pour un '0' (et réciproquement). Cette propriété peut être testée avec notre analyse de corrélations en vérifiant qu'à chaque bit parfaitement corrélé à un bit d'attaque peut être associé un bit inversement corrélé à ce dernier.

5.3.3.1 RTL

A ce niveau d'abstraction, l'ensemble R^M est constitué des bits du bloc de chiffrement des données ($|R^M| = 15854$), le syndrome S émis par les bits de R^M est le poids de Hamming et l'ensemble ζ correspond aux instants d'échantillonnage en fin des deux premières rondes ($\zeta = \{\delta_0, \delta_1\}$). En effet, l'AES Dual a été conçu à partir de l'AES **V2** lequel ne présente pas de bits potentiellement sensibles pendant la troisième ronde (section 5.3.2.1). Le calcul des corrélations à la fin de la première ronde $Corr_{T,HW,HW,CP}(r^M, k_0, \delta_0, SB_0[i], k_0, \delta'_0)$ a été réalisé pour $i \in [0, \ldots, 7]$.

Bit d'attaque	$SB_0[0]$	$SB_0[1]$	$SB_0[2]$	$SB_0[3]$	$SB_0[4]$	$SB_0[5]$	$SB_0[6]$	$SB_0[7]$	Total
Nbr. bits sensibles	28	38	28	28	38	38	38	44	280

Tab. 5.8 – Nombre de bits sensibles pour l'AES dual niveau RTL pendant la première ronde

On constate dans le Tableau 5.8 que le nombre total de bits potentiellement sensibles calculés pendant la première ronde est multiplié par 2 par rapport à l'AES **V2** 5.3.

Pour tester la sécurité de l'AES dual, il est nécessaire de savoir s'il existe une dualité entre les bits corrélés du chemin régulier et ceux du chemin dual, c'est-à-dire vérifier qu'à chaque bit parfaitement corrélé (resp. inversement corrélé) du chemin normal correspond un bit inversement corrélé (resp. parfaitement corrélé) du chemin dual. Grâce à la liste des bits corrélés, fourni par notre outil, une recherche de correspondance entre les noms de bits corrélés du chemin régulier et ceux du chemin dual est effectuée. Pour le bit d'attaque $SB_0[0]$, par exemple, la liste des bits corrélés est dressé dans le Tableau 5.9.
La symétrie entre les noms des bits corrélés dans le chemin normal et ceux du chemin dual permet de conclure que l'AES dual est résistant aux attaques CPA au niveau RTL.

5.3.3.2 Niveau portes rétro- annotée

Dans ce cas, l'ensemble R^M est constitué des bits du bloc de chiffrement des données ($|R^M| = 47807$), le syndrome S émis par les bits de R^M est le poids de Hamming et l'ensemble ζ est formé des pas de temps multiples de $100ps$ entre le début de la première ronde et la fin de la troisième ronde ($|zeta| = 380$).
Le calcul des corrélations $Corr_{T,HW,HW,CP}(r^M, k_0, \delta, SB_0[i], k_0, \delta'_0)$ a été réalisé pour $i \in [0, \ldots, 7]$. Le nombre de bits potentiellement sensibles est donné par la Table 5.10.

Chemin régulier	Chemin dual
$byte\_1/byte\_o[120]$ (+1)	$byte\_2/byte\_o[120]$ (-1)
$shift\_1/shift\_o[120]$ (+1)	$shift\_2/shift\_o[120]$ (-1)
$mix\_1/mix2[121]$ (+1)	$mix\_2/mix2[121]$ (-1)
$mix\_1/s0[120]$ (+1)	$mix\_2/s0[120]$ (-1)
$mix\_1/s1[120]$ (+1)	$mix\_2/s1[120]$ (-1)
$mix\_1/s2[120]$ (+1)	$mix\_2/s2[120]$ (-1)
$mix\_1/s3[120]$ (+1)	$mix\_2/s3[120]$ (-1)
$mix\_1/s4[120]$ (+1)	$mix\_2/s4[120]$ (-1)
$mix\_1/mix\_o[56]$ (+1)	$mix\_2/mix\_o[56]$ (-1)
$mix\_1/mix\_o[88]$ (-1)	$mix\_2/mix\_o[88]$ (+1)
$mux\_1/mux\_o[56]$ (+1)	$mux\_2/mux\_o[56]$ (-1)
$mux\_1/mux\_o[88]$ (-1)	$mux\_2/mux\_o[88]$ (+1)
$state\_1/state\_o[56]$ (+1)	$state\_2/state\_o[56]$ (-1)
$state\_1/state\_o[88]$ (-1)	$state\_2/state\_o[88]$ (+1)

Tab. 5.9 – Répartition des bits sensibles entre le chemin régulier et le chemin dual pour le bit $SB_0[0]$ de l'AES dual au niveau RTL, valeur de corrélation entre parenthèses

Bit d'attaque	$SB_0[0]$	$SB_0[1]$	$SB_0[2]$	$SB_0[3]$	$SB_0[4]$	$SB_0[5]$	$SB_0[6]$	$SB_0[7]$
Nbr. bits sensibles	58	56	40	36	54	50	50	54

Tab. 5.10 – Nombre de bits sensibles pour les 8 bits en sortie de la première S-Box de l'AES Dual au niveau portes avec rétro-annotation

Comme aucune contrainte particulière n'a été imposée à l'outil de synthèse, les chemins logiques qui relient les bits duaux (définis au niveau RTL) ne sont pas forcément symétriques, au sens où leurs structures ne sont pas forcément superposables et même si cela était le cas, les portes appariées, si elles sont bien complémentaires, n'ont pas forcément les mêmes caractéristiques électriques (les sortances peuvent être différentes). Les chemins logiques réguliers et duaux qui relient les bits $mixi[126]$ et $mixo[62]$ (corrélés au bit d'attaque $SB_0[6]$) sont, par exemple, représentés respectivement en haut et en bas de la Figure 5.10. Ils ne sont pas superposables car l'outil de synthèse a choisi d'utiliser deux multiplexeurs simples (en rouge) dans le chemin régulier et un seul, plus complexe (en bleu), dans le chemin dual. Cette dissymétrie structurelle crée une différence entre le nombre de bits corrélés au bit d'attaque sur le chemin régulier et le chemin dual (respectivement 6 et 4 dans ce cas, en gras sur la Figure 5.10).

Pour faire apparaitre ce type de dissymétries engendrées lors de la phase de synthèse, la différence entre le nombre de bits parfaitement corrélés et le nombre de bits inversement corrélés est calculée à chaque pas de temps. Les résultats sont reportés sur la Figure 5.11 pour le bit d'attaque $SB_0[0]$. La première différence apparait à l'instant $271800ps$. Cet instant correspond au calcul des S-Boxes dans le chemin normal. Par contre, dans le chemin dual, le calcul des sorties des S-Boxes ne commence qu'à partir de $273300ps$.

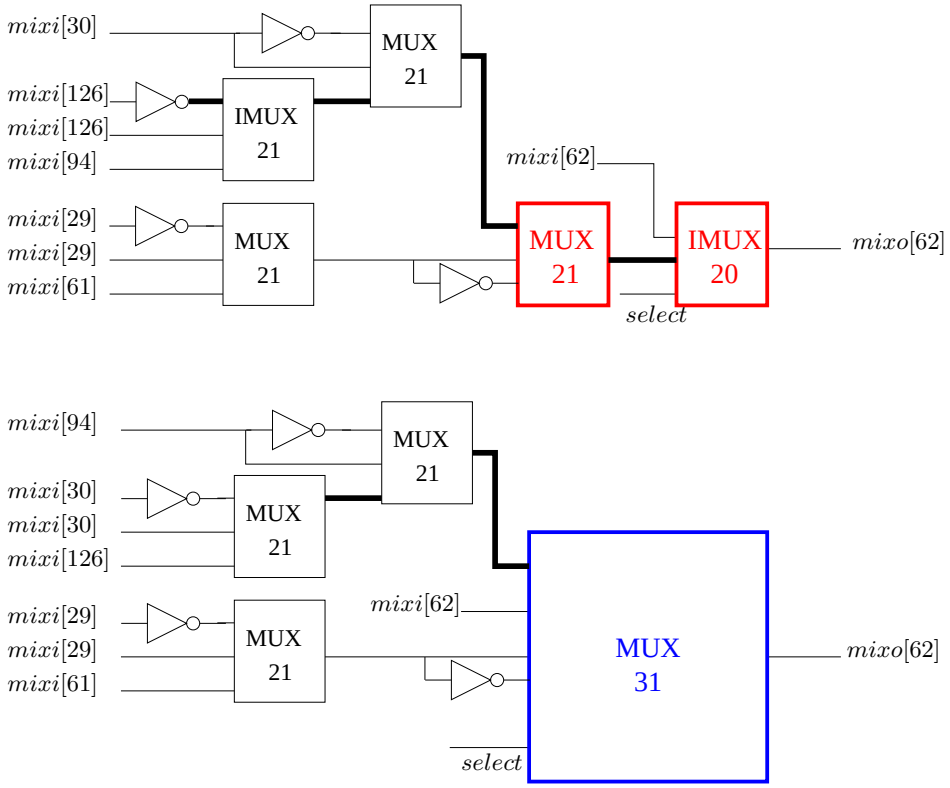

Fig. 5.10 – Vue partielle du bloc MixColumns (haut : chemin normal, bas : chemin dual)

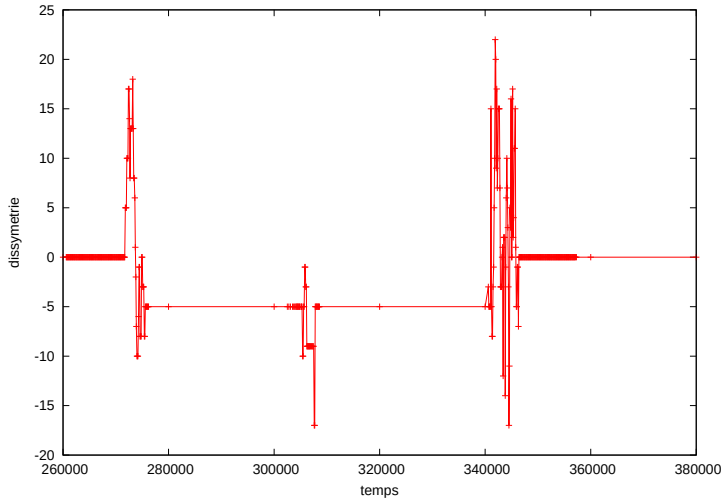

Fig. 5.11 – Valeurs de dissymétries obtenues pour le bit d'attaque $SB_0[0]$ pendant les 3 premières rondes de l'AES Dual

Ces dissymétries engendrent des fuites d'information et rend par conséquent l'AES dual vulnérable aux attaques CPA. Ce constat est confirmé par l'évaluation de l'AES dual en appliquant la méthode classique des attaques CPA. La Figure 5.12 représente l'ensemble des courbes $\{\delta, Corr^{k'}_{T,Id,HW,CP}(r^M, k_0, \delta, SB_0[0], \delta'_0)\}$ pour l'ensemble des hypothèses de clefs partielles et pour tous les pas de temps. Les instants d'apparition des pics CPA sont en accord avec les pics de dissymétries visualisés en Figure 5.11.

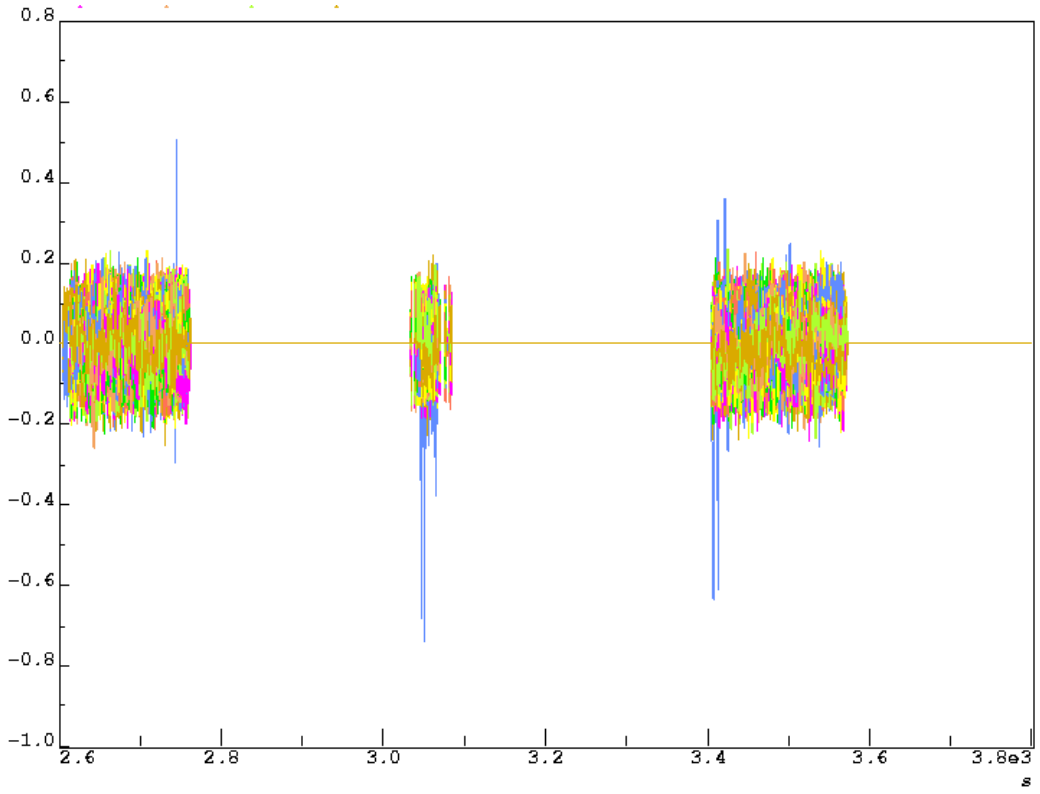

Fig. 5.12 – Courbes CPA de l'AES Dual

5.3.4 Étude de l'AES masqué

5.3.4.1 Syndrome de poids de Hamming

L'évaluation de l'AES masqué est réalisée sur une description "portes" avec rétro-annotation obtenue après synthèse automatique hiérarchique à partir du modèle RTL décrit dans le paragraphe 5.2.4.2. L'ensemble R^M est constitué des bits du bloc de chiffrement des données ($|R^M| = 56614$), le syndrome S émis par les bits de R^M est le poids de Hamming et l'ensemble ζ est formé des pas de temps multiples de $100ps$ entre le début de la première ronde et la fin de la troisième ronde ($|zeta| = 592$). Le masque utilisé pour chiffrer les données est choisi aléatoirement et modifié à chaque exécution.

Le maximum de la fonction $Corr_{T,HW,HW,CP}(r^M, k_0, \delta, SB_0[i], k_0, \delta'_0)$ calculée pour tout $\delta \in \zeta$ et pour tout $r^M \in R^M$, est reporté dans le Tableau 5.11 pour $i \in [0, \ldots, 7]$.

Les valeurs maximales varient entre 0.43 (pour le bit $SB_0[6]$) et 0.6 (pour le bits $SB_0[5]$), ce qui montre que, dans ces conditions, il n'existe pas de bits parfaitement corrélés à des bits d'attaque connus.

Bit d'attaque	$SB_0[0]$	$SB_0[1]$	$SB_0[2]$	$SB_0[3]$	$SB_0[4]$	$SB_0[5]$	$SB_0[6]$	$SB_0[7]$
Max corrélation	0.5	0.58	0.56	0.47	0.51	0.6	0.43	0.59

Tab. 5.11 – Valeurs maximales de corrélations pour l'AES masqué niveau portes avec rétro-annotation pendant les 3 premières rondes pour les 8 bits en sortie de la première S-Box

La Figure 5.13 représente le maximun de corrélations obtenu par l'ensemble des bits de R^M avec le bit d'attaque $SB_0[5]$ en fonction du temps. On constate que le maximum de corrélations, de 0.6, se produit à l'instant $683ns$, fin du calcul de la première ronde. Ce maximum est obtenu pour les 4 bits $state\_o[58]$, $state\_o[61]$, $state\_o[123]$ et $state\_o[126]$, lesquels apparaissent comme des bits potentiellement sensibles.

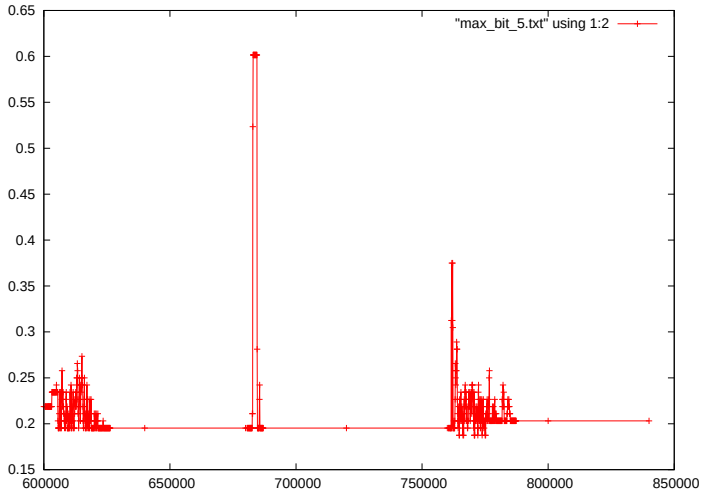

Fig. 5.13 – Valeurs maximales de corrélations pour le bit d'attaque $SB_0[5]$, clef k_0

Le calcul du vecteur $\Theta = (r^M, r^{M'} = SB_0^0[5], k' \in K')$ a été réalisé pour l'ensemble de ces bits à l'instant $\delta = 683ns$. La Figure 5.14 représente l'ensemble des corrélations ainsi obtenu pour le bit $state\_o[123]$. On constate que la clef $2B_h$ présente le maximum de corrélations. Ce bit, comme nous l'avons vérifié pour les 3 autres bits $state\_o[58]$, $state\_o[61]$ et $state\_o[126]$, est donc bien un bit sensible pour cette clef.

L'analyse de corrélations dans l'AES masqué est réalisée avec une autre clef de chiffrement. Le calcul de la fonction de corrélation $Corr_{T,HW,HW,CP}(r^M, k_1, 683, SB_0[5], k_1, \delta'_0)$ pour la clef de chiffrement $k1 = 07260189F4A549E601EA6C763663085A$ donne des va-

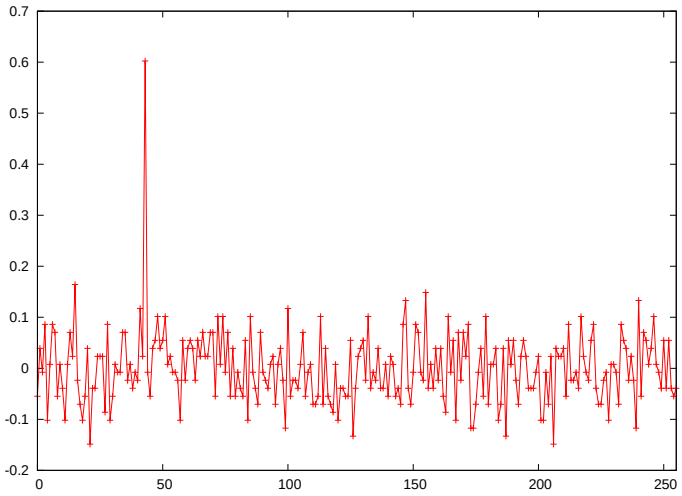

Fig. 5.14 – Valeurs de corrélations pour le bit $state\_o[123]$ avec $SB_0^0[5]$ à l'instant $683ns$

leurs de corrélations qui varient entre -0.3 et 0.2. Cette étude révèle que, pour cette clef, l'AES masqué ne présente pas de bits sensibles.

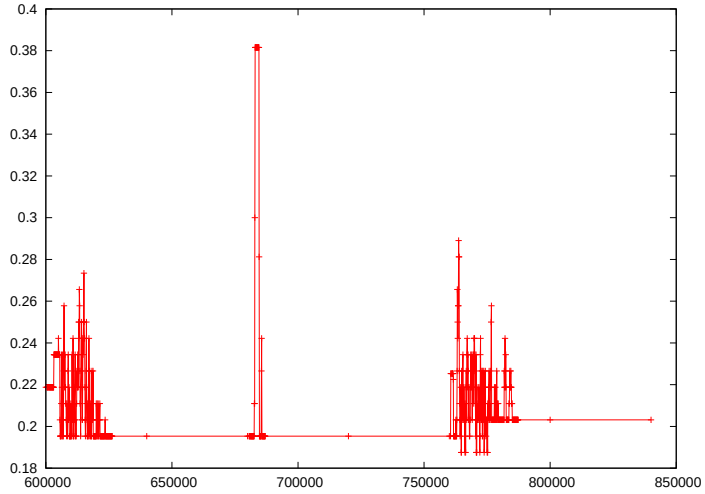

Fig. 5.15 – Valeurs maximales de corrélations pour le bit d'attaque $SB_0[5]$, clef k_1

Les deux analyses réalisées montrent que la sensibilité de certains bits internes à l'AES dépend de la clef qui est utilisée pour le chiffrement. Cette dépendance est certainement due à des calculs anticipés ou à des calculs intermédiaires parasites ("glitchs"), phénomènes temporels susceptibles d'apparaître pour des jeux de données particuliers.

5.3.4.2 Syndrome de distance de Hamming

Pour cette étude, le syndrome S émis par les bits de R^M est la distance de Hamming. Le calcul de la fonction de corrélations $Corr_{T,HD,HW,CP}(r^M, k_0, \delta, r^{M'}, k_0, \delta'_0)$ pendant les 3 premières rondes de l'AES masqué avec une résolution égale à $100ps$ met en valeur des bits parfaitement corrélés et inversement corrélés. Ces bits sont manipulés au début de la deuxième ronde et sont reportés dans le Tableau 5.12.

Bit d'attaque (première ronde)	Bits sensibles (seconde ronde)
$SB_0[0]$	$SB_4/SBox\_in[0]$, $SB_8/SBox\_in[0]$
$SB_0[1]$	$SB_4/SBox\_in[1]$, $SB_8/SBox\_in[1]$
$SB_0[2]$	$SB_4/SBox\_in[2]$, $SB_8/SBox\_in[2]$
$SB_0[3]$	$SB_4/SBox\_in[3]$, $SB_8/SBox\_in[3]$
$SB_0[4]$	$SB_4/SBox\_in[4]$, $SB_8/SBox\_in[4]$
$SB_0[5]$	$SB_4/SBox\_in[5]$, $SB_8/SBox\_in[5]$
$SB_0[6]$	$SB_4/SBox\_in[6]$, $SB_8/SBox\_in[6]$
$SB_0[7]$	$SB_4/SBox\_in[7]$, $SB_8/SBox\_in[7]$

Tab. 5.12 – Bits sensibles pour l'AES masqué

En première ronde, l'octet i en entrée des S-Boxes vérifie la relation suivante :

$$SB_i/SBox\_in \quad = \quad k_i^0 \oplus m_i^0 \oplus X_i^0 \tag{5.11}$$

k_i^0, m_i^0 X_i^0 désignent respectivement les octets i de la clef, du texte clair et du masque en début de première ronde.

En deuxième ronde, l'octet i en entrée des S-Boxes vérifie la relation suivante :

$$SB_i/SBox\_in \quad = \quad k_i^1 \oplus MC_i^0 \oplus X_i^0 \oplus X_i^1$$

k_i^1 et X_i^1 sont respectivement la clef et le masque en début de seconde ronde.

Rappelons que pour $i \in \{0, 4, 8, 12\}$ (indices correspondant à la première colonne en sortie du `MixColumns`), on a :

$$\begin{aligned}
MC_0^0 &= 02 \bullet (SR_0^0 \oplus SR_4^0) \oplus SR_4^0 \oplus SR_8^0 \oplus SR_{12}^0 \\
MC_4^0 &= SR_0^0 \oplus 02 \bullet (SR_4^0 \oplus SR_8^0) \oplus SR_8^0 \oplus SR_{12}^0 \\
MC_8^0 &= SR_0^0 \oplus SR_4^0 \oplus 02 \bullet (SR_4^0 \oplus SR_8^0) \oplus SR_{12}^0 \\
MC_{12}^0 &= SR_0^0 \oplus SR_4^0 \oplus SR_4^0 \oplus 02 \bullet (SR_8^0 \bullet SR_{12})^0
\end{aligned}$$

Avec

$$\begin{aligned}
SR_0^0 &= SB^{'}(k_0^0 \oplus m_0^0 \oplus X_0^0) \\
SR_4^0 &= SB^{'}(k_5^0 \oplus m_5^0 \oplus X_5^0) \\
SR_8^0 &= SB^{'}(k_{10}^0 \oplus m_{10}^0 \oplus X_{10}^0) \\
SR_{12}^0 &= SB^{'}(k_{15}^0 \oplus m_{15}^0 \oplus X_{15}^0)
\end{aligned}$$

Or, d'après la définition des blocs $SB^{'}$ modifiés, pour tout octet i et ronde j, on a :

$$SB^{'}(k_i^j \oplus m_i^j \oplus X_i^j) \quad = \quad SB(k_i^j \oplus m_i^j) \oplus X_i^j$$

donc

$$\begin{aligned} SR_0^0 &= SB(k_0^0 \oplus m_0^0) \oplus X_0^0 \\ SR_4^0 &= SB(k_5^0 \oplus m_5^0) \oplus X_5^0 \\ SR_8^0 &= SB(k_{10}^0 \oplus m_{10}^0) \oplus X_{10}^0 \\ SR_{12}^0 &= SB(k_{15}^0 \oplus m_{15}^0) \oplus X_{15}^0 \end{aligned}$$

Or :

$$\begin{aligned} X_0^1 &= 02 \bullet (X_0^0 \oplus X_5^0) \oplus X_5^0 \oplus X_{10}^0 \oplus X_{15}^0 \\ X_4^1 &= X_0^0 \oplus 02 \bullet (X_5^0 \oplus X_{10}^0) \oplus X_{10}^0 \oplus X_{15}^0 \\ X_8^1 &= X_0^0 \oplus X_5^0 \oplus 02 \bullet (X_5^0 \oplus X_{10}^0) \oplus X_{15}^0 \\ X_{12}^1 &= X_0^0 \oplus X_5^0 \oplus X_5^0 \oplus 02 \bullet (X_{10}^0 \bullet X_{15})^0 \end{aligned}$$

L'octet 0 en entrée de la S-Box pendant la deuxième ronde vérifie donc l'équation suivante :

$$\begin{aligned} SB_0/SBox\_in &= k_0^1 \oplus X_0^0 \oplus MC_0^0 \oplus X_0^1 \\ SB_0/SBox\_in &= k_0^1 \oplus X_0^0 \\ &\oplus \quad 02 \bullet (SR_0^0 \oplus SR_4^0) \oplus SR_4^0 \oplus SR_8^0 \oplus SR_{12}^0 \\ &\oplus \quad 02 \bullet (X_0^0 \oplus X_5^0) \oplus X_5^0 \oplus X_{10}^0 \oplus X_{15}^0 \\ SB_0/SBox\_in &= k_0^1 \oplus X_0^0 \\ &\oplus \quad 02 \bullet ((SB(k_0^0 \oplus m_0^0) \oplus X_0^0) \oplus (SB(k_5^0 \oplus m_5^0) \oplus X_5^0)) \\ &\oplus \quad SB(k_5^0 \oplus m_5^0) \oplus X_5^0 \\ &\oplus \quad SB(k_{10}^0 \oplus m_{10}^0) \oplus X_{10}^0 \\ &\oplus \quad SB(k_{15}^0 \oplus m_{15}^0) \oplus X_{15}^0 \\ &\oplus \quad 02 \bullet (X_0^0 \oplus X_5^0) \oplus X_5^0 \oplus X_{10}^0 \oplus X_{15}^0 \\ SB_0/SBox\_in &= k_0^1 \oplus X_0^0 \\ &\oplus \quad 02 \bullet ((SB(k_0^0 \oplus m_0^0)) \oplus (SB(k_5^0 \oplus m_5^0))) \\ &\oplus \quad (SB(k_5^0 \oplus m_5^0)) \\ &\oplus \quad (SB(k_{10}^0 \oplus m_{10}^0)) \\ &\oplus \quad (SB(k_{15}^0 \oplus m_{15}^0)) \end{aligned}$$

On a k_0^1, k_5^0, k_{10}^0, k_{15}^0, m_5^0, m_{10}^0 et m_{15}^0 qui sont constants pour toutes les réalisations mais comme X_0^0 ne l'est pas, contrairement à la version de l'AES non masqué, aucun des bits de l'octet $SB_0/SBox\_in$ n'est corrélé aux bits d'attaque $SB(k_0^0 \oplus m_0^0)$.

L'équation ci-dessus et 5.11 permettent de calculer la distance de Hamming du premier octet en entrée de la seconde ronde :

$$\begin{aligned} HD(SB_0/SBox\_in) &= k_0^0 \oplus m_0^0 \oplus k_0^1 \\ &\oplus \quad 02 \bullet ((SB(k_0^0 \oplus m_0^0)) \oplus (SB(k_5^0 \oplus m_5^0))) \\ &\oplus \quad SB(k_5^0 \oplus m_5^0) \\ &\oplus \quad SB(k_{10}^0 \oplus m_{10}^0) \\ &\oplus \quad SB(k_{15}^0 \oplus m_{15}^0) \end{aligned}$$

On a k_5^0, k_{10}^0, k_{15}^0, k_0^1, m_5^0, m_{10}^0 et m_{15}^0 qui sont constants pour toutes les réalisations mais comme m_0^0 ne l'est pas aucun des bits de l'octet $HD(SB_0/SBox\_in)$ n'est corrélé

aux bits d'attaque $SB(k_0^0 \oplus m_0^0)$. Cependant, 5 des bits de cette distance de Hamming est corrélée à $SB(k_0^0 \oplus m_0^0) \oplus m_0^0 \oplus k_0^0$, grandeur qui peut être prédite par l'attaquant car m_0^0 est connu.

De la même manière, l'octet 4 de la S-Box pendant la deuxième ronde vérifie l'équation suivante :

$$\begin{aligned}
SB_4/SBox_in &= k_4^1 \oplus X_4^0 \oplus MC_4^0 \oplus X_4^1 \\
SB_4/SBox_in &= k_4^1 \oplus X_4^0 \\
&\oplus\ SR_0^0 \oplus 02 \bullet (SR_4^0 \oplus SR_8^0) \oplus SR_8^0 \oplus SR_{12}^0 \\
&\oplus\ X_0^0 \oplus 02 \bullet (X_5^0 \oplus X_{10}^0) \oplus X_{10}^0 \oplus X_{15}^0 \\
SB_4/SBox_in &= k_4^1 \oplus X_4^0 \\
&\oplus\ SB(k_0^0 \oplus m_0^0) \oplus X_0^0 \\
&\oplus\ 02 \bullet (SB(k_5^0 \oplus m_5^0) \oplus X_5^0 \oplus SB(k_{10}^0 \oplus m_{10}^0) \oplus X_{10}^0) \\
&\oplus\ SB(k_{10}^0 \oplus m_{10}^0) \oplus X_{10}^0 \\
&\oplus\ SB(k_{15}^0 \oplus m_{15}^0) \oplus X_{15}^0 \\
&\oplus\ X_0^0 \oplus 02 \bullet (X_5^0 \oplus X_{10}^0) \oplus X_{10}^0 \oplus X_{15}^0 \\
SB_4/SBox_in &= k_4^1 \oplus X_4^0 \\
&\oplus\ SB(k_0^0 \oplus m_0^0) \\
&\oplus\ 02 \bullet (SB(k_5^0 \oplus m_5^0) \oplus SB(k_{10}^0 \oplus m_{10}^0)) \\
&\oplus\ SB(k_{10}^0 \oplus m_{10}^0) \\
&\oplus\ SB(k_{15}^0 \oplus m_{15}^0)
\end{aligned}$$

On a k_5^0, k_{10}^0, k_{15}^0, m_5^0 m_{10}^0 et m_{15}^0 qui sont constants pour toutes les réalisations mais comme X_0^0 ne l'est pas,contrairement à la version de l'AES non masqué, aucun des bits de l'octet $SB_0/SBox\_in$ n'est corrélé aux bits d'attaque $SB(k_0^0 \oplus m_0^0)$.

L'équation ci-dessus et 5.11 permettent de calculer la distance de Hamming du cinquième octet en entrée de la seconde ronde :

$$\begin{aligned}
HD(SB_4/SBox_in) &= k_4^0 \oplus m_4^0 \oplus k_4^1 \\
&\oplus\ SB(k_0^0 \oplus m_0^0) \\
&\oplus\ 02 \bullet (SB(k_5^0 \oplus m_5^0) \oplus SB(k_{10}^0 \oplus m_{10}^0)) \\
&\oplus\ SB(k_{10}^0 \oplus m_{10}^0) \\
&\oplus\ SB(k_{15}^0 \oplus m_{15}^0)
\end{aligned}$$

Comme k_4^0, k_5^0, k_{10}^0, k_{15}^0, k_4^1, m_4^0, m_5^0, m_{10}^0 et m_{15}^0 sont constants pour toutes les réalisations, la distance de Hamming de tout bit i de l'octet $SB_4/SBox\_in$ est corrélée au bit d'attaque $SB(k_0^0 \oplus m_0^0)[i]$. De même (les calculs réalisée ci-dessus sont strictement identiques), la distance de Hamming calculée sur tout bit i de l'octet $SB_8/SBox\_in$ est corrélée au bit d'attaque $SB(k_0^0 \oplus m_0^0)[i]$.

Par conséquent, l'AES masqué tel qu'il est décrit en début de chapitre, s'il est implanté dans une technologie qui "consomme" une distance de Hamming des données manipulées, est non résistant à la DPA d'ordre 1 "textes choisis", et ce indépendemment de tout calcul intermédiaire parasite ("glitch") qui se propage dans le circuit.

La présence de bits potentiellement sensibles dans l'AES masqué montre que la contre-mesure basée sur le masquage additif n'est pas résistante aux attaques CPA. Une analyse CPA (basée sur la méthode classique) permet également de constater ce résultat (Figure 5.16). Néanmoins, notre approche permet, en plus, de localiser l'origine de la fuite d'information. Ce qui permet de mieux comprendre les faiblesses de la contre-mesure "masquage" et d'essayer de l'améliorer.

5.3.4.3 Amélioration de la contre-mesure masquage

D'après les équations du paragraphe précédent, la distance de Hamming du bit $SB_4/SBox\_in[0]$ est corrélé à $SB_0[0]$ en raison de la simplification de la valeur du masque lors du calcul de la distance de Hamming. En effet, un seul masque est utilisé pour tout l'AES, le masque de chaque ronde étant une fonction linéaire du masque initial. Afin d'éviter cette simplification du masque lors du calcul de la distance de Hamming, l'idée est d'utiliser un masque différent à chaque ronde.

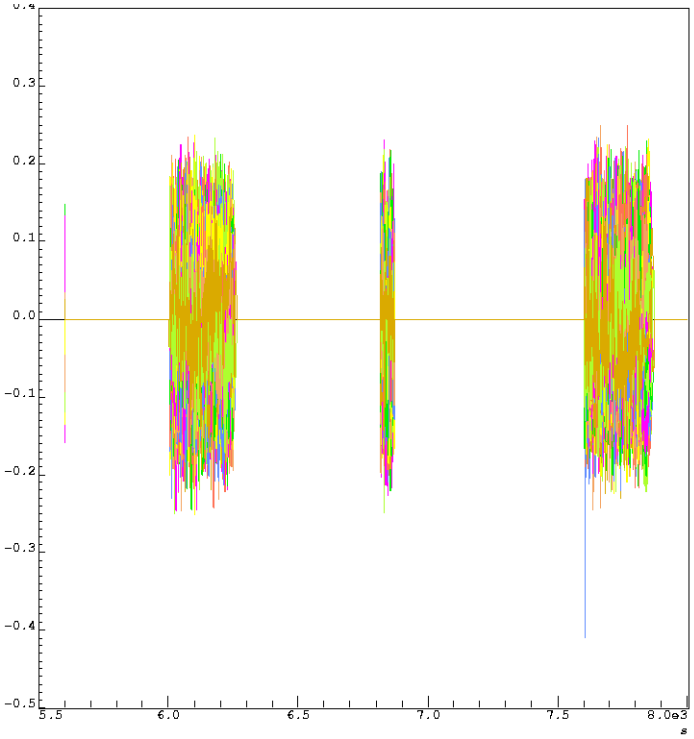

Fig. 5.16 – Courbes CPA sur l'AES masqué

5.4 Bits d'attaque

Le but de cette section est de détecter des nouveaux bits d'attaques non connus dans la littérature et qui sont suceptibles d'être utilisés par un attaquant afin de mener des attaques par corrélation. La recherche de ces nouveaux bits d'attaque se fait conformément aux définitions du bit d'attaque introduit dans le chapitre précédent. Dans ce cas, le modèle M' prédit par l'attaquant est égal au modèle du concepteur M. Par ailleurs, en raison de la complexité de la matrice de corrélations, la recherche de bits d'attaques se fera sur un sous ensemble de bits de M. Ce sous ensemble est formé par les bits d'une des 16 S-Box de l'AES. Deux S-Boxes sont étudiées : la première est celle décrite dans 5.2.2.1 et est basée sur des calculs dans $GF(2^4)$. La seconde est basée sur l'utilisation d'une *Look-Up Table* (LUT).

Les paramètres suivants sont utilisés pour la recherche des bits d'attaque :
- $M = M'$: Description niveau portes de l'AES.
- T : Ensemble des textes clairs qui parcourent l'entrée de la première S-Box soit 256 textes clairs.
- K : Clef de chiffrement k_0 qui est égale à `2B 7E 15 16 28 AE D2 A6 AB F7 15 88 09 CF 4F 3C`.
- K' : Ensemble des 256 hypothèses de clefs partielles qui parcourent l'entrée de la première S-Box.
- $R^M = R^{M'}$: Ensemble des bits de la première S-Box de l'AES. Ce choix se justifie par le fait que beaucoup d'architectures des S-Box ont été rendus publiques, la prédiction des bits d'attaque est donc possible par un attaquant.
- $\zeta = \zeta'$: Fin de la première ronde noté δ_0.
- $S = S'$: Syndrome poids de Hamming.

5.4.1 S-Box $GF(2^4)$

Pour cette étude, l'ensemble R^M est formé de 347 bits. La fonction de corrélation $Correl_{T,HW,HW,CP}(r^M, k_0, \delta_0, r^M, k', \delta_0)$ est calculée pour les différentes hypothèses de clefs partielles. 97 bits de l'ensemble R^M vérifient la relation $|Correl_{T,HW,HW,CP}(r^M, k_0, \delta_0, r^M, k_0, \delta_0)| > Correl_{T,HW,HW,CP}(r^M, k_0, \delta_0, r^M, k', \delta_0)$ pour $k' \neq k_0$. La S-Box présente donc 97 bits d'attaque pour le modèle M, ce qui représente à peu près 28% des bits de la S-Box. Ces bits appartiennent aux blocs "inv_0", "mult_1", "mult_2", "mapping_inverse" et "transformation affine". Ces blocs sont mis en couleur sur la Figure 5.17. Parmi les 97 bits d'attaques, 28 bits sont des entrées/sorties de ces blocs. Il s'agit donc de bits d'attaque qui peuvent être prédits par un attaquant qui ne dispose que d'une description RTL de la S-Box.
La Figure 5.18 est un exemple de bit d'attaque non connu dans la littérature, il s'agit d'un bit en entrée du bloc "mult_2". Il s'agit d'un bit qui peut être prédit par l'attaquant.

5.4.2 S-Box LUT

Le calcul de la fonction de corrélation $Correl_{T,HW,HW,CP}(r^M, k_0, \delta_0, r^{M'}, k', \delta_0)$ pour les 856 bits de l'ensemble $R^M = R^{M'}$ et pour les 256 hypothèses de clefs partielles révèle 118 bits d'attaque. La Figure 5.19 est un exemple de bit d'attaque à l'intérieur de la S-Box. La courbe $\{r^M, Correl_{T,HW,HW,CP}(r^M, k_0, \delta_0, n858, k', \delta_0)\}$ présente deux pics : le premier correspond à la manipulation du bit d'attaque et le second est un bit sensible à ce dernier.

13.75% des bits qui constitutent la S-Box LUT sont des bits d'attaque, elle est donc plus robuste que la S-Box $GF(2^4)$ laquelle a 38% de ses bits qui sont des bits d'attaque.

5.5 Conclusion

Dans ce chapitre, la résistance de différents modèles du crypto-processeur AES avec et sans contre-mesures a été évaluée. L'étude de 3 implémentations de l'AES qui différent de

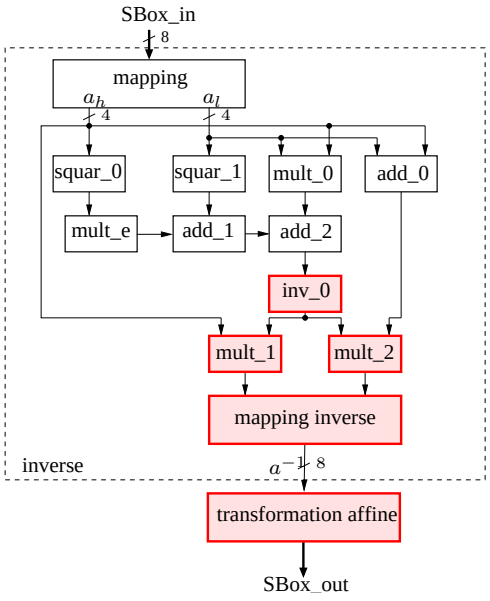

Fig. 5.17 – Bits d'attaques dans la S-Box $GF(2^4)$

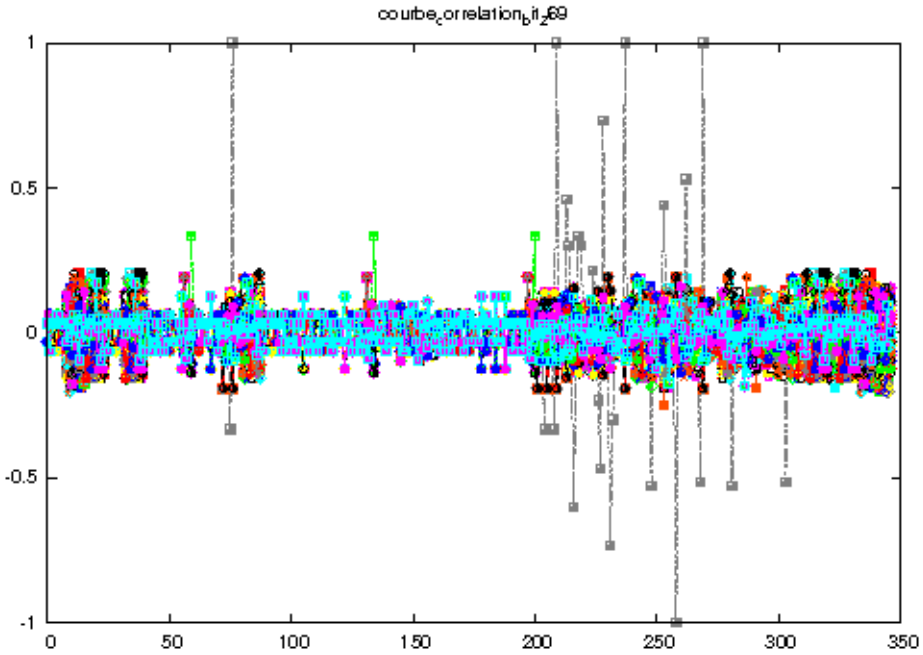

Fig. 5.18 – Courbes $\{r^M, Correl_{T,HW,HW,CP}(r^M, k_0, \delta_0, mult\_2/mult\_in1[3], k', \delta_0)\}$ pour la S-Box $GF(2^4)$

l'architecture de la transformation `MixColumns` au niveau RTL et portes a révélé l'impact du numéro du bit d'attaque et de l'architecture du `MixColumns` sur le niveau de sécurité

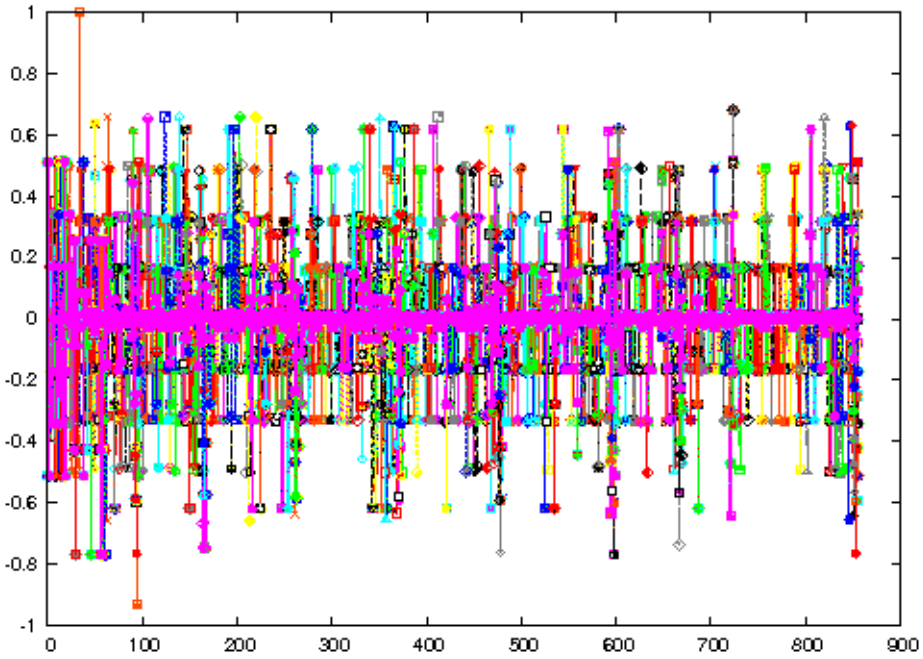

FIG. 5.19 – Courbes $\{r^M, Correl_{T,HW,HW,CP}(r^M, k_0, \delta_0, n858, k', \delta_0)\}$ pour la S-Box LUT

de ces derniers.
L'évaluation de la contre-mesure "dual" après synthèse logique en tenant compte du fichier d'annotations temporelles a montré des problèmes de dissymétries structurelles dans l'AES. La contre-mesure "masquage", telle qu'elle a été implantée dans notre étude, est elle aussi vulnérable aux attaques CPA. En effet, l'étude d'une description niveau portes d'un AES masqué révèle des bits sensibles dans ce dernier.
Enfin, la recherche de nouveaux bits d'attaque dans la S-Box [71] a révélé des bits d'attaques à l'intérieur de cette dernière. Parmi eux, des bits peuvent être prédits par l'attaquant s'il dispose simplement d'une description RTL de la S-Box.

Conclusion et perspectives

Le talon d'achille des circuits cryptographiques réside incontestablement dans leurs implémentations matérielles. En effet, les faiblesses des implémentations physiques des algorithmes cryptographiques ont été exploitées pour mener des attaques et retrouver les secrets. Nous avons expliqué que parmi ces attaques physiques, celles par analyse de "corrélations" représentent un véritable danger pour la sécurité des circuits intégrés dans la mesure où elles permettent, à un coût relativement réduit, d'obtenir des informations secrètes sur les algorithmes cryptographiques comme le DES et l'AES. Ces attaques regroupent les attaques par analyse différentielle de courant (DPA, CPA), d'émissions électromagnétiques (DEMA), de comportement (DBA) et les attaques par micro-sondage.

Nous avons également listé les solutions qui existent pour protéger les circuits intégrés contre ces attaques. Parmi les contre-mesures proposées, certaines ont pour but de réduire la corrélation entre les données manipulées et les signaux compromettants, en équilibrant les données notamment grâce au codage double rail, d'autres cherchent à rendre non prédictibles les variables intermédiaires par le biais de techniques de masquage. Cependant, ces contre-mesures présentent un coût matériel important. En effet, nous avons montré que les contre-mesures basées sur le masquage et le dual multiplient la surface du circuit par deux et détériorent leur vitesse de fonctionnement.

Pour évaluer l'efficacité de ces contre-mesures, des outils d'évaluation sécuritaire ont été proposés dans la littérature. Ceux-ci peuvent être divisés en deux catégories : la première, dite fonctionnelle, consiste à réaliser des attaques sur des signaux "globaux" comme des courbes de courant ou d'émissions électromagnétiques issues de simulations numériques ou électriques. Le nombre de courbes nécessaires pour récupérer la clef est alors considéré comme critère de sécurité. La seconde catégorie, dite structurelle, évalue la robustesse des circuits implantés en logique double rail à partir de recherche de déséquilibres "locaux" susceptibles d'apparaître sur les rails (dues, par exemple, à des différences entre les capacités de charges, les temps de transition ou les temps d'arrivées des signaux). L'approche proposée dans le cadre de cette thèse emprunte l'aspect "analyse fonctionnelle" à la première catégorie et l'aspect recherche "locale" des fuites d'information de la seconde : Il s'agit en pratique de réaliser des attaques sur un ensemble de signaux élémentaires pris un à un.

Dans un premier temps, nous avons proposé un formalisme commun à toutes les attaques par corrélation et à l'évaluation a priori de contre-mesures à ces dernières. Ce formalisme a mis en lumière la complexité des calculs nécessaires à cette évaluation a priori, notamment pour des modèles décrits au niveau RTL ou portes. Des simplifications

ont alors été proposées, permettant ainsi de se focaliser sur l'ensemble des bits qu'un attaquant est susceptible de prédire, appelés bits d'attaque et sur l'ensemble des bits corrélés à ces derniers, appelés bits sensibles.

Dans un second temps, l'efficacité de notre approche de recherche de fuite d'information "locale" a été démontrée sur un modèle de l'AES ne comportant aucune contre-mesure. Elle nous a permis de détecter très précisement l'ensemble des bits qui fuient de l'information. Nous avons également constaté que certaines de ces fuites n'étant pas détectables par analyse classique de courbes CPA.

Dans un troisième temps, l'analyse de corrélations , réalisée sur plusieurs modèles de l'AES avec et sans contre-mesures et décrits au niveau RTL et portes, a permis :

- de mettre en évidence l'influence de l'architecture choisie pour implémenter la fonction `MixColumns` sur la résistance de l'AES face aux attaques en corrélation.
- de confirmer que la phase de synthèse logique est une étape cruciale lors de la conception. En effet, non seulement les options de synthèse jouent grandement sur la sécurité (par exemple, la synthèse "à plat", en simplifiant des calculs intermédiaires, sources potentielles de fuites d'information, augmente le niveau de sécurité) mais aussi que cette étape peut mettre à mal les efforts de symétrisation engagés au niveau RTL.
- d'identifier des bits d'attaque autres que ceux en sorties des S-Box. Ces nouveaux bits d'attaques peuvent être prédits par un attaquant qui dispose d'une description RTL de la S-Box.

Une suite de ces travaux de recherche consiste à observer l'impact de la phase de placement routage sur la sécurité des circuits cryptographiques. Une application de notre outil à des niveaux d'abstraction plus bas est également envisageable. En effet, une recherche de corrélations entre les noeuds d'une netlist SPICE peut être réalisée en se basant sur les niveaux de courant ou de tension au lieu des niveaux logiques.
Enfin, un circuit comportant le crypto-processeur AES est en cours d'évaluation, ce qui permettra de confronter les résultats de simulation aux résultats obtenus sur silicium.

Bibliographie

[1] *ISO/IEC 7816-1*, chapter Identification Cards-Integrated Circuit(s) Cards with Contacts – Part 1 : Physical characteristics. 1998.

[2] *ISO/IEC 7816-2*, chapter Identification Cards-Integrated Circuit(s) Cards with Contact – Part 2 : Dimensions and location of the contacts. 2007.

[3] D. Agrawal, B. Archambeault, J.R. Rao, and P. Rohatgi. The EM Side-Channel (s). *Cryptographic Hardware and Embedded Systems-Ches 2002 : 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002 : Revised Papers*, 2002.

[4] H. Bar-El, H. Choukri, D. Naccache, M. Tunstall, C. Whelan, D.T. Ltd, and I. Rehovot. The sorcerer's apprentice guide to fault attacks. *Proceedings of the IEEE*, 94(2) :370–382, 2006.

[5] Guido Bertoni, Luca Breveglieri, Israel Koren, Paolo Maistri, and Vincenzo Piuri. On the propagation of faults and their detection in a hardware implementation of the advanced encryption standard. In *ASAP '02 : Proceedings of the IEEE International Conference on Application-Specific Systems, Architectures, and Processors*, page 303, Washington, DC, USA, 2002. IEEE Computer Society.

[6] E. Biham and A. Shamir. Differential cryptanalysis of DES-like cryptosystems. *Journal of Cryptology*, 4(1) :3–72, 1991.

[7] Eli Biham and Adi Shamir. Differential fault analysis of secret key cryptosystems. In *CRYPTO*, pages 513–525. Springer, 1997.

[8] D. Binder, EC Smith, AB Holman, et al. Satellite anomalies from galactic cosmic rays. *IEEE Trans. Nucl. Sci*, 22(6) :2675–2680, 1975.

[9] J. Blöemer and J.-P. Seifert. Fault based cryptanalysis of the advanced encryption standard. Cryptology ePrint Archive, Report 2002/075, 2002. `http://eprint.iacr.org/`.

[10] Johannes Blomer and Volker Krummel. Fault based collision attacks on aes. In *FDTC*, pages 106–120, 2006.

[11] D. Boneh, R.A. DeMillo, and R.J. Lipton. On the importance of checking cryptographic protocols for faults. *Lecture Notes in Computer Science*, 1233 :37–51, 1997.

[12] G. Fraidy Bouesse, Marc Renaudin, Sophie Dumont, and Fabien Germain. Dpa on quasi delay insensitive asynchronous circuits : formalization and improvement. *CoRR*, abs/0710.3443, 2007.

[13] E. Brier, C. Clavier, and F. Olivier. Correlation Power Analysis with a Leakage Model. *Cryptographic Hardware and Embedded Systems–CHES 2004 : 6th International Workshop, Cambridge, MA, USA, August 11-13, 2004 : Proceedings*, 2004.

[14] M. Bucci, M. Guglielmo, R. Luzzi, and A. Trifiletti. A Power Consumption Randomization Countermeasure for DPA-Resistant Cryptographic Processors. *Proc. Int. l Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS'04)*, pages 481–490.

[15] M. Bucci, R. Luzzi, F. Menichelli, R. Menicocci, M. Olivieri, and A. Trifiletti. Testing power-analysis attack susceptibility in register-transfer level designs. *Information Security, IET*, 1(3) :128–133, 2007.

[16] S. Chari, J.R. Rao, and P. Rohatgi. Template Attacks. *Cryptographic Hardware and Embedded Systems-Ches 2002 : 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002 : Revised Papers*, 2002.

[17] H. Choukri and M. Tunstall. Round reduction using faults. *2 ndInternational Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC05), Edinburgh, Scotland, September*, 2005.

[18] C. Clavier, J.S. Coron, and N. Dabbous. Differential Power Analysis in the Presence of Hardware Countermeasures. *Cryptographic Hardware and Embedded Systems–CHES 2000 : Second International Workshop, Worcester, MA, USA, August 17-18, 2000 : Proceedings*, 2000.

[19] J.F. Dhem, F. Koeune, P.A. Leroux, P. Mestre, J.J. Quisquater, and J.L. Willems. A Practical Implementation of the Timing Attack. *Proceedings of CARDIS*, pages 167–182, 1998.

[20] W. Diffie and M. Hellman. New directions in cryptography. *Information Theory, IEEE Transactions on*, 22(6) :644–654, 1976.

[21] P. Dusart, G. Letourneux, and O. Vivolo. Differential Fault Analysis on AES. *Applied Cryptography and Network Security : First International Conference, Acns 2003, Kunming, China, October 16-19, 2003 : Proceedings*, 2003.

[22] Olivier Faurax. Pafi : Outil d'analyse de circuit pour l'accélération de l'injection de fautes en simulation. In *Actes des 10èmes Journées Nationales du Réseau Doctoral en Microélectronique (JNRDM)*, May 2007.

[23] Pierre-Alain Fouque, Gwenaëlle Martinet, and Guillaume Poupard. Attacking Unbalanced RSA–CRT Using SPA. *j-LECT-NOTES-COMP-SCI*, 2779 :254–268, 2003.

[24] J. J. A. Fournier, S. Moore, H. Li, R. Mullins, and G. Taylor. Security Evaluation of Asynchronous Circuits. *Proceedings of Cryptographic Hardware and Embedded Systems-CHES2003*, pages 137–151, 2003.

[25] K. Gandolfi, C. Mourtel, and F. Olivier. Electromagnetic Analysis : Concrete Results. *Cryptographic hardware and embedded systems-CHES 2001 : Third International Workshop, Paris, France, May 14-16, 2001 : Proceedings*, 2001.

[26] Christophe Giraud. Dfa on aes. In *AES Conference*, pages 27–41. Springer, 2004.

[27] Jovan Dj. Golic and Christophe Tymen. Multiplicative masking and power analysis of aes. In *CHES '02 : Revised Papers from the 4th International Workshop on Cryptographic Hardware and Embedded Systems*, pages 198–212, London, UK, 2003. Springer-Verlag.

[28] Sylvain Guilley, Philippe Hoogvorst, Yves Mathieu, and Renaud Pacalet. The "backend duplication" method. In *CHES*, pages 383–397, 2005.

[29] Sylvain Guilley, Philippe Hoogvorst, Yves Mathieu, Renaud Pacalet, and Jean Provost. Cmos structures suitable for secured hardware. In *DATE '04 : Proceedings of the conference on Design, automation and test in Europe*, page 21414, Washington, DC, USA, 2004. IEEE Computer Society.

[30] Marc Joye, Pascal Manet, and Jean-Baptiste Rigaud. Strengthening hardware aes implementations against fault attacks. *IET Information Security*, 1(3) :106–110, September 2007.

[31] Ramesh Karri, Grigori Kuznetsov, and Michael Gössel. Parity-based concurrent error detection of substitution-permutation network block ciphers. In *CHES*, pages 113–124, 2003.

[32] Ramesh Karri, Kaijie Wu, Piyush Mishra, and Yongkook Kim. Concurrent error detection schemes for fault-based side-channel cryptanalysis of symmetric block ciphers. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 21(12) :1509–1517, 2002.

[33] N. Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48(177) :203–209, 1987.

[34] P. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. *Advances in Cryptology–CRYPTO*, 96 :104–113, 1996.

[35] P.C. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis. *Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, pages 388–397, 1999.

[36] O. Kommerling and M.G. Kuhn. Design Principles for Tamper-Resistant Smartcard Processors. *USENIX Workshop on Smartcard Technology*, pages 9–20, 1999.

[37] T. H. Le, J. Clédière, C. Servière, and J. L. Lacoume. Higher order statistics for side channel analysis enhancement. *Proceedings of e-Smart 2006*, 2006.

[38] H. Li and S. Moore. Security evaluation at design time against optical fault injection attacks. *IEE Proceedings - Information Security*, 153(1) :3–11, 2006.

[39] Fernanda Lima, Luigi Carro, and Ricardo Reis. Designing fault tolerant systems into sram-based fpgas. In *DAC '03 : Proceedings of the 40th annual Design Automation Conference*, pages 650–655, New York, NY, USA, 2003. ACM.

[40] F. Mace, F. Standaert, and J. J. Quisquater. Information Theoretic Evaluation of Side-Channel Resistant Logic Styles. *LECTURE NOTES IN COMPUTER SCIENCE*, 4727 :427, 2007.

[41] Tal Malkin, François-Xavier Standaert, and Moti Yung. A comparative cost/security analysis of fault attack countermeasures. In Luca Breveglieri, Israel Koren, David Naccache, and Jean-Pierre Seifert, editors, *FDTC*, volume 4236 of *Lecture Notes in Computer Science*, pages 159–172. Springer, 2006.

[42] S. Mangard. A Simple Power-Analysis (SPA) Attack on Implementations of the AES Key Expansion. *Proceedings of the 5th International Conference on Information Security and Cryptology-ICISC*, pages 28–29, 2002.

[43] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power Analysis Attacks : Revealing the Secrets of Smart Cards (Advances in Information Security)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.

[44] Stefan Mangard, Thomas Popp, and Berndt M. Gammel. Side-Channel Leakage of Masked CMOS Gates. In Alfred Menezes, editor, *Topics in Cryptology - CT-RSA 2005, The Cryptographers' Track at the RSA Conference 2005, San Francisco, CA, USA, February 14-18, 2005, Proceedings*, volume 3376 of *Lecture Notes in Computer Science*, pages 351–365. Springer, 2005.

[45] M. Matsui. Linear cryptanalysis method for DES cipher. *Workshop on the theory and application of cryptographic techniques on Advances in cryptology table of contents*, pages 386–397, 1994.

[46] TC May and MH Woods. Alpha-particle-induced soft errors in dynamic memories. *Electron Devices, IEEE Transactions on*, 26(1) :2–9, 1979.

[47] Francesco Menichelli, Renato Menicocci, Mauro Olivieri, and Alessandro Trifiletti. High-level side-channel attack modeling and simulation for security-critical systems on chips. *IEEE Transactions on Dependable and Secure Computing*, 5(3) :164–176, 2008.

[48] Thomas S. Messerges, Ezzy A. Dabbish, and Robert H. Sloan. Investigations of power analysis attacks on smartcards. In *In USENIX Workshop on Smartcard Technology*, pages 151–162, 1999.

[49] V.S. Miller. Use of elliptic curves in cryptography. *Lecture notes in computer sciences ; 218 on Advances in cryptology—CRYPTO 85 table of contents*, pages 417–426, 1986.

[50] Y. Monnet, M. Renaudin, and R. Leveugle. Asynchronous circuits transient faults sensitivity evaluation. *Proceedings of the 42nd annual conference on Design automation*, pages 863–868, 2005.

[51] Yannick Monnet, Marc Renaudin, and Regis Leveugle. Designing resistant circuits against malicious faults injection using asynchronous logic. *IEEE Transactions on Computers*, 55(9) :1104–1115, 2006.

[52] Roman Novak. SPA-Based Adaptive Chosen-Ciphertext Attack on RSA Implementation. In *PKC '02 : Proceedings of the 5th International Workshop on Practice and Theory in Public Key Cryptosystems*, pages 252–262, London, UK, 2002. Springer-Verlag.

[53] V. Ocheretnij, G. Kouznetsov, R. Karri, and M. Gossel. On-line error detection and bist for the aes encryption algorithm with different s-box implementations. In *IOLTS '05 : Proceedings of the 11th IEEE International On-Line Testing Symposium*, pages 141–146, Washington, DC, USA, 2005. IEEE Computer Society.

[54] Siddika Berna Örs, Frank Gürkaynak, Elisabeth Oswald, and Bart Preneel. Power-analysis attack on an asic aes implementation. In *ITCC '04 : Proceedings of the International Conference on Information Technology : Coding and Computing (ITCC'04) Volume 2*, page 546, Washington, DC, USA, 2004. IEEE Computer Society.

[55] Elisabeth Oswald, Stefan Mangard, and Norbert Pramstaller. A side-channel analysis resistant description of the aes s-box. In *Fast Software Encryption 2005, LNCS 3557*, pages 413–423. Springer, 2005.

[56] Gilles Piret and Jean-Jacques Quisquater. A differential fault attack technique against spn structures, with application to the aes and khazad. In *CHES*, pages 77–88. Springer, 2003.

[57] J.J. Quisquater and D. Samyde. ElectroMagnetic Analysis (EMA) : Measures and Counter-Measures for Smart Cards. *Proceedings of the International Conference on Research in Smart Cards : Smart Card Programming and Security*, pages 200–210, 2001.

[58] Alin Razafindraibe, Michel Robert, and Philippe Maurine. Analysis and improvement of dual rail logic as a countermeasure against dpa. In *PATMOS*, pages 340–351, 2007.

[59] RL Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications*, 1978.

[60] B. Robisson and P. Manet. Differential Behavioral Analysis. *LECTURE NOTES IN COMPUTER SCIENCE*, 4727 :413, 2007.

[61] B. Robisson, P. Manet, and S. Laabidi. Differential Behavioral Analysis Application to an asynchronous crypto-processor. *Presented at USE-IT workshop, Toulouse, July 2007*.

[62] K. Schramm, G. Leander, P. Felke, and C. Paar. A Collision-Attack on AES Combining Side Channel-and Differential-Attack. *Cryptographic Hardware and Embedded Systems–CHES 2004 : 6th International Workshop, Cambridge, MA, USA, August 11-13, 2004 : Proceedings*, 2004.

[63] K. Schramm, T. Wollinger, and C. Paar. A New Class of Collision Attacks and Its Application to DES. *Fast Software Encryption : 10th International Workshop, FSE 2003, Lund, Sweden, February 24-26, 2003 : Revised Papers*, 2003.

[64] C.E. Shannon. Communication theory of secrecy systems. *Bell System Technical Journal*, 28(4) :656–715, 1949.

[65] Claude E. Shannon. Communication theory of secrecy systems. *Bell Systems Technical Journal*, 28(4) :656—715, 1949.

[66] S.P. Skorobogatov and R.J. Anderson. Optical Fault Induction Attacks. *Cryptographic Hardware and Embedded Systems-Ches 2002 : 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002 : Revised Papers*, 2002.

[67] D.E. Standard. Data Encryption Standard. *National Bureau of Standards, US Department of Commerce, Washington DC, Jan*, 1977.

[68] N.F. Standard. Announcing the ADVANCED ENCRYPTION STANDARD (AES). *Federal Information Processing Standards Publication*, 197, 2001. http ://csrc.nist.gov/publications/fips/fips197/fips-197.pdf.

[69] K. Tiri and I. Verbauwhede. A VLSI Design Flow for Secure Side-Channel Attack Resistant ICs. *Design, Automation, and Test in Europe : Proceedings of the conference on Design, Automation and Test in Europe-*, 3 :58–63, 2005.

[70] K. Tiri and I. Verbauwhede. Simulation models for side-channel information leaks. *Proceedings of the 42nd annual conference on Design automation*, pages 228–233, 2005.

[71] Johannes Wolkerstorfer, Elisabeth Oswald, and Mario Lamberger. An asic implementation of the aes sboxes. In *CT-RSA '02 : Proceedings of the The Cryptographer's Track at the RSA Conference on Topics in Cryptology*, pages 67–78, London, UK, 2002. Springer-Verlag.

[72] Kaijie Wu, Ramesh Karri, Grigori Kuznetsov, and Michael Goessel. Low cost concurrent error detection for the advanced encryption standard. In *ITC '04 : Proceedings of the International Test Conference on International Test Conference*, pages 1242–1248, Washington, DC, USA, 2004. IEEE Computer Society.

[73] S.M. Yen and M. Joye. Checking before output may not be enough against fault-based cryptanalysis. *IEEE Transactions on Computers*, 49(9) :967–970, 2000.

[74] Sung-Ming Yen, Dongryeol Kim, and Sang-Jae Moon. Cryptanalysis of two protocols for rsa with crt based on fault infection. In *FDTC*, pages 53–61, 2006.

[75] Sung-Ming Yen, Seungjoo Kim, Seongan Lim, and SangJae Moon. Rsa speedup with residue number system immune against hardware fault cryptanalysis. In *ICISC '01 : Proceedings of the 4th International Conference Seoul on Information Security and Cryptology*, pages 397–413, London, UK, 2002. Springer-Verlag.

[76] JF Ziegler and WA Lanford. Effect of Cosmic Rays on Computer Memories. *Science*, 206(4420) :776–788, 1979.