Sécurité adaptative et énergétiquement efficace dans l'Internet des Objets

Maxime Montoya

▶ **To cite this version:**

Maxime Montoya. Sécurité adaptative et énergétiquement efficace dans l'Internet des Objets. Autre. Université de Lyon, 2019. Français. NNT : 2019LYSEM032 . tel-02966640

HAL Id: tel-02966640
https://theses.hal.science/tel-02966640

Submitted on 14 Oct 2020

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.

N°d'ordre NNT : 2019LYSEM032

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

Ecole Doctorale N° 488
Sciences, Ingénierie, Santé

Spécialité de doctorat : Microélectronique

Soutenue publiquement le 06/12/2019, par :
Maxime Montoya

Sécurité adaptative et énergétiquement efficace dans l'Internet des Objets

---

Devant le jury composé de :

Rouzeyre, Bruno	Professeur	Univ. Montpellier 2	Président
Tisserand, Arnaud	Directeur de recherche	CNRS	Rapporteur
Hély, David	Maître de conférences	Grenoble-INP	Rapporteur
Guilley, Sylvain	Professeur	TELECOM-ParisTech	Examinateur
Bossuet, Lilian	Professeur	Univ. St-Etienne	Examinateur
Moro, Nicolas	Ingénieur de recherche	IMEC	Examinateur
Fournier, Jacques	Ingénieur de recherche	CEA	Directeur de thèse
Bacles-Min, Simone	Ingénieure de recherche	CEA	Encadrante
Molnos, Anca	Ingénieure de recherche	CEA	Invitée

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

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

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

Mise à jour : 30/08/2018

Remerciements

Je souhaite tout d'abord remercier Arnaud Tisserand, David Hély, Sylvain Guilley, Lilian Bossuet, Bruno Rouzeyre, et Nicolas Moro pour avoir accepté de faire partie de mon jury de thèse, et pour les discussions que nous avons eues suite à la présentation de mes travaux. En particulier, merci à Arnaud Tisserand et David Hély pour leur relecture de l'ensemble de mon manuscrit de thèse et pour leurs retours.

Mes travaux ont été encadrés par trois personnes aux sensibilités et aux domaines d'expertise différents, mais complémentaires. Je souhaite les remercier tous les trois pour avoir grandement contribué à la réussite de ma thèse. Jacques Fournier, grâce à son expertise en cybersécurité, a contribué à éprouver mes idées et à faire mûrir ma pensée au cours de ces trois années, ainsi qu'à améliorer la qualité scientifique des documents rédigés durant cette période. Anca Molnos m'a fait réfléchir à l'utilité des solutions proposées et aux différents cas d'usage possibles, m'a aidé à mieux mettre en perspective mes travaux, et m'a été d'une aide précieuse pour la rédaction de documents en anglais. Enfin, je remercie particulièrement Simone Bacles-Min pour ses conseils tant sur le plan humain que technique, son temps passé à discuter mes idées et leur mise en œuvre, et plus généralement pour sa disponibilité, son enthousiasme, et son énergie.

Cette thèse s'est déroulée dans le laboratoire d'intégration sur silicium des architectures numériques du CEA LETI. J'aimerais remercier tous mes collègues pour leur accueil chaleureux au sein du laboratoire. J'ai énormément appris à leur contact au cours des trois dernières années, que ce soit sur le flot de conception des circuits intégrés, sur les architectures matérielles, ou plus généralement sur la recherche en microélectronique. Je voudrais particulièrement remercier Nirmal, David, Jean-Fred, Ivan, Sylvain, Emmanuel, Roman, Thanos et Yvain pour m'avoir formé sur des points particuliers, pour m'avoir aidé quand je rencontrais des problèmes techniques, ou tout simplement pour les discussions très enrichissantes que nous avons pu avoir. Enfin, j'ai eu la chance de pouvoir co-encadrer un stagiaire investi et intéressé, Alexandre, que je souhaite également remercier car j'ai beaucoup appris de cette expérience.

Au cours de ma thèse, j'ai également eu l'occasion d'échanger avec des personnes d'autres laboratoires du CEA LETI, dont le laboratoire d'informatique et le laboratoire de cybersécurité. J'aimerais remercier toutes les personnes de ces laboratoires avec qui j'ai eu des discussions souvent très intéressantes. En particulier, je remercie Thomas, grâce à qui j'ai pu effectuer des attaques par analyse des canaux auxiliaires sur FPGA, et qui m'a formé à la méthodologie et au matériel utilisés dans son laboratoire.

Je remercie aussi les amis exceptionnels que j'ai rencontrés au CEA pour tous les bons moments passés ensemble, que ce soit lors des repas de midi, ou en dehors du CEA autour d'un verre, d'un bon repas ou d'un jeu de société. J'espère que nous resterons en contact où que nos chemins nous mènent.

Pour conclure, je souhaite remercier mes proches, qui m'ont soutenu non seulement au cours de ces trois années, mais également depuis bien plus longtemps. Je remercie tout particulièrement mes parents et grands-parents, qui m'ont toujours encouragé dans mes études et poussé à aller de l'avant pour que je m'épanouisse dans mon travail. Et enfin, car l'épanouissement professionnel doit s'accompagner d'une vie privée également épanouie, je remercie ma fiancée Julie, qui m'a soutenu activement dans les bons comme dans les mauvais moments depuis plusieurs années.

Table des matières

Table des matières v

Liste des figures vii

Liste des tableaux ix

1 Introduction 1
- 1.1 L'Internet des Objets . . . 2
- 1.2 Circuits intégrés pour l'Internet des Objets . . . 3
- 1.3 Enjeux de sécurité dans l'Internet des Objets . . . 11
- 1.4 Positionnement de la thèse : de la sécurisation de mécanismes de gestion de l'énergie à l'optimisation énergétique de mécanismes de sécurisation . . . 19
- 1.5 Publications et valorisation . . . 21

2 Sécurisation de mécanismes de gestion de l'énergie : le cas de la radio de réveil 23
- 2.1 Analyse des méthodes existantes . . . 24
- 2.2 Hypothèses effectuées . . . 26
- 2.3 Génération du code de réveil . . . 27
- 2.4 Processus de réveil . . . 29
- 2.5 Analyse de la sécurité . . . 31
- 2.6 Implémentation et évaluation . . . 35
- 2.7 Conclusion . . . 38

3 Optimisation énergétique de mécanismes de sécurisation : le cas des contre-mesures contre les attaques matérielles 41
- 3.1 Attaques par analyse des canaux auxiliaires . . . 42
- 3.2 Cas d'étude : fuites par canaux auxiliaires de l'algorithme Trivium . . . 46
- 3.3 Attaques par injection de fautes . . . 52
- 3.4 Protections contre les attaques par canaux auxiliaires . . . 55
- 3.5 Protections contre les attaques par injection de fautes . . . 68
- 3.6 Protections mixtes . . . 70
- 3.7 Conclusion . . . 73

4 L'encodage dynamique, une contre-mesure mixte pour les algorithmes à base de registres à décalage 75
- 4.1 Encodage dynamique des registres à décalage . . . 76
- 4.2 Sécurité contre les attaques par injection de fautes . . . 79
- 4.3 Encodage dynamique aléatoire . . . 82
- 4.4 Extension à la logique combinatoire . . . 83
- 4.5 Caractérisation et évaluation de l'encodage dynamique . . . 87
- 4.6 Conclusion . . . 98

5 Concept et étude des contre-mesures adaptatives **101**
5.1 Masquage TI de Trivium 103
5.2 Masquage semi-adaptatif, énergétiquement efficace 105
5.3 Masquage adaptatif 106
5.4 Encodage dynamique adaptatif 109
5.5 Caractérisation et évaluation des contre-mesures 110
5.6 Implémentation de sécurité adaptative pour divers cas d'usage 117
5.7 Conclusion 118

6 Évaluation des contre-mesures matérielles sur cibles physiques **121**
6.1 Intégration sur ASIC 123
6.2 Implémentation et évaluation sur FPGA 125
6.3 Conclusion 132

7 Conclusion et perspectives **133**
7.1 Conclusion 133
7.2 Perspectives 136

Liste des figures

1.1 Exemples de domaines d'application de l'IoT [2]. 2
1.2 Différentes topologies de réseaux dans l'IoT. 3
1.3 Contraintes de temps de propagation d'un signal dans la logique combinatoire [6]. . 4
1.4 Architecture de la plateforme L-IoT. 10
1.5 L'algorithme de chiffrement par flot Trivium. 14
1.6 Différentes *couches*, sur lesquelles des attaques peuvent être effectuées. 16
1.7 Nécessité d'un compromis entre consommation d'énergie, sécurité et performances dans l'IoT. 17

2.1 Diagramme de séquence de messages entre deux noeuds lorsque le code de réveil est calculé à partir d'un compteur de réveil, noté SN_A->B [56]. 25
2.2 Diagramme de séquence de messages entre une station centrale A et un nœud B avec notre processus de réveil, pour un fonctionnement normal. 30
2.3 Diagramme de séquence de messages entre une station centrale A et un nœud B avec notre processus de réveil, dans le cas où une resynchronisation est nécessaire. . . . 31
2.4 Architecture du circuit WARRIOR. 35

3.1 Exemple d'évolution temporelle du coefficient de corrélation pour quatre hypothèses de clé (courtoisie du laboratoire LSOSP). 45
3.2 Exemple de trace de consommation pour le Trivium non protégé durant deux cycles d'horloge, avec un point toutes les 10 ps, à une fréquence de 100 MHz. 48
3.3 Évolution du coefficient de corrélation pour une implémentation matérielle de Trivium. 49
3.4 T-tests spécifiques et T-test non-spécifique durant les 1500 premiers cycles d'exécution de Trivium, en simulations post-layout. 51
3.5 Injection de glitchs d'horloge dans un circuit intégré. 53
3.6 Génération de *bruit non corrélé* (à gauche), et de *bruit corrélé* (à droite) [92]. 56
3.7 Contre-mesure consistant à isoler le circuit de l'alimentation externe avec deux capacités qui commutent à tour de rôle [102]. 57
3.8 Le style de logique WDDL [109]. 59
3.9 Encodage différentiel temporel d'un "1" et d'un "0" [114]. 60
3.10 Protection fondée sur des bascules à débit de données double [117]. 61
3.11 Protection d'un registre à décalage grâce à l'ajout de bascules permettant d'équilibrer la distance de Hamming [120]. 63
3.12 Principe du masquage algorithmique [44]. 64
3.13 Contre-mesure mixte, à base d'un masquage algorithmique des données et de codes MAC [143]. 72

4.1 Exemples d'un état interne encodé avec un encodage dynamique 1-sur-4 et un encodage dynamique 2-sur-4. 79
4.2 Exemples de fautes injectées sur exactement deux parts parmi quatre, pour un encodage 1-sur-4. 80
4.3 Tables de vérités pour les fonctions ET, XOR et INV encodées. 85

4.4 Application de l'encodage dynamique à la protection des registres à décalage de Trivium. 88
4.5 Cinq exemples de traces de consommation, en énergie par cycle d'horloge, pour l'implémentation TRI_ENC_COMB. 93
4.6 Valeurs du T-test pour les différentes versions de Trivium, avec et sans encodage dynamique. 95
4.7 Valeurs du T-test pour les différentes versions du registre à décalage, avec et sans encodage dynamique. 95
4.8 Évolution du coefficient de corrélation pour les différentes implémentations de Trivium protégées par l'encodage dynamique. 96
4.9 Évolution du coefficient de corrélation pour les différentes versions du registre à décalage, avec et sans encodage dynamique. 97

5.1 Masquage TI au premier ordre de trivium (en noir), et modifications apportées pour le masquage énergétiquement efficace (en rouge). 106
5.2 Masquage TI au second ordre de trivium (en noir), et modifications apportées pour le masquage adaptatif (en rouge). 107
5.3 Mécanisme permettant d'additionner un masque M^1 au contenu du premier chemin de données, entre les k-ième et $(k+1)$-ième bascules. 108
5.4 Évolution de la consommation des différentes contre-mesures adaptatives par rapport à d'autres contre-mesures ayant un niveau de protection similaire. 112
5.5 Valeurs du T-test pour le Trivium de référence, et lorsque les protections adaptatives sont désactivées. 113
5.6 Valeurs du T-test pour les différentes contre-mesures adaptatives appliquées à Trivium. 114
5.7 T-test bivarié sur le masquage au second ordre de Trivium. 115
5.8 Évolution du coefficient de corrélation pour les contre-mesures adaptatives. 116

6.1 Architecture du bloc fonctionnel contenant trois versions de Trivium, sur le circuit WARRIOR. 124
6.2 Banc de mesure des émissions électromagnétiques sur FPGA (courtoisie du laboratoire LSOSP). 126
6.3 Partitionnement sur le FPGA. 126
6.4 Exemple de trace d'émissions EM pour le masquage adaptatif à l'ordre 2. 127
6.5 T-tests spécifiques et non-spécifiques durant les 1500 premiers cycles d'exécution de Trivium, sur FPGA. 128
6.6 Valeurs du T-test pour les circuits évalués sur FPGA, aux différents niveaux de protection. 129
6.7 T-test bivarié sur le masquage adaptatif au second ordre de Trivium, sur FPGA. 129
6.8 Évolution du coefficient de corrélation pour les circuits évalués sur FPGA, aux différents niveaux de protection. 130

Liste des tableaux

2.1 Temps moyen pour trouver un code de réveil par force brute, en fonction de la taille du code de réveil, pour une radio de réveil ayant un débit en réception de 10 kbps. . 32
2.2 Consommation d'énergie journalière et durée de vie d'un nœud avec et sans protection, et avec et sans attaque par déni de sommeil. 37
2.3 Caractéristiques des méthodes de génération de codes de réveil existantes, et comparaison avec notre solution. 37

3.1 Équations pour σ durant les 81 premiers cycles d'horloge. 48

4.1 Principe de l'encodage dynamique d'un bit x. 78
4.2 Exemple de table d'encodage dynamique 1-sur-4 d'un bit x. 78
4.3 Deux exemples de tables d'encodage dynamique 2-sur-4 d'un bit x. 79
4.4 Équations simplifiées pour les portes ET et XOR, en fonction de la parité relative des entrées $\mathscr{F}_c(x)$ et $\mathscr{F}_c(y)$. 86
4.5 Liste des implémentations étudiées. 90
4.6 Surface après synthèse (GE) pour différentes variantes du registre à décalage, avec et sans encodage dynamique. 91
4.7 Surface après synthèse (GE) pour différentes variantes de Trivium, avec et sans encodage dynamique. 91
4.8 Consommation moyenne simulée (µW) pour différentes variantes du registre à décalage, avec et sans encodage dynamique. 92
4.9 Consommation moyenne simulée (µW) pour différentes variantes de Trivium, avec et sans encodage dynamique. 92
4.10 NED et NSD maximaux. 93
4.11 Maximum du coefficient de corrélation pour les implémentations de référence et les implémentations protégées par l'encodage dynamique. 96

5.1 Surface après synthèse (GE) pour les contre-mesures adaptatives. 111
5.2 Consommation moyenne simulée (µW) pour les contre-mesures adaptatives. 111
5.3 Maximum du coefficient de corrélation pour les contre-mesures adaptatives. 117

6.1 Maximum du coefficient de corrélation, sur FPGA. 131

Chapitre 1

Introduction

Sommaire

1.1 L'Internet des Objets 2
1.2 Circuits intégrés pour l'Internet des Objets 3
1.2.1 Introduction aux circuits intégrés 3
1.2.2 Contraintes sur les circuits intégrés pour l'IoT 8
1.2.3 Exemple de système sur puce pour l'IoT : la plateforme L-IoT 9
1.3 Enjeux de sécurité dans l'Internet des Objets 11
1.3.1 Principes de la sécurité 11
1.3.2 Chiffrement léger pour l'IoT 13
1.3.3 Des attaques à différents niveaux 15
1.3.4 La sécurité, une contrainte supplémentaire pour les circuits intégrés 17
1.4 Positionnement de la thèse : de la sécurisation de mécanismes de gestion de l'énergie à l'optimisation énergétique de mécanismes de sécurisation 19
1.4.1 Sécurité de la radio de réveil pour une gestion efficace de l'énergie 19
1.4.2 Efficacité énergétique des contre-mesures matérielles 20
1.5 Publications et valorisation 21

1.1 L'Internet des Objets

L'Internet des Objets, appelé par la suite IoT pour *Internet of Things*, peut être considéré comme l'extension des principes de l'Internet classique à virtuellement tous les objets qui nous entourent. Ces objets, alors appelés *objets connectés*, traitent des données et communiquent, que ce soit à travers divers types de réseaux privés et publiques ou directement sur Internet. L'IoT est en pleine expansion : selon un rapport de Gartner publié en 2017, on peut s'attendre à avoir 20,4 milliards d'objets connectés d'ici 2020, avec des investissements atteignant déjà 2.000 milliards de dollars en 2017 [1].

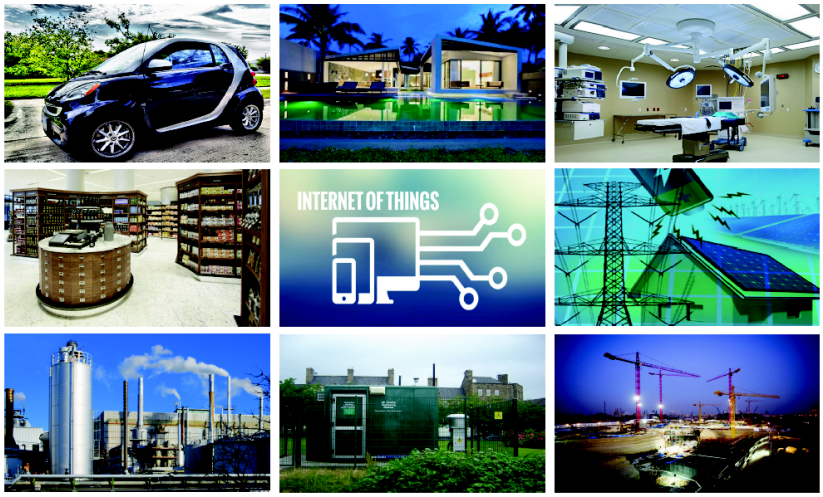


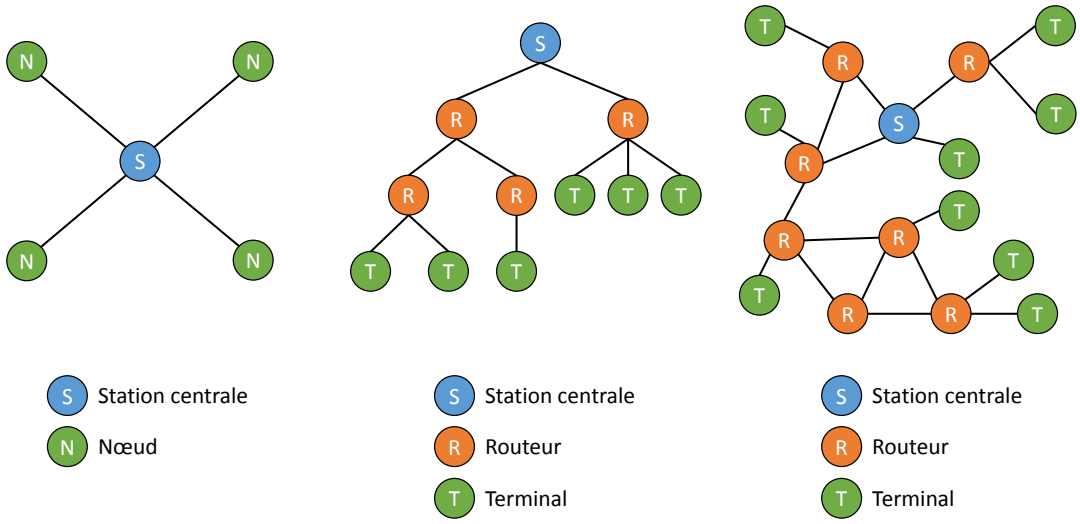
FIGURE 1.1 – Exemples de domaines d'application de l'IoT [2].

L'IoT couvre aujourd'hui une large gamme d'applications dans de nombreux domaines. La figure 1.1, tirée de [2], montre une partie de ces applications dans différents secteurs : les véhicules connectés, les bâtiments intelligents et la domotique, la santé connectée, la gestion de chaînes logistiques, la gestion de l'énergie, l'industrie 4.0, les mesures environnementales, et la gestion de grands chantiers. Cette liste est loin d'être exhaustive, et tous les aspects de notre vie sont en passe d'être touchés par le développement de l'IoT.

Les objets connectés partagent pour la plupart un ensemble de caractéristiques communes. Ainsi, la durée de vie de ces objets peut être longue, allant jusqu'à plusieurs années. De plus, ils opèrent généralement de manière totalement ou partiellement autonome, c'est-à-dire que leur fonctionnement n'est pas supervisé en permanence. Enfin, ces objets étant généralement produits en très grande quantité, leur coût doit être faible.

Une partie d'entre eux disposent de ressources limitées, et notamment d'une quantité d'énergie finie s'ils fonctionnent grâce à une batterie. C'est notamment le cas des objets dits « portables » ou « mettables », également appelés *wearables*, qui peuvent notamment servir à mesurer des caractéristiques physiologiques et des données liées à la santé d'un individu. On peut également citer le cas des réseaux de capteurs sans fil, ou WSN pour *Wireless Sensor Networks* [3], dont chaque entité est constituée au minimum d'un capteur, d'une unité de calcul et d'une radio. Ces objets fonctionnent souvent de manière non supervisée, et une maintenance limitée est apportée à ceux-ci ; en particulier, leur batterie n'est généralement pas remplacée ou rechargée durant toute leur durée de vie.

Les objets connectés font généralement partie de réseaux d'objets. Selon les applications, ces réseaux peuvent être constitués de seulement deux objets (par exemple, un capteur qui communique avec un smartphone), ou de plusieurs milliers, voire millions d'objets, comme c'est le cas pour les réseaux de capteurs sans fil [3, 4]. Au sein d'un tel réseau, un objet connecté est appelé un *nœud*. Les objets sont répartis dans ces réseaux selon plusieurs *topologies* [5], représentées sur la figure 1.2. Dans la topologie *en étoile*, tous les objets connectés communiquent avec une seule station centrale ; les nœuds peuvent avoir des ressources limitées et fonctionner sur batterie, tandis



(a) Topologie en étoile (b) Topologie en arbre (c) Topologie en réseau maillé

FIGURE 1.2 – Différentes topologies de réseaux dans l'IoT.

que cette station centrale est connectée au réseau électrique et dispose de ressources importantes afin de communiquer efficacement avec les différents nœuds. Dans les topologies *en arbre* ou *en réseau maillé*, les nœuds peuvent communiquer entre eux. Les nœuds faisant la liaison entre la station centrale et les autres nœuds sont alors appelés des *routeurs*; en plus d'établir la communication entre plusieurs nœuds, ceux-ci sont généralement capables de réaliser des mesures ou d'effectuer des actions, tout comme les nœuds situés aux extrémités du réseau et appelés *terminaux*.

1.2 Circuits intégrés pour l'Internet des Objets

Dans cette section, nous introduisons tout d'abord les notions sur les circuits intégrés nécessaires pour la compréhension de cette thèse. Nous décrivons ensuite les contraintes qui s'appliquent aux circuits intégrés conçus pour un usage dans l'IoT. Enfin, nous présentons un exemple de circuit intégré pour l'IoT, dont l'architecture est prise comme référence dans nos travaux.

1.2.1 Introduction aux circuits intégrés

Logique combinatoire et logique séquentielle

Les circuits intégrés sont composés de milliers, voire de millions de transistors. Ces transistors sont assemblés pour constituer des portes logiques, qui effectuent des opérations logiques sur des signaux binaires. Il y a deux types de portes logiques, selon qu'elles fassent partie de la *logique combinatoire*, ou de la *logique séquentielle*. Les portes logiques élémentaires qui constituent la logique combinatoire sont utilisées pour effectuer divers calculs sur des bits; ces portes permettent par exemple d'implémenter les opérations booléennes ET, OU, OU EXCLUSIF (notée XOR, pour *eXclusive OR*) ou encore des inversions INV. Ces portes logiques élémentaires peuvent être combinées pour effectuer des opérations plus complexes, telles que des additions ou des multiplications sur des octets. Par la suite, nous appellerons *bloc combinatoire*, ou *fonction combinatoire*, les fonctions réalisées avec de la logique combinatoire. La logique séquentielle contient des portes logiques qui mémorisent l'information. La porte logique élémentaire de la logique séquentielle synchrone est la bascule, qui mémorise le bit qui lui est présenté à chaque front montant d'un signal appelé horloge. La valeur de ce signal d'horloge alterne régulièrement entre "0" et "1", et la succession de ces deux valeurs constitue un cycle d'horloge; ce signal est partagé par l'ensemble

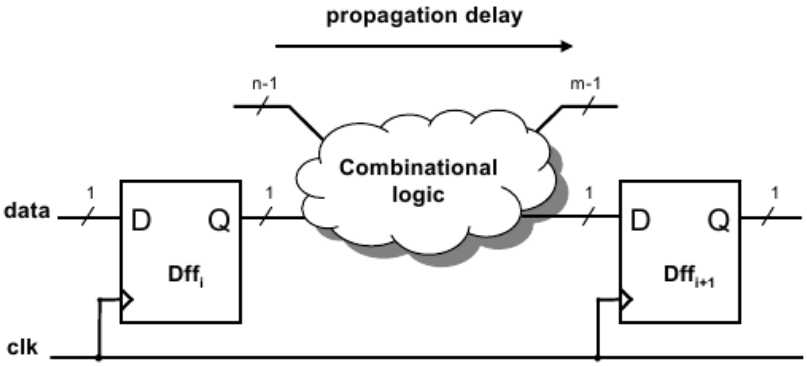


FIGURE 1.3 – Contraintes de temps de propagation d'un signal dans la logique combinatoire [6].

d'un circuit, ou du moins d'une sous-partie de ce circuit, et permet une mise à jour synchrone des données dans toutes les bascules. Lorsque plusieurs bascules sont assemblées pour contenir plusieurs bits, on parle de registre. Dans un registre à décalage, toutes les bascules, sauf la première et la dernière, sont reliées, c'est-à-dire que la sortie de chaque bascule sert d'entrée à la bascule suivante ; le chargement des données dans ce registre à décalage se fait en série, bit par bit, et les bits contenus dans celui-ci sont décalés d'une position vers la sortie à chaque cycle d'horloge.

Un circuit intégré est constitué par plusieurs blocs logiques combinatoires et séquentiels. Les calculs sont effectués dans les blocs logiques combinatoires, et échantillonnés en entrée des blocs séquentiels à chaque front montant de l'horloge. Or, le courant traverse les transistors en un certain temps, non nul, et le temps que met chaque bloc combinatoire à traiter une information dépend donc du nombre de transistors contenus dans ce bloc combinatoire. Ce temps de propagation à travers les blocs combinatoires doit être inférieur à la période de l'horloge, c'est-à-dire au délai entre deux fronts montants du signal d'horloge, faute de quoi les données mémorisées dans les bascules seraient incorrectes. La figure 1.3 représente le fonctionnement d'un tel circuit : à chaque cycle d'horloge, l'information stockée dans la première bascule est mise à jour, et des calculs sont effectués par la logique combinatoire sur cette nouvelle information ; ces calculs doivent être terminés avant leur mémorisation dans la seconde bascule. Le temps de propagation est différent pour chaque bloc combinatoire ; celui qui présente le temps de propagation le plus long est appelé *chemin critique*. Le temps de propagation au sein de celui-ci détermine la période minimale de l'horloge, et donc la fréquence maximale à laquelle le circuit intégré peut fonctionner.

Circuits spécifiques et circuits programmables

Certains circuits intégrés sont conçus pour une application spécifique, et sont appelés ASIC pour *Application-Specific Integrated Circuit*. Ils sont optimisés pour cette application, et offrent donc une surface et une consommation minimale ainsi qu'une fréquence maximale pour l'exécution de celle-ci. Un ASIC contient uniquement les portes logiques élémentaires nécessaires pour une application donnée.

D'autres circuits, appelées FPGA pour *Field-Programmable Gate Arrays*, sont programmables, et diverses applications peuvent donc être effectuées sur ces circuits. Ils sont constitués de nombreuses cellules identiques qui contiennent généralement des tables de correspondances ou des multiplexeurs. Chacune de ces cellules peut être programmée afin d'effectuer une opération logique élémentaire, et les interconnexions entre ces cellules peuvent également être programmées. Du fait de l'utilisation de cellules reprogrammables, comprenant chacune de nombreuses portes logiques, les applications implémentées sur FPGA sont plus coûteuses que si elles étaient intégrées sur ASIC, à la fois en termes de consommation, de surface et de temps d'exécution. Par exemple, les auteurs de [7] ont estimé en 2007 qu'une même application implémentée sur ASIC et sur FPGA a une surface 20 à 35 fois plus élevée sur FPGA, ainsi qu'une consommation supérieure d'au moins 14 fois et des temps de propagation trois à quatre fois plus longs.

Les circuits utilisés en grande quantité pour une application spécifique nécessitant de bonnes performances et une faible consommation, tels que ceux utilisés dans certains réseaux de capteurs, sont donc généralement des ASIC. Les FPGA sont utilisés dans le cas où les performances et la consommation importent moins, mais où il est nécessaire de pouvoir reprogrammer le circuit. Les FPGA permettent également de faire du prototypage avant une implémentation sur ASIC. Par la suite, le terme *circuit intégré* désigne des circuits spécifiques ASIC. Les circuits FPGA seront désignés explicitement en tant que tels.

Conception et fabrication des circuits intégrés

La conception d'un circuit intégré suit plusieurs étapes, appelées *flot de conception*. En premier lieu, les spécifications du circuit sont établies. Celles-ci comprennent le comportement attendu dans tous les cas possibles, les interfaces du circuit, et les attentes concernant les performances du circuit.

Une fois les spécifications établies, l'architecture du circuit est décrite au niveau algorithmique, ou RTL pour *Register Transfer Level*. Cette description, qui porte sur les différents blocs logiques, sur les signaux échangés entre ces blocs et sur les opérations effectuées sur ces signaux, est effectuée grâce à un langage de description matériel, appelé HDL pour *Hardware Description Language*. Le comportement de cette description du circuit en HDL est validé avec des simulations fonctionnelles, effectuées grâce à un outil spécialisé tel que Mentor Questa. Cet outil permet de visualiser les variations des signaux internes du circuit en réponse à divers signaux d'entrée, ou *vecteurs de test*, spécifiés dans un fichier (ou ensemble de fichiers) appelé *banc de test*.

Cette description au niveau RTL est ensuite transformée, lors de l'étape de synthèse logique, en une description sous forme d'un assemblage de cellules standard ; celles-ci peuvent être des portes logiques élémentaires qui effectuent des opérations booléennes, des éléments de logique séquentielle, ou des fonctions plus complexes comme des additionneurs ou des multiplexeurs. La liste des cellules standard et de leurs interconnexions est appelée *netlist*. Ces cellules standard sont issues d'une *bibliothèque de cellules*, c'est-à-dire d'une liste de cellules dans un nœud technologique donné fournie par un fabricant de circuits intégrés, également appelé *fonderie*. Des données relatives à ce nœud technologique, comme la consommation et le dimensionnement de ces portes logiques, sont disponibles dans cette bibliothèque de cellules. La synthèse optimise la description RTL d'un circuit selon plusieurs critères, afin de réduire la surface occupée par les portes logiques, la longueur des chemins de données, et la consommation du circuit. Dans nos travaux, la synthèse est effectuée avec l'outil Synopsys Design Compiler.

À l'issue de la synthèse, une représentation physique du circuit est élaborée lors de l'étape de placement et routage. Au cours de celle-ci, l'emplacement de toutes les cellules standard sur le circuit est déterminé, et celles-ci sont reliées par des fils sur différentes couches de métal superposées. Certains de ces fils sont utilisés pour la transmission de l'information entre les cellules et avec les interfaces vers l'extérieur du circuit intégré, et d'autres permettent d'alimenter ces cellules en courant. Afin d'optimiser divers paramètres tels que la surface, la consommation ou la fréquence maximale de fonctionnement d'un circuit intégré, l'outil de placement-routage peut remplacer certaines cellules standard de la netlist fournie par l'outil de synthèse par d'autres cellules standard, ayant la même fonctionnalité mais des caractéristiques physiques différentes. Par exemple, en fonction des dimensions des transistors dans une cellule standard, sa consommation et le temps de propagation d'un signal au sein de celle-ci peuvent varier. La description du circuit placé et routé est appelée *layout*. Dans notre flot de conception, cette étape est effectuée avec l'outil Cadence Innovus.

Diverses simulations peuvent être effectuées sur le circuit afin de valider que celui-ci correspond aux spécifications préalablement établies. Ces simulations effectuées sur le layout sont appelées simulations *post-layout*. Elles utilisent les données fournies par la fonderie sur la consommation et le temps de propagation de l'information dans les fils et les cellules standard, et peuvent permettre de valider fonctionnellement le circuit, ou d'évaluer sa consommation ou encore les temps de propagation au sein de celui-ci. Une fois le layout du circuit validé, celui-ci est envoyé

en fonderie afin d'être fabriqué. Plusieurs étapes sont nécessaires pour la fabrication d'un circuit, au cours desquelles différents *masques* sont utilisés pour graver le silicium et les différentes couches de métal superposées. Au cours de ces étapes, les dispersions du processus de fabrication entraînent des variations des paramètres physiques d'un circuit à l'autre, tels que les dimensions des pistes de métal ou l'épaisseur des différents matériaux. Ces dispersions, généralement très faibles, sont appelées par la suite *variations de processus*.

Nous avons décrit ici la conception d'un circuit intégré avec des cellules standard, issues d'une bibliothèque de cellules fournie par une fonderie. C'est ce qu'on appelle un flot de conception standard. Ce flot standard est utilisé dans la majorité des cas, mais il est parfois nécessaire d'utiliser des portes logiques différentes de celles fournies par les fabricants de circuits intégrés, comme nous le verrons dans la section 3.4.1. Dans ce cas, le flot de conception doit être modifié, et nous parlerons par la suite de flot *dédié*, ou *spécialisé*. Les outils de conception assistée par ordinateur, et notamment ceux nécessaires à la synthèse, et au placement et routage du circuit, sont adaptés pour un flot standard et effectuent de nombreuses opérations de manière automatique ; dans un flot spécialisé, plusieurs étapes doivent être effectuées manuellement, de la conception des portes logiques à leur intégration dans le circuit. Un flot spécialisé nécessite donc des efforts importants, ce qui peut avoir un impact non négligeable sur le coût de conception des circuits intégrés.

Évaluation de la surface d'un circuit intégré

Un circuit intégré occupe une certaine surface, également appelée *surface silicium* ou encore *coût silicium* par la suite. Il est généralement nécessaire de connaître cette surface lors de la conception d'un circuit intégré, car celle-ci conditionne une partie du coût de fabrication des circuits, et certains circuits sont limités en surface. Les choix effectués lors de la conception d'un circuit peuvent donc être modifiés après une première évaluation de la surface, afin de diminuer cette dernière.

Une première estimation de la surface du circuit intégré est donnée en sortie de synthèse. Celle-ci est la somme des surfaces des cellules standard qui constituent ce circuit et qui figurent dans la netlist. À l'issue de l'élaboration du layout du circuit, les outils de conception donnent une surface précise correspondant à la surface réelle occupée par le circuit après fabrication. Cette surface prend en compte de nombreux facteurs, tels que la densité de placement, le routage entre les cellules, et leur placement relatif les unes aux autres. Ces paramètres découlent eux-mêmes de l'optimisation à la fois en temps d'exécution, surface et consommation qui est réalisée par l'outil de placement-routage. Ce processus étant en grande partie automatisé et géré par l'outil, ces paramètres ne sont pas tous maîtrisés par l'utilisateur. Par exemple, la densité de placement des transistors, c'est-à-dire la quantité de transistors sur une surface donnée, peut varier entre deux circuits, voire pour un même circuit avec deux placements différents des cellules standard. La surface fournie par les outils après le placement et le routage du circuit n'est donc pas une métrique équitable afin de comparer la complexité de deux architectures de circuits intégrés. À l'inverse, la surface fournie par l'outil de synthèse, qui dépend uniquement des cellules standard et n'est pas liée au placement et au routage de ces cellules, ne donne pas avec précision la taille d'un circuit tel qu'il sera fabriqué, mais elle est plus adaptée pour la comparaison des tailles de plusieurs circuits.

Pour cette raison, nous utiliserons par la suite les surfaces fournies par l'outil de synthèse pour comparer les différents circuits conçus au cours de nos travaux. Ces surfaces sont exprimées en surface équivalente des portes logiques, notée GE pour *Gate Equivalent*. Tandis que la surface du circuit en μm^2 dépend du nœud technologique pour lequel le circuit a été synthétisé, la surface en GE est a priori indépendante de celui-ci, et elle permet donc la comparaison de la taille de plusieurs circuits synthétisés pour des nœuds technologiques différents. La surface d'un circuit en GE est obtenue en divisant sa surface, initialement exprimée en μm^2, par la surface en μm^2 d'une porte NON-ET (ou *NAND*) à deux entrées prise dans la même bibliothèque de cellules que celle ayant servi à créer la netlist du circuit intégré. Cette surface en GE est donc égale au nombre de portes NON-ET nécessaires pour occuper la même surface. Par la suite, nous utiliserons pour

le calcul de la surface en GE une porte NON-ET prise dans une bibliothèque de cellules standard pour le nœud technologique 28 nm FDSOI. Cette porte NON-ET a une surface de 0,4896 μm^2, ce qui signifie que les surfaces en μm^2 des différentes implémentations étudiées dans les chapitres 3 à 5 peuvent être obtenues en multipliant par 0,4896 les surfaces fournies en GE.

Consommation des circuits intégrés

La consommation de puissance dans un circuit intégré est la somme de deux composantes : une consommation *statique*, et une consommation *dynamique*. La consommation dynamique est liée à l'*activité* au sein de ce circuit, c'est-à-dire à la commutation des portes logiques entre "0" et "1". Plus l'activité est importante, et plus cette consommation est élevée. La consommation statique est due aux fuites de courants dans les transistors. Celle-ci est généralement négligeable par rapport à la consommation dynamique : par exemple, dans un circuit développé en 2018 au CEA LETI, cette consommation statique représente seulement 0,12 % de la consommation totale de puissance. La consommation d'énergie durant un temps donné est l'intégrale de la puissance dissipée sur ce temps. De manière générale, la consommation est plus élevée dans la logique séquentielle que dans la logique combinatoire ; par exemple, une bascule a une consommation de puissance 10,5 fois plus élevée qu'une porte ET et 6,7 plus élevée qu'une porte XOR dans le nœud technologique 0,18 µm, selon [8].

De même que pour la surface, il est généralement nécessaire d'avoir une estimation de la consommation d'un circuit au cours de sa conception. L'outil de placement-routage fournit une estimation de la consommation statique et de la consommation dynamique pour une configuration arbitraire du circuit. Afin d'avoir une évaluation de la consommation qui dépend réellement de l'activité d'un circuit dans un cas d'usage donné, cette consommation peut être évaluée suite à des simulations post-layout effectuées avec des vecteurs de test reflétant les conditions souhaitées. La simulation pour chaque circuit utilise alors la netlist post-layout de ce circuit, ainsi que le fichier SDF (*Standard Delay Format*) contenant toutes les informations de temps de propagation du circuit. Cela permet de générer un fichier d'activité, ou fichier VCD (*Value Change Dump*), qui contient le temps exact des commutations de toutes les cellules du circuit pour une simulation donnée. Dans notre cas, ce fichier VCD est ensuite traité avec le logiciel Synopsys PrimeTime. Cet outil combine les informations contenues dans les bibliothèques de cellules standard, telles que la consommation statique et dynamique de chaque cellule, à celles provenant du fichier VCD afin de calculer la consommation du circuit à chaque instant de la simulation. Une telle simulation des variations de la consommation peut nécessiter un certain temps, généralement compris entre plusieurs minutes et plusieurs heures, selon la taille du circuit simulé, la durée simulée (en cycles d'horloge), et la résolution temporelle souhaitée.

Diverses méthodes peuvent être mises en œuvre afin de réduire la puissance dissipée par un circuit lors de son fonctionnement. L'ajustement dynamique de la fréquence et de la tension d'alimentation, ou DVFS pour *Dynamic Voltage and Frequency Scaling*, consiste à moduler la fréquence et la tension d'un circuit en fonction des besoins en énergie et en performances. Une réduction de ces paramètres permet de réduire la consommation, au prix d'une diminution des performances de calcul, et dépend donc des opérations effectuées par le circuit. De plus, l'ensemble des blocs logiques d'un circuit ne sont pas constamment actifs, et une désactivation des blocs inactifs permet de réduire la consommation de puissance. Deux méthodes peuvent être mises en œuvre dans ce but. La désactivation de l'horloge, ou *clock gating*, peut être appliquée sélectivement à certains blocs de logique séquentielle afin de stopper l'activité de ceux-ci, et d'annuler ainsi leur consommation dynamique. La mise hors tension de certains blocs combinatoires ou séquentiels, appelée *power gating*, permet de totalement couper l'alimentation de ces blocs, ce qui permet d'importants gains en consommation. Le *power gating* est généralement plus complexe à mettre en œuvre que le *clock gating*, et nécessite la séparation du circuit en plusieurs domaines d'alimentation distincts, chacun pouvant être désactivé indépendamment des autres.

1.2.2 Contraintes sur les circuits intégrés pour l'IoT

Consommation d'énergie

Généralement, les objets connectés doivent avoir une durée de vie élevée, pouvant aller jusqu'à plusieurs années, voire plusieurs dizaines d'années, selon [3, 9]. Cela impacte fortement les choix effectués lors de leur conception. Ainsi, la batterie d'un objet connecté peut ne pas être rechargée ou remplacée durant plusieurs années. Dans ce cas, la consommation d'énergie sur l'ensemble de cette durée de vie est un paramètre majeur à prendre en compte lors de la conception de circuits intégrés pour l'IoT. Cette consommation d'énergie dépend non seulement de la consommation instantanée de puissance, mais également du temps d'exécution des différentes fonctionnalités d'un objet connecté. En particulier, il est intéressant du point de vue de l'énergie d'activer les fonctionnalités ayant une consommation instantanée élevée uniquement lorsque celles-ci sont nécessaires, et de les mettre en veille le reste du temps. On notera que la communication par radio consomme généralement plus que les calculs effectués sur un circuit intégré et doit donc être minimisée [4, 5], ce qui n'empêche pas de réduire également le coût énergétique de ces calculs.

Performances

Les objets connectés doivent disposer d'une puissance de calcul suffisante, adaptée aux applications visées. Ainsi, les tâches critiques ayant un impact sur la sécurité des biens et des personnes, par exemple dans le domaine des véhicules connectés ou de l'industrie 4.0, doivent généralement être effectuées quasiment en temps réel. Dans une moindre mesure, même les systèmes fortement contraints en énergie comme les réseaux de capteurs doivent effectuer efficacement certaines tâches, telles que la compression d'image ou l'agrégation de données.

Flexibilité et reconfiguration

Du fait de la durée de vie élevée d'un objet connecté, l'application pour laquelle il a été initialement conçu, mais également les besoins des utilisateurs, peuvent évoluer au cours du temps. Selon les auteurs de [9], il est impossible de prévoir toutes les utilisations possibles d'un objet connecté, pour l'ensemble de la durée de vie de cet objet. En effet, de nouveaux objets connectés sont constamment développés pour des usages très divers ; si ces objets communiquaient entre eux, de nouvelles applications pourraient voir le jour. Les auteurs donnent ainsi l'exemple d'un réfrigérateur connecté et d'une voiture autonome, dont nous ne pouvons pas à l'heure actuelle prévoir toutes les interactions possibles pour effectuer diverses actions. Les circuits intégrés à l'intérieur de ces objets connectés doivent donc pouvoir être reconfigurés après leur déploiement, afin de supporter de nouvelles applications et de nouveaux modes de fonctionnement.

Coût des circuits intégrés pour l'IoT

Selon [3], le coût des circuits intégrés pour l'IoT doit être faible, afin de permettre un déploiement à grande échelle de ceux-ci. Plusieurs facteurs sont à prendre en compte pour calculer le coût d'un circuit intégré. D'un côté, le coût d'un circuit intégré est lié à sa taille, c'est-à-dire à la surface occupée sur silicium. La surface de chaque circuit, mais également la mémoire disponible sur celui-ci, est donc limitée dans l'IoT. D'un autre côté, ce coût provient en grande partie du coût de conception, c'est-à-dire de l'effort effectué pour concevoir ce circuit, et du coup de fabrication, c'est-à-dire du prix des masques nécessaires à la production de ce circuit en fonderie. Afin de réduire le coût par circuit, il est donc nécessaire de réduire les coûts de conception et de fabrication, rapportés au nombre de circuits fabriqués. Pour cela, il faut produire le plus de circuits identiques possible.

Plutôt que de produire des circuits pour une seule application donnée, il peut donc être plus rentable de créer des circuits couvrant plusieurs applications. Chaque circuit est ensuite configuré

pour effectuer de manière optimale une application donnée. Cela rejoint les besoins en flexibilité décrits précédemment. Au final, les circuits intégrés pour l'IoT doivent donc non seulement être configurables au cours de leur durée de vie afin de couvrir les évolutions d'une application donnée, mais ils doivent également être configurables dès leur sortie d'usine afin d'être adaptés à diverses applications.

Circuits intégrés et électronique durable

Plusieurs projets ont vu le jour récemment afin de proposer une approche durable de la micro-électronique, regroupés en Europe dans le réseau ENCOS (*European Nanoelectronics COnsortium for Sustainability*) [10]. Outre l'utilisation de nouveaux matériaux ou la modification des procédés de fabrication, cette vision de la micro-électronique durable impose plusieurs contraintes sur les circuits intégrés eux-mêmes. Ceux-ci doivent avoir une faible consommation et une durée de vie élevée, et doivent pouvoir être ré-utilisés pour une autre application une fois leur utilisation terminée. Par exemple, lorsqu'un objet connecté devient obsolète ou qu'il cesse de fonctionner, il serait possible de récupérer le circuit intégré au sein de celui-ci, et de le reconfigurer pour une autre application. Le concept d'électronique durable est donc compatible avec les contraintes, énoncées précédemment, de faible consommation énergétique et de flexibilité des circuits intégrés.

1.2.3 Exemple de système sur puce pour l'IoT : la plateforme L-IoT

Un circuit intégré pour l'IoT est donc soumis à de nombreuses contraintes, et un compromis doit être trouvé entre celles-ci. En particulier, un tel circuit doit avoir une faible consommation énergétique, et des performances suffisantes pour une application donnée. De plus, les applications peuvent évoluer au cours de la vie d'un circuit intégré, et celui-ci doit donc pouvoir être mis à jour et s'adapter à diverses situations.

Une application peut être implémentée sur FPGA, ou sur ASIC. Les FPGA ont l'avantage d'être reconfigurables, mais les applications implémentées sur FPGA sont plus coûteuses que si elles étaient intégrées sur ASIC, à la fois en termes de consommation, de surface et de temps d'exécution. Dans nos travaux, nous étudions principalement les implémentations sur ASIC, plus légères et donc plus adaptées pour une utilisation dans l'IoT.

Enfin, les applications peuvent également être développées en logiciel, et implémentées sur un processeur lui-même intégré sur silicium. Ces implémentations logicielles offrent une grande flexibilité et permettent d'effectuer des applications très variées. Cependant, pour une application donnée, une implémentation matérielle sera plus rapide et moins consommatrice en énergie que l'implémentation logicielle correspondante. Une application critique du point de vue des performances ou effectuée régulièrement gagnera donc à être implémentée directement sur silicium. Un système sur puce comprend généralement un processeur communiquant avec plusieurs *accélérateurs matériels*. Ces accélérateurs sont des implémentations matérielles de certaines fonctionnalités, comme la compression d'image ou le chiffrement. Cette architecture permet donc d'effectuer les fonctions critiques rapidement et avec une faible consommation, tandis que le processeur exécute toutes les autres applications et assure la flexibilité du système. Divers types de capteurs ou d'interfaces sont également intégrés à ce système sur puce en fonction des applications.

Un exemple d'un tel système sur puce pour l'IoT est la plateforme L-IoT (*Low-power IoT*) [11], développée au sein du CEA LETI dans le département DACLE (Département Architecture, Conception, et Logiciel Embarqué). Elle regroupe de nombreuses contributions originales répondant à plusieurs scénarios applicatifs dans l'IoT ; par exemple, elle comprend divers capteurs, des moyens de calculs légers en énergie, et plusieurs mécanismes de gestion de la consommation. Ce système sur puce est prévu pour une utilisation sur batterie, avec un budget énergétique limité et une durée de vie potentiellement importante. L'architecture de L-IoT est donnée sur la figure 1.4. On notera que ce circuit est partitionné en deux sous-systèmes, notés respectivement *On-Demand* et *Always-Responsive*.

Le sous-système *On-Demand* comprend un coeur RISC-V [12], qui communique avec des blocs mémoires, des capteurs, une radio, et des accélérateurs matériels pour le traitement d'images, la fusion de données, ou le chiffrement des données. Ce sous-système *On-Demand* a donc une consommation instantanée élevée. Il est mis en veille lorsqu'il n'est pas utilisé, puis réveillé à nouveau en cas de besoin ; lorsqu'il est en veille, son alimentation est coupée avec du *power gating*, ce qui permet des économies d'énergie importantes.

La gestion de la veille et du réveil du sous-système *On-Demand* est gérée par le sous-système *Always-Responsive*. Ce dernier a une consommation très faible, et peut fonctionner en permanence sans impacter la consommation globale. Il comprend en particulier une *radio de réveil* [13], qui est une radio fonctionnant seulement en réception, avec un très faible débit, mais qui a l'avantage de consommer très peu d'énergie. Cette radio peut recevoir des *codes de réveil* envoyés par d'autres nœuds d'un même réseau, dont le but est de réveiller la partie *On-Demand* lorsque cela est nécessaire, par exemple lorsqu'un autre nœud souhaite communiquer avec celui-ci. Pour cela, chaque code de réveil reçu est comparé à un code de référence ; si ces deux codes sont identiques, la partie *On-Demand* sort de veille. Cette comparaison est effectuée dans la partie *Always-Responsive*, et son coût en énergie est négligeable. Du fait du faible débit de la radio de réveil, ces codes de réveil sont généralement courts, avec une taille comprise entre 8 et 64 bits selon les cas. De plus, ils sont généralement pré-définis et constants, c'est-à-dire identiques pour tous les réveils, et peuvent par exemple correspondre à l'adresse d'un nœud dans un réseau [14, 15, 16].

Par la suite, nous appellerons *radio principale* la radio contenue dans la partie *On-Demand*, afin de distinguer celle-ci de la radio de réveil. La communication sur la radio principale est effectuée avec un protocole de communication standard pour l'IoT comme le Wi-Fi ou sa version faible consommation Wi-Fi HaLow [17], le Bluetooth Low Energy [18], ou encore les protocoles issus de la famille de réseaux LRWPAN (*Low Rate Wireless Personal Area Network*) définis par la norme IEEE 802.15.4 [3, 5] tels que ZigBee ou WirelessHART. Ces protocoles transmettent des paquets de données qui contiennent, outre les messages échangés entre les nœuds, de nombreuses informations telles que les adresses des nœuds émetteur et récepteur et le numéro de séquence du paquet. Ces protocoles nécessitent l'envoi d'un nombre élevé de bits par paquet, et parfois de plusieurs paquets par session de communication ; ils ne sont donc pas adaptés pour une utilisation sur la radio de réveil, qui a un débit de données très faible. Il n'existe pas à ce jour de protocole de communication standardisé et largement répandu pour la radio de réveil, et les codes de réveil sont généralement envoyés de manière brute, sans traitement ni rajout de diverses informations qui augmenteraient la taille d'une transmission vers cette radio et donc la durée de cette transmission.

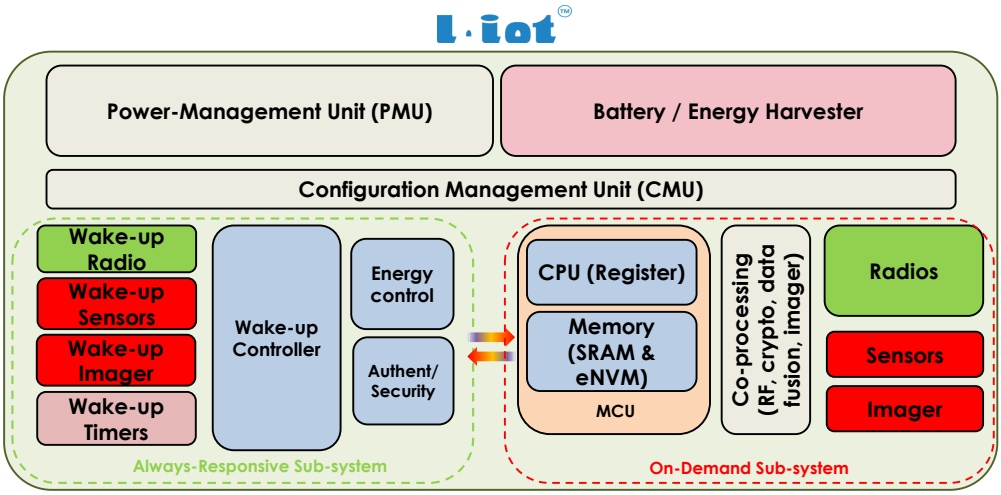


FIGURE 1.4 – Architecture de la plateforme L-IoT.

1.3 Enjeux de sécurité dans l'Internet des Objets

Les objets connectés sont de plus en plus présents dans nos vies, et traitent une quantité croissante de données devant être sécurisées. Ceux-ci collectent, stockent, et envoient de nombreuses données personnelles. Or, selon une étude menée par HP en 2014 sur un échantillon d'objets connectés [19], bien que 90% des objets analysés collectent des données personnelles, 70% de ceux-ci ne chiffrent pas ces données avant de les transmettre. Ces données privées peuvent alors être consultées par une personne malintentionnée.

De plus, l'IoT couvre de nombreux domaines critiques, tels que la santé connectée, les véhicules connectés, ou encore l'industrie 4.0. Pour ces applications, la sécurité physique des biens et des personnes est directement liée à la sécurité des objets et des réseaux dans l'IoT. Par exemple, en 2015, des chercheurs ont montré la possibilité de prendre le contrôle à distance d'une Jeep Cherokee [20], afin de mettre ses freins ou son accélérateur hors service, ou encore de suivre celle-ci à distance en récupérant ses coordonnées GPS en temps réel. Les auteurs de [21] et [22], quant à eux, ont montré la possibilité d'extraire des informations et de prendre le contrôle de, respectivement, une pompe à insuline connectée et un pacemaker, à condition d'être situé à proximité de ceux-ci (quelques dizaines de mètres pour la pompe à insuline, et quelques centimètres pour le pacemaker).

Ces quelques exemples montrent que diverses attaques sont possibles contre l'IoT, dont les conséquences peuvent varier de la diffusion d'informations personnelles à la perte de nombreuses vies humaines. Dans cette section, nous définissons tout d'abord les principes de base de la sécurité dans l'IoT, et les moyens pouvant être mis en œuvre afin d'assurer cette sécurité. Le chiffrement est un de ces moyens, et nous décrivons son implémentation efficace sur un circuit intégré contraint en ressources. Nous détaillons ensuite les attaques pouvant être menées à différents niveaux contre l'IoT, puis nous expliquons pourquoi la sécurité des circuits intégrés est particulièrement importante, et en quoi celle-ci s'ajoute aux différentes contraintes pour les circuits intégrés pour l'IoT listées dans la section 1.2.2.

1.3.1 Principes de la sécurité

La sécurité dans l'IoT repose sur plusieurs principes de base, que sont la confidentialité, l'intégrité, l'authentification, et la disponibilité des données [23]. Ceux-ci sont décrits dans cette section, ainsi que les moyens permettant de mettre ces principes en application.

Confidentialité

La confidentialité des données consiste à garantir que seuls les utilisateurs autorisés ont accès à celles-ci. Elle est assurée par le chiffrement des données, et par une politique de sécurité garantissant des droits d'accès, et donc des clés de chiffrement, aux utilisateurs légitimes. Ces clés doivent elles-mêmes être distribuées, stockées et mises à jour de manière sécurisée, c'est-à-dire sans possibilité pour un attaquant de récupérer ou d'altérer celles-ci. Le principe du chiffrement est que l'algorithme de chiffrement utilisé est connu de tous, mais que seuls les utilisateurs disposant des clés de chiffrement peuvent chiffrer et déchiffrer des données.

Il existe deux types d'algorithmes de chiffrement : le chiffrement asymétrique, et le chiffrement symétrique. Pour le chiffrement symétrique, tous les interlocuteurs utilisent la même clé secrète à la fois pour le chiffrement et le déchiffrement d'un message. Cela nécessite donc que cette clé secrète ait été distribuée préalablement à chacun de manière sécurisée. Avec le chiffrement asymétrique, chaque interlocuteur dispose d'une paire de clés : une clé privée et une clé publique. La clé publique est transmise à tous, tandis que la clé privée est propre à chaque interlocuteur, et connue de lui seul. Le propriétaire d'une clé privée est le seul à pouvoir déchiffrer les messages ayant été chiffrés avec la clé publique correspondante. Le chiffrement asymétrique ne requiert donc pas la connaissance d'un secret commun à tous les interlocuteurs.

La stéganographie est une autre méthode visant à assurer la confidentialité d'une donnée. Son principe est de dissimuler une donnée dans une autre donnée ; par exemple, un message peut être dissimulé dans une image. Cependant, cette méthode repose sur le fait qu'un attaquant ne connaît pas la méthode utilisée pour cacher la donnée, voire ne sait pas qu'une donnée a été cachée ; on appelle cela de la sécurité par obscurité. Au contraire du chiffrement, la sécurité de la stéganographie ne peut pas être prouvée de manière formelle.

Intégrité

L'intégrité des données consiste à garantir que celles-ci ne peuvent être altérées par une personne non autorisée. L'intégrité est assurée en calculant une *empreinte* d'une donnée lors de la création de cette donnée. Une nouvelle empreinte peut être calculée à tout moment à partir de cette donnée, et comparée à l'empreinte de référence, afin de s'assurer que la donnée n'a pas été modifiée : si les deux empreintes sont identiques, cela signifie que l'intégrité de la donnée a été préservée. Par exemple, lors de la transmission d'un message, l'empreinte de ce message est transmise avec celui-ci ; le récepteur calcule l'empreinte du message reçu, et la compare à l'empreinte reçue.

Cette empreinte est généralement calculée avec une fonction de hachage cryptographique, et sera également appelée *valeur de hachage* dans nos travaux. Une fonction de hachage cryptographique donnée calcule une empreinte de taille fixe, quelle que soit la longueur du message en entrée de cette fonction. C'est une fonction à sens unique : le calcul de la valeur de hachage s'effectue facilement, mais il est très difficile, voire impossible avec les moyens de calcul actuels, de retrouver la valeur d'entrée de la fonction en connaissant la valeur de hachage. Une modification quelconque du message en entrée de cette fonction, que cette modification porte sur seulement un bit ou sur une grande partie du message, résulte généralement en une modification importante de la valeur de hachage. Il n'est donc pas possible, en connaissant la valeur de hachage, d'obtenir des informations exploitables sur le message original. Enfin, une telle fonction doit garantir que deux messages différents ont quasi-systématiquement des empreintes différentes.

Plusieurs attaques sont possibles contre une fonction de hachage cryptographique. Une attaque par pré-image consiste à retrouver l'entrée x d'une fonction de hachage en connaissant sa sortie $h(x)$. Une attaque par seconde pré-image consiste à trouver, pour des valeurs x et $h(x)$ données, une seconde entrée y qui permet d'obtenir la même valeur de sortie $h(y) = h(x)$. Enfin, une attaque par collision consiste à trouver deux valeurs d'entrées x et y différentes telles que $h(x) = h(y)$. La résistance à ces trois attaques permet de quantifier la sécurité d'une fonction de hachage cryptographique.

Authentification

L'authentification consiste à vérifier l'identité d'un interlocuteur. Ce dernier peut par exemple être une personne réelle, ou un objet connecté. L'authentification est nécessaire pour des applications variées, comme le contrôle d'accès à un service, ou la vérification de l'identité de l'expéditeur d'un message. Différents facteurs peuvent permettre l'authentification d'un interlocuteur : celui-ci peut être identifié par *ce qu'il sait* (par exemple, un mot de passe), *ce qu'il a* (par exemple, une carte à puce), ou *ce qu'il est* (cela inclut par exemple les données biométriques).

En particulier, l'authenticité et l'intégrité d'un message reçu peuvent toutes deux être garanties en utilisant une clé secrète lors du calcul de l'empreinte de ce message. Cette clé secrète permet d'attester de l'identité de l'émetteur. Plusieurs méthodes peuvent être employées : la fonction de hachage cryptographique utilisée pour générer l'empreinte peut être utilisée avec une clé, combinée à un algorithme de chiffrement, voire totalement remplacée par un algorithme de chiffrement. On parle de code MAC, pour *Message Authentication Code*, lorsque l'empreinte authentifiée est calculée avec une clé secrète partagée par différents interlocuteurs. Lorsque l'empreinte est calculée avec une clé privée, donc avec du chiffrement asymétrique, on parle plutôt de *signature* d'un message.

Disponibilité

La disponibilité d'une donnée ou d'un service, c'est-à-dire la possibilité d'accéder à ceux-ci lorsque c'est nécessaire, garantit le fonctionnement ininterrompu d'un objet ou d'un réseau entier. Les attaques visant à rendre indisponibles cette donnée ou ce service sont généralement appelées attaques par déni de service. Il existe plusieurs manières d'assurer la disponibilité, qui dépendent directement des modèles d'attaques et des moyens physiques mis en œuvre afin d'effectuer ces attaques. Un exemple d'attaque par déni de service et d'une contre-mesure protégeant un système contre cette attaque est présenté dans le chapitre 2.

1.3.2 Chiffrement léger pour l'IoT

Afin d'assurer la confidentialité des communications dans l'IoT, tous les messages échangés par radio et sur Internet doivent être chiffrés. De plus, certaines données sensibles stockées sur un circuit intégré doivent également être chiffrées, afin d'empêcher un attaquant de récupérer ces données en lisant sa mémoire. Enfin, le chiffrement, lorsqu'il est combiné à une fonction de hachage cryptographique, permet de générer des codes MAC qui sont utilisés pour démontrer l'authenticité et l'intégrité des messages échangés. Le chiffrement est donc une opération critique, effectuée très fréquemment lors du fonctionnement d'un circuit intégré. Sur un système sur puce tel que L-IoT, le chiffrement est effectué avec un accélérateur matériel afin d'optimiser les performances et la consommation de cette opération. Dans cette section, nous abordons l'implémentation des différents types d'algorithmes de chiffrement, et nous décrivons un algorithme de chiffrement léger particulièrement adapté pour une intégration sur un circuit pour l'IoT contraint en surface et/ou en énergie.

Le chiffrement peut être symétrique, lorsque tous les interlocuteurs partagent la même clé secrète, ou asymétrique lorsque chaque interlocuteur dispose d'une clé publique et d'une clé privée. Les algorithmes de chiffrement asymétrique requièrent actuellement des clés bien plus grosses que les algorithmes de chiffrement symétrique, pour un niveau de sécurité équivalent. Ainsi, l'Agence Nationale de la Sécurité des Systèmes d'Informations (ANSSI) recommandait en 2014 [24] des clés secrètes de 128 bits pour le chiffrement symétrique, et d'au moins 2048 bits pour les algorithmes de chiffrement asymétrique à base de factorisation de nombre premiers tels que le RSA [25]. Du fait de la complexité des opérations effectuées pour le chiffrement asymétrique et de la taille des nombres traités, ce chiffrement s'effectue généralement dans un temps bien plus long qu'un chiffrement symétrique, et avec une consommation supérieure. Le chiffrement symétrique est donc généralement préféré pour chiffrer les messages dans l'IoT, le chiffrement asymétrique étant quant à lui utilisé pour établir de manière sécurisée des clés symétriques partagées.

Il existe de nombreux algorithmes de chiffrement symétrique légers, c'est-à-dire ayant une faible surface et/ou une faible consommation. Une liste représentative de ceux-ci, établie en 2015, est dressée dans [26]. On distingue à nouveau deux types d'algorithmes : les algorithmes de chiffrement par bloc, et les algorithmes de chiffrement par flot. Le chiffrement par bloc consiste à découper les données à chiffrer en blocs de taille fixe, par exemple 128 bits pour l'*Advanced Encryption Standard* (AES) [27]. Si la taille des données n'est pas multiple de celle des blocs, le dernier bloc est complété en concaténant les données à chiffrer avec une constante, souvent égale à 0 ; cette opération est appelée *padding*. Ces blocs sont chiffrés les uns à la suite des autres. Le chiffrement et le déchiffrement avec un tel algorithme constituent souvent deux opérations distinctes, qui doivent alors être implémentées séparément. Le chiffrement par flot consiste à générer un flot pseudo-aléatoire de données, appelé flot de clé, celui-ci étant ensuite additionné bit par bit, par exemple avec une porte logique XOR, aux données à chiffrer. Cette addition étant inversible, le déchiffrement s'effectue de la même manière et ne nécessite pas de mécanisme supplémentaire, ce qui est un avantage par rapport à la majorité des algorithmes de chiffrement par bloc. De plus, lors du chiffrement par flot, il n'est pas nécessaire d'effectuer un *padding* sur les données. Ces deux dernières propriétés rendent le chiffrement par flot intéressant pour une application contrainte en ressources dans l'IoT.

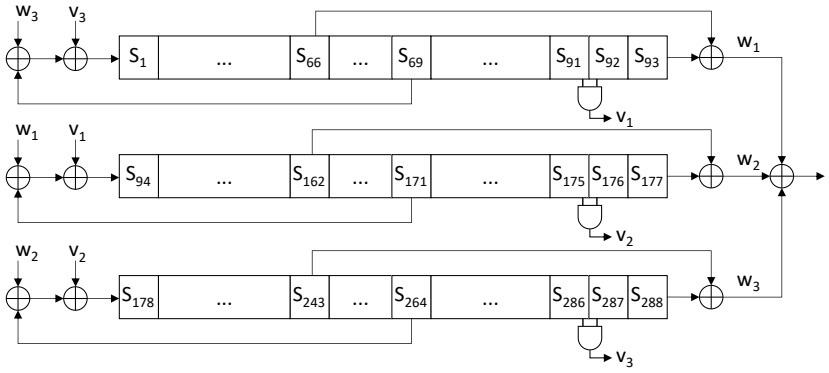


FIGURE 1.5 – L'algorithme de chiffrement par flot Trivium.

En 2008, la compétition *eStream*, ou *ECRYPT Stream Cipher Project* [28], a désigné des algorithmes de chiffrement par flot efficaces et compacts, adaptés pour une implémentation sur des circuits contraints en surface et en consommation. Parmi ceux-ci, Trivium [29], standardisé dans la norme ISO/IEC 29192-3 :2012 qui porte sur la cryptographie légère [30], présente de nombreux avantages. Trivium n'a pour l'instant fait l'objet d'aucune cryptanalyse, contrairement à la première version de Grain [31], l'un des autres algorithmes sélectionnés pour la compétition eStream. De plus, Trivium offre beaucoup de possibilités d'implémentations différentes afin de privilégier, au choix, plutôt la surface, les performances ou la consommation, ce qui permet de couvrir une large gamme d'applications potentielles avec un seul algorithme. Dans [29], les auteurs proposent ainsi une implémentation en série générant un seul bit de clé par cycle d'horloge, ou plusieurs implémentations parallèles avec 2 à 64 bits de parallélisme. Les auteurs de [32] ont analysé en 2008 les implémentations sur silicium de huit familles d'algorithmes de chiffrement par flot. Ils concluent alors que Trivium est non seulement l'algorithme offrant l'espace de conception le plus étendu, c'est également celui qui présente la plus faible consommation, en énergie par bits, pour une implémentation en parallèle. Enfin, depuis la compétition eStream, deux variantes de Trivium utilisant des clés de 128 bits ont été proposées [33, 34], ainsi qu'une version ayant une consommation plus faible [35], qui viennent donc compléter les possibilités offertes par Trivium.

La conception de Trivium a été réalisée, selon ses auteurs [29], de façon aussi simple que possible sans sacrifier la sécurité. La version en série de Trivium est décrite ici, le passage aux versions parallèles se faisant aisément tel que décrit dans [29]. Cette version est représentée sur la figure 1.5. Par la suite, nous nous plaçons dans le corps de Galois GF(2), dans lequel sont effectuées les opérations booléennes bit à bit ; dans ce corps, les opérations XOR et ET sont équivalentes à l'addition et à la multiplication, et nous adoptons donc ces notations. Trivium est constitué de trois registres à décalage à rétroaction non-linéaire (notés NLFSR, pour *Non-Linear Feedback Shift Registers*). Son état interne contient 288 bits, notés $S_1, ..., S_{288}$ et répartis sur trois registres à décalage de longueurs respectives de 93, 84 et 111 bits. À chaque cycle d'horloge, l'état interne est décalé d'un bit dans chaque registre à décalage, et mis à jour en entrée de ces registres grâce à une fonction combinatoire non-linéaire utilisant 3 portes XOR et une porte ET par registre. Cette fonction combinatoire, notée par la suite *fonction de rétroaction*, est décrite par l'algorithme 1.

Préalablement à l'exécution de Trivium, une clé et un vecteur d'initialisation de 80 bits chacun, ainsi qu'une valeur constante de 128 bits, sont chargés dans l'état interne de l'algorithme, ce qui est également décrit dans l'algorithme 1. Le vecteur d'initialisation, noté IV par la suite pour *Initialization Vector*, change à chaque réinitialisation de l'algorithme, tandis que la clé peut être utilisée plusieurs fois de suite.

Trivium fonctionne en deux phases : une phase d'initialisation au cours de laquelle le flot de clé n'est pas généré, suivie par une phase de chiffrement. L'objectif de la phase d'initialisation est de mélanger l'état interne afin de créer un état pseudo-aléatoire, à partir duquel un attaquant ne pourrait pas retrouver la clé de chiffrement ou l'IV. Elle dure 4×288 cycles d'horloge, soit 1152 cycles. Une fois cette phase terminée, le flot de clé est généré à raison d'un bit par cycle d'horloge.

Algorithme 1: Chargement des valeurs initiales de Trivium, et mise à jour de son état interne avec la *fonction de rétroaction*

Entrée : K, IV (clé et vecteur d'initialisation)
Sortie : S (état interne)
Paramètre : N (nombre de cycles)

$S = (0, \ldots, 0)$;
$(S_1, \ldots, S_{80}) = K$;
$(S_{94}, \ldots, S_{173}) = IV$;
$(S_{286}, \ldots, S_{288}) = (1, 1, 1)$;
for $i = 1$ **to** N **do**
  $(S_2, \ldots, S_{93}) = (S_1, \ldots, S_{92})$;
  $(S_{95}, \ldots, S_{177}) = (S_{94}, \ldots, S_{176})$;
  $(S_{179}, \ldots, S_{288}) = (S_{178}, \ldots, S_{287})$;
  $S_1 = S_{243} + S_{288} + S_{69} + S_{286}S_{287}$;
  $S_{94} = S_{66} + S_{93} + S_{171} + S_{91}S_{92}$;
  $S_{178} = S_{162} + S_{177} + S_{264} + S_{175}S_{176}$;
end

1.3.3 Des attaques à différents niveaux

La connexion à Internet, la communication par radio, et l'accessibilité physique des objets connectés sont autant de chemins d'attaques potentiels. Ainsi, diverses attaques peuvent être effectuées à différents niveaux dans l'IoT, allant du circuit intégré à l'application, en passant par la communication avec la radio. Nous distinguons ici les attaques effectuées sur 5 niveaux différents, inspirés du modèle OSI (*Open Systems Interconnection*) et également appelés couches. Ces couches sont représentées sur la figure 1.6. La couche *physique* est en charge de la transmission de signaux radio entre deux nœuds et concerne les moyens physiques mis en œuvre dans ce but. La couche *liaison* gère la transmission de données entre deux nœuds disposant d'une liaison directe entre eux. La couche *réseau* est en charge de l'établissement de la connexion entre deux nœuds distants, et gère en particulier le routage et l'adressage dans le réseau. La couche *transport* gère la bonne transmission des messages eux-mêmes entre ces deux nœuds. Enfin, nous considérons ici que les autres tâches, liées à l'application elle-même, font partie de la couche *application*; cette simplification par rapport au modèle OSI, également faite par les auteurs de [36], revient à regrouper les couches *session*, *présentation* et *application* de ce modèle.

Il existe autant d'attaques sur la couche *application* que d'applications elles-mêmes. Ces attaques peuvent viser les serveurs, les applications elles-mêmes, ou encore la gestion des mises à jour. En particulier, de nombreux objets disposent d'une adresse IP et sont connectés à Internet en permanence; certains outils, comme le moteur de recherche Shodan, permettent de découvrir tous ces objets, et en particulier ceux présentant des vulnérabilités [37]. Or, de nombreux objets disposent de mots de passe par défaut, ce qui a été exploité par le logiciel Mirai en 2012 [38] afin de créer des réseaux d'objets connectés zombies, qui ont ensuite été utilisés pour effectuer plusieurs attaques par déni de service distribuées.

Les attaques sur les couches *réseau* et *transport* sont également très variées, et un aperçu de celles-ci est donné dans [39, 36]. Elles sont particulièrement efficaces contre les réseaux comprenant un certain nombre d'objets, avec une topologie en arbre ou en réseau maillé. Une partie d'entre elles nécessite l'insertion d'un nœud malveillant dans un réseau, ou la corruption d'un nœud existant dans un réseau. Certaines attaques visent les données, par exemple en ne transmettant pas les données transitant par un nœud corrompu, ou en transmettant seulement une partie de celles-ci. D'autres attaques visent à perturber le routage et l'intégrité du réseau, en mettant hors service un nœud routeur, en transmettant les données aux mauvais nœuds, en changeant l'identifiant d'un nœud corrompu, ou encore en envoyant des requêtes erronées telles que

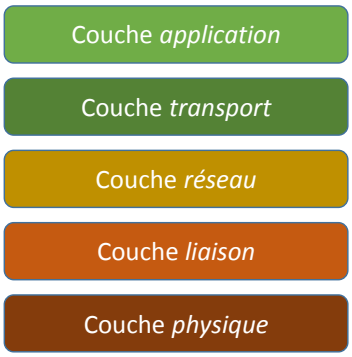


FIGURE 1.6 – Différentes *couches*, sur lesquelles des attaques peuvent être effectuées.

des messages d'erreurs ou des demandes de mise à jour du réseau.

Une liste des principales attaques sur la couche *liaison* peut être trouvée dans [2, 39, 36]. Un attaquant disposant des clés de chiffrement utilisées pour sécuriser la communication peut espionner les messages échangés, voire altérer ceux-ci ou forger et insérer de nouveaux messages. Un attaquant ne disposant pas de ces clés peut par exemple rejouer des messages échangés précédemment, c'est-à-dire ré-utiliser des messages authentiques échangés entre deux nœuds, afin de perturber le réseau. L'efficacité d'une telle attaque dépend fortement du protocole de communication utilisé. Diverses manières d'exécuter des dénis de service sont également possibles, en attaquant soit un nœud en particulier, soit la communication entre deux nœuds. Par exemple, l'envoi de nombreuses requêtes à un nœud peut aboutir à une décharge de sa batterie, tandis que la dégradation des signaux radio peut empêcher la communication entre deux nœuds. Enfin, de nombreuses attaques ont été publiées contre des protocoles de communication particuliers, que nous ne détaillerons pas ici.

Les attaques sur la couche *physique* visent soit la radio d'un objet, par exemple en brouillant les signaux émis et reçus, soit l'objet connecté lui-même [2]. Certaines attaques sur l'objet connecté ont pour objectif de récupérer son logiciel embarqué ou de le modifier afin de prendre le contrôle de l'objet. De telles attaques peuvent exploiter par exemple les interfaces de test, ou des vulnérabilités dans le logiciel embarqué et dans son intégration. D'autres attaques, appelées par la suite *attaques matérielles*, sont réalisées contre le circuit intégré contenu dans l'objet connecté. Elles peuvent avoir plusieurs objectifs, comme la récupération de données secrètes contenues dans le circuit intégré, la perturbation du fonctionnement de ce circuit, ou encore la rétro-ingénierie de celui-ci afin d'obtenir des informations sur sa conception.

Il existe plusieurs types d'attaques matérielles : celles-ci peuvent être destructives et irréversibles, ou non-destructives. Les attaques destructives consistent à décapsuler et découper le circuit intégré, grâce à divers moyens chimiques ou physiques. Elles sont généralement effectuées en laboratoire avec un matériel coûteux, et requièrent une certaine expertise et des efforts importants. Les attaques non-destructives sont séparées en deux catégories : les attaques par injection de fautes [6, 40, 41, 42], et les attaques par analyse des canaux auxiliaires [43, 44]. Les attaques par injection de fautes reposent sur divers mécanismes permettant d'introduire des fautes dans un circuit intégré et d'exploiter ensuite le résultat obtenu. Les attaques par analyse des canaux auxiliaires consistent à mesurer certains effets physiques corrélés aux opérations effectuées et aux données manipulées. Ces effets physiques peuvent être par exemple les variations de la consommation ou des émissions électromagnétiques d'un circuit. Les attaques matérielles non-destructives peuvent être menées sur place, c'est-à-dire sans déplacer les circuits intégrés, avec un matériel peu coûteux, et elles requièrent peu d'expertise. Elles peuvent donc être particulièrement efficaces pour attaquer certains objets connectés qui opèrent de manière non supervisée et qui sont physiquement accessibles.

Les vulnérabilités sur ces différentes couches peuvent être combinées afin de mettre en place des attaques plus complexes. Par exemple, dans [45], les auteurs démontrent la possibilité d'extraire les clés de chiffrement d'une ampoule connectée grâce à une attaque par canaux auxiliaires,

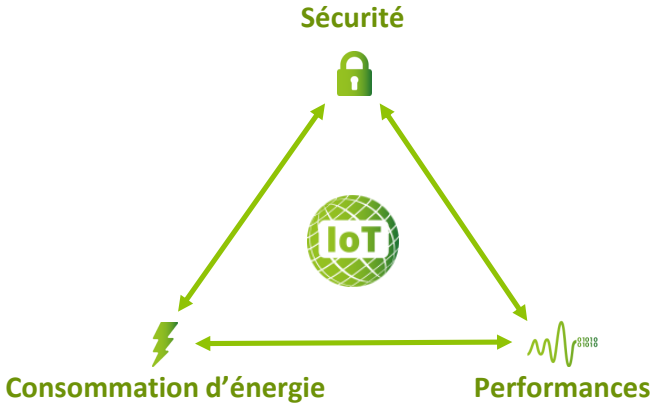


FIGURE 1.7 – Nécessité d'un compromis entre consommation d'énergie, sécurité et performances dans l'IoT.

afin de modifier le logiciel embarqué de l'ampoule et de prendre le contrôle de celle-ci. Grâce à une faille dans le protocole de communication ZigBee [5], ce logiciel embarqué malveillant peut alors être transmis par radio à toutes les ampoules environnantes, créant une réaction en chaîne permettant à un attaquant de prendre le contrôle de l'ensemble des ampoules d'une ville. Cet attaquant pourrait alors non seulement éteindre toutes les lampes ou les faire clignoter de manière à créer des crises d'épilepsie, mais également brouiller les fréquences radio utilisées par certains protocoles de communication comme le Wi-Fi. On constate avec cet exemple que la corruption d'un seul nœud grâce à une attaque matérielle peut compromettre l'ensemble d'un réseau, voire l'ensemble des nœuds à portée de signal radio même si ces nœuds appartiennent à des réseaux différents.

De manière générale, de nombreuses attaques sur la couche *liaison* et sur les couches supérieures reposent sur l'introduction de nœuds malveillants dans un réseau, ou sur la corruption de nœuds légitimes faisant déjà partie d'un réseau. Pour cela, un attaquant doit être en possession des clés secrètes utilisées afin de sécuriser les communications et d'authentifier les nœuds dans un réseau. La récupération de clés secrètes avec des attaques matérielles permet donc d'effectuer ensuite divers types d'attaques sur ces couches supérieures.

1.3.4 La sécurité, une contrainte supplémentaire pour les circuits intégrés

Compromis entre sécurité, consommation d'énergie, et performances

Un objet connecté doit être sécurisé contre les attaques sur les différentes couches mentionnées précédemment. Or, la sécurité n'est pas en soi une fonctionnalité d'un objet connecté, dans le sens où elle ne consiste pas directement à remplir les tâches pour lesquelles cet objet a été conçu. C'est une sur-couche qui est rajoutée afin de protéger cet objet et les applications pour lesquelles il a été conçu contre diverses attaques. En particulier, afin d'assurer la sécurité des échanges radio, le circuit intégré contenu dans cet objet doit contenir plusieurs primitives cryptographiques, c'est-à-dire des fonctions de base telles que des algorithmes de chiffrement et de hachage cryptographique. L'implémentation et l'exécution de ces primitives cryptographiques sur un circuit intégré a pour conséquence une augmentation du nombre d'opérations à effectuer par ce circuit, et donc une augmentation de la consommation instantanée de celui-ci. De plus, les circuits intégrés doivent être protégés contre les attaques matérielles ; la mise en place de protections adéquates contre ces attaques résulte également en une augmentation de la consommation instantanée et/ou du temps d'exécution des opérations protégées.

Or, nous avons expliqué dans la section 1.2.2 qu'un circuit intégré pour l'IoT doit avoir des performances de calcul suffisantes, et une faible consommation d'énergie. Un compromis doit donc être trouvé entre la sécurité, la consommation d'énergie et les performances d'un objet connecté, illustré sur la figure 1.7. À ce jour, peu de recherches ont été menées sur l'optimisation conjointe

de l'efficacité énergétique et de la sécurité des circuits intégrés. Ainsi, le développement de protections ayant une faible consommation énergétique, ou au contraire la sécurisation de mécanismes de gestion de l'énergie, n'ont été que très peu étudiés jusqu'à présent. Ce problème a notamment été pointé par les auteurs de [9] en 2016.

Sécurité et flexibilité : nécessité d'une sécurité adaptative

Dans la section 1.2.2, nous avons expliqué que les applications et les attentes vis-à-vis d'un objet connecté peuvent évoluer sur le long terme, au cours de la durée de vie de cet objet. L'évolution de ces applications entraîne une évolution des besoins en sécurité pour les circuits intégrés : certaines applications peuvent requérir une augmentation du niveau de sécurité, tandis que d'autres peuvent au contraire nécessiter une sécurité plus faible. De plus, les contraintes en sécurité peuvent évoluer lors de la découverte de nouvelles vulnérabilités et menaces au cours de la vie d'un objet connecté. Afin de répondre à d'éventuels futurs besoins en sécurité, cette sécurité doit être soit surdimensionnée lors de la conception des circuits intégrés, soit reconfigurable en fonction des besoins. La seconde solution devrait apporter d'importants gains en énergie sur l'ensemble de la durée de vie des circuits.

Les besoins en sécurité peuvent également varier à court terme, et adapter le niveau de sécurité au cas par cas pour chaque circuit, durant le fonctionnement de celui-ci, permettrait alors d'optimiser la consommation énergétique. Ainsi, les données sensibles, telles que les mots de passe ou les mises à jour, doivent être protégées efficacement, tandis qu'une faible consommation peut être privilégiée lors du traitement d'autres données telles que des mesures de capteurs. Un autre exemple est l'usage de plusieurs types de clés de chiffrement dans certains réseaux de communication : des *clés maîtresses* ayant une durée de vie élevée permettent de générer de nombreuses *clés de sessions* éphémères, elles-mêmes utilisées pour le chiffrement d'un ou de plusieurs messages au cours d'une session de communication. Dans ce cas, un effort plus élevé doit être fourni pour la protection des *clés maîtresses*, tandis que la sécurité mise en place pour la protection des *clés de sessions* peut être plus faible.

Enfin, le fonctionnement en autonomie de ces objets peut parfois empêcher la détection et la réaction à diverses attaques par des opérateurs humains, ces tâches de détection et de réaction étant alors laissées à la charge des objets eux-mêmes. Pour cela, de nombreux mécanismes de détection d'attaques existent, et peuvent être exploités pour la mise en place d'une sécurité dynamique. Par exemple, plusieurs méthodes, détaillées dans le chapitre 3, permettent de détecter les attaques matérielles par injection de fautes. Il est également possible de détecter les attaques matérielles par analyse des canaux auxiliaires, par exemple grâce à la détection d'une éventuelle sonde électromagnétique placée à proximité d'un circuit intégré [46], ou avec une mesure embarquée des fluctuations de la tension d'alimentation qui révèlerait la présence d'un appareil externe de mesure de cette tension [47, 48]. Des méthodes logicielles permettent également de détecter des attaques par analyse de canaux auxiliaires, grâce à l'utilisation de fausses clés [49]. D'autres solutions, tels que des capteurs de lumière [6], permettent de détecter la décapsulation d'un circuit intégré préalable à d'autres attaques matérielles. En exploitant plusieurs de ces mécanismes de détection d'attaques, les contre-mesures contre diverses attaques matérielles pourraient être activées uniquement lors de la détection de ces attaques, ce qui permettrait un gain important en énergie le reste du temps.

Ces observations montrent que la mise en place d'une sécurité adaptative pour les circuits intégrés peut permettre de répondre aux besoins variables au cours du temps, en assurant un niveau de protection suffisant à chaque instant avec un coût énergétique réduit au minimum. Cela revient à optimiser de manière dynamique le compromis entre sécurité et consommation énergétique.

1.4 Positionnement de la thèse : de la sécurisation de mécanismes de gestion de l'énergie à l'optimisation énergétique de mécanismes de sécurisation

De nombreuses attaques contre des objets ou des réseaux dans l'IoT ont été évoquées dans la section 1.3. Les attaques sur la couche *application*, ainsi que les protections contre ces attaques, sont en général propres à chaque application. Les attaques sur les couches *réseau*, *transport* et *liaison* dépendent du protocole de communication, et peuvent généralement être contrées par la mise en œuvre d'une sécurité adéquate au niveau de ce protocole. Le chiffrement, l'authentification et éventuellement l'intégrité des données échangées dans ces protocoles reposent sur l'utilisation de clés secrètes ; ces clés sont stockées sur les circuits intégrés, et peuvent être obtenues grâce à des attaques matérielles contre ces circuits. Un attaquant en possession de ces clés pourrait alors contourner la sécurité mise en place au niveau des protocoles et des applications. D'un autre côté, la mise hors service d'un seul circuit intégré, par exemple avec une attaque par déni de service ciblée, peut avoir des conséquences sur le routage et la transmission des messages à l'échelle d'un réseau entier. La mise en place d'une sécurité efficace sur la couche *physique*, au niveau du circuit intégré lui-même, est donc primordiale afin d'éviter divers types d'attaques à plusieurs autres niveaux. Dans nos travaux, nous étudions la sécurité d'un circuit pour l'IoT ayant une structure similaire à celle de L-IoT, présenté dans la section 1.2.3.

De nombreuses contraintes sont imposées aux circuits intégrés conçus pour une utilisation dans l'IoT. En particulier, ceux-ci doivent avoir une faible consommation énergétique, de bonnes performances et un niveau de sécurité élevé. Bien que la mise en place d'une sécurité appropriée soit souvent coûteuse en énergie, très peu de solutions permettent actuellement de concilier ces deux contraintes. Dans nos travaux, nous traitons donc de la relation entre sécurité et efficacité énergétique à travers l'étude de divers compromis entre ces deux contraintes. Dans un premier temps, nous nous intéressons à la sécurisation d'un mécanisme de gestion de l'énergie contre des attaques par déni de service. Dans un second temps, nous abordons l'efficacité énergétique de contre-mesures contre les attaques matérielles, qui visent l'implémentation matérielle de primitives cryptographiques.

1.4.1 Sécurité de la radio de réveil pour une gestion efficace de l'énergie

Tandis que la majorité des attaques sur la radio principale peuvent être contrées avec l'utilisation d'un protocole de communication sécurisé, ce n'est pas le cas des potentielles attaques sur la radio de réveil (décrite dans la section 1.2.3), pour laquelle il n'existe pas de protocole de communication communément utilisé à l'heure actuelle. Cette radio ne reçoit ou ne transmet pas de message confidentiel, mais elle occupe tout de même une fonction critique : elle permet de contrôler la sortie de veille de l'ensemble du circuit intégré. C'est donc un mécanisme important pour la gestion de l'énergie d'un circuit intégré. Or, le code envoyé sur cette radio afin de réveiller un circuit est généralement prédéfini et identique pour tous les réveils. Un attaquant peut alors ré-utiliser ce code constant en le renvoyant continuellement au circuit visé, ce qui aurait pour effet de réveiller constamment celui-ci et donc de vider rapidement sa batterie. Une telle attaque, décrite pour la première fois dans [50], est appelée une attaque par déni de sommeil, et fait partie de la catégorie des attaques par déni de service.

Pour contrer cette attaque, les codes de réveil doivent être *dynamiques* : ceux-ci doivent être différents à chaque réveil, et imprévisibles pour un attaquant, c'est-à-dire qu'un attaquant connaissant un ou plusieurs codes de réveil passés ne doit pas être en mesure de prédire la valeur d'un futur code de réveil. Plusieurs méthodes existent afin de générer ces codes, et sont décrites et analysées dans le chapitre 2. Elles ont plusieurs désavantages, tels qu'un coût énergétique élevé ou une mise en œuvre complexe. Nous proposons donc dans le chapitre 2 une nouvelle méthode permettant de générer des codes de réveil dynamiques pour un coût négligeable en énergie, qui tire pleinement partie de la séparation du circuit en deux sous-parties *Always-Responsive* et *On-*

Demand et de l'usage de deux types de radios différentes. Une analyse de la sécurité de cette méthode montre qu'il n'existe pas d'attaque plus efficace contre celle-ci qu'une attaque par force brute, et qu'une telle attaque est irréalisable en pratique. De plus, nous proposons également un processus de réveil et de communication robuste, qui serait compatible avec de nombreux protocoles de communication sur la radio principale.

1.4.2 Efficacité énergétique des contre-mesures matérielles

Les attaques matérielles par injection de fautes et par analyse des canaux auxiliaires ont des objectifs variés, allant de la récupération des clés de chiffrement stockées dans les circuits intégrés à l'altération des données stockées ou du fonctionnement normal d'un algorithme. Elles sont peu coûteuses à mettre en place et requièrent peu d'expertise. Elles constituent donc une menace importante pour la sécurité dans l'IoT. Dans le chapitre 3, nous décrivons les principes physiques et la mise en œuvre de ces attaques, et nous évaluons leur efficacité contre l'implémentation de l'algorithme léger de chiffrement par flot Trivium, qui est particulièrement adapté pour un usage dans l'IoT. De nombreuses protections, présentées dans le chapitre 3, ont été publiées afin de sécuriser l'implémentation d'un algorithme contre l'une ou l'autre de ces attaques, ou contre les deux à la fois. La mise en place de ces protections introduit généralement une augmentation de la consommation instantanée lors de l'exécution de cet algorithme, et/ou une augmentation du temps d'exécution de celui-ci. De plus, de nombreuses contre-mesures requièrent la génération continue de bits aléatoires, ce qui implique également une augmentation de la consommation. Au final, elles ont donc un impact important sur la consommation d'énergie, et ce coût énergétique peut être prohibitif pour une intégration dans l'IoT. Afin de concilier sécurité matérielle et efficacité énergétique, des contre-mesures plus efficaces d'un point de vue énergétique ont été développées au cours de cette thèse.

Dans le chapitre 4, nous proposons une nouvelle contre-mesure, appelée *encodage dynamique*, qui permet de contrer efficacement les attaques matérielles par injection de fautes et par analyse des canaux auxiliaires. Celle-ci est particulièrement adaptée pour la protection d'algorithmes légers tels que Trivium. Cette contre-mesure multiplie environ par quatre la consommation instantanée d'un algorithme, n'augmente pas son temps d'exécution, et requiert entre 0 et 8 bits aléatoires à chaque exécution de l'algorithme. Elle a donc un coût énergétique bien plus faible que les contre-mesures existantes contre ces deux types d'attaques, et nous montrons qu'elle permet de protéger efficacement l'implémentation d'un algorithme contre les attaques existantes.

Pour aller plus loin dans l'efficacité énergétique des contre-mesures matérielles, nous avons évoqué dans la section 1.3.4 la possibilité de reconfigurer de manière dynamique le niveau de sécurité, durant le fonctionnement d'un circuit intégré. La mise en place de contre-mesures adaptatives, activées ou non selon le contexte et les besoins variables en sécurité et en consommation d'énergie, permet de réduire au minimum le coût énergétique de la sécurité. De plus, cette flexibilité des contre-mesures permet de garantir un niveau de sécurité adéquat durant toute la durée de vie d'un circuit intégré, avec la possibilité d'augmenter le niveau de protection lors de la découverte de nouvelles attaques, et elle répond également aux exigences de durabilité et de réutilisation des circuits intégrés. Dans le chapitre 5, nous proposons donc plusieurs contre-mesures matérielles adaptatives. Ce concept est appliqué à l'encodage dynamique, mais également au masquage, qui est une contre-mesure existante largement étudiée. Plusieurs variantes de ces contre-mesures adaptatives sont décrites, afin de couvrir des cas d'usage variés, et une analyse de la sécurité de celles-ci est effectuée.

Enfin, dans le chapitre 6, nous détaillons l'implémentation matérielle sur cibles physiques des contre-mesures proposées précédemment, dans le but d'évaluer expérimentalement leur efficacité contre les attaques matérielles. Plusieurs implémentations sur ASIC et sur FPGA sont décrites, et une évaluation des fuites d'information par canaux auxiliaires sur FPGA est effectuée.

1.5 Publications et valorisation

Conférences avec actes

— M. Montoya, S. Bacles-Min, A. Molnos, and J. J. A. Fournier, « SWARD : A Secure WAke-up RaDio Against Denial-of-Service on IoT Devices », in *Proceedings of the 11th ACM Conference on Security & Privacy in Wireless and Mobile Networks (WiSec)*, 2018, p. 190–195.

— M. Montoya, T. Hiscock, S. Bacles-Min, A. Molnos, and J. J. A. Fournier, « Energy-efficient Masking of the Trivium Stream Cipher », in *2018 25th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, 2018, p. 393-396.

— M. Montoya, T. Hiscock, S. Bacles-Min, A. Molnos, and J. J. A. Fournier, « Adaptive Masking : a Dynamic Trade-off between Energy Consumption and Hardware Security », in *2019 37th IEEE International Conference on Computer Design (ICCD)*, 2019.

Brevets

— M. Montoya, S. Bacles-Min, A. Molnos, and J. J. A. Fournier, « Dispositif d'émission/ réception à radio de réveil résistant aux attaques par déni de sommeil », 2017

— M. Montoya, S. Bacles-Min, A. Molnos, and J. J. A. Fournier, « Registre à décalage protégé contre les attaques physiques », 2018

Présentation orale

— M. Montoya, S. Bacles-Min, A. Molnos, and J. J. A. Fournier, « Lightweight and secure scheme to mitigate Denial-of-Service on wake-up radios for IoT devices », in *Workshop on Practical Hardware Innovations in Security Implementation and Characterization (PHISIC)*, Gardanne, France, 2018.

Poster

— M. Montoya, S. Bacles-Min, A. Molnos, and J. J. A. Fournier, « Lightweight and secure scheme to mitigate Denial-of-Sleep on wake-up radios for IoT devices », in *Design Automation Conference (DAC)*, San Francisco, USA, 2018.

Chapitre 2

Sécurisation de mécanismes de gestion de l'énergie : le cas de la radio de réveil

Sommaire

2.1 Analyse des méthodes existantes 24
2.2 Hypothèses effectuées 26
2.3 Génération du code de réveil 27
2.3.1 Génération récursive 28
2.3.2 Initialisation 29
2.4 Processus de réveil 29
2.5 Analyse de la sécurité 31
2.5.1 Attaque par force brute et taille du code de réveil 32
2.5.2 Discussion sur l'existence d'attaques plus efficaces 33
2.5.3 Fonction de hachage 34
2.6 Implémentation et évaluation 35
2.6.1 Implémentation de notre méthode de génération de codes de réveil 35
2.6.2 Évaluation de la durée de vie d'un nœud dans plusieurs cas 36
2.7 Conclusion 38

Des mécanismes de gestion de l'énergie pour les circuits intégrés sont parfois développés sans tenir compte de la sécurité de ces circuits. Ces mécanismes peuvent alors introduire de nouveaux chemins d'attaques contre ces circuits. Par exemple, CLKSCREW [51] est une attaque publiée en 2017 qui exploite le fait que la fréquence et la tension d'alimentation d'un circuit intégré peuvent être contrôlées par le logiciel embarqué exécuté sur celui-ci, afin d'injecter des fautes lors du fonctionnement de ce circuit qui peuvent ensuite être exploitées de diverses manières. Les radios de réveil constituent un autre exemple d'un tel mécanisme de gestion de l'énergie qui n'est pas sécurisé.

Une radio de réveil est un mécanisme de gestion de l'énergie pour certains circuits intégrés pour l'IoT. Ces circuits sont séparés en deux sous-parties, notées *Always-Responsive* et *On-Demand*, comme cela est décrit dans la section 1.2.3. La partie *On-Demand* contient une radio principale et de nombreux blocs fonctionnels et est en veille la plupart du temps afin d'économiser de l'énergie ; elle est réveillée lors de la réception par la radio de réveil, dans la partie *Always-Responsive*, d'un code de réveil spécifique. Ce code de réveil est généralement identique à chaque réveil, et une telle radio est donc sensible aux attaques par déni de sommeil, au cours desquelles ce code de réveil est renvoyé continuellement par un attaquant afin de réveiller le circuit visé et de vider sa batterie.

Ces attaques peuvent être contrées en générant des codes de réveil *dynamiques*, c'est-à-dire différents à chaque réveil et imprévisibles pour un attaquant. Ces codes de réveil sont générés dans la partie *On-Demand*, qui dispose de ressources de calcul plus importantes que la partie *Always-Responsive*, et sont ensuite stockés dans cette dernière. Bien que plusieurs méthodes existent afin de générer des codes de réveil dynamiques, celles-ci présentent de nombreux inconvénients. Dans ce chapitre, nous proposons donc une nouvelle méthode pour protéger les radios de réveil contre les attaques par déni de sommeil, afin d'optimiser la durée de vie d'un circuit intégré en milieu hostile.

Dans un premier temps, nous décrivons et nous analysons les méthodes existantes pour la génération de codes de réveil dynamiques, puis nous détaillons les hypothèses effectuées dans nos travaux. Nous présentons ensuite une nouvelle méthode récursive de génération de codes de réveil, qui exploite le fait que deux radios différents sont utilisées sur le même circuit intégré : la radio principale sur laquelle les échanges de messages sont effectués de manière sécurisée, et la radio de réveil, qui ne peut pas émettre de messages et qui a un faible débit en réception. Nous décrivons un processus de réveil et d'échange de messages avec ces deux radios, compatible avec de nombreux protocoles de communication sur la radio principale. Une analyse de la sécurité de cette solution montre la difficulté d'effectuer diverses attaques contre celle-ci. Enfin, nous étudions l'effet d'une attaque par déni de sommeil sur la consommation énergétique d'un nœud fonctionnant sur batterie, et nous montrons que notre solution permet de contrer ces attaques pour un coût négligeable en énergie.

2.1 Analyse des méthodes existantes

Plusieurs méthodes ont été proposées afin d'établir des codes de réveil différents à chaque réveil et imprévisibles pour un attaquant. Dans ce chapitre, nous considérons qu'à un instant donné, un attaquant a connaissance d'une partie ou de tous les codes de réveil ayant permis de réveiller le nœud par le passé, mais que cet attaquant doit être incapable de calculer de futurs codes de réveil. De plus, l'attaquant peut envoyer des messages à la radio de réveil, et tenter ainsi de réveiller le circuit intégré si l'un de ces messages est un code de réveil valide.

La première méthode, proposée dans [50], consiste à générer le code de réveil seulement sur l'un des nœuds qui participent à la communication, puis à envoyer ce code via la radio principale aux autres interlocuteurs. Cette transmission doit être effectuée sur un canal sécurisé afin que les attaquants ne puissent pas obtenir ce code. Cette solution introduit donc des échanges de messages supplémentaires sur la radio principale, ce qui est très coûteux en énergie : selon [4], la transmission de données a un coût énergétique équivalent à l'exécution de plusieurs centaines

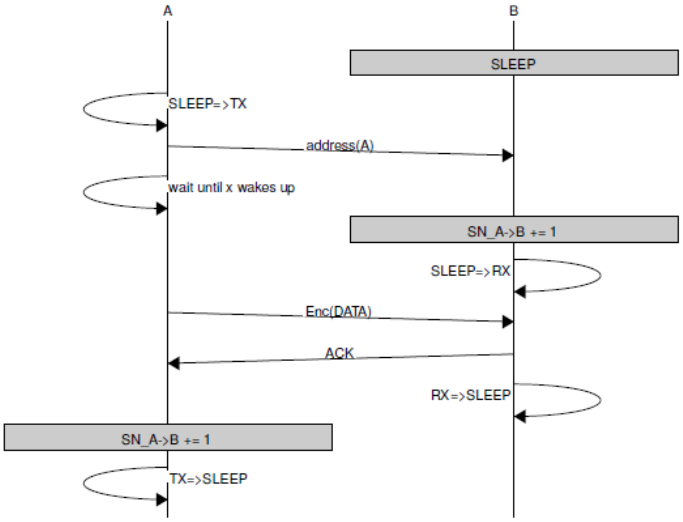


FIGURE 2.1 – Diagramme de séquence de messages entre deux noeuds lorsque le code de réveil est calculé à partir d'un compteur de réveil, noté SN_A->B [56].

d'instructions sur un circuit intégré. Pour une meilleure efficacité énergétique, les codes de réveil devraient être générés localement sur chaque nœud impliqué dans la communication, à partir de données communes, sans créer d'échanges radio supplémentaires ; c'est l'approche suivie dans les autres méthodes existantes, et dans notre solution.

Les auteurs de [52] et ceux de [53] proposent l'utilisation de mots de passe à usage unique, générés sur chaque nœud. Ces mots de passe à usage unique sont calculés à partir de la date et de l'heure actuelle. Dans [52], ces données sont chiffrées, tandis que dans [53], les auteurs proposent l'utilisation de l'algorithme TOTP (Time-based One-Time Password) [54] qui consiste à appliquer une fonction de hachage cryptographique à une clé secrète et à l'heure actuelle. Dans les deux cas, une clé secrète, dédiée au processus de réveil, est requise pour la génération du code de réveil. Ces mots de passe à usage unique calculés à partir d'une heure donnée sont valables durant une durée limitée, et doivent être mis à jour régulièrement. Par exemple, dans [53], un nouveau code de réveil est calculé par tous les nœuds d'un réseau toutes les 30 secondes, ce qui implique un réveil régulier de ces nœuds. De plus, cette méthode requiert la mise à jour continuelle de l'heure dans la partie *Always-Responsive* et une synchronisation temporelle précise de l'ensemble des nœuds d'un réseau, et donc un mécanisme de resynchronisation pour compenser les dérives de leurs horloges respectives au cours du temps. Par conséquent, cette solution a un coût en énergie important.

Enfin, les travaux publiés dans [55] et [56] reprennent le principe du mot de passe à usage unique, en remplaçant la date et l'heure par un compteur de réveil. Ce compteur est incrémenté lors de chaque réveil. Ces propositions s'affranchissent donc des désavantages liés à l'utilisation d'une heure précise et à la nécessaire synchronisation temporelle qui en découle. Dans [55], la valeur du compteur de réveil est concaténée aux adresses des nœuds émetteur et récepteur, et ces données sont chiffrées afin de générer un code imprévisible pour un attaquant ne disposant pas de la clé de chiffrement. Dans [56], une fonction de hachage cryptographique est appliqué à une concaténation de plusieurs données : le compteur de réveil, une clé secrète, et les adresses des nœuds émetteur et récepteur. La figure 2.1 montre le diagramme d'échange de messages entre deux nœuds, avec la méthode de génération de codes de réveil de [56]. Lorsque le nœud A envoie un code de réveil valide au nœud B, ce dernier se réveille, incrémente son compteur de réveil, et attend la réception d'un message de A sur sa radio principale. Après la réception du message, B envoie un acquittement à A pour signifier le succès de cette réception, puis retourne en mode veille. Lors de la réception de cet acquittement, A incrémente également son compteur de réveil, et peut ensuite se mettre en veille.

Les méthodes consistant à calculer des mots de passe à usages uniques, que ceux-ci soient calculés à partir de l'heure ou de compteurs de réveils, nécessitent toutes l'utilisation d'une clé

secrète dédiée à la génération de codes de réveil. L'échange sécurisé de cette clé est généralement complexe et très coûteux en énergie, et peut nécessiter plusieurs échanges de messages sur la radio principale ainsi que l'utilisation de certificats fournis par une tierce partie de confiance, comme expliqué dans [56]. De plus, cette clé doit être stockée et mise à jour de manière sécurisée, ce qui ajoute également un surcoût en énergie. Si cette clé était découverte par un attaquant, que ce soit lors de son échange, de son stockage, de son utilisation ou de sa mise à jour, celui-ci serait en mesure d'effectuer une attaque par déni de sommeil.

Enfin, parmi les méthodes qui consistent à calculer le code de réveil localement sur chaque nœud, sans nécessiter de communication supplémentaire, seule celle proposée dans [52] repose sur une primitive cryptographique *légère* afin de calculer ce code. Les auteurs de ces travaux proposent d'utiliser l'algorithme de chiffrement léger TEA, pour *Tiny Encryption Algorithm* [57]. Les primitives cryptographiques légères sont généralement plus adaptées pour un usage dans l'IoT, afin de diminuer la consommation, mais également le coût d'implémentation, c'est-à-dire la surface silicium pour une implémentation matérielle ou la taille du code en mémoire pour une implémentation logicielle. Dans [55], le code de réveil est calculé avec l'algorithme de chiffrement AES [27], et les fonctions de hachage cryptographiques SHA-1 et SHA-2 [58] sont utilisées dans [53] et [56] ; toutes ces primitives cryptographiques ne sont généralement pas considérées comme étant légères. Par exemple, la solution proposée dans [53], qui utilise l'algorithme SHA-1, occupe 56,6 % de la mémoire d'un microcontrôleur MSP430F1612 [59], ce qui laisse peu de place pour d'autres applications.

2.2 Hypothèses effectuées

Nos travaux reposent sur trois hypothèses, qui sont présentées dans cette section.

Hypothèse 1 *La communication sur la radio principale est sécurisée, c'est-à-dire que les messages sont chiffrés et authentifiés, et que leur intégrité est vérifiée.*

Cette hypothèse est également effectuée dans la solution de [50] qui nécessite des échanges de messages additionnels sur la radio principale, mais c'est une hypothèse supplémentaire par rapport aux autres travaux existants qui ne requièrent pas ces échanges additionnels [52, 53, 55, 56]. Cependant, cette hypothèse est raisonnable, car elle est nécessaire pour garantir la sécurité du nœud ou du réseau contre d'autres attaques. En particulier, une attaque par déni de sommeil pourrait également être effectuée sur une radio principale non sécurisée, en envoyant des requêtes à un nœud pour l'empêcher de retourner en mode veille une fois ses tâches légitimes terminées. Cette dernière attaque est généralement contrée par l'authentification et la vérification de l'intégrité des messages reçus, ce qui permet de rejeter les messages non légitimes, et par la vérification du numéro de séquence des messages, ce qui empêche un attaquant de renvoyer des anciens messages légitimes [60]. Plus généralement, la sécurité sur la radio principale permet de contrer diverses attaques, visant par exemple à espionner les messages échangés, à modifier ces messages ou à en forger de nouveaux. Ainsi, si les échanges sur la radio principale n'étaient pas sécurisés, un attaquant pourrait effectuer des attaques bien plus efficace sur cette radio principale que sur la radio de réveil, et il serait donc secondaire de protéger cette dernière.

Hypothèse 2 *La communication sur la radio principale est robuste face à la perte de messages : lors de l'envoi d'un message par un émetteur, le récepteur de ce message y répond par l'envoi d'un acquittement, qui atteste de la bonne réception du message.*

C'est également une hypothèse supplémentaire par rapport aux travaux existants, excepté ceux de [56] qui nécessitent l'utilisation d'un tel procédé pour assurer la synchronisation des compteurs de réveil. L'envoi d'acquittements pour valider la réception de messages est généralement nécessaire afin de garantir une bonne qualité de service au sein d'un réseau. Cela permet de détecter divers problèmes dans le réseau, et notamment certaines attaques par déni de service,

par exemple lorsqu'un attaquant brouille les communications ou envoie des messages simultanément et sur les mêmes fréquences que les messages légitimes envoyés par les interlocuteurs autorisés dans le but de perturber la réception de ces messages légitimes.

Hypothèse 3 *Le réseau a une topologie en étoile : des nœuds contraints en ressources et séparés en deux sous-systèmes Always-Responsive et On-Demand sont regroupés autour d'une station centrale disposant de ressources de calcul importantes et d'une réserve d'énergie conséquente, voire illimitée.*

Cette hypothèse nous permet de nous placer dans un cas d'étude simple, où chaque nœud peut être réveillé uniquement par la station centrale. Ainsi, un seul code de réveil doit être stocké dans chaque nœud à chaque instant. La méthode proposée pourrait être généralisée à d'autres topologies, en utilisant un code de réveil distinct pour chaque interlocuteur d'un nœud donné, comme décrit dans [56]. Cela nécessiterait des ajustements à la fois pour la génération du code de réveil et pour le protocole de réveil, qui ne sont pas détaillés dans ces travaux. Par la suite, nous considérons donc uniquement le processus de réveil d'un unique nœud par une station centrale. Le réveil d'autres nœuds du réseau par la même station centrale s'effectue de manière identique, tous les nœuds ayant a priori des codes de réveil distincts, ce qui permet de sélectionner le nœud à réveiller.

Ces trois hypothèses sont compatibles avec de nombreux protocoles de communication largement répandus et standardisés, tels que le Wi-Fi et le Wi-Fi HaLow, le Bluetooth et le Bluetooth Low Energy, et les protocoles définis par la norme IEEE 802.15.4 comme ZigBee ou WirelessHART. Par exemple, en ce qui concerne le protocole ZigBee, l'algorithme de chiffrement AES peut être utilisé à la fois pour chiffrer les messages transmis, et pour générer un code MAC afin de valider l'intégrité de ces messages. Outre la sécurisation des échanges, ces protocoles ont également d'autres caractéristiques intéressantes qui permettent leur utilisation dans des environnements hostiles avec une mauvaise qualité du signal radio. Par exemple, ils fournissent une certaine robustesse face au re-jeu ou à la perte de messages, grâce à l'utilisation de numéros de séquence permettant de déterminer l'ordre des messages échangés et à l'envoi d'acquittements pour informer l'expéditeur d'un message de sa bonne réception. Enfin, tous ces protocoles peuvent fonctionner avec une topologie en étoile.

2.3 Génération du code de réveil

Comme nous l'avons vu dans la section 2.1, les méthodes existantes pour la génération de codes dynamiques présentent de nombreux désavantages. Certaines s'appuient sur des mécanismes coûteux en énergie, comme des échanges supplémentaires sur la radio principale, ou un réveil régulier des nœuds afin de mettre à jour le code de réveil. Seule l'une d'entre elles utilise une primitive cryptographique légère, adaptée pour un usage dans l'IoT. Enfin, celles qui ne requièrent pas d'échanges supplémentaires sur la radio principale nécessitent toutes l'utilisation de clés secrètes spécifiques au processus de réveil, qui viennent s'ajouter à celles déjà nécessaires pour la sécurisation des échanges sur la radio principale. La gestion de ces clés additionnelles doit être réalisée de manière sécurisée, ce qui peut avoir un impact sur la consommation du circuit sur le long terme.

Nous proposons donc une nouvelle méthode de génération de codes de réveil dynamiques qui ne souffre pas de ces désavantages. Celle-ci tire partie du fait que deux radios sont présentes sur le circuit intégré, et que la communication sur la radio principale est nécessairement sécurisée, c'est-à-dire que les messages sont chiffrés et authentifiés, et que leur intégrité est vérifiée. Contrairement à la proposition de [50], qui repose également sur la sécurisation des échanges sur la radio principale, notre solution ne requiert par l'échange de messages supplémentaires sur cette radio, et le code de réveil est calculé localement à la fois sur la station centrale et sur le nœud contraint en énergie, en utilisant des données connues seulement de ceux-ci. En s'appuyant sur la sécurité de la radio principale, sans compromettre cette sécurité, notre méthode ne requiert pas l'usage de clés secrètes supplémentaires dédiées à la génération des codes de réveil. Notre méthode est

récursive, c'est-à-dire qu'à chaque réveil, elle utilise des données calculées au réveil précédent. Seules ces données doivent être stockées d'un réveil à l'autre, et il n'est donc pas nécessaire pour notre méthode de maintenir à jour un compteur de réveil ou une heure précise. À chaque réveil, ces données du réveil précédent sont combinées à des données relatives à la session en cours, afin d'augmenter le niveau de sécurité. Ainsi, notre méthode de génération de codes de réveil ne repose pas sur une seule donnée secrète, mais utilise plusieurs données indépendantes inconnues des attaquants, et elle est donc plus robuste que les méthodes existantes.

2.3.1 Génération récursive

Par la suite, nous notons CR un code de réveil, et le code pour le n-ième réveil est appelé CR_n. Deux types de codes sont calculés à chaque réveil : un code *court* et un code *long*. Le code court CR_n est une partie tronquée du code long, et c'est le code de réveil effectif qui est transmis sur la radio de réveil. N'importe quelle fonction de troncature peut être utilisée afin d'obtenir ce code court à partir du code long; on peut par exemple sélectionner les bits de poids faible, ou les bits de poids fort du code long. Selon les radios de réveil, la taille de ce code court peut être comprise entre 8 et 64 bits; une analyse de la longueur de ce code est fournie dans la section 2.5.1. Le code long, noté CR_n^{long}, est utilisé uniquement pour le calcul récursif. Il n'est jamais transmis en entier par radio, et est calculé localement à la fois sur le nœud et sur la station centrale. L'utilisation d'un code long est nécessaire pour garantir la sécurité de cette méthode contre différentes attaques, ce qui est expliqué dans la section 2.5. En notant $tronc$ la fonction de troncature utilisée, on a donc l'équation suivante :

$$\mathrm{CR}_n = tronc(\mathrm{CR}_n^{long}) \tag{2.1}$$

Au n-ième réveil, le code pour le $(n+1)$-ième réveil est calculé récursivement en utilisant le code long CR_n^{long}; celui-ci est stocké à chaque fois que le nœud est mis en veille, et mis à jour lors du calcul du prochain code de réveil. On considère qu'à chaque réveil, M messages sont échangés sur la radio principale, que ceux-ci soient émis ou reçus par le nœud, avec $\mathrm{M} \geq 1$. Un ou plusieurs messages échangés sur la radio principale au cours du n-ième réveil sont également utilisés pour le calcul de ce code de réveil, et sont notés $\mathrm{M}sg_n$. La communication étant sécurisée sur cette radio par hypothèse, ces messages sont échangés de manière chiffrée, et sont donc inconnus des attaquants. En notant $hash$ une fonction de hachage cryptographique telle que SHA-256 [58] ou SPONGENT [61], et || la concaténation, le $(n+1)$-ième code long peut être calculé suivant l'équation :

$$\mathrm{CR}_{n+1}^{long} = hash(\mathrm{CR}_n^{long} \,||\, \mathrm{M}sg_n) \tag{2.2}$$

Ce code long est ensuite tronqué selon l'équation (2.1) pour obtenir le code court, qui permettra de réveiller le nœud au $(n+1)$-ième réveil. Cette équation utilise à la fois le code long CR_n^{long}, qui n'est jamais transmis par radio est qui est donc a priori inconnu d'un attaquant, et un ou plusieurs messages $\mathrm{M}sg_n$ ayant été échangés de manière chiffrée et donc également inconnus d'un attaquant. $\mathrm{M}sg_n$ peut par exemple être le premier ou le dernier message échangé au cours du n-ième réveil, ou encore une combinaison de plusieurs messages qui pourraient être tronqués et concaténés. Le choix du ou des messages à sélectionner dépend du protocole sur la radio principale et de l'application, et doit être effectué au cas par cas, lors de l'initialisation d'un réseau.

Une fonction de hachage cryptographique est une fonction à sens unique : le calcul de CR_{n+1}^{long} ne demande pas beaucoup de ressources, mais il est très difficile, avec les moyens de calcul actuels, de retrouver CR_n^{long} et $\mathrm{M}sg_n$ en connaissant CR_{n+1}^{long}. L'usage de cette fonction de hachage est nécessaire afin de ne pas compromettre la confidentialité des messages échangés sur la radio principale. Une analyse de la sécurité fournie par cette fonction de hachage est effectuée dans la section 2.5.3.

Enfin, il est possible qu'aucun message ne soit échangé sur la radio principale durant le réveil d'un nœud, c'est-à-dire que $\mathrm{M} = 0$. Cela peut par exemple se produire si la qualité de la communication sur le canal radio n'est pas suffisamment bonne, ou si un attaquant brouille intention-

nellement ce canal radio. Dans ce cas, le code de réveil long du prochain réveil est tout simplement calculé en considérant un message vide dans l'équation (2.2), ce qui résulte en l'équation suivante :

$$\mathrm{CR}_{n+1}^{long} = hash(\mathrm{CR}_{n}^{long}) \tag{2.3}$$

La sécurité du mécanisme de génération du code de réveil dans les différents cas est discutée dans la section 2.5.2.

2.3.2 Initialisation

L'équation récursive (2.2) (ou l'équation (2.3) en l'absence de messages sur la radio principale) dépend d'un premier code de réveil CR_1^{long}. Celui-ci doit être inconnu de potentiels attaquants, et généré lors de l'initialisation du réseau ou lors de l'ajout d'un nœud au réseau, avant la première mise en veille de ce nœud. Lors de cette phase d'initialisation, le nœud échange une clé secrète avec la station centrale, afin de communiquer avec cette dernière via la radio principale. Le calcul du premier code de réveil est effectué juste après cet échange de clés, et tire à nouveau partie de la sécurité existante sur la radio principale. En notant K la clé secrète permettant de chiffrer les communications sur la radio principale, et *addr* l'adresse unique du nœud dans le réseau, ce code est calculé de la manière suivante :

$$\mathrm{CR}_1^{long} = hash(\mathrm{K} \,||\, addr) \tag{2.4}$$

Ce code long est ensuite tronqué afin d'obtenir le code qui sera envoyé sur la radio de réveil pour réveiller le nœud une première fois, CR_1. L'utilisation de l'adresse du nœud permet de s'assurer que chaque nœud communiquant avec la radio principale a un code de réveil distinct, et qu'il est donc possible de réveiller chaque nœud indépendamment des autres, même dans le cas où la clé K serait partagée par plusieurs nœuds du réseau.

Cette méthode présente l'avantage de ré-utiliser la même fonction de hachage cryptographique que celle utilisée par la suite lors de chaque réveil. Cependant, d'autres méthodes de génération du premier code de réveil peuvent être envisagées. Par exemple, il est possible d'utiliser la fonction utilisée pour le chiffrement avec la clé K sur la radio principale, notée $\mathrm{E_K}()$. Dans ce cas, le premier code de réveil pourrait être calculé de la manière suivante :

$$\mathrm{CR}_1^{long} = \mathrm{E_K}(addr) \tag{2.5}$$

2.4 Processus de réveil

La figure 2.2 présente un diagramme de séquence de messages entre une station centrale A et un nœud B avec notre processus de réveil. Le nœud B est tout d'abord en mode veille, et le prochain réveil est le n-ième. Lorsque la station A doit communiquer avec le nœud B, elle lui envoie le code de réveil CR_n. Lorsque la radio de réveil sur le nœud B reçoit ce code, elle vérifie sa validité puis réveille le nœud, qui envoie un acquittement ACK à la station A, via la radio principale, pour indiquer le succès du réveil. Cela initie la communication entre les deux entités, avec un certain nombre de messages échangés dans un sens ou dans l'autre, selon l'application et le protocole de communication sur la radio principale. Lorsque la communication est terminée, A et B calculent chacun le code long CR_{n+1}^{long} ainsi que le prochain code de réveil CR_{n+1}, en utilisant certains des messages échangés sur la radio principale. Cette génération du prochain code se fait dans la partie *On-Demand* dans le nœud B, celle-ci ayant un accès direct aux messages échangés sur la radio principale et de meilleures capacités de calcul que la partie *Always-Responsive*. Ce code est ensuite stocké dans la partie *Always-Responsive*, tandis que la partie *On-Demand* retourne en mode veille.

Les nœuds A et B peuvent se désynchroniser, c'est-à-dire que ces deux interlocuteurs peuvent calculer et stocker des codes de réveil différents pour le prochain réveil de B. Cette situation peut a priori survenir dans plusieurs cas : suite à la perte d'un message ou d'un acquittement, lors de la mauvaise réception du code de réveil par le nœud B, ou après un réveil illégitime de celui-ci. Un tel

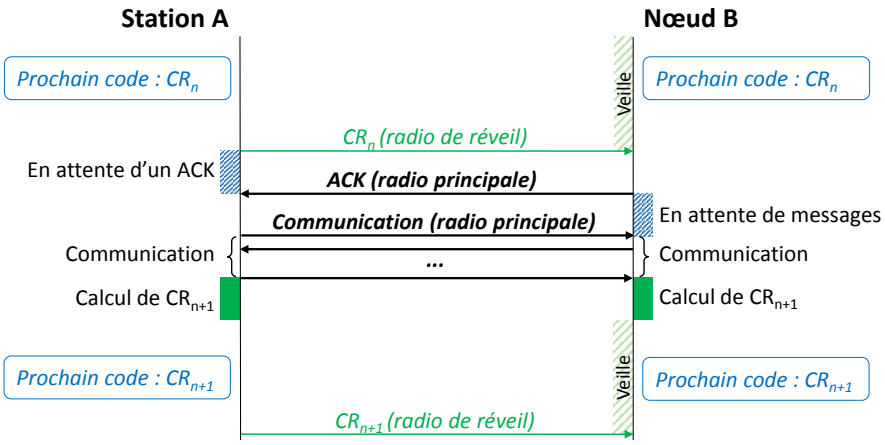


FIGURE 2.2 – Diagramme de séquence de messages entre une station centrale A et un nœud B avec notre processus de réveil, pour un fonctionnement normal.

réveil illégitime peut être non intentionnel et avoir lieu si deux nœuds dans le même réseau partagent le même code de réveil ; plus le réseau est grand ou plus le code de réveil est court, et plus cette situation a de chances de se produire. Par exemple, avec un code de réveil de 8 bits, cette situation a plus d'une chance sur deux de se présenter avec un réseau de seulement 19 nœuds communiquant avec une même station centrale, selon le paradoxe des anniversaires. Pour un code de réveil de 32 bits, il faut au moins 77163 nœuds dans le réseau pour que la probabilité que deux nœuds aient le même code de réveil soit supérieure à 0,5. Un autre cause de réveil illégitime serait un attaquant parvenant à réveiller un nœud en devinant un code de réveil, par exemple grâce à une attaque par force brute.

Afin d'empêcher un blocage, où le nœud B ne pourrait plus être réveillé, il est nécessaire d'utiliser un protocole de communication sur la radio principale dans lequel des acquittements sont envoyés pour valider la bonne réception de chaque message, ce qui correspond à l'hypothèse 2 définie dans la section 2.2. Seuls certains messages sont alors considérés pour le calcul des codes de réveil. Ainsi, l'émetteur d'un message peut seulement utiliser ce message dans le calcul du code de réveil s'il a reçu un acquittement validant la bonne réception de ce message ; le récepteur d'un message peut utiliser celui-ci dans le calcul, après avoir envoyé l'acquittement correspondant. Cela permet de s'assurer que les deux interlocuteurs ont eu connaissance de chaque message pouvant potentiellement être utilisé dans le calcul du code de réveil. Une resynchronisation est alors toujours possible, et la perte de messages ne peut pas être une cause de désynchronisation.

Une désynchronisation est constatée par la station A si après plusieurs tentatives pour réveiller B avec un code de réveil donné, elle ne reçoit aucun acquittement ou aucun autre message de B. Dans ce cas, il existe un nombre limité de situations ayant pu conduire à cette désynchronisation : un ou plusieurs acquittements ont pu être perdus au cours de la session de communication précédente entre A et B, le code de réveil a pu être perdu, ou B a pu être réveillé de manière illégitime. La station A calcule alors un code de réveil pour B pour chacune de ces possibilités, et teste chacun de ces codes de réveil à tour de rôle. On constate que l'ensemble du coût du processus de resynchronisation est porté par la station A, qui calcule et envoie divers codes de réveil possibles. Durant ce temps, le nœud B est en mode veille et attend le prochain code de réveil valide, et sa consommation est donc réduite au minimum. Ainsi, la resynchronisation n'induit pas de surconsommation d'énergie pour le nœud B.

Si, après avoir testé toutes les possibilités, A ne reçoit toujours pas de réponse de B, cela signifie que la communication vers B est temporairement ou définitivement impossible. La station A répète donc le processus complet un certain nombre de fois, après un certain délai, avant de conclure que B est définitivement déconnecté. La station A peut alors lever une alerte, par exemple pour qu'un opérateur aille vérifier physiquement la situation du nœud B ; cette réaction dépend de nombreux paramètres tels que l'application et le coût des nœuds, et sort du cadre de notre étude.

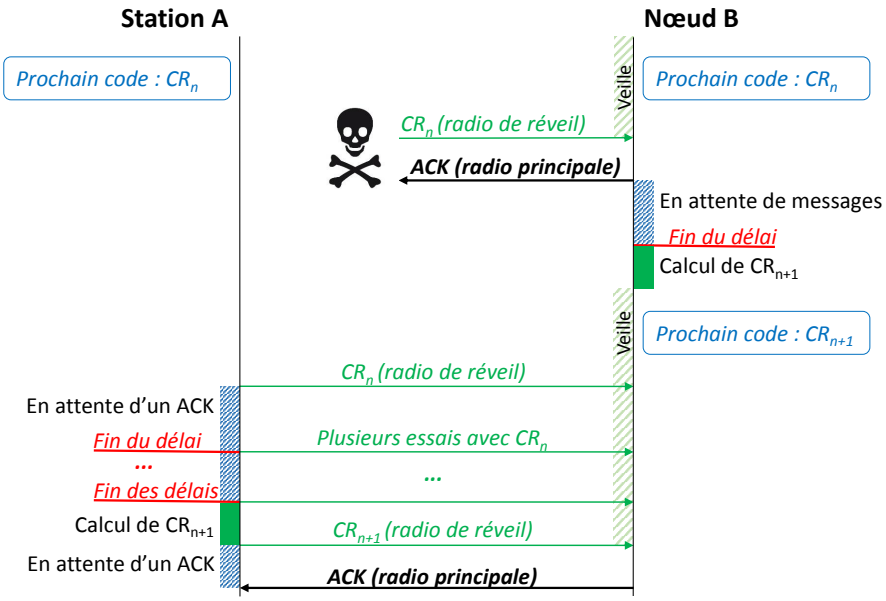


FIGURE 2.3 – Diagramme de séquence de messages entre une station centrale A et un nœud B avec notre processus de réveil, dans le cas où une resynchronisation est nécessaire.

Ce processus est représenté sur la figure 2.3, dans le cas où B est réveillé de manière illégitime, et où un attaquant empêche la transmission de son acquittement de réveil à la station A. Après un certain délai sans communication authentifiée sur sa radio principale, le nœud B calcule un nouveau code de réveil à l'aide du code précédent selon l'équation (2.3) et retourne en mode veille, ce qui résulte en une désynchronisation entre les codes stockés dans A et B. Le processus de resynchronisation décrit plus haut est alors effectué par la station A après plusieurs tentatives infructueuses pour réveiller B sans succès.

On notera que ce problème de désynchronisation est présent dans toutes les propositions existantes pour la génération de codes de réveil dynamiques. Ainsi, lorsque le code du prochain réveil est transmis par radio sur un canal sécurisé, comme dans [50], mais que le message contenant ce code est perdu, la resynchronisation peut se révéler complexe, voire impossible. Le calcul de codes de réveil à partir d'une heure précise, proposé dans [52, 53], souffre naturellement des problèmes de dérives d'horloges sur les différents nœuds, et des mécanismes coûteux doivent être mis en place afin de synchroniser régulièrement l'horloge interne de ces nœuds. Enfin, le calcul de ces codes à partir d'un compteur de réveil, proposé dans [55, 56], peut également souffrir d'une désynchronisation de ce compteur entre les deux entités, auquel cas un processus de resynchronisation similaire à celui proposé dans cette section pourrait être appliqué.

2.5 Analyse de la sécurité

Dans la section précédente, nous avons décrit une méthode générique permettant de calculer récursivement un code de réveil différent lors de chaque réveil. Nous analysons ici plusieurs aspects de la sécurité de cette méthode. Dans un premier temps, nous analysons la résistance aux attaques par force brute visant à deviner le code de réveil court, en fonction de la taille de ce code. Nous discutons ensuite de la possibilité d'effectuer des attaques plus efficaces pour réveiller un nœud. Enfin, nous discutons de la sécurité liée au choix de la fonction de hachage, et nous proposons l'utilisation d'une fonction de hachage légère, adaptée à la génération de codes de réveil sur un nœud contraint en ressources.

Taille du code de réveil (bits)	Temps moyen pour une attaque par force brute
8	0,1 seconde
16	52,4 secondes
24	5,6 heures
32	2,6 mois
64	$1,9 \times 10^9$ années

TABLEAU 2.1 – Temps moyen pour trouver un code de réveil par force brute, en fonction de la taille du code de réveil, pour une radio de réveil ayant un débit en réception de 10 kbps.

2.5.1 Attaque par force brute et taille du code de réveil

Les radios de réveil utilisent un code de réveil court, avec une taille de quelques bits. Dans cette section, nous évaluons le temps nécessaire pour qu'un attaquant devine ce code de réveil par force brute, en envoyant successivement des codes différents sur la radio de réveil. L'efficacité d'une telle attaque est évaluée pour des codes compris entre 8 et 64 bits.

On considère ici que la radio de réveil a un débit en réception de 10 kbps (10.000 bits par seconde). Bien que des débits plus élevés existent pour certaines radio de réveil publiées [15, 14], un haut débit n'est pas nécessaire pour la réception d'un code de réveil de seulement quelques bits, et la plupart des travaux considèrent un débit plus faible, ce qui permet de garantir une faible consommation. Ainsi, le débit est compris entre 1 et 10 kbps dans les travaux de [56] et est inférieur à 8,2 kbps dans [53], tandis qu'un débit de 1 kbps est « raisonnable » selon [14]. Enfin, la radio de réveil AMS AS3932 [62], disponible dans le commerce, autorise un débit maximal de 8,2 kbps, mais fonctionne de manière optimale avec des débits plus faibles. Par la suite, on néglige le temps de réveil du circuit intégré, et le temps nécessaire pour l'envoi d'un acquittement de réveil qui signalerait le succès de l'attaque. Ces temps seraient à prendre en compte dans une attaque réelle, et rallongeraient légèrement celle-ci. De plus, on considère que l'attaquent envoie successivement les différents codes possibles à la radio de réveil jusqu'au succès de l'attaque, et qu'il n'y a aucun délai entre la réception de deux codes.

Dans ces conditions, le temps moyen nécessaire pour le succès d'une attaque par force brute sur la radio de réveil dépend uniquement du débit D et de la taille des codes de réveil courts, notée L. Ce temps moyen T est obtenu avec l'équation suivante :

$$T = \frac{2^{L-1} \times L}{D} \tag{2.6}$$

Le tableau 2.1 donne le temps nécessaire, en moyenne, pour le succès d'une attaque par force brute en fonction de la taille du code de réveil, c'est-à-dire le temps moyen pour trouver le bon code de réveil. On constate qu'avec un débit sur la radio de réveil de 10 kbps, le temps moyen pour trouver un code de 8 ou 16 bits est inférieur à une minute. Ce temps est de quelques heures pour un code de 24 bits, puis de quelques mois pour un code de 32 bits, et quelques années pour un code de 64 bits. Une fois le nœud réveillé, l'attaquant ne pourra pas communiquer avec la radio principale de celui-ci sans disposer des clés secrètes permettant d'authentifier ses messages ; on considère donc qu'après un certain temps sans communication sur la radio principale, le nœud calcule le prochain code de réveil à partir du code long précédent, selon l'équation (2.3), puis retourne en mode veille. Ce code long étant inconnu de l'attaquant, le prochain code de réveil est également inconnu de celui-ci, et il devra effectuer une nouvelle attaque par force brute afin de réveiller à nouveau le nœud.

En se basant sur ces observations, l'utilisation d'un code de réveil de 32 bits semble suffisante pour garantir la sécurité contre les attaques par force brute ; cette taille de code de réveil est choisie pour la suite de nos travaux. En effet, le coût d'une telle attaque est alors trop important par rapport au résultat obtenu : un attaquant doit envoyer des codes pendant plusieurs mois, afin de réveiller une seule fois le nœud, ce qui a un impact limité sur sa batterie. De plus, le code de réveil change à chaque réveil du nœud ; ainsi, à chaque réveil légitime, c'est-à-dire à chaque réveil initié

par la station centrale, l'attaque par force brute serait à recommencer. Par exemple, si le nœud est réveillé une fois par jour par la station centrale, seuls 0,6 % des codes possibles pourraient être testés entre deux réveils légitimes lors d'une attaque par force brute.

2.5.2 Discussion sur l'existence d'attaques plus efficaces

Avec des codes de réveil courts de 32 bits, une attaque par force brute sur la radio de réveil demande des efforts importants pour réveiller le nœud une seule fois. Dans cette section, nous discutons la possibilité d'effectuer une attaque plus efficace afin de réveiller le nœud. Pour effectuer une telle attaque, un attaquant doit connaître certaines données secrètes, qui dépendent d'une part de l'application, et d'autre part du protocole de communication sur la radio principale. Deux cas en particulier sont distingués, en fonction du contenu des messages échangés sur la radio principale. Dans tous les cas, on considère qu'un attaquant est capable d'obtenir le code de réveil CR_n transmis sur la radio de réveil pour le n-ième réveil, et qu'il cherche à calculer le code CR_{n+1}.

Cas 1 : Messages prévisibles, ou absence de messages

Il peut arriver qu'aucun message ne soit échangé avec succès sur la radio principale durant toute la durée de réveil d'un nœud, pour diverses raisons. Dans ce cas, l'équation (2.3) est utilisée afin de générer le prochain code long. Il est également possible que les messages échangés sur la radio principale soient prévisibles pour un attaquant. Par exemple, ces messages peuvent être des mesures issues d'un capteur, ou des commandes envoyées au nœud par la station centrale ; dans les deux cas, le contenu des messages fait partie d'un nombre limité de possibilités connues d'un attaquant. L'équation (2.2) est alors utilisée pour calculer le prochain code long, mais on considère que l'attaquant connaît le contenu de $\mathrm{M}sg_n$.

Ces cas sont similaires du point de vue de l'attaquant, pour qui la seule donnée inconnue lors du n-ième réveil est le code long CR_n^{long}. Cette situation présente donc un niveau de sécurité similaire aux méthodes existantes pour la génération d'un code de réveil [52, 53, 55, 56], avec une seule donnée secrète inconnue d'un attaquant. Cependant, contrairement au méthodes existantes, cette donnée secrète est ici générée à partir d'autres données secrètes, et ne nécessite pas d'échange sécurisé de clés de chiffrement supplémentaires.

Une manière de retrouver le code de réveil long serait d'obtenir la clé secrète K utilisée lors de l'initialisation du réseau, et de calculer récursivement les codes de réveil longs jusqu'au n-ième. Une autre manière consiste à deviner directement CR_n^{long}, par exemple en exploitant la connaissance de CR_n. En effet, CR_n est une partie tronquée de CR_n^{long}. En notant L et L^{long} les longueurs respectives de ces codes, un attaquant connaît L bits des L^{long} bits du code long CR_n^{long}, et doit donc faire des hypothèses sur les $(\mathrm{L}^{long} - \mathrm{L})$ bits restants. Chaque hypothèse sur CR_n^{long} permet de calculer un hypothétique code de réveil CR_{n+1}, dont la validité doit être testée sur la radio de réveil. Par conséquent, plus la différence $(\mathrm{L}^{long} - \mathrm{L})$ est élevée, et plus l'attaquant devra calculer et tester d'hypothèses. Si $(\mathrm{L}^{long} - \mathrm{L}) > \mathrm{L}$, c'est-à-dire si le code de réveil long fait au moins deux fois la taille du code de réveil court, le nombre de bits à deviner est supérieur au nombre de bits L du code court CR_n. Le nombre d'hypothèses à effectuer et à tester est alors supérieur à 2^{32}, ce qui signifie qu'un attaquant devra vraisemblablement effectuer autant d'essais sur la radio de réveil que pour une attaque par force brute. On peut donc raisonnablement choisir un code long CR_n^{long} avec une longueur $\mathrm{L}^{long} \geq 64$ bits, de sorte à ce que la complexité d'une telle attaque soit au moins égale à celle d'une attaque par force brute.

Cas 2 : Messages imprévisibles

La complexité de l'attaque augmente encore lorsque les messages échangés sur la radio principale ne sont pas prévisibles. Dans ce cas, la connaissance de CR_n^{long} ne suffit pas à calculer CR_{n+1}^{long} ; un attaquant doit également être en mesure d'intercepter et de déchiffrer les messages échangés

sur la radio principale. Afin de calculer récursivement le code de réveil CR_{n+1}, un attaquant doit donc non seulement obtenir la clé K utilisée lors de l'initialisation du réseau, mais également intercepter, stocker et déchiffrer tous les messages échangés durant les n premiers réveils. Ainsi, un attaquant qui ne serait pas présent lors des premiers échanges de messages serait incapable de calculer ce code de manière récursive.

De plus, les clés utilisées pour chiffrer les messages peuvent différer d'un réveil à l'autre. En effet, deux sessions de communication différentes peuvent être établies lors de deux réveils distincts. .Or, certains protocoles de communication utilisent des clés de session éphémères et différentes pour chaque session de communication. Dans ce cas, un attaquant souhaitant calculer (CR_{n+1} devrait non seulement obtenir la clé secrète K, mais également tous les messages échangés depuis l'initialisation du nœud, et toutes les clés de sessions permettant de déchiffrer ces messages. Une telle attaque est irréalisable en pratique, d'autant plus qu'une clé de session n'est généralement pas conservée une fois la session terminée.

2.5.3 Fonction de hachage

L'usage d'une fonction de hachage cryptographique pour le calcul des codes de réveil longs répond à deux objectifs, décrits dans les deux paragraphes suivants. Nous proposons ensuite une fonction de hachage cryptographique avec un niveau de sécurité suffisant, au vu de ces objectifs, pour la génération des codes de réveil.

Lors de l'initialisation, la clé secrète nécessaire pour sécuriser les communications sur la radio principale est utilisée pour le calcul du code de réveil. Lors des réveils suivants, les messages échangés sur cette radio sont utilisés dans ce calcul. Afin de ne pas compromettre la confidentialité des échanges sur la radio principale, toutes ces données ne doivent pas être connues d'un attaquant. La connaissance d'un ou plusieurs codes de réveil ne doit donc pas permettre à un attaquant de calculer celles-ci. C'est pourquoi une fonction à sens unique doit être utilisée pour traiter ces données. Le principe d'une telle fonction est qu'il est très complexe de calculer la valeur d'entrée de celle-ci (ici, le code précédent et les messages) en connaissant sa valeur de sortie. Une fonction de hachage cryptographique est généralement une bonne fonction à sens unique, car une des caractéristiques d'une telle fonction de hachage est sa résistance élevée aux attaques par pré-image ; pour rappel, une telle attaque consiste à retrouver une entrée x d'une fonction de hachage en connaissant sa sortie $h(x)$.

Il n'est pas possible d'utiliser n'importe quelle fonction à sens unique pour le calcul des codes de réveil longs. En effet, le code généré doit être imprévisible pour un attaquant, à condition que cet attaquant ne connaisse pas le code long précédent et/ou le contenu des messages échangés sur la radio principale. Ce code ne serait pas suffisamment imprévisible avec n'importe quelle fonction à sens unique. C'est ici la résistance aux attaques par collision, généralement élevée pour les fonctions de hachage cryptographiques, qui nous intéresse. Pour rappel, une attaque par collision consiste à trouver deux valeurs x et y différentes telles que $h(x) = h(y)$. Autrement dit, pour deux données quelconques en entrée, une fonction de hachage cryptographique doit calculer deux valeurs de hachage différentes, imprévisibles pour un attaquant ne connaissant pas les données en entrée.

Nous avons établi précédemment que le code de réveil long, généré par la fonction de hachage cryptographique, doit avoir une taille supérieure ou égale à 64 bits. Par conséquent, l'utilisation d'une fonction de hachage cryptographique légère est suffisante pour respecter cette propriété. La fonction de hachage cryptographique SPONGENT-88 [61] génère une valeur de hachage de 88 bits. Elle est légère et adaptée pour une implémentation matérielle ou logicielle dans l'IoT ; elle a principalement été conçue pour une implémentation matérielle compacte, mais son implémentation logicielle consomme également peu de ressources, avec environ deux fois moins de lignes de code et de mémoire RAM que l'algorithme SHA-256 selon [63].

SPONGENT-88 a une résistance à la pré-image de 80 bits, ce qui signifie qu'en connaissant la valeur de hachage CR_{n+1}^{long}, une attaque par force brute aurait une complexité de 2^{80} pour retrouver

l'entrée de la fonction de hachage, c'est-à-dire CR_n^{long} et Msg_n. Cependant, avec un code de réveil court de 32 bits, seuls 32 bits sur les 88 de la valeur de hachage sont connus d'un attaquant. Par conséquent, un attaquant souhaitant effectuer une attaque par pré-image afin de retrouver une entrée de la fonction de hachage devrait en fait effectuer 2^{56} attaques sur les 2^{56} valeurs possibles de CR^{long}, chacune de ces attaques ayant une complexité de 2^{80}. Une telle attaque par pré-image n'est pas réalisable avec les moyens de calcul actuels. De plus, SPONGENT-88 a une résistance aux collisions de 40 bits. Cette résistance aux collisions garantit qu'une attaque sur cette fonction afin de trouver un code de réveil long serait moins efficace qu'une attaque par force brute sur la radio de réveil ayant une complexité de 2^{32}. Ainsi, l'utilisation de la fonction de hachage SPONGENT-88 permet d'économiser des ressources, tout en garantissant que la confidentialité des données d'entrée de cette fonction n'est pas compromise par le calcul du code de réveil, et en s'assurant que la meilleure stratégie pour un attaquant reste une attaque par force brute sur la radio de réveil.

2.6 Implémentation et évaluation

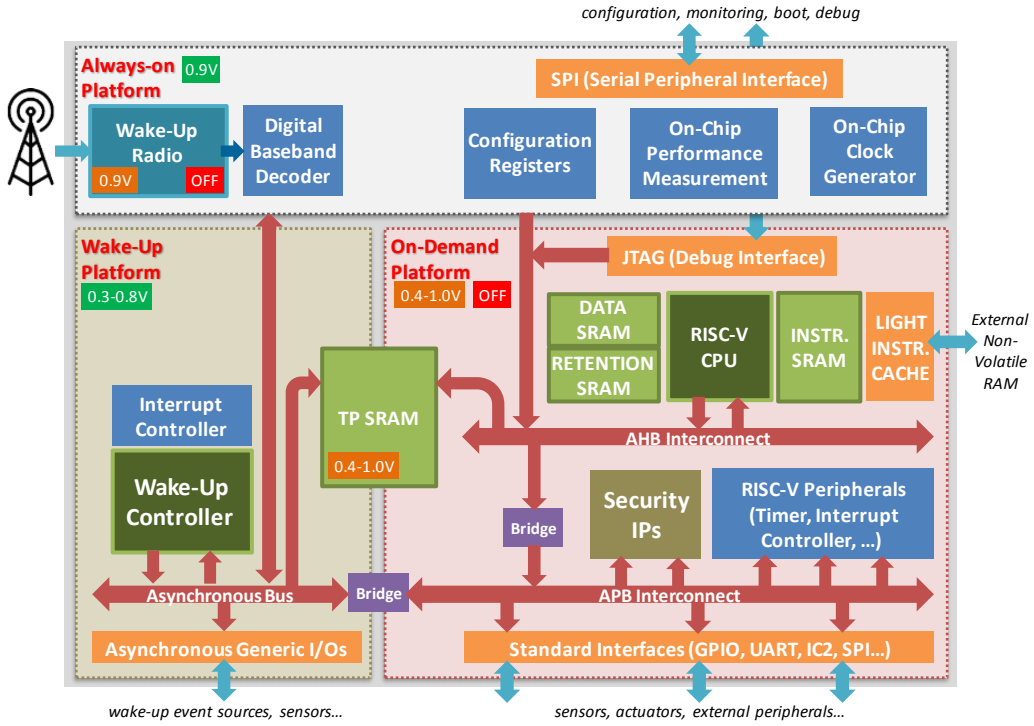


FIGURE 2.4 – Architecture du circuit WARRIOR.

2.6.1 Implémentation de notre méthode de génération de codes de réveil

Afin d'évaluer le coût énergétique et le gain apporté par notre méthode de génération de codes de réveil en présence d'attaques par déni de sommeil, nous effectuons des simulations de l'implémentation de celle-ci sur un circuit intégré. Ces simulations sont réalisées pour une implémentation sur le circuit WARRIOR, fabriqué dans le nœud technologique 28 nm FDSOI, et développé par le CEA LETI comme preuve de concept de la plateforme L-IoT décrite dans la section 1.2.3. Ce circuit est partitionné en deux sous-parties *On-Demand* et *Always-Responsive*. La première contient de nombreux blocs logiques, dont un cœur RISC-V et une interface vers une radio principale (extérieure au circuit intégré), et consomme une quantité importante d'énergie ; elle est donc mise hors tension la plupart du temps, afin d'économiser de l'énergie. La seconde est toujours en activité, et contient notamment une radio de réveil. L'architecture de ce circuit est représentée sur la figure 2.4.

La radio de réveil utilisée a un débit de données en réception configurable, entre 1 kbps et plusieurs dizaines de kbps [13]. Par la suite, on considère un débit de 10 kbps, ce qui est suffisant pour la réception d'un code de réveil de quelques bits sans trop augmenter la consommation. Cette radio a une sensibilité de -60 dBm et peut fonctionner sur les bandes de fréquences comprises entre 433 MHz et 2,4 GHz, et elle peut donc être utilisée conjointement à de nombreux protocoles de communication existants pour l'IoT sur la radio principale. Enfin, elle a une consommation moyenne de puissance de 48 µW.

Lorsque l'ensemble du circuit fonctionne, sa consommation de puissance est de 34,23 mW, selon des simulations post-layout. On notera que cette consommation est seulement celle du circuit intégré ; la radio principale étant extérieure à celui-ci, sa consommation devrait être additionnée à la consommation du circuit intégré lorsque celui-ci est réveillé afin d'obtenir la consommation totale du nœud. En l'absence de valeurs pour la consommation du circuit en mode veille, on peut estimer que celle-ci est environ égale à la consommation de la radio de réveil, soit 48 µW. On considère par la suite que le circuit est alimenté à 0,9 V : bien que plusieurs niveaux de tension d'alimentation soient possibles, c'est la tension d'alimentation optimale pour une utilisation dans des conditions standard, à la fois pour la radio de réveil et pour la partie *On-Demand*.

Les codes de réveil sont calculés par l'algorithme SPONGENT-88, implémenté en logiciel sur le cœur RISC-V. D'après des simulations fonctionnelles au niveau RTL, l'exécution de l'algorithme SPONGENT-88 sur le cœur à une fréquence de 150 MHz s'effectue en 60,3 millisecondes. On estime donc que le coût énergétique de notre méthode de génération de codes de réveil correspond à la consommation du circuit intégré durant ces 60,3 millisecondes, qui s'ajoutent au temps de calcul nécessaire pour les autres opérations effectuées pendant le réveil du nœud, et pendant lesquelles le nœud ne peut pas retourner en mode veille. On considère que l'énergie dépensée pour stocker le code long dans la partie *On-Demand* et le code de réveil court dans la partie *Always-Responsive* est négligeable. Par conséquent, la génération du code de réveil consomme 2,06 mJ à chaque réveil. On notera que ce coût est calculé au pire cas, en considérant que le cœur RISC-V effectue continuellement des calculs pendant l'ensemble du réveil d'un nœud, ce qui justifie d'additionner le temps de calcul du code de réveil au temps de réveil total.

2.6.2 Évaluation de la durée de vie d'un nœud dans plusieurs cas

Dans cette partie, la consommation d'énergie par jour et la durée de vie sont calculées à la fois pour un nœud non protégé, et pour un nœud protégé avec notre méthode de génération de codes de réveil, avec et sans attaques par déni de sommeil. Elles sont évaluées selon un cas d'usage typique pour l'IoT, issu de [55] : le nœud est réveillé par la station centrale 10 fois par jour, et chaque réveil dure en moyenne 60 secondes, ce qui comprend le temps de réveil, diverses opérations telles que des mesures effectués par un capteur intégré et de l'agrégation de données, la communication avec la station centrale via la radio principale, et la mise en veille. On considère de plus que le nœud est alimenté par une pile de 1600 mAh.

On considère que le réveil et la mise en veille sont instantanés ; en réalité, ces deux étapes nécessitent un certain temps, au cours duquel la consommation de puissance est comprise entre 48 µW et 34,23 mW. Du fait de cette approximation, on considère également qu'un nœud non protégé sur lequel une attaque par déni de sommeil est effectuée reste toujours éveillé. Les consommations d'énergie fournies dans cette section sont donc des estimations au pire cas, à la fois pour le nœud non protégé et pour le nœud protégé.

En notant P_{veille} et $P_{réveil}$ la puissance moyenne durant respectivement la veille et le réveil, et $T_{réveil}$ le temps de chaque réveil, soit 60 secondes, l'énergie consommée par jour par le circuit non protégé sans attaque, E_{std}, est obtenue par la formule suivante :

$$E_{std} = 10 \times T_{réveil} \times P_{réveil} + (24 \times 60 \times 60 - 10 \times T_{réveil}) \times P_{veille} \tag{2.7}$$

La consommation d'énergie par jour de ce circuit lorsqu'une attaque par déni de sommeil est effectuée sur celui-ci est notée E'_{std} et est donnée par la formule :

$$E'_{std} = 24 \times 60 \times 60 \times P_{réveil} \tag{2.8}$$

Circuit	Attaque DoS	Consommation journalière (J)	Durée de vie (jours)
Non protégé	*Non*	24,656	210,25
	Oui	2457,5	1,7528
Protégé	-	24,677	210,07

TABLEAU 2.2 – Consommation d'énergie journalière et durée de vie d'un nœud avec et sans protection, et avec et sans attaque par déni de sommeil.

L'analyse de la sécurité réalisée dans la section 2.5 permet d'estimer qu'un attaquant n'est pas en mesure de réveiller le nœud protégé. En effet, la stratégie la plus efficace est une attaque par force force brute sur le code de réveil. Or, une telle attaque requiert en moyenne 2,6 mois pour trouver un code de réveil, tandis que la durée de vie de ces codes est inférieure à 2,4 heures. La consommation d'énergie journalière d'un nœud protégé par notre méthode de génération de codes de réveil est donc identique en présence et en l'absence d'une attaque sur la radio de réveil. Celle-ci est notée E_{prot} et est donnée par la formule suivante :

$$E_{prot} = 10 \times (T_{réveil} + 0,0603) \times P_{réveil} + (24 \times 60 \times 60 - 10 \times (T_{réveil} + 0,0603)) \times P_{veille} \quad (2.9)$$

Enfin, l'énergie totale que la batterie peut fournir au circuit peut être calculée de la manière suivante :

$$\begin{aligned} E_{dispo} &= 1600\ mAh \times 0,9\ V \\ &= 5184\ J \end{aligned} \quad (2.10)$$

En l'absence d'un mécanisme de recharge de la batterie, la durée de vie de la batterie est définie comme le temps avant la décharge complète de celle-ci. Cette durée de vie, en jours, est donc le ratio entre l'énergie totale disponible dans la batterie, et la consommation d'énergie journalière du nœud. Le tableau 2.2 donne la consommation d'énergie par jour et la durée de vie des circuits protégés et non protégés, en présence ou non d'une attaque par déni de sommeil notée DoS pour *Denial-of-Sleep*. On constate que l'utilisation de notre méthode de génération de codes de réveil augmente la consommation journalière de $10 \times 2,06$ mJ, soit 20,6 mJ. Cela représente une augmentation de 0,08 % de cette consommation. Une attaque par déni de sommeil, qui maintient éveillé le nœud non protégé, a pour effet de multiplier par 120 la consommation d'énergie journalière de ce nœud, et de réduire d'autant sa durée de vie, qui est alors inférieure à deux jours. Cette attaque n'a pas d'effet sur le nœud protégé. Par conséquent, au prix d'un surcoût négligeable en énergie, notre méthode de génération de codes de réveil permet d'augmenter considérablement la durée de vie d'un nœud face à une attaque par déni de sommeil.

	Envoi sur la radio principale [50]	**Date et heure actuelle** [52, 53]	**Compteur de réveil** [55, 56]	**Notre solution**
Communication supplémentaire	✔	✘	✘	✘
Clés secrètes supplémentaires	*Inconnu*	✔	✔	✘
Synchronisation temporelle	✘	✔	✘	✘
Algorithme complexe	*Inconnu*	[52] : ✘ [53] : ✔	✔	✘

TABLEAU 2.3 – Caractéristiques des méthodes de génération de codes de réveil existantes, et comparaison avec notre solution.

2.7 Conclusion

Dans ce chapitre, nous présentons une nouvelle méthode de génération de codes de réveil dynamiques afin de contrer les attaques par déni de sommeil sur la radio de réveil. Notre solution ne souffre pas des problèmes inhérents aux méthodes de génération de codes de réveil existantes. Les caractéristiques principales de ces méthodes et de notre solution sont représentées dans le tableau 2.3 ; dans celui-ci, l'utilisation par une méthode d'un *algorithme complexe* signifie que la méthode en question n'utilise pas de primitive cryptographique légère pour calculer les codes de réveil. Toutes les caractéristiques listées dans ce tableau ont un effet négatif sur la consommation ; la présence d'une de ces caractéristiques est représentée par une coche rouge, et son absence par une croix verte. Les méthodes existantes ont généralement un coût élevé en énergie, du fait de communications supplémentaires sur la radio principale, de la mise en place d'un mécanisme de synchronisation temporelle, ou de calculs complexes. De plus, la plupart d'entre elles requièrent l'échange et la maintenance de clés secrètes dédiées au processus de génération de codes de réveil ; cela a un certain coût énergétique et rajoute des contraintes sur le circuit pour la gestion sécurisée de ces clés.

Notre méthode de génération de codes de réveil utilise une fonction de hachage cryptographique légère, et ne nécessite pas de maintenir une clé secrète dédiée au processus de réveil ; de plus, les codes sont générés localement sur chaque nœud, et leur calcul ne requiert pas d'échanges supplémentaires sur la radio principale. Elle repose sur deux hypothèse supplémentaires par rapport aux propositions existantes : la communication sur la radio principale doit être sécurisée, et robuste face à la perte de messages. Ces deux hypothèses sont raisonnables, cette communication sécurisée et robuste étant nécessaire pour la protection du nœud et du réseau contre de nombreuses autres attaques. Nous nous plaçons également dans un cas d'étude simple avec une topologie en étoile, bien que cette hypothèse ne soit pas strictement nécessaire et que notre méthode puisse être étendue à la protection de nœuds dans d'autres types de réseaux en suivant la méthodologie décrite dans [56].

Notre méthode repose sur deux types de codes, qui sont générés de manière récursive : le code long sert uniquement au calcul du code suivant et n'est jamais transmis par radio, tandis que le code court est transmis à la radio de réveil afin de sortir le nœud de veille. La génération des codes de réveil utilise à la fois les codes longs, et des messages échangés de manière chiffrés sur la radio principale à chaque réveil. Les codes longs comme les messages sont inconnus des attaquants, ce qui empêche ceux-ci de calculer les codes de réveil. Ces données sont traitées par une fonction de hachage cryptographique, qui est une fonction à sens unique, afin d'empêcher un attaquant d'en prendre connaissance, ce qui poserait des problèmes de confidentialité.

Une analyse de la sécurité de cette méthode montre qu'il n'existe pas d'attaque plus efficace qu'une attaque par force brute sur la radio de réveil. Une telle attaque n'est pas réalisable en pratique pour un code de réveil de 32 bits, le temps moyen nécessaire pour deviner un seul code étant largement supérieur à la durée de vie de ce code avant son renouvellement. L'analyse de la sécurité de notre méthode montre également que la fonction de hachage cryptographique légère SPONGENT-88 peut être utilisée pour la génération des codes de réveil. En particulier, cette fonction protège efficacement ses données d'entrée contre une attaque par pré-image, et empêche ainsi un attaquant de remonter au contenu des messages échangés sur la radio principale, voire à la clé secrète utilisée pour chiffrer ces messages. De plus, la résistance de SPONGENT-88 aux attaques par collision permet de garantir que le code de réveil généré est imprévisible pour un attaquant n'ayant pas connaissance des messages et de la clé secrète. L'usage d'une autre fonction de hachage cryptographique pourrait également être envisagé, afin d'explorer divers compromis entre le niveau de sécurité fourni, la consommation et le coût d'implémentation (en surface silicium pour une implémentation ASIC, et en occupation de la mémoire pour une implémentation logicielle).

Nous proposons également un processus de réveil et de communication entre un nœud et une station centrale disposant de ressources importantes. Ce processus est compatible avec l'usage sur la radio principale de nombreux protocoles de communication communément utilisés dans l'IoT.

En cas de désynchronisation de ces deux interlocuteurs, c'est-à-dire si les codes de réveil stockés par ceux-ci diffèrent à un moment donné, notre processus de réveil permet une resynchronisation efficace, dont le coût est entièrement supporté par la station centrale. Ce processus n'impacte donc pas la consommation énergétique du nœud contraint en énergie.

Enfin, une simulation de l'implémentation de notre méthode sur silicium montre que celle-ci a un coût en énergie négligeable par rapport à la consommation énergétique journalière d'un circuit intégré. En considérant un cas d'usage typique dans l'IoT, tandis que la durée de vie d'un nœud non protégé est réduite par 120 lors d'une attaque par déni de sommeil, notre solution permet de protéger efficacement le nœud contre ces attaques pour une augmentation de seulement 0,08 % de sa consommation d'énergie par jour.

Les travaux présentés dans ce chapitre ont donné lieu à un dépôt de brevet en 2017 et à un article figurant dans les actes de la conférence WiSec 2018 qui s'est déroulée à Stockholm. Une communication orale a également été effectuée lors de la conférence PHISIC 2018 à Gardanne. Enfin, un poster sur ce sujet a été présenté à la conférence DAC 2018 à San Francisco.

Chapitre 3

Optimisation énergétique de mécanismes de sécurisation : le cas des contre-mesures contre les attaques matérielles

Sommaire

3.1 Attaques par analyse des canaux auxiliaires 42
3.1.1 Différents types de canaux auxiliaires 43
3.1.2 Exploitation des canaux auxiliaires 44
3.2 Cas d'étude : fuites par canaux auxiliaires de l'algorithme Trivium 46
3.2.1 Implémentation matérielle de Trivium 46
3.2.2 Attaques par canaux auxiliaires contre Trivium 47
3.2.3 Évaluation de l'ensemble des fuites d'information par canaux auxiliaires . . 49
3.2.4 Méthodologie : corrélation ou T-test? 51
3.3 Attaques par injection de fautes 52
3.3.1 Moyens d'injection 52
3.3.2 Types de fautes obtenues 53
3.3.3 Exploitation des fautes 54
3.4 Protections contre les attaques par canaux auxiliaires 55
3.4.1 Réduction du rapport signal sur bruit 55
3.4.2 Masquage des données 62
3.4.3 Combinaison de plusieurs contre-mesures 67
3.5 Protections contre les attaques par injection de fautes 68
3.5.1 Détection et prévention de l'injection physique de fautes 68
3.5.2 Détection et correction des fautes 68
3.6 Protections mixtes 70
3.7 Conclusion 73

Dans le chapitre précédent, nous avons traité de la sécurisation d'un mécanisme de gestion de l'énergie d'un circuit intégré. Dans ce chapitre, ainsi que dans les chapitres 4, 5 et 6, nous abordons l'efficacité énergétique des contre-mesures matérielles, ce qui représente un autre aspect du compromis à effectuer entre sécurité et consommation d'énergie pour les circuits intégrés pour l'IoT.

De nombreux circuits intégrés pour l'IoT opèrent de manière non supervisée et sont physiquement accessibles. Une personne malintentionnée peut donc effectuer des attaques matérielles sur ces circuits. En particulier, contrairement aux attaques matérielles destructives, les attaques par analyse des canaux auxiliaires et par injection de fautes peuvent être effectuées avec un matériel peu coûteux, et elles requièrent peu d'expertise et de connaissances sur le layout du circuit visé. Ces attaques ont des objectifs variés, allant de la récupération d'informations sur le circuit intégré et sur les données manipulées, à l'altération des données stockées ou du fonctionnement normal d'un algorithme. Ces attaques exploitent l'implémentation physique des primitives cryptographiques, et non pas des faiblesses mathématiques dans la conception des algorithmes eux-mêmes. Par exemple, elles peuvent permettre à un attaquant d'obtenir les clés secrètes manipulées lors de l'exécution d'une primitive cryptographique. Une fois en possession de ces clés, un attaquant peut effectuer divers types d'attaques contre le circuit intégré lui-même ou contre d'autres objets appartenant au même réseau. Il est donc nécessaire de protéger efficacement les circuits intégrés contre les attaques par analyse des canaux auxiliaires et par injection de fautes.

Nous nous intéressons plus particulièrement à la protection des implémentations matérielles sur ASIC des primitives cryptographiques. En effet, les implémentations logicielles et les implémentations matérielles sur FPGA présentent généralement une consommation instantanée et un temps d'exécution plus élevés, et sont donc moins adaptées pour un usage dans l'IoT avec de fortes contraintes sur la consommation énergétique. Les principes généraux des attaques et des contre-mesures présentées dans ce chapitre peuvent également être appliqués pour des implémentations sur FPGA. D'autres protections peuvent être mises en œuvre sur FPGA, qui consistent à reprogrammer partiellement ou complètement le circuit à chaque exécution d'une application protégée [64]. Ces contre-mesures sortent du cadre de nos travaux et ne sont pas détaillées par la suite.

Nos travaux visent à proposer des contre-mesures génériques, applicables en particulier pour la protection de primitives cryptographiques légères dans le cadre de l'IoT. Les objectifs de ce chapitre sont donc de décrire les attaques contre l'implémentation de ces primitives et les contre-mesures existantes, ce qui permet de comprendre les protections proposées dans les chapitres suivants. Par conséquent, les contre-mesures dédiées à certains algorithmes de chiffrement asymétrique, dont elles exploitent les spécificités architecturales, sortent de ce cadre et ne seront pas décrites par la suite.

Dans un premier temps, nous décrivons la mesure de divers effets physiques et son exploitation afin de mettre en œuvre des attaques par analyse des canaux auxiliaires. Nous présentons comme cas d'étude les attaques existantes contre l'algorithme léger de chiffrement par flot Trivium, qui est représentatif des primitives cryptographiques légères que nous cherchons à sécuriser dans nos travaux. Nous mettons ces attaques en application avec des simulations post-layout d'une implémentation de Trivium et nous analysons les fuites d'information par canaux auxiliaires de cette implémentation. Nous décrivons ensuite les principes physiques sur lesquels se fondent les attaques par injection de fautes, ainsi que l'exploitation de ces fautes. Enfin, nous détaillons les contre-mesures matérielles publiées à ce jour contre ces deux attaques.

3.1 Attaques par analyse des canaux auxiliaires

De nombreux effets physiques sont observables lors du fonctionnement d'un circuit intégré, dont certains sont directement liés aux données manipulées et aux opérations effectuées. Ces effets sont appelés *canaux auxiliaires*. Les attaques par analyse des canaux auxiliaires consistent à mesurer au moins l'un de ces effets afin d'obtenir des informations sur les calculs effectués par le

circuit. Elles peuvent permettre de récupérer des données secrètes telles que des clés de chiffrement, ou des informations sur les opérations effectuées et donc sur les algorithmes exécutés par le circuit.

3.1.1 Différents types de canaux auxiliaires

Il y a plusieurs types de canaux auxiliaires pouvant être exploités afin de mener une attaque avec succès. Comme nous l'avons expliqué dans la section 1.2.1, la consommation de courant dépend de l'activité d'un circuit, c'est-à-dire de la commutation des transistors au sein de celui-ci, et donc des données manipulées. La mesure de cette consommation peut donc permettre de retrouver ces données. Cette mesure peut par exemple être effectuée au niveau de l'alimentation externe d'un circuit intégré, auquel cas elle requiert un matériel peu coûteux.

Le passage du courant dans les couches de métal qui constituent un circuit intégré crée des émissions électromagnétiques, ces couches métalliques se comportant comme des antennes. Ces émissions peuvent être mesurées avec une sonde électromagnétique placée à proximité immédiate du circuit intégré. L'amplitude des émissions électromagnétiques mesurées décroit avec le carré de la distance entre la sonde et le circuit, selon [44]. Comme ces émissions proviennent du passage du courant dans le circuit intégré, elles sont hautement corrélées à la consommation du circuit, et ces deux grandeurs sont directement liées à l'activité d'un circuit. Les auteurs de [65] ont montré expérimentalement que lorsque des fuites d'information par canaux auxiliaires sont observées en mesurant la consommation d'un circuit, des fuites d'information similaires sont observées en mesurant les émissions électromagnétiques de ce circuit. Cependant, du fait que l'amplitude des émissions décroit rapidement avec la distance, la mesure de celles-ci peut donner des informations sur des variations locales de l'activité d'un circuit, tandis que la mesure de courant au niveau de l'alimentation externe de ce circuit permet uniquement de connaître l'activité totale de ce circuit.

Le temps que met un algorithme à calculer un résultat peut dépendre à la fois des données traitées et des opérations effectuées sur ces données [66]. Ce temps d'exécution peut donc également être exploité lors d'une attaque par canaux auxiliaires. En particulier, cette méthode peut être appliquée lors de l'attaque d'implémentations logicielles ; selon [44], dans le cas d'implémentations matérielles synchrones, le temps d'exécution d'un algorithme est généralement constant, car le nombre d'opérations effectuées ne dépend pas des données traitées, et chaque opération est effectuée dans un temps limité du fait de l'utilisation de logique séquentielle. Nos travaux portent essentiellement sur la protection de primitives cryptographiques légères implémentées sur silicium, et nous ne décrirons donc pas ces attaques plus en détails.

Des émissions de lumière visible ou infrarouge sont observables lors du fonctionnement de circuits intégrés. Selon [44], ces émissions optiques sont rares, ce qui impose de répéter un nombre élevé de fois chaque opération effectuée par le circuit intégrée avant d'obtenir une cartographie des émissions pour chaque opération.

Des émissions acoustiques ont été observées lors du fonctionnement d'un ordinateur [67]. Elles sont principalement observables dans les blocs qui fournissent l'alimentation aux circuits intégrés tels que les régulateurs de tension, et sont dues à des vibrations mécaniques de certains composants électriques comme les condensateurs. Elles sont donc liées au courant consommé par les circuits intégrés, et les contre-mesures contre l'analyse de la consommation devraient également permettre d'empêcher l'analyse des émissions acoustiques. À ce jour, elles ont uniquement été observées sur des systèmes complexes ayant une consommation élevée, tels que des ordinateurs, et il n'est pas certain qu'elles puissent être exploitées avec les moyens actuels sur des objets connectés ayant une faible consommation.

Du fait du peu d'attaques publiées exploitant l'analyse des émissions acoustiques et optiques, mais également de la complexité de mener ces attaques en pratique, nous ne détailleront pas l'exploitation de ces canaux auxiliaires par la suite.

3.1.2 Exploitation des canaux auxiliaires

La mesure des canaux auxiliaires peut être exploitée de diverses manières, par exemple pour déterminer quelle primitive cryptographique est exécutée sur un circuit intégré et à quel moment. Dans cette section, nous nous intéressons aux attaques visant l'obtention de données secrètes manipulées par un algorithme, telles que des clés de chiffrement. En particulier, nous détaillons les attaques exploitant la mesure de la consommation de courant ou des émissions électromagnétiques, car celles-ci peuvent être menées avec un matériel peu coûteux, sans décapsuler le circuit intégré, et elles ne requièrent pas de connaissance préalable sur le layout du circuit intégré. Pour rappel, ces deux canaux auxiliaires sont fortement corrélés et liés aux mêmes phénomènes physiques, i.e. la commutation des transistors. Tandis que la mesure de ces deux canaux auxiliaires diffère, le traitement des données mesurées est similaire dans ces deux cas. Nous détaillerons donc uniquement par la suite les attaques exploitant la mesure de la consommation de courant. Dans cette section, le terme *consommation* désigne la consommation de courant, afin d'alléger les notations.

Au cours de ces attaques, un attaquant mesure un certain nombre de traces de consommation au moment de la manipulation de données secrètes, par exemple lors de l'exécution d'un algorithme de chiffrement. Une trace de consommation est constituée de plusieurs points, chacun étant une mesure de la consommation du circuit à un instant donné ; elle montre donc l'évolution de la consommation durant l'exécution d'un algorithme. Il existe deux types d'attaques par analyse de la consommation. Les attaques dites *simples*, pour lesquelles seulement une seule trace de consommation est nécessaire, consistent généralement en une analyse visuelle de cette trace. Ces attaques ne sont possibles que dans certains cas, et requièrent généralement une connaissance précise de l'implémentation de l'algorithme ciblé ainsi qu'un rapport signal sur bruit élevé [43, 44]. Les attaques statistiques, introduites par Kocher *et al.* en 1999 [68], consistent à mesurer un certain nombre de traces de consommation et à appliquer des méthodes statistiques pour traiter ces traces et en extraire une information sur les données secrètes. Elles requièrent très peu de connaissances préalables sur l'implémentation de l'algorithme, et peuvent être mises en œuvre contre la majorité des algorithmes de chiffrement, parfois même en présence de contre-mesures. Un exemple d'attaque statistique est l'analyse par corrélation de la consommation, appelée par la suite CPA pour *Correlation Power Analysis*. Une CPA peut se décomposer en cinq étapes selon [43], décrites ci-après.

Étape 1 : Mesure de la consommation. Au cours de cette étape, l'attaquant acquiert un certain nombre D de traces de consommation lors de l'exécution de l'algorithme de chiffrement sur D données d'entrée différentes, notées d. Une hypothèse raisonnable est que l'attaquant connaît les données d'entrée de l'algorithme, notées *textes en clair* dans le cas du chiffrement par bloc. L'attaque s'effectue de manière similaire si l'attaquant connaît les données de sortie, chiffrées, de cet algorithme, et ce cas ne sera pas détaillé ici. Chaque trace de consommation comprend T points, chacun correspondant à une mesure à un instant donné. On obtient donc une matrice M de mesures, de taille D×T.

Étape 2 : Choix d'une valeur intermédiaire de l'algorithme de chiffrement. Cette valeur correspond au résultat d'une opération $f(d,k)$ sur deux données : une valeur secrète k, et un texte en clair d. La fonction f est appelée fonction de sélection. La valeur secrète est généralement une partie de la clé de chiffrement, que nous appellerons sous-clé par la suite. En divisant la clé en plusieurs sous-clés et en attaquant chacune de ces sous-clés séparément, une CPA repose sur une approche de « diviser pour mieux régner » afin de réduire le nombre de possibilités et donc la complexité de l'attaque. Par exemple, pour une clé de 128 bits, un attaquant doit faire 2^{128} hypothèses lors d'une attaque par force brute, ce qui est impossible en pratique avec les moyens de calculs actuels et dans un temps raisonnable. En décomposant la clé en 16 sous-clés d'un octet chacune, et en obtenant chaque sous-clé successivement avec une CPA, l'attaquant doit faire uniquement 256 hypothèses par sous-clé, soit 2^{12} hypothèses pour la clé complète. Les étapes suivantes sont détaillées uniquement pour attaquer une sous-clé donnée, celles-ci étant ensuite répétées pour chaque sous-clé.

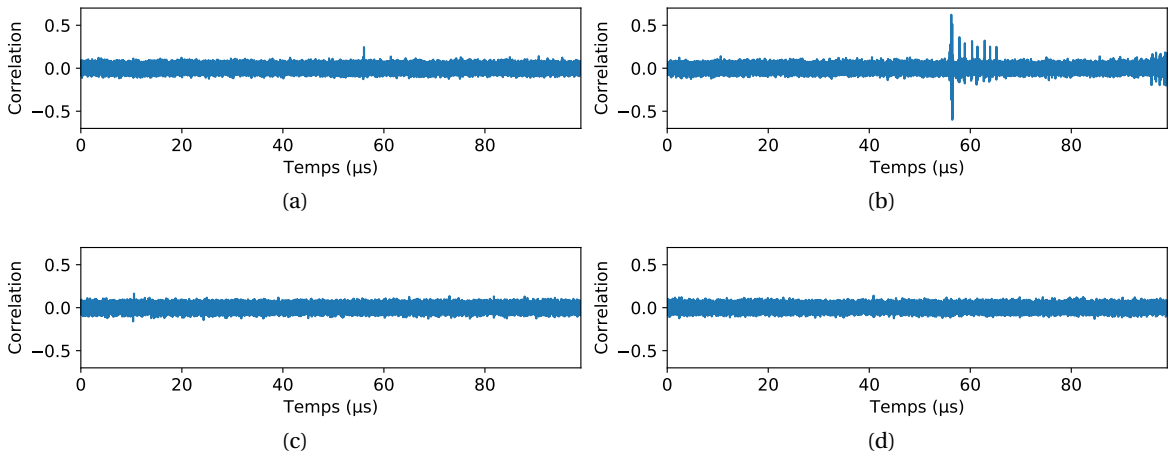


FIGURE 3.1 – Exemple d'évolution temporelle du coefficient de corrélation pour quatre hypothèses de clé (courtoisie du laboratoire LSOSP).

Étape 3 : Calcul de valeurs intermédiaires hypothétiques. L'attaquant émet K hypothèses correspondant à toutes les valeurs possibles pour la sous-clé attaquée k. À partir de ces K hypothèses, l'attaquant calcule les K valeurs intermédiaires possibles pour chaque texte en clair d grâce à la fonction $f(d,k)$. Chaque valeur intermédiaire hypothétique est donc associée à une sous-clé hypothétique.

Étape 4 : Calcul de valeurs de consommation hypothétiques. À partir de chaque valeur intermédiaire hypothétique, l'attaquant calcule une consommation hypothétique. Pour cela, il utilise un modèle de consommation représentant au mieux la consommation réelle de l'implémentation de l'algorithme attaqué. Deux modèles de consommation sont généralement utilisés : le poids et la distance de Hamming. La distance de Hamming correspond au nombre de commutations de "0" vers "1" et inversement, c'est-à-dire à l'activité du circuit, et le poids de Hamming d'une valeur intermédiaire correspond au nombre de bits valant "1" dans celle-ci. Au final, l'attaquant aura donc K consommations hypothétiques pour chaque texte en clair d, soit une matrice H de dimensions D×K de consommations hypothétiques dépendant à la fois des textes en clair connus et des sous-clés hypothétiques.

Étape 5 : Comparaison des consommations hypothétiques avec la consommation réelle. Au cours de cette étape, l'attaquant calcule la corrélation entre, d'une part, chacune des consommations hypothétiques, et d'autre part, la consommation réelle mesurée. Cela revient à calculer la corrélation entre chaque colonne des matrices H et M. On note ici $h_{n,i}$ l'élément de la n-ième ligne et de la i-ième colonne de H, c'est-à-dire la consommation hypothétique calculée pour la i-ième clé hypothétique parmi K et le n-ième texte en clair parmi D. De même, $m_{n,j}$ désigne le j-ième point de la n-ième trace de consommation, obtenue avec le n-ième texte en clair. En notant respectivement $\overline{h_i}$ et $\overline{m_j}$ les moyennes de la i-ième colonne de H et de la j-ième colonne de M, le coefficient de corrélation est calculé pour chaque clé hypothétique et à chaque instant par la formule suivante :

$$c_{i,j} = \frac{\sum_{n=1}^{\mathrm{D}} (h_{n,i} - \overline{h_i})(m_{n,j} - \overline{m_j})}{\sqrt{\sum_{n=1}^{\mathrm{D}} (h_{n,i} - \overline{h_i})^2 \sum_{n=1}^{\mathrm{D}} (m_{n,j} - \overline{m_j})^2}} \tag{3.1}$$

On obtient une matrice R de taille K×T, dont chaque ligne donne l'évolution temporelle de la corrélation entre une consommation hypothétique et la consommation réelle. À condition d'avoir un rapport signal sur bruit suffisamment élevé et un modèle de consommation adapté, le coefficient de corrélation maximal de cette matrice devrait correspondre à la bonne hypothèse de clé, et donner l'instant où la valeur intermédiaire ciblée est manipulée par l'algorithme. En traçant les corrélations pour les différentes hypothèses de clés en fonction du temps, on observe un pic de

corrélation pour la bonne clé au moment où la valeur intermédiaire est manipulée. Ce résultat est illustré sur la figure 3.1, qui montre la corrélation en fonction du temps pour quatre hypothèses de clés lors de l'exécution de l'AES sur un microcontrôleur STM32 [69]. Dans ce cas, la bonne hypothèse est la seconde, représentée sur la figure 3.1b, et l'octet de clé correspondant est manipulé par l'algorithme après 56 µs.

On notera qu'il est possible d'utiliser d'autres méthodes pour comparer les matrices M et H, n'utilisant pas le coefficient de corrélation. Parmi ces méthodes, on retrouve notamment la différence des moyennes, proposée par Kocher *et al.* en 1999 [68]. Celle-ci consiste à séparer les traces mesurées en deux lots, en fonction d'une consommation hypothétique calculée à partir d'une hypothèse de clé. La différence des consommations moyennes entre ces deux lots est alors calculée ; si celle-si dépasse une certaine valeur à un instant donné, cela signifie que les deux lots sont distinguables, et que l'hypothèse effectuée est correcte. Dans le cas contraire, une autre hypothèse est effectuée sur la clé afin de répartir différemment les traces dans ces deux lots, et le processus est répété jusqu'à ce que la clé soit obtenue. Cette méthode nécessite que le modèle de consommation soit binaire, c'est-à-dire que seules deux valeurs de consommations hypothétiques soient possibles. Un tel modèle de consommation décrit la consommation réelle moins précisément que le poids ou la distance de Hamming, qui peuvent prendre plusieurs valeurs distinctes ; cette méthode est donc mois efficace que l'utilisation du coefficient de corrélation.

3.2 Cas d'étude : fuites par canaux auxiliaires de l'algorithme Trivium

Dans cette section, nous analysons la présence de fuites d'information par canaux auxiliaires et leur exploitation pour une implémentation matérielle de l'algorithme de chiffrement par flot Trivium. En effet, celui-ci est représentatif des primitives cryptographiques légères pour l'IoT. Une description de Trivium est fournie dans la section 1.3.2.

Ces fuites sont analysées avec des simulations post-layout de la consommation de puissance, obtenues pour une implémentation de Trivium sur ASIC dans le nœud technologique 28 nm FD-SOI. Ces simulations permettent une étude totalement reproductible et plus précise que des mesures effectuées sur circuits réels. En particulier, elles s'affranchissent du bruit inhérent aux mesures sur circuits réels, et permettent d'étudier la consommation de seulement certaines parties d'un circuit tandis que les mesures ne permettent parfois que d'obtenir la consommation totale d'un circuit. Après avoir décrit le flot de conception et l'implémentation matérielle de Trivium, nous détaillons les simulations effectuées pour réaliser une CPA contre Trivium, ainsi qu'une méthodologie permettant d'évaluer l'ensemble des fuites d'information par canaux auxiliaires, que celles-ci soient exploitables ou non avec une CPA.

3.2.1 Implémentation matérielle de Trivium

Nous implémentons Trivium suivant un flot de conception standard, avec des paramètres de conception typiques choisis de sorte à refléter la conception d'un circuit intégré pour l'IoT dans le nœud 28 nm FDSOI, tel que le circuit WARRIOR développé par le CEA LETI et décrit dans la section 2.6. Cela permet d'évaluer la surface et la consommation, mais également les fuites d'information par canaux auxiliaires, dans des conditions standard, sans contraindre les outils afin d'optimiser l'un ou l'autre des paramètres.

Les étapes de synthèse et de placement-routage sont effectuées en considérant une tension de 0,9V, une température de 25 degrés, et des variations de processus typiques pour les transistors NMOS et PMOS. Une bibliothèque de cellules standard à faible tension de seuil, ou *Low Voltage Threshold (LVT)*, est utilisée. Cette bibliothèque est utilisée par défaut pour de nombreux circuits intégrés afin de réduire les temps de propagation aux dépens de la consommation statique. Enfin, la période d'horloge minimale visée est de 3 ns, ce qui correspond à une fréquence de 330 MHz. Cette fréquence maximale est suffisante pour la plupart des application dans l'IoT.

Comme nous l'avons expliqué dans la section 1.2.1, la surface est obtenue à l'issue de l'étape

de synthèse. Notre implémentation de Trivium a une surface de 2808 GE (*Gate Equivalent*), soit 1375 μm^2.

Les simulations post-layout pour l'évaluation de la consommation de puissance sont effectuées selon la méthodologie décrite dans la section 1.2.1. Une fréquence de 100 MHz est choisie pour ces simulations ; celle-ci est inférieure à la fréquence maximale autorisée mais est suffisante pour la génération de quelques dizaines voire quelques centaines de bits de flot de clé. On rappelle ici que l'état interne de Trivium contient 288 bits, et que lors du chargement de la clé et de l'IV dans celui-ci, au moins 125 bits consécutifs de cet état interne valent "0". Au fur et à mesure de la phase d'initialisation, l'état interne est progressivement mélangé, et la consommation de Trivium augmente donc pendant quelques dizaines de cycles d'horloge, avant de se stabiliser autour d'une valeur moyenne. Afin d'obtenir une valeur moyenne de consommation représentative du fonctionnement global de Trivium, les simulations sont effectuées après cette phase transitoire de montée de la consommation. Dix simulations avec des clés et des IV différents sont effectuées, sur une durée de 100 cycles d'horloge chacune. La consommation moyenne de puissance de Trivium sur ces 10 simulations est de 241 µW.

3.2.2 Attaques par canaux auxiliaires contre Trivium

Dans cette section, nous détaillons les attaques par canaux auxiliaires existantes contre Trivium, puis nous présentons notre méthodologie pour l'étude du coefficient de corrélation et la mise en place d'une CPA contre l'implémentation de Trivium décrite précédemment.

Attaques publiées contre Trivium

Plusieurs attaques par analyse des canaux auxiliaires ont été publiées contre l'algorithme Trivium [70, 71, 72, 73, 74]. Pour rappel, Trivium est constitué de 288 bascules, 10 portes XOR, et 3 portes ET. La consommation dynamique des bascules étant généralement bien supérieure à celle de la logique combinatoire, les modèles d'attaques considèrent que cette dernière est négligeable dans la consommation globale de Trivium. La consommation des bascules provient principalement de leurs commutations, c'est-à-dire des changements d'état de "0" vers "1" ou inversement, le maintien d'un état à "0" ou "1" ayant une consommation négligeable. Le modèle de consommation en distance de Hamming est donc utilisé dans tous les travaux existants afin de décrire la consommation de Trivium. C'est le modèle généralement utilisé pour attaquer l'implémentation des algorithmes de chiffrement par flot basés sur des registres à décalage [75].

Afin d'effectuer une CPA contre un algorithme de chiffrement par flot, la donnée d différente pour chaque trace et connue de l'attaquant est l'IV, et non le texte en clair comme dans le cas du chiffrement par bloc. En effet, le texte en clair est additionné au flot de clé généré par l'algorithme, et il n'est pas traité directement par cet algorithme.

Toutes les attaques par canaux auxiliaires connues contre Trivium sont effectuées au début de la phase d'initialisation de l'algorithme, et ce, pour diverses raisons. Lors du chargement de la clé et de l'IV dans Trivium, l'ensemble de l'état interne de l'algorithme est connu de l'attaquant, hormis la clé qui occupe 80 bits consécutifs des 288 bits de cet état interne. La consommation de Trivium à ce moment-là dépend donc directement de la clé secrète et de l'IV connu, ce qui permet de réaliser des attaques avec succès avec seulement 550 traces mesurées sur ASIC dans le nœud technologique 0,18 µm [71], et 1200 traces en simulations dans le nœud technologique 90 nm [73]. L'attaque lors du début de l'initialisation permet de cibler certains bits de la clé uniquement, et d'appliquer ainsi l'approche de « diviser pour mieux régner » permettant de garder une complexité faible. Selon une étude menée en 2008 [75] sur la vulnérabilité aux attaques par analyse des canaux auxiliaires des algorithmes présentés à la compétition eStream, dont Trivium fait partie, une attaque après cette phase d'initialisation serait plus complexe à mettre en œuvre, du fait de la diffusion de bits de clé dans les 288 bits de l'état interne. Enfin, effectuer les attaques lors du début de l'initialisation permet de limiter le temps nécessaire à l'acquisition des traces ; si une attaque était effectuée après cette phase d'initialisation, il serait nécessaire d'attendre du-

TABLEAU 3.1 – Équations pour σ durant les 81 premiers cycles d'horloge.

Cycle i	**Équation pour σ**
1 à 12	$\sigma_i = k_{67-i}$
13	$\sigma_{13} = k_{54} + k_{79}k_{80}$
14 à 66	$\sigma_i = k_{67-i} + (k_{92-i}k_{93-i}) + k_{94-i}$
67	$\sigma_{67} = k_{69} + (k_{25}k_{26}) + k_{27}$
68 à 69	$\sigma_i = \overline{k}_{136-i} + (k_{92-i}k_{93-i}) + k_{94-i}$
70 à 81	$\sigma_i = k_{136-i} + (k_{92-i}k_{93-i}) + k_{94-i}$

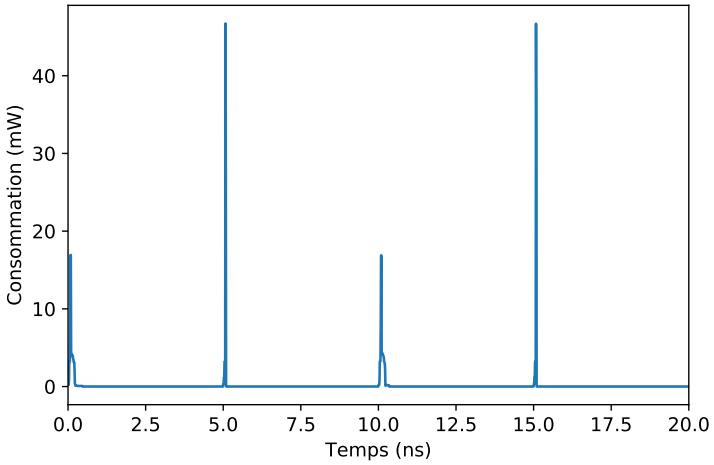


FIGURE 3.2 – Exemple de trace de consommation pour le Trivium non protégé durant deux cycles d'horloge, avec un point toutes les 10 ps, à une fréquence de 100 MHz.

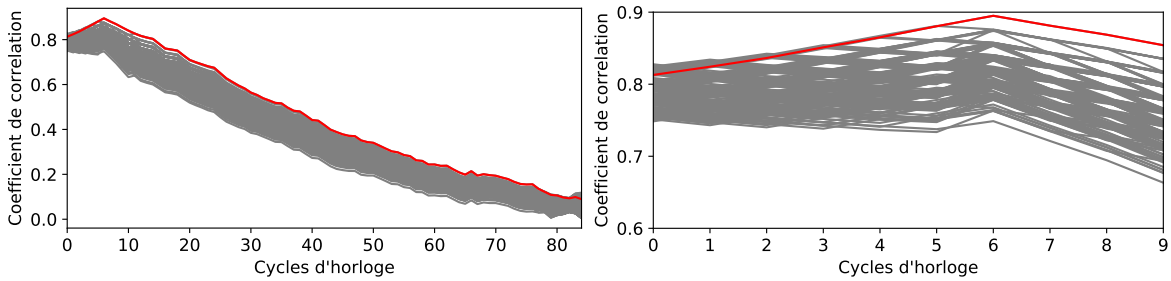
rant les 1152 cycles d'horloge de cette phase avant la mesure de chaque trace, ce qui rallongerait considérablement le temps d'acquisition.

Les attaques CPA contre Trivium visent la valeur intermédiaire S_{94}, c'est-à-dire le $94^{ème}$ bit de l'état interne de Trivium qui est également le premier bit du second registre à décalage constituant Trivium. Ce bit est mis à jour à chaque cycle selon la formule : $S_{94} = S_{66} + S_{93} + S_{171} + S_{91}S_{92}$. Lors de l'initialisation, on peut transformer cette formule pour obtenir $S_{94} = \sigma + f(IV)$, f étant une fonction dépendant uniquement de certains bits d'IV et σ une valeur dépendant uniquement de certains bits de clé. Il suffit alors à un attaquant d'obtenir au minimum 80 valeurs de σ différentes, à 80 cycles d'horloge différents, et de résoudre le système de 80 équations obtenues afin de remonter aux 80 bits de clé. À titre d'exemple, le tableau 3.1 montre les 81 valeurs de σ permettant de retrouver la clé d'après [74], notées σ_i. On notera que la $13^{ème}$ équation n'est pas indépendante des autres, et n'est donc pas nécessaire pour résoudre ce système.

Chaque σ_i est obtenu itérativement en effectuant une CPA sur le i-ième cycle d'horloge, les valeurs de σ_j pour $j < i$ obtenues lors des cycles précédents étant utilisées pour le calcul de la consommation hypothétique de ce i-ième cycle d'horloge. Comme cette CPA itérative est effectuée sur un bit à la fois, seules deux consommations hypothétiques sont calculées en fonction des deux valeurs possibles pour σ_i. La consommation hypothétique présentant la corrélation la plus élevée avec la consommation mesurée lors du i-ième cycle d'horloge est celle correspondant à la bonne valeur intermédiaire σ_i.

Mise en œuvre

La CPA décrite précédemment est mise en œuvre sur les simulations post-layout de la consommation de puissance de l'implémentation de Trivium dans le nœud technologique 28 nm FDSOI. Les traces de consommation utilisées par la suite sont générées avec un intervalle de 10 ps entre deux points, c'est-à-dire entre deux valeurs simulées de la consommation. Cet intervalle de 10 ps permet une étude équivalente, voire plus fine, que ce qui est réellement observable avec un oscil-



(a) Corrélation durant 85 cycles d'horloge

(b) Agrandissement, 10 premiers cycles d'horloge

FIGURE 3.3 – Évolution du coefficient de corrélation pour une implémentation matérielle de Trivium.

loscope moderne ; par exemple, le Rohde & Schwarz RTO2014 [76] a une fréquence d'échantillonnage maximale de 10 GSa/s (soit 10 milliards de valeurs mesurées par seconde), ce qui correspond à un intervalle de 100 ps entre deux points. Une trace de consommation de deux cycles d'horloge est représentée sur la figure 3.2 pour l'exécution de Trivium. On constate qu'il y a deux pics par cycles d'horloge, correspondants aux fronts montants et descendants de l'horloge sur lesquels est concentrée l'essentiel de l'activité des portes logiques du circuit. En dehors de ces pics, seule la consommation statique est observable ; celle-ci est négligeable devant la consommation dynamique et son étude sort du cadre d'une CPA classique. Afin d'extraire uniquement l'information utile pour une CPA, chaque trace de consommation est donc compressée en conservant uniquement la valeur de consommation maximale de chaque cycle d'horloge.

100.000 traces de consommation sont simulées avec des IV différents pour chaque trace, générés aléatoirement avec un script en Python. La clé de chiffrement est identique pour toutes les traces et est choisie aléatoirement. Toutes les traces de consommation sont simulées sur les 85 premiers cycles d'horloge de l'exécution de Trivium. Cette durée est suffisante pour étudier la possibilité d'effectuer une CPA, les attaques existantes nécessitant entre 76 et 81 cycles d'horloge [70, 71, 72, 73, 74]. Les valeurs intermédiaires ciblées sont les σ_i décrits précédemment, et le modèle de consommation est le modèle en distance de Hamming.

Dans un premier temps, le coefficient de corrélation est étudié pour les 8 premiers σ_i, ce qui correspond à un octet de la clé. Il y a 256 hypothèses possibles pour cet octet de clé. Pour chaque hypothèse, la corrélation entre la consommation hypothétique correspondante et la consommation réelle est calculée. La figure 3.3 montre l'évolution du coefficient de corrélation pour ces 256 hypothèses ; la corrélation pour la bonne hypothèse de clé est affichée en rouge. On observe un pic de corrélation au cours des 8 premiers cycles d'horloge, ce qui correspond au moment où les 8 bits de valeurs intermédiaires sont manipulés. Les courbes pour les différentes hypothèses sont proches les unes des autres, car seuls quelques bits sur les 288 de l'état interne diffèrent entre ces hypothèses, mais la courbe de la bonne hypothèse est tout de même distinguable, avec une valeur maximale supérieure à la corrélation pour les autres hypothèses et égale à 0,8950. Cette corrélation élevée, proche de 1, permet de valider que le modèle de consommation en distance de Hamming reflète avec précision la consommation obtenue avec les simulations post-layout.

Dans un second temps, une CPA itérative permettant de retrouver bit par bit les 81 valeurs de σ, telle que décrite dans la section 3.1, est effectuée. La clé de chiffrement de 80 bits est retrouvée avec cette CPA en utilisant seulement 100 traces de consommation simulées.

3.2.3 Évaluation de l'ensemble des fuites d'information par canaux auxiliaires

Nous avons décrit plus haut l'étude de la corrélation entre une consommation hypothétique et la consommation réelle (ou simulée), pour certaines valeurs intermédiaires données, afin d'effectuer une attaque par canaux auxiliaires. Cette approche permet uniquement d'évaluer la vulnérabilité d'un algorithme par rapport à cette attaque en particulier. Elle repose sur le choix judicieux de certaines valeurs intermédiaires qui dépendent à la fois de la clé et de valeurs connues d'un

attaquant, et sur l'utilisation d'un modèle de consommation qui représente au mieux la consommation réelle du circuit. Elle ne permet pas d'évaluer l'ensemble des fuites d'information potentielles, éventuellement exploitables avec d'autres types d'attaques qui utiliseraient d'autres valeurs intermédiaires ou un autre modèle de consommation. Nous complétons donc cette étude par une évaluation des fuites d'information par canaux auxiliaires grâce au T-test de Welch.

Principe

Le T-test de Welch permet d'évaluer l'ensemble des fuites d'information potentielles par canaux auxiliaires, que ces fuites soient ensuite exploitables en pratique pour effectuer une CPA ou non [77, 78]. Le principe de ce test est de déterminer si deux lots de traces sont distinguables l'un de l'autre, grâce à leur moyenne et leur variance. Pour cela, ce test évalue la probabilité que la moyenne de ces deux lots soit différente. Si les deux lots, obtenus avec des paramètres différents (clé et/ou IV), sont distinguables, cela signifie qu'il y a une fuite d'information. On appelle hypothèse nulle le cas où les deux lots sont indistinguables avec ce test.

On considère deux lots de traces X_0 et X_1, les traces dans chaque lot ayant toutes la même durée, c'est-à-dire le même nombre de points. En notant μ_0 et μ_1 les moyennes de chaque lot, s_0 et s_1 leurs variances et n_0 et n_1 le nombre d'éléments dans chaque lot, la statistique de test t est obtenue avec la formule :

$$t = \frac{\mu_0 - \mu_1}{\sqrt{\frac{s_0}{n_0} + \frac{s_1}{n_1}}} \tag{3.2}$$

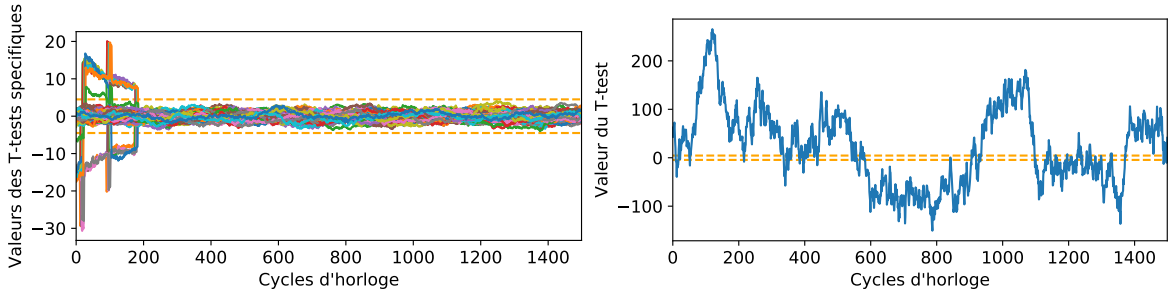
Cette statistique de test t contient autant de points que les traces utilisées pour le calcul. Si elle dépasse un certain seuil à un instant donné, cela signifie que l'hypothèse nulle est rejetée, c'est-à-dire que les traces sont issues de deux populations différentes, et que les lots sont distinguables à cet instant. Une valeur de seuil de $\pm 4{,}5$ est généralement choisie, celle-ci correspondant à une probabilité de rejeter l'hypothèse nulle supérieure à 0,99999 selon [77, 78]. Par la suite, cette valeur de seuil sera représentée par des lignes oranges en pointillés sur toutes les figures montrant des résultats de T-tests.

Les deux lots peuvent être obtenus de différentes manières. On distingue ainsi les T-tests *spécifiques* des T-tests *non-spécifiques*. Dans un T-test spécifique, les deux lots sont constitués en fonction de certaines valeurs intermédiaires spécifiques ciblées par une attaque CPA donnée. Ce test permet d'évaluer la vulnérabilité d'un circuit à cette attaque, sans qu'il soit nécessaire d'effectuer l'attaque et de recouvrer la clé. Par exemple, dans le cas de Trivium, un évaluateur peut fixer un IV et faire varier la clé de manière à séparer les deux lots selon les valeurs de σ utilisées pour la CPA. Un T-test non-spécifique permet d'évaluer les fuites d'information sans se baser sur de telles hypothèses, et couvre donc l'ensemble des fuites possibles. Ce test, également appelé T-test fixe-vs-aléatoire, consiste à utiliser des paramètres fixes dans l'un des deux lots de traces, et aléatoires dans l'autre. Dans le cas de Trivium, l'un des deux lots contient uniquement des traces générées avec un même IV, tandis que l'autre lot contient des traces initialisées avec des IV différents. La clé est fixée et identique pour les deux lots.

Application à Trivium

Nous étudions les fuites d'information par canaux auxiliaires de Trivium durant les 1500 premiers cycles d'horloge de l'exécution de l'algorithme. Cela correspond aux 1152 cycles de l'initialisation, suivis des 348 premiers cycles du chiffrement, au cours desquels 348 bits de flot de clé sont générés. Des T-tests spécifiques et non-spécifiques sont effectués, afin d'étudier à la fois les fuites d'information par canaux auxiliaires de manière générale, et les fuites exploitables avec les attaques publiées à l'heure actuelle.

Pour les T-tests spécifiques, 20.000 traces de consommation sont simulées avec un IV fixé pour toutes les traces et une clé générée aléatoirement pour chaque trace. Au total, 81 T-tests sont effectués avec ces traces, pour les 81 valeurs de σ_i ciblées par une CPA. Les deux lots de traces pour chacun de ces 81 T-tests sont constitués en séparant les traces selon la valeur correspondante de



(a) T-tests spécifiques

(b) T-test non-spécifique

FIGURE 3.4 – T-tests spécifiques et T-test non-spécifique durant les 1500 premiers cycles d'exécution de Trivium, en simulations post-layout.

σ_i. Il y a donc environ 10.000 traces par lot pour chaque test. Ces T-tests sont représentés sur la figure 3.4a, et montrent que les attaques effectuées sur les σ_i, dépendant chacune de seulement un à quatre bits de clé, ne peuvent pas être effectuées après environ 200 cycles d'horloge. Cela permet de constater l'efficacité du mélange de l'état interne de Trivium au cours de la phase d'initialisation.

Pour le T-test non-spécifique, deux lots de 10.000 traces chacun sont simulés, l'un avec une clé et un IV fixe, et l'autre avec la même clé fixe et des IV aléatoires. Ce T-test non-spécifique, dont les résultats sont représentés sur la figure 3.4b, montre des fuites d'information importantes durant toute la durée de l'initialisation, mais également lors du chiffrement. Bien que des attaques CPA aient seulement été publiées à ce jour contre la phase d'initialisation de Trivium, ces résultats montrent qu'en théorie, des attaques pourraient également être effectuées contre la phase de chiffrement en exploitant ces fuites d'information.

3.2.4 Méthodologie : corrélation ou T-test ?

Le T-test non-spécifique permet d'évaluer la présence ou non de potentielles fuites d'information par canaux auxiliaires, et n'est pas restreint à un modèle de consommation donné ou à des valeurs intermédiaires spécifiques. À l'inverse, une CPA est limitée à la détection de fuites d'information selon un unique modèle de fuites, prenant en compte certaines valeurs intermédiaires et un modèle de consommation. En ce qui concerne l'évaluation des fuites d'information, le T-test est donc plus efficace que l'étude du coefficient de corrélation.

Cependant, si le T-test non-spécifique permet d'observer la présence de fuites d'information, il ne permet pas de conclure sur la possibilité d'exploiter ces fuites en pratique avec une CPA donnée. En effet, le fait de pouvoir distinguer deux lots de traces avec le T-test n'est pas forcément une information suffisante pour retrouver la clé de chiffrement utilisée. Afin d'effectuer une CPA avec succès sur les fuites observables au T-test, un attaquant doit être en mesure de déterminer des valeurs intermédiaires pertinentes et un modèle de consommation adapté. Cela peut nécessiter un travail important, voire être irréalisable avec les moyens de calcul actuels. Par exemple, bien que des fuites soient visibles au T-test non-spécifique après la phase d'initialisation de Trivium, il n'existe à l'heure actuelle aucune attaque permettant d'exploiter ces fuites pour retrouver la clé de chiffrement.

Par conséquent, l'observation de potentielles fuites d'information avec un T-test non-spécifique et l'étude du coefficient de corrélation dans le but de mettre en place une attaque CPA sont deux approches complémentaires et nécessaires afin d'évaluer la vulnérabilité d'un circuit intégré aux attaques par analyse des canaux auxiliaires. Dans les chapitres suivants, nous évaluons tout d'abord la présence de fuites par canaux auxiliaires avec un T-test non-spécifique, puis nous estimons la possibilité d'exploiter ces fuites avec les attaques publiées à l'heure actuelle à travers l'étude des variations du coefficient de corrélation.

3.3 Attaques par injection de fautes

Les attaques par injection de fautes sont un autre type d'attaques matérielles. Comme leur nom l'indique, ces attaques consistent à injecter des fautes dans un circuit intégré afin de perturber son fonctionnement. Ces fautes peuvent être injectées puis exploitées de différentes manières.

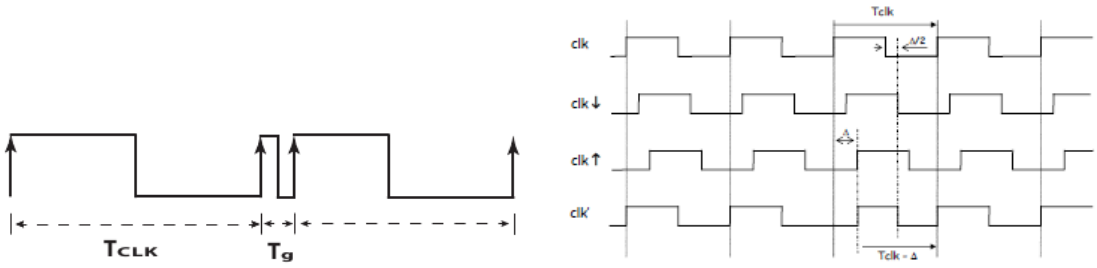
3.3.1 Moyens d'injection

Un circuit intégré numérique est constitué de blocs logiques combinatoires et séquentiels. Nous avons expliqué dans le chapitre 1 que l'information met un certain temps à se propager au sein des blocs combinatoires, en fonction de la longueur des différents chemins de données. Le chemin de données présentant le temps de propagation le plus long est appelé chemin critique, et ce temps de propagation conditionne la fréquence maximale à laquelle le circuit peut fonctionner sans erreurs. Un attaquant capable de modifier la fréquence de fonctionnement du circuit pourrait augmenter cette fréquence au-delà de la fréquence maximale, et créer l'apparition de fautes dans le circuit intégré [79]. En général, lors d'une telle attaque, des fautes apparaissent non seulement dans le chemin critique, mais également dans tous les chemins de données dont les contraintes de temps de propagation sont violées, et le nombre de blocs combinatoires touchés augmente donc avec l'augmentation de la fréquence. Cette attaque est classée parmi les attaques à bas coût par Barenghi *et al.* [40], c'est-à-dire celles qui peuvent s'effectuer avec un matériel dont le coût total est inférieur à 3000 $. La fréquence peut généralement être modifiée par un attaquant ayant un accès physique à l'horloge externe d'un circuit intégré. Dans certains cas, cette attaque peut être réalisée à distance, en exploitant le logiciel embarqué exécuté sur le circuit intégré, si ce logiciel permet de contrôler la fréquence de fonctionnement de ce circuit [51].

L'attaque précédente crée des fautes dans le circuit intégré tant que sa fréquence de fonctionnement est supérieure à la fréquence maximale autorisée. Afin d'augmenter la précision temporelle de cette attaque, il est possible de viser un seul cycle d'horloge. Deux approches permettent d'effectuer cette attaque, appelée *glitch d'horloge* [6, 41], et sont représentées sur la figure 3.3.1. La première consiste à réduire la durée d'un cycle d'horloge à un temps T_g plus court que le temps de propagation minimal, et la seconde revient à retarder par un délai Δ un front montant de l'horloge sans changer la durée totale de ce cycle d'horloge.

Le temps de propagation au sein des blocs combinatoires dépend de plusieurs facteurs, et est notamment conditionné par la tension d'alimentation et la température du circuit intégré. Le fait de réduire la tension d'alimentation ou d'augmenter la température résulte, dans les deux cas, en une augmentation du temps de propagation. Un attaquant ayant accès au circuit intégré peut donc choisir de mettre en œuvre l'un de ces deux mécanismes afin de violer les contraintes de temps de propagation au sein de ce circuit, sans modifier la fréquence de l'horloge. Ces attaques nécessitent un matériel peu coûteux et peu de connaissances sur le circuit attaqué [40]. En particulier, il est possible de créer des *glitchs de tension* dans un circuit intégré, c'est-à-dire des variations transitoires de cette tension, afin de contrôler le moment de l'injection de fautes [80]. De même que pour les attaques visant l'horloge d'un circuit intégré, il est possible, dans certains cas, de contrôler la tension d'alimentation grâce au logiciel embarqué, et donc d'effectuer une telle attaque à distance [51].

L'exposition d'un circuit intégré à une source lumineuse peut également permettre d'injecter des fautes dans celui-ci. Si l'énergie transmise par cette source lumineuse est suffisante, on assiste à la création de paires électrons-trous dans les transistors du circuit intégré par effet photoélectrique, ce qui génère un courant dans ces transistors [6]. Ce courant entraîne l'apparition d'un bref pic de tension, qui dure quelques centaines de picosecondes. Un tel pic de tension dans la logique séquentielle peut résulter en une modification des valeurs stockées. S'il est injecté dans un bloc combinatoire, il peut également induire l'apparition d'une faute, à condition de se propager au sein de ce bloc et d'être échantillonné en sortie de celui-ci par de la logique séquentielle. Bien qu'il soit possible d'utiliser diverses sources lumineuses telles qu'une lampe à ultraviolets ou le flash d'un appareil photo [6, 40], l'utilisation d'un laser permet une meilleure précision spatiale et



(a) Réduction de la durée d'un cycle d'horloge [41]

(b) Insertion d'un délai avant le front montant de l'horloge [6]

FIGURE 3.5 – Injection de glitchs d'horloge dans un circuit intégré.

temporelle, au prix d'un coût plus élevé [40].

Il est possible d'injecter des fautes dans un circuit intégré en exposant celui-ci à un rayonnement électromagnétique. Ce rayonnement électromagnétique crée un courant de Foucault dans le circuit intégré, ce courant créant lui-même un pic de tension qui peut induire des fautes de la même manière que le pic de tension causé par l'exposition d'un circuit à une source lumineuse. Les impulsions électromagnétiques peuvent être crées par une sonde électromagnétique à proximité du circuit intégré, afin de contrôler le moment précis, et, dans une certaine mesure, la localisation spatiale de la faute. Ces attaques sont peu coûteuses, selon Barenghi *et al.* [40].

Enfin, d'autres attaques avancées nécessitent un matériel extrêmement coûteux afin d'altérer le circuit intégré de manière permanente ou semi-permanente, avec une très grande précision. Ces attaques nécessitent une expertise importante, et une très bonne connaissance du layout du circuit attaqué. Par exemple, un faisceau d'ions focalisés (FIB, *Focused Ion Beam*) peut permettre de modifier la structure d'un circuit en coupant ou en ajoutant des fils entre divers blocs logiques, avec une précision de l'ordre de 2,5 nm selon [40]. Un faisceau de rayons X focalisés grâce à l'utilisation d'un synchrotron peut également permettre de viser un transistor précis, et de modifier l'état de ce transistor ou son implémentation physique [81].

3.3.2 Types de fautes obtenues

Selon la méthode choisie pour l'injection de fautes parmi celles décrites précédemment, ces fautes peuvent prendre des formes variées, correspondant à divers modèles de fautes. Ainsi, la taille des fautes peut varier selon le moyen d'injection : une faute peut être injectée sur un seul bit, ou sur plusieurs bits. Par exemple, en utilisant un laser avec un faisceau suffisamment fin, il est possible d'obtenir des fautes locales, sur un seul bit ou sur un faible nombre de bits traités ou stockés dans la même zone d'un circuit [6, 40]. Au contraire, la violation des contraintes de temps de propagation via une perturbation de la tension d'alimentation, de l'horloge ou de la température aura plutôt tendance à créer des fautes globales, qui affectent l'ensemble du circuit. Une telle violation des contraintes de temps de propagation peut également permettre d'obtenir une faute locale, sur un seul bit, en visant le chemin critique du circuit intégré et en augmentant la fréquence suffisamment pour qu'une faute se produise sur ce chemin critique, mais pas sur les autres chemins de données [79].

La durée des effets d'une injection de fautes sur un circuit dépend également de la méthode d'injection utilisée. Certaines fautes sont *permanentes*, car elles sont dues à une modification partielle du circuit intégré lui-même, grâce à une méthode dite *destructive*. Ces fautes ne sont pas corrigées lors de la ré-initialisation du circuit. Selon [40], de telles fautes permanentes peuvent par exemple être injectées en utilisant un laser avec une forte puissance ou un faisceau d'ions focalisés (FIB, *focused ion beam*). Une faute permanente injectée sur une bascule affectera par la suite toutes les valeurs stockées successivement dans cette bascule, de même qu'une telle faute sur une porte logique modifiera les valeurs de sortie de cette porte logique. Les fautes *éphémères*, au contraire, ne persistent pas après la ré-initialisation d'un circuit ou le chargement de nouvelles

valeurs dans celui-ci. La précision temporelle de l'injection de ces fautes éphémères varie selon les moyens d'injection utilisés. Dans nos travaux, nous définissons les fautes *temporellement précises* comme les fautes visant un cycle d'horloge en particulier, et dont la durée de l'injection ne dépasse pas ce cycle d'horloge. Par opposition, les fautes *temporellement imprécises* sont injectées durant plusieurs cycles d'horloge. Par exemple, nous considérons que les lasers et les glitchs d'horloge permettent d'injecter des fautes temporellement précises, tandis que la modification de la température de fonctionnement crée des fautes temporellement imprécises.

Enfin, certaines fautes inversent la valeur d'un bit (ou d'un groupe de bits) et seront notées par la suite fautes *en commutation*, tandis que d'autres fautes consistent à fixer la valeur d'un bit (ou groupe de bits) à "0" ou à "1", indépendamment de sa valeur initiale. Ces dernières, appelées par la suite *mise à "0"* et *mise à "1"*, produisent donc un effet visible uniquement lorsque la valeur initiale non fautée est différente de la valeur fixée par la faute : une faute de mise à "1" n'aura aucun effet sur un bit valant déjà "1".

3.3.3 Exploitation des fautes

Une fois injectées, les fautes peuvent être exploitées de différentes manières. Certaines fautes visent à modifier le comportement du circuit intégré, par exemple en empêchant l'exécution de certaines instructions sur un processeur, ou en modifiant les instructions exécutées. Il est également possible de modifier le fonctionnement global d'un algorithme de chiffrement, que celui-ci soit implémenté en matériel ou en logiciel, afin de réduire sa complexité et d'effectuer ensuite une cryptanalyse classique. Par exemple, l'algorithme AES repose sur une répétition un certain nombre de fois des mêmes transformations, chaque transformation étant effectuée sur le résultat de la transformation précédente. Ces transformations sont appelées *tour*, et un certain nombre de tours est nécessaire afin de mélanger suffisamment le texte et la clé et empêcher ainsi une cryptanalyse visant à obtenir cette clé. Le nombre de tours effectués est comptabilisé par un compteur, qui est incrémenté à chaque tour. Une injection de fautes visant ce compteur peut permettre de réduire le nombre de tours, et d'effectuer ainsi une cryptanalyse sur le résultat chiffré obtenu [82].

D'autres attaques, appelées *safe-error* [83], exploitent le simple fait d'observer si le résultat d'un algorithme est modifié ou non après une injection de fautes. En effet, le succès de l'injection de fautes peut dépendre de la valeur de la clé secrète de l'algorithme, et donc révéler des informations sur cette clé. Un exemple simple est une injection de fautes fixant à "0" certains bits de clé : si le résultat du calcul après cette injection de fautes n'est pas modifié, cela signifie que ces bits valaient déjà "0".

Enfin, une dernière catégorie d'attaques est l'analyse différentielle en fautes, ou DFA pour *Differential Fault Analysis*. Ce type d'attaques, introduit en 1997 [84], nécessite qu'un attaquant soit en possession du résultat chiffré non fauté d'un algorithme de chiffrement, et d'un ou plusieurs résultats fautés obtenus avec la même clé de chiffrement. Avec la connaissance du modèle de fautes et des caractéristiques de l'algorithme visé, la cryptanalyse des différents résultats permet de déterminer un certain nombre de clés possibles ayant permis de générer ceux-ci. Cela réduit donc l'espace de recherche afin de retrouver la clé ayant servi au chiffrement.

Toutes les attaques décrites ici sont adaptées au cas par cas pour chaque algorithme, et s'appuient selon les cas sur l'un des différents modèles de fautes évoqués précédemment. La précision temporelle et spatiale requise est propre à chaque attaque, allant de l'injection maîtrisée à un endroit et un instant connus, à des injections aléatoires à la fois en temps et en localisation.

À titre d'exemple, plusieurs études ont été menées pour la mise en œuvre de DFA sur Trivium [85, 86, 87]. Un étude théorique de la possibilité de réaliser ces attaques a été menée dans [85, 86]. Hojsik *et al.* [85] considèrent un modèle de fautes sur un seul bit injecté dans un des trois registres de Trivium. Hu *et al.* [86] étendent cette attaque en proposant un modèle de fautes moins restrictif, avec des fautes sur un à huit bits consécutifs dans l'un des registres. Selon cette dernière étude, une attaque en fautes effectuée au cours des 32 premiers cycles d'horloge de l'initialisation de Trivium nécessite en moyenne moins de 16 injections de fautes avant que l'attaquant ne soit en mesure de retrouver la clé secrète. Selon les auteurs, le nombre d'injections à effectuer augmente

avec le nombre de cycles d'horloge écoulés avant ces injections, et il est donc nécessaire d'effectuer celles-ci lors du début de la phase d'initialisation de Trivium. Potestad-Ordóñez *et al.* [87] ont démontré expérimentalement sur FPGA la possibilité de l'injection de fautes au sein des registres de Trivium en utilisant des glitchs d'horloge. Cependant, la reproductibilité de cette injection pour deux implémentations de Trivium au sein d'un même FPGA comme entre deux FPGA différents reste à améliorer.

Remarque : Dans la suite de nos travaux, nous n'évaluons pas la vulnérabilité de Trivium à des attaques en fautes à partir de simulations. En effet, une telle étude serait totalement dépendante de la sélection arbitraire d'un modèle de fautes parmi ceux présentés dans la section 3.3.2. Le choix d'un modèle de fautes identique à ceux des attaques théoriques de Hojsik *et al.* [85] ou de Hu *et al.* [86] permettrait d'obtenir les fautes souhaitées afin de réussir ces attaques, tandis que le choix d'un modèle différent empêcherait l'exploitation des fautes obtenues. Dans les deux cas, les résultats ne reflèteraient pas nécessairement une attaque réelle mais uniquement le modèle choisi.

3.4 Protections contre les attaques par canaux auxiliaires

Nous détaillons ici les protections existantes contre les attaques matérielles décrites précédemment. Les contre-mesures contre les attaques par canaux auxiliaires visent soit à réduire le rapport signal sur bruit, noté SNR pour *Signal to Noise Ratio*, soit à agir directement sur le signal mesuré afin de le rendre inexploitable. D'un côté, la réduction du SNR peut se faire de différentes manières, par exemple en augmentant le bruit, ou en réduisant l'amplitude du signal lié aux données secrètes et exploitable lors d'une attaque. D'un autre côté, il est possible d'ajouter de l'aléa aux valeurs intermédiaires, afin de décorréler les mesures effectuées des données secrètes. Les différentes catégories de contre-mesures sont détaillées dans les paragraphes suivants.

Il existe également des protections au niveau système, comme la limitation du nombre de chiffrements par minute, ou un changement fréquent des clés de chiffrement [88]. Le principe de ces protections est que de nombreuses traces d'exécution sont généralement nécessaires pour récupérer une donnée secrète avec une attaque CPA, et qu'il suffit donc d'empêcher l'acquisition de ces traces afin d'empêcher l'attaque. Cependant, ces protections au niveau système présentent des désavantages importants du point de vue fonctionnel : la vitesse d'exécution est réduite, et le coût énergétique du changement fréquent des clés est généralement important. Une mise à jour fréquente des clés nécessite une bonne synchronisation de tous les acteurs impliqués dans la communication, afin que toutes les clés soient changées simultanément au sein d'un réseau, ce qui est complexe et coûteux à mettre en œuvre. De plus, certaines attaques plus élaborées requièrent très peu de traces [43], et ces protections sont donc inefficaces contre ce type d'attaques. Pour ces raisons, ces protections ne seront pas décrites plus en détails par la suite.

3.4.1 Réduction du rapport signal sur bruit

Génération algorithmique de bruit

La génération de bruit en amplitude est une méthode intuitive permettant de réduire le SNR. De nombreux auteurs proposent le rajout de modules supplémentaires dont l'unique fonction est de générer du bruit. Par exemple, Robisson *et al.* [89] proposent de générer du bruit avec un générateur de nombres aléatoires, Liu *et al.* [90] grâce à des oscillateurs en anneau, et Shan *et al.* [91] en utilisant des blocs fonctionnels identiques à ceux utilisés pour le chiffrement. Cependant, le rajout de bruit aléatoire, ayant une distribution gaussienne, ne permet de contrer efficacement les attaques telles que la CPA. En effet, ces attaques sont basées sur des statistiques sur un certain nombre de traces d'exécution, et elles filtrent donc ce bruit à condition d'avoir un nombre suffisant de traces. C'est pourquoi plusieurs travaux portent sur la génération de *bruit corrélé*, c'est-à-dire de bruit dépendant des données chiffrées. Par exemple, les auteurs de [92] proposent d'effectuer une partie du chiffrement avec une fausse clé en parallèle à celui effectué avec la clé

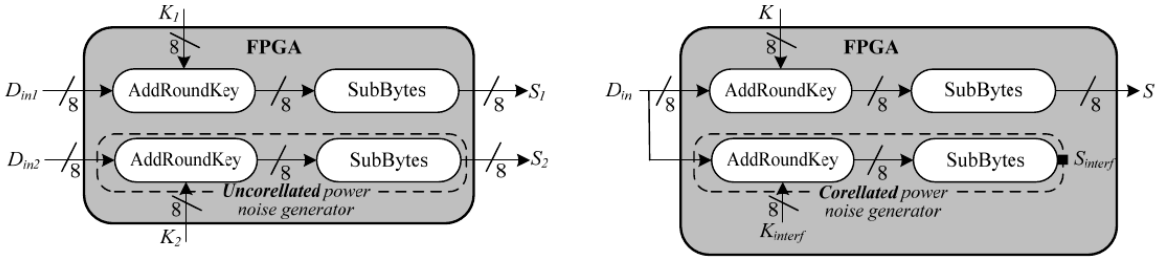


FIGURE 3.6 – Génération de *bruit non corrélé* (à gauche), et de *bruit corrélé* (à droite) [92].

réelle, sur le même texte d'entrée. La figure 3.6, extraite de [92], montre la génération de bruit non corrélé et corrélé pour une partie de l'exécution de l'algorithme AES. Ce bruit corrélé ne peut pas être filtré avec une CPA classique ; cependant, selon les auteurs de [93], un traitement adapté des traces permet toujours de récupérer la clé réelle, et cette protection n'est donc pas suffisante en elle-même.

Il est également possible de changer à chaque exécution les caractéristiques de consommation d'un circuit donné, en modifiant son implémentation. Les auteurs de [94] proposent ainsi d'implémenter en parallèle plusieurs blocs fonctionnellement équivalents et ayant des profils de consommation différents. Le choix du bloc à utiliser pour chaque chiffrement est effectué de manière aléatoire, afin de rendre également la consommation aléatoire. Cependant, aucune analyse de la sécurité n'est effectuée par les auteurs. Cette approche est limitée : un nombre fini de blocs différents sont implémentés en parallèle, et on a donc un nombre fini de profils de consommation différents. De plus, comme les consommations moyennes des différents blocs sont différentes, un attaquant peut aisément déterminer quel bloc est utilisé à chaque moment, et donc sélectionner les traces correspondant à un bloc donné. De la même manière, les auteurs de [95] proposent d'utiliser des chemins de données de manière aléatoire, en stockant les valeurs intermédiaires dans des bascules choisies aléatoirement plutôt que toujours dans les mêmes bascules. De même que pour la proposition précédente, un nombre limité de combinaisons est possible, et une attaque peut donc être effectuée avec un nombre suffisant de traces. Ces deux solutions introduisent un coût en surface important, qui croît avec le nombre de blocs fonctionnels ou de bascules.

Outre le bruit en amplitude, le bruit temporel a également été étudié afin de contrer les attaques par canaux auxiliaires. Plusieurs méthodes existent pour cela [91, 96, 64, 97], qui consistent à insérer des délais aléatoires au sein des opérations de chiffrement, à mélanger l'ordre des opérations lorsque celles-ci sont interchangeables et indépendantes, ou encore à effectuer des opérations supplémentaires inutiles d'un point de vue fonctionnel. De même que celles basées sur la génération de bruit en amplitude, ces protections complexifient les attaques, sans les rendre impossibles pour autant. Ainsi, dès 2000, Clavier *et al.* [98] ont montré qu'avec un traitement adapté des traces, il est possible de mener une CPA face à ce type de contre-mesures.

Au final, les contre-mesures basées sur de la génération de bruit, que celui-ci soit temporel ou en amplitude, peuvent rendre plus difficiles les attaques par canaux auxiliaires et augmenter le nombre de traces nécessaires afin de mener celles-ci à bien, sans pour autant contrer totalement ces attaques. Ces contre-mesures ne permettent donc pas d'assurer une protection suffisante en elles-mêmes, mais peuvent être associées à d'autres types de contre-mesures. De plus, elles sont généralement coûteuses. Le bruit en amplitude doit couvrir les variations du signal utile, ce qui implique souvent une consommation au moins multipliée par deux par rapport à une implémentation non protégée. Le bruit temporel implique une augmentation significative du temps d'exécution, ce qui impacte également la consommation d'énergie.

Protections externes au circuit protégé

Plusieurs propositions visant à réduire le SNR consistent à agir directement sur la tension d'alimentation ou l'horloge du circuit. Leur principal avantage est que ces protections sont appliquées de manière externe au circuit à protéger, et qu'il n'est donc pas nécessaire de modifier celui-ci ou

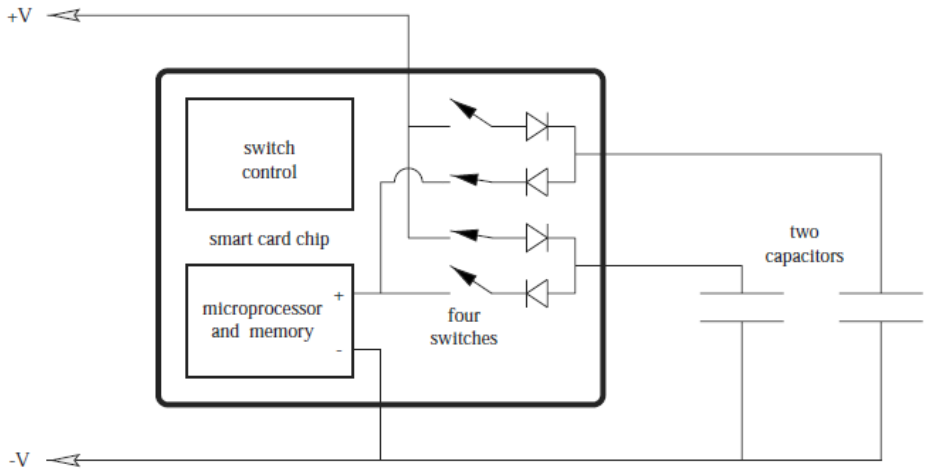


FIGURE 3.7 – Contre-mesure consistant à isoler le circuit de l'alimentation externe avec deux capacités qui commutent à tour de rôle [102].

de rajouter des blocs logiques.

L'ajustement dynamique de la fréquence et de la tension d'alimentation, ou DVFS pour *Dynamic Voltage and Frequency Scaling*, habituellement utilisé afin de réduire la consommation, peut être effectué de manière aléatoire [99]. Cette technique revient à générer du bruit afin de perturber une attaque par analyse des canaux auxiliaires. Dans [100], les auteurs montrent que les changements de fréquence produisent des variations transitoires de la tension, et qu'il est donc possible de détecter ces changements en mesurant cette tension. Le couple tension-fréquence peut alors être déterminé à chaque instant, rendant possible les attaques. Les auteurs proposent donc de maintenir la fréquence constante et de réaliser uniquement un ajustement dynamique aléatoire de la tension. Un tel ajustement dynamique de la tension est efficace seulement si l'intervalle entre les différents niveaux de tension est suffisamment grand, comme expliqué dans [101] ; la réduction de la tension d'alimentation externe a pour effet de réduire ces intervalles, et donc de faciliter les attaques contre ce type de contre-mesures.

Au lieu de générer du bruit, il est également possible de lisser la consommation. Une technique consiste à isoler le circuit à protéger de l'alimentation externe grâce à plusieurs capacités, et à utiliser ces capacités à tour de rôle afin d'alimenter le circuit. Ces capacités commutent à tour de rôle : lorsqu'une capacité se décharge dans le circuit, une autre capacité se charge avec l'alimentation [102, 103]. Cette contre-mesure avec deux capacités est représentée sur la figure 3.7, et a pour effet de lisser la consommation globale du circuit. Tokunaga *et al.* [102, 103] montrent qu'avec seulement deux capacités, des fuites d'information par canaux auxiliaires sont toujours mesurables, et préconisent donc d'utiliser au moins trois capacités.

D'autres travaux portent sur la mesure embarquée en temps réel de la consommation du circuit à protéger, et l'utilisation d'une boucle de rétroaction permettant d'ajuster la consommation globale. Le principe est de définir préalablement un niveau de courant à consommer, strictement supérieur au courant maximal consommé par le circuit à protéger, et de s'assurer que la consommation se maintient à ce niveau. Ainsi, Ratanpal *et al.* [104] proposent de court-circuiter le courant en surplus avec un transistor en parallèle du circuit à protéger, tandis que Attaran *et al.* [105] proposent de charger une capacité avec ce courant en surplus. L'utilisation d'une boucle de rétroaction présente plusieurs inconvénients. Le temps de réponse dans un tel dispositif n'est pas nul, et il y a donc un délai qui pourrait être exploité par un attaquant entre la détection d'une variation de la consommation et la réaction appropriée. De plus, le contrôle n'étant pas parfait, la consommation tend à osciller légèrement autour d'une valeur moyenne.

L'utilisation de composants analogiques et de capacités pour le lissage de la consommation rend généralement l'intégration de ces protections difficile et coûteuse sur un circuit intégré. Plus récemment, les auteurs de [106] proposent pour lisser la consommation d'utiliser des régulateurs

de tension intégrés, plus petits et plus efficaces que les méthodes existantes. Cette protection doit être combinée avec un ajustement dynamique aléatoire de la fréquence et de la tension d'alimentation afin de ne pas présenter de fuites pour une attaque CPA avec 50.000 traces.

Toutes ces méthodes de lissage de la consommation sont appliquées de manière externe au circuit protégé, et sont donc efficaces uniquement contre un attaquant mesurant la tension d'alimentation externe. Un attaquant capable de mesurer la tension locale directement à l'intérieur du bloc protégé, par exemple en plaçant une sonde au niveau du régulateur de tension intégré, pourrait contourner ces protections et mener à bien une CPA. De plus, celles-ci ne protègent pas contre l'exploitation des autres canaux auxiliaires ; ainsi, les travaux présentés dans [107] démontrent qu'un tel lissage de la consommation est inefficace contre un attaquant mesurant les émissions électromagnétiques avec une sonde placée au-dessus du circuit protégé. Pour contrer cette vulnérabilité, les auteurs proposent un routage spécifique de l'algorithme à protéger et de la protection en n'utilisant que les couches de métal basses d'un circuit intégré.

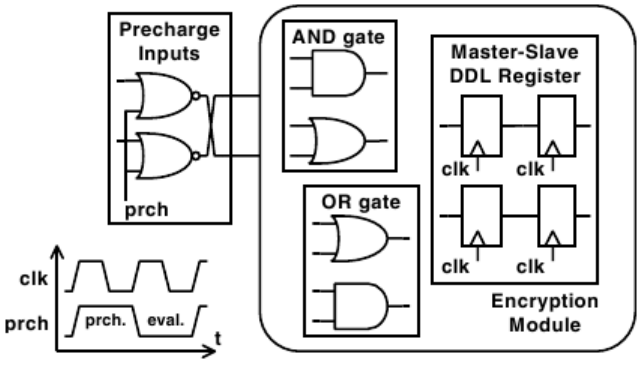
Enfin, le coût énergétique de ces protections est généralement important. D'un côté, le lissage de la consommation implique de maintenir cette consommation à un niveau élevé de manière permanente : celle-ci doit être au moins égale au maximum de consommation instantanée observable durant le fonctionnement du circuit non protégé. D'un autre côté, le bruit généré par le DVFS doit être suffisant afin de masquer les variations de consommation liées aux données secrètes. De plus, le DFVS est généralement mis en place afin de réduire la consommation d'un circuit intégré ; en utilisant cette méthode de manière aléatoire pour la sécurité du circuit intégré, on perd le bénéfice d'utiliser ce dispositif pour l'optimisation de la consommation.

Équilibrage de la consommation au niveau des portes logiques

De nombreux travaux portent sur le lissage de la consommation directement au niveau des portes logiques. Leur principe est d'équilibrer la consommation de chaque porte logique, afin que chacune ait une consommation constante quelles que soient ses entrées.

Plusieurs de ces propositions rentrent dans la famille de la logique différentielle avec précharge, également appelée logique à double rail avec précharge. Pour un bit donné, la logique différentielle consiste à manipuler simultanément ce bit et son inverse, sur deux chemins de données complémentaires en parallèle. Cela revient par exemple à coder un bit à "0" en "01", et un "1" en "10". Cette manipulation est effectuée au niveau de chaque porte logique. Cette technique fonctionne en deux phases : une phase de précharge, suivie d'une phase d'évaluation. Durant la précharge, les entrées et sorties de toutes les portes logiques du circuit sont mises à zéro (ou à un). Durant l'évaluation, les données réelles et complémentaires sont traitées. On observe alors exactement une transition de "0" vers "1" (ou inversement, selon la valeur de précharge) pour chaque porte logique protégée du circuit, celle-ci étant due soit à la donnée réelle, soit à son complément.

Un des premiers travaux sur ce type de logique est appelé *Sense Amplifier Based Logic*, ou SABL [108]. Ce style de logique requiert une conception manuelle de chaque porte logique, au niveau des transistors. Or, ce flot de conception dédié est très complexe à mettre en place. C'est pourquoi les auteurs de SABL proposent également la logique *Wave Dynamic Differential Logic*, ou WDDL [109], qui s'inspire des travaux réalisés sur le SABL mais pouvant être implémentée avec des portes logiques standard, pour un coût en surface et en énergie plus important. Afin de manipuler simultanément les valeurs réelles et complémentées en entrée tout en conservant une sortie différentielle, les portes logiques sont elles-mêmes implémentées de manière complémentée, en utilisant les lois de De Morgan. Par exemple, le complémentaire d'une porte OU telle que $Z = X + Y$ est $\overline{Z} = \overline{X}.\overline{Y}$, et ces deux versions complémentaires sont implémentées en parallèle. Enfin, contrairement au SABL où toutes les portes sont connectées à l'horloge et préchargées simultanément, la précharge est effectuée pour le WDDL avec une « vague » de précharge : elle est appliquée en entrée de chaque bloc combinatoire, et est transmise en cascade aux différentes portes. Cela pose un problème si des inverseurs font partie du bloc combinatoire, le signal de précharge étant également inversée. La figure 3.8a montre un exemple d'implémentation de la logique WDDL [109], avec la représentation de portes ET et OU ainsi qu'une bascule modifiée permettant de stocker



(a) Implémentation du style de logique WDDL

$$(A.B).\overline{prch} \leftrightarrow (\overline{A}+\overline{B}).\overline{prch}$$

A	B	$\overline{A}$	$\overline{B}$	prch	Z	$\overline{Z}$
0	0	1	1	0	0	1
0	1	1	0	0	0	1
1	0	0	1	0	0	1
1	1	0	0	0	1	0
X	X	X	X	1	0	0

(b) Table de vérité d'une porte ET en logique WDDL

FIGURE 3.8 – Le style de logique WDDL [109].

à la fois les données contenues sur les deux rails et les signaux de précharge et d'évaluation. La figure 3.8b représente la table de vérité d'une porte ET protégée par la logique WDDL [109] ; on constate que la sortie de celle-ci présente exactement une transition de "0" vers "1" lors de la phase d'évaluation, quelles que soient ses valeurs d'entrée.

Plusieurs problèmes rendent la logique WDDL, ainsi que le SABL dans une moindre mesure, vulnérable aux attaques par canaux auxiliaires. Ainsi, le routage de la donnée réelle et de la donnée complémentaire doivent être parfaitement équilibrés pour chaque porte logique protégée. Cela signifie que les capacités et les résistances doivent être identiques sur ces deux chemins de données, ce qui est difficilement réalisable en pratique. De plus, les différences de temps de propagation au sein des portes logiques standard, qui peuvent créer des glitchs ou une évaluation prématurée des signaux, permettent de réaliser une CPA avec succès contre le WDDL [110, 111]. De nombreuses variantes de la logique différentielle à double rail ont été proposées afin de répondre au moins partiellement à ces problèmes. On compte par exemple l'utilisation d'un troisième rail, ou la réalisation de cellules asynchrones à double rail avec précharge [112]. Une liste relativement exhaustive et une description de ces différentes variantes a été dressée dans [111, 113]. Par soucis de concision, ces méthodes ne sont pas détaillées ici. Parmi celles-ci, celles qui parviennent à répondre à l'un ou plusieurs des problèmes énoncés ci-dessus reposent en majorité sur des flots de conception spécialisés et introduisent des surcoûts importants.

Plus récemment, Bellizia *et al.* [114] ont proposé une implémentation avec un encodage différentiel *temporel* des données, appelé TEL pour *Time Enclosed Logic*, qui ne semble pas être affecté par les problèmes de déséquilibres et de temps de propagation de la logique différentielle classique. Cet encodage temporel repose également sur l'utilisation de deux rails et d'un signal de précharge. Cependant, les rails effectuent tous deux une transition de "0" vers "1" lors de la phase d'évaluation ; cette transition n'est pas simultanée pour les deux rails, et c'est donc l'ordre dans lequel les rails l'effectuent qui permet d'encoder l'information utile. L'encodage temporel pour un "1" et un "0" est représenté sur les figure 3.9a et 3.9b, respectivement. L'hypothèse sous-jacente et nécessaire pour le fonctionnement de ce type de logique est que le circuit est capable de détecter les délais de transition différents sur les deux rails, tandis que l'attaque est effectuée avec une résolution temporelle et une bande passante limitées. Un flot de conception spécialisé est nécessaire pour mettre en place ce type de logique.

Il est également possible de lisser la consommation sans utiliser de logique différentielle avec précharge. Par exemple, la logique adiabatique, ou logique à récupération de charge, a été initialement introduite afin de réduire la consommation d'un circuit. Le principe de ce style de logique est de récupérer le courant normalement dissipé vers la masse, afin de le réutiliser pour l'alimentation du circuit. Cela nécessite l'utilisation d'une source de tension spécialement conçue générant une tension variable selon le temps, ainsi que la conception manuelle des portes logiques au niveau des transistors avec un flot de conception spécialisé. Au prix d'efforts de conception et d'un

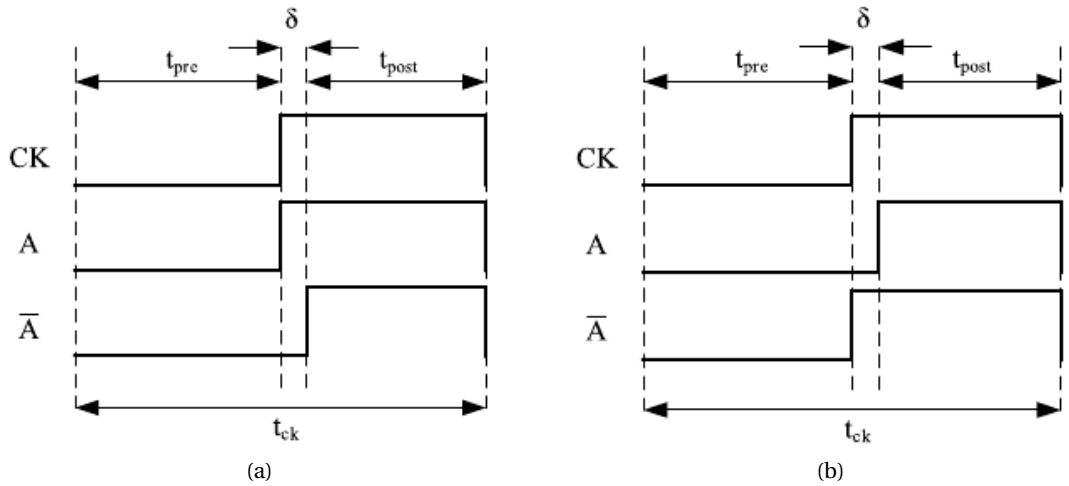


FIGURE 3.9 – Encodage différentiel temporel d'un "1" et d'un "0" [114].

coût en surface important, la logique adiabatique permet donc une réduction partielle des fuites d'information par canaux auxiliaires pour une consommation bien plus faible que les autres styles de logique.

De manière générale, le lissage de la consommation au niveau logique est très coûteux, à la fois en surface, en consommation et en temps d'exécution. Par exemple, deux implémentation de SABL sur Trivium dans les nœuds technologiques 0,13 µm et 0,35 µm [115], présentent respectivement une augmentation par 1,7 et 1,5 fois de la consommation instantanée, et par 2,7 fois en surface, par rapport à un Trivium non protégé. De plus, l'utilisation d'un signal de précharge devrait avoir un effet important sur le temps d'exécution, non évalué dans l'article. Ce coût est à mettre en relation avec les efforts de conception nécessaires pour l'implémentation de SABL, mais également avec la sécurité fournie : des variations de courant sont toujours observables sur les deux implémentations protégées.

Ce coût est plus élevé pour les styles de logique WDDL et TEL. Ceux-ci présentent les avantages par rapport au SABL, respectivement, de pouvoir être implémenté avec des cellules standard (WDDL), et de mieux résister à certaines attaques (TEL). Ainsi, le WDDL augmente de 4 fois la surface et de 3,5 fois la consommation d'un circuit dans le nœud technologique 0,18 µm [109], tandis que le TEL augmente de 3,5 fois la surface et de 4,4 la consommation pour une implémentation en 65 nm.

Ce coût provient de différents facteurs. D'un côté, la complexité de la logique combinatoire est élevée, avec des chemins de données complémentaires dans chaque porte logique protégée. D'un autre côté, le coût est généralement au moins multiplié par 4 pour la logique séquentielle. En effet, comme représenté sur la figure 3.8a, il est nécessaire de doubler le nombre de bascules en parallèle afin de stocker à la fois les valeurs réelles et leurs compléments, mais il faut également doubler le nombre de bascules en série pour stocker à la fois les signaux de précharge et d'évaluation. De plus, la logique de contrôle et la génération du signal de précharge introduisent également un coût important.

Enfin, l'alternance entre une phase de précharge et une phase d'évaluation a un impact important sur le temps d'exécution d'un algorithme. Certaines propositions partent du principe que la précharge et l'évaluation sont évaluées dans le même cycle d'horloge, l'une étant effectuée sur le front montant de l'horloge et l'autre sur le front descendant, et elles revendiquent donc ne pas avoir d'effets sur ce temps d'exécution. Cependant, les bascules doivent tout de même être dédoublées en série afin de stocker à la fois les signaux de précharge et d'évaluation. De la même manière, la logique combinatoire commute lors de chaque changement de phase, ce qui diminue par deux la fréquence de fonctionnement maximale si ces deux phases sont effectuées en un seul cycle d'horloge, tandis que la complexité accrue de la logique contribue également à réduire cette

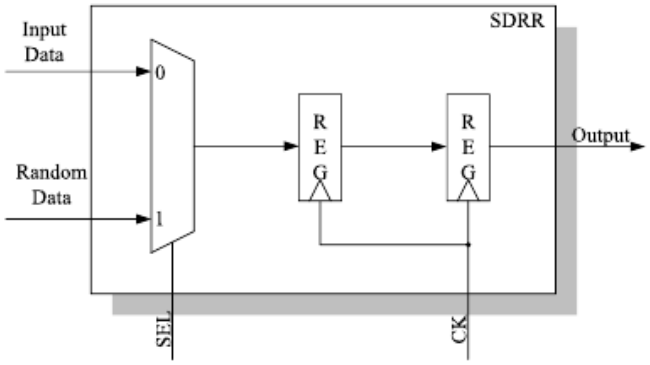


FIGURE 3.10 – Protection fondée sur des bascules à débit de données double [117].

fréquence. De manière générale, on peut considérer que l'existence d'une phase de précharge diminue d'au moins deux fois la vitesse d'exécution maximale d'un circuit [113].

Équilibrage algorithmique de la consommation

Entre le lissage de la consommation au niveau des portes logiques et celui réalisé de manière externe pour l'ensemble du circuit protégé, le lissage au niveau de l'algorithme présente un bon compromis. Les protections algorithmiques sont réalisées au niveau RTL, et requièrent donc moins d'effort d'intégration que le lissage au niveau des portes logiques ou du circuit intégré. Cependant, certaines requièrent tout de même un placement particulier lors de l'implémentation physique.

Dans [116], les auteurs proposent d'équilibrer le poids de Hamming en sortie d'une fonction logique complète, et non uniquement d'une porte logique. Cette solution repose sur une phase de précharge suivie d'une phase d'évaluation afin de présenter également une distance de Hamming constante. La consommation interne de chaque fonction doit également être équilibrée, ce qui est réalisé au cas par cas et d'une manière propre à chaque fonction. Cela résulte en un effort important pour adapter cette solution à chaque fonction à protéger. Selon les auteurs, cette méthode ne peut pas être appliquée à un algorithme entier, et n'est adaptée que pour protéger certaines fonctions, comme des opérations de substitution sur des données de 4 ou 8 bits.

Les auteurs de [117] proposent l'utilisation de bascules à débit de données double, notées SDRR pour *Secure Double Rate Registers*, comme représenté sur la figure 3.10. Chaque bascule est dédoublée, et alimentée alternativement avec une donnée réelle et une donnée aléatoire. Ainsi, la consommation ne dépend plus directement de la distance de Hamming entre deux données intermédiaires consécutives. Cette contre-mesure parvient donc à masquer une partie de la consommation due aux valeurs intermédiaires. Cependant, le poids de Hamming n'est pas affecté par cette protection. Cette protection appliquée à l'algorithme de chiffrement AES a un surcoût en consommation instantanée de 2,8 fois celle d'une implémentation non protégée, et de 33 % en surface. On notera que ce coût serait bien plus élevé pour un algorithme tel que Trivium pour lequel le nombre de registres est plus important. Enfin, cette contre-mesure souffre d'un temps d'exécution environ deux fois plus élevé qu'une implémentation non protégée, à cause de la duplication en série de l'ensemble de la logique séquentielle.

Les auteurs de [118] proposent une méthode plus générique, et plus facilement intégrable dans un flot de conception classique. Leur solution consiste à quadrupler tous les chemins de données (combinatoires et séquentiels) d'un algorithme donné. Afin d'obtenir une consommation constante, chacune des quatre instances d'un chemin de données traite des données partiellement complémentaires ; par exemple, pour protéger le traitement d'une donnée D et une clé K, les quatre instances traiteront simultanément les combinaisons $((D,K),(D,\overline{K}),(\overline{D},K),(\overline{D},\overline{K}))$. Afin d'atténuer les déséquilibres d'implémentation entre les différents chemins de données, chaque chemin doit être capable de traiter chaque combinaison de la donnée et de la clé, et ces combi-

naisons sont donc permutées de manière cyclique entre chaque bloc pour chaque nouveau chiffrement. Cette méthode permet d'équilibrer le poids de Hamming en sortie de chaque fonction combinatoire, et nécessite une précharge afin d'équilibrer la distance de Hamming. Appliquée à l'AES, elle présente une surface 6,5 fois plus élevée que celle d'une implémentation non protégée, une consommation 4 fois plus élevée, et un temps d'exécution deux fois plus long du fait de la précharge.

Dans [119], les auteurs améliorent la méthode précédente en rendant inutile la phase de précharge. Pour cela, des permutations spécifiques entre les données complémentées contenues sur les quatre chemins de données sont imposées après chaque opération non-linéaire. Avec des permutations judicieusement choisies, il est possible d'avoir à la fois un poids et une distance de Hamming totaux équilibrés au niveau algorithmique. Cette méthode résulte en une implémentation plus complexe que celle de [118], avec des conditions restrictives sur les entrées des différentes fonctions combinatoires et sur la représentation des données à l'intérieur de ces fonctions. Elle est donc plus difficile à intégrer à un flot de conception classique et doit être adaptée au cas par cas pour chaque fonction à protéger. Elle présente également un coût en surface plus important : deux implémentations différentes de celle-ci sur FPGA ont respectivement une surface 8 et 32 fois plus élevées qu'une implémentation non protégée de l'AES.

L'équilibrage de la consommation au niveau algorithmique a été étudié spécifiquement pour les registres à décalage dans [120, 121]. Ces deux propositions s'appuient sur le fait que la distance de Hamming est le meilleur modèle pour décrire la consommation d'un registre à décalage, et proposent donc uniquement une compensation de celle-ci et non du poids de Hamming. Dans [120], une bascule supplémentaire est rajoutée pour chaque bascule contenue dans le registre à décalage à protéger. Chaque bascule supplémentaire est contrôlée de sorte à commuter si la bascule non protégée correspondante ne commute pas, et inversement à garder sa valeur si la bascule non protégée commute, de sorte à maintenir une distance de Hamming totale constante. Cette solution est représentée sur la figure 3.11, où les bascules S sont les bascules originales du registre à décalage, et les bascules T sont les bascules supplémentaires permettant d'équilibrer le nombre de commutations. On remarquera sur cette figure que le rajout de logique de contrôle avec une modification du chemin de l'horloge est nécessaire pour chaque bascule. Les auteurs de [121], quant à eux, proposent de rajouter un second registre à décalage à rétroaction linéaire (ou LFSR, pour *Linear Feedback Shift Register*) en parallèle du LFSR à protéger, ce second registre ayant une fonction de rétroaction complémentaire à celle du premier afin d'équilibrer les commutations dans les deux registres. Cependant, cette dernière proposition n'est possible que pour les registre à décalage à rétroaction linéaire, tandis que les algorithmes de chiffrement nécessitent une rétroaction non-linéaire, par exemple avec l'usage de portes ET. Elle a donc un intérêt limité contre les attaques par canaux auxiliaires.

3.4.2 Masquage des données

Le principe du masquage est de rendre aléatoires les valeurs intermédiaires d'un algorithme. Ainsi, les données manipulées sont décorrélées des valeurs secrètes, et l'obtention de ces données par une mesure des canaux auxiliaires est alors inexploitable pour un attaquant souhaitent retrouver ces données secrètes. Le masquage peut être effectué au niveau des portes logiques, ou de l'algorithme. Le masquage au niveau des portes logiques est décrit dans les paragraphes qui suivent, et le reste de cette section détaille ensuite différents aspects du masquage algorithmique.

Masquage au niveau des portes logiques

Plusieurs exemples de réalisation du masquage au niveau des portes logiques ont été proposés, dont le DRSL (*Dual-rail Random Switching Logic*) [122] et le iMDPL (*improved Masked Dual-rail Pre-charge Logic*) [123]. Ces deux familles de logique exploitent le principe de la logique différentielle avec précharge afin de lisser la consommation. Elles sont plus résistantes face aux attaques par canaux auxiliaires, notamment en présence de capacités non équilibrées entre les

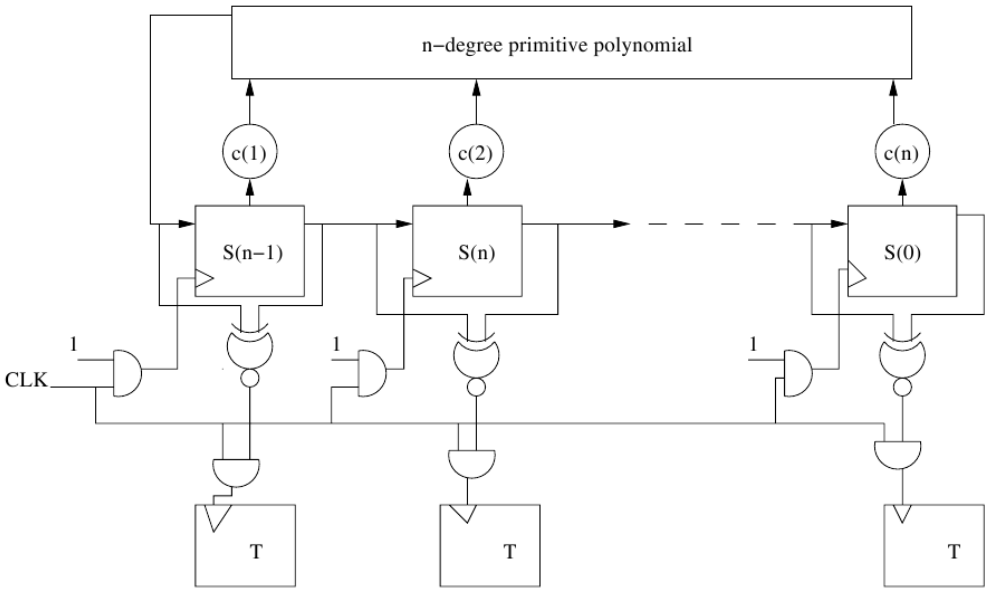


FIGURE 3.11 – Protection d'un registre à décalage grâce à l'ajout de bascules permettant d'équilibrer la distance de Hamming [120].

chemins de donnés complémentés, grâce à un masquage aléatoire de toutes les portes logiques. Tandis que le iMDPL suit les mêmes principes que le style de logique WDDL et peut être implémenté avec des cellules standard en adaptant un flot de conception classique, ce n'est pas le cas du DRSL qui nécessite un flot dédié. L'implémentation de ces protections est coûteuse. Dans le cas du DRSL, la surface des portes ET et OU est multipliée par 5,42, celle des portes NON-ET et NON-OU par 7,21, et celle des bascules D par 2,56, par rapport à des portes standard. Selon [124], un circuit implémenté en logique iMPDL présente une surface d'environ 18 à 19 fois celle d'un circuit non protégé, tandis que la consommation est multipliée par un facteur 5 à 10 et que la fréquence maximale est également significativement réduite. De plus, ces deux méthodes requièrent toutes deux la génération d'un nouveau bit de masque aléatoire à chaque cycle d'horloge. Plusieurs attaques ont été publiées contre ces contre-mesures [125, 126, 127]. Elles démontrent que le masquage au niveau des portes logiques peut être contourné et n'est pas suffisant pour compenser les fuites d'information des styles de logiques différentielles.

Principe du masquage algorithmique

Le masquage algorithmique présente l'avantage d'être réalisé intégralement au niveau RTL. En effet, son objectif étant de rendre les valeurs intermédiaires aléatoires et non pas de lisser la consommation, il n'est pas nécessaire d'équilibrer les différents chemins de données ou d'effectuer d'autres tâches spécifiques lors de l'implémentation matérielle. Cette protection s'intègre donc très bien dans le flot de conception d'un circuit intégré. Le principe du masquage algorithmique, présenté sur la figure 3.12 extraite de [44], est d'ajouter avant le chiffrement un ou plusieurs masques aux données traitées, et de retirer ces masques à la fin du chiffrement. Les masques sont générés aléatoirement, et sont différents pour chaque chiffrement. Afin de garantir l'exactitude du résultat, les masques sont traités suivant les mêmes opérations que les données masquées.

Dans nos travaux, nous considérons uniquement les masques booléens, ajoutés puis retirés avec un XOR bit à bit aux données. Ces masques sont en effet plus légers que les masques arithmétiques, et sont plus adaptés pour les implémentations matérielles. Une représentation possible du masquage d'une donnée x est donc la séparation de cette donnée en n parts x^i telles que $x = \sum_i x^i$, cette somme étant effectuée dans le corps de Galois GF(2) grâce à des portes XOR. Pour une donnée à masquer d et un unique masque m, on peut par exemple initialiser les deux parts de la

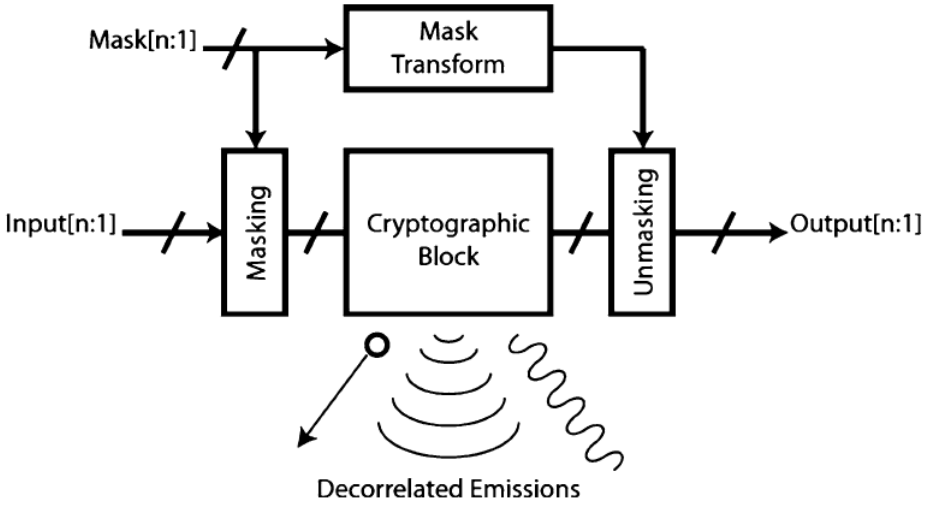


FIGURE 3.12 – Principe du masquage algorithmique [44].

manière suivante : $(x^1, x^2) = (d + m, m)$. De cette façon, la donnée d non masquée n'est pas traitée directement par l'algorithme. Selon les implémentations du masquage, les opérations linéaires sont généralement effectuées indépendamment et en parallèle sur toutes les parts, tandis que les opérations non-linéaires peuvent être effectuées sur plusieurs parts à la fois, voire nécessiter un re-masquage des données avec l'utilisation de nouveaux masques aléatoires.

Remarque : Le numéro correspondant à la part masquée est généralement indiqué en indice dans les publications sur le masquage. Dans nos travaux, nous indiquons ce numéro en exposant, afin de ne pas introduire de confusion par rapport à la notation choisie pour représenter l'état interne d'un registre à décalage, où S_i est le i-ième bit d'état interne selon la section 1.3.2.

Attaques et masquage d'ordres supérieurs

Un attaquant capable de récupérer les données traitées sur les différentes parts peut remonter aux données non masquées, et donc effectuer une CPA. Afin de réaliser une telle attaque, les parts doivent être combinées de manière non-linéaire [128]. Pour récupérer les informations contenues sur d parts traitées en parallèle, les traces doivent être traitées en élevant chaque point de celles-ci à la puissance d, ce qui permet d'exhiber leur d-ième moment statistique. La CPA est ensuite effectuée de manière classique sur les traces traitées de cette manière. Une telle attaque est appelée une *CPA d'ordre d*, ou *CPA à l'ordre d* ; par la suite, on désigne par *CPA d'ordre supérieur* une CPA dont l'ordre est supérieur à 1. Une CPA classique, d'ordre 1, permet de différencier les moyennes de plusieurs lots de traces non masquées selon les hypothèses effectuées sur la clé, comme décrit dans la section 3.1. Une attaque d'ordre 2 permet d'attaquer une implémentation de masquage sur deux parts, c'est-à-dire avec un seul masque, en exploitant la variance des traces plutôt que leur moyenne, et une attaque d'ordre 3 permet de récupérer les informations contenues sur 3 parts en exploitant le coefficient d'asymétrie des traces.

Les attaques précédentes sont dites *univariées* car elles sont effectuées sur une seule variable, c'est-à-dire sur un seul instant des traces à la fois. Elles sont particulièrement adaptées aux implémentations matérielles du masquage, où le masque est traité en parallèle des données. Des attaques dites *multivariées* existent également. Plutôt que d'élever les traces à une certaine puissance afin de mettre en avant les phénomènes non-linéaires à chaque instant, celles-ci combinent de manière non-linéaire les mesures effectuées à plusieurs instants. Par exemple, une attaque multivariée d'ordre 2 consiste à multiplier les mesures effectuées à deux instants différents. Ces attaques multivariées sont généralement utilisées pour attaquer les implémentations logicielles où les données masquées et les masques sont traités successivement [43], mais elles peuvent également être utilisées contre des implémentations matérielles dans certains cas [78].

Afin de contrer les attaques d'ordre supérieur, il faut mettre en place un masquage d'ordre supérieur. En effet, il faut au minimum $d+1$ parts, c'est-à-dire d masques aléatoires indépendants, pour contrer une attaque d'ordre d. La difficulté de l'attaque et notamment le nombre de traces nécessaires augmente avec l'ordre du masquage. Ainsi, selon Standaert [129], le bruit, qu'il soit naturellement présent lors de la mesure ou généré en tant que protection supplémentaire, est amplifié lors de l'exécution d'attaques d'ordre supérieur. Le nombre de traces nécessaires pour réaliser une attaque augmente alors de façon exponentielle avec l'ordre du masquage en présence de bruit, ce qui rend les attaques irréalisables en pratique à partir d'un certain ordre.

Le masquage en *Threshold Implementation*

De nombreuses méthodes de masquage existent. De manière générale, celles-ci peuvent être considérées comme des approches différentes basées sur les mêmes principes fondamentaux [130]. Cependant, leurs implémentations ne fournissent pas toutes le même niveau de sécurité. Ainsi, de nombreuses propositions, comme celle d'Ishiai *et al.* [131] ou l'implémentation compacte du masquage de Canright *et al.* [132], reposent sur l'hypothèse de portes logiques idéales et ne prennent pas en compte l'influence des glitchs dans ces portes. Or, ces implémentations du masquage présentent des fuites importantes en présence de glitchs dans un circuit réel, comme montré dans [133]. Ces glitchs sont inévitables sans rajouter un coût silicium et des efforts de conception importants, et un masquage qui ne fuite pas d'informations en présence de glitchs doit donc être sélectionné afin de protéger efficacement le circuit.

Le masquage en *Threshold Implementation* [134], noté par la suite TI, présente de nombreux avantages. Certaines méthodes de masquage nécessitent un re-masquage des données traitées au cours de l'exécution de l'algorithme protégé, c'est-à-dire l'ajout de bits aléatoires supplémentaires à ces données ; le masquage TI requiert, selon les cas, moins de re-masquage des données au cours du calcul que d'autres implémentations, et utilise donc moins de bits aléatoires. La génération de ces bits aléatoires étant coûteuse d'un point de vue énergétique, ce gain est important pour une utilisation dans l'IoT. De plus, le masquage TI ne repose sur aucune hypothèse sur l'implémentation des portes logiques, et sa sécurité peut être prouvée de manière formelle même en présence de glitchs [135]. Enfin, cette implémentation du masquage a été très étudiée depuis sa publication en 2006, avec des implémentations et évaluations sur de nombreux algorithmes tels que l'AES [136], KATAN [135], PRESENT [137] ou encore Trivium [138], et plusieurs propositions de combinaisons de cette protection avec d'autres contre-mesures afin d'augmenter la sécurité contre les attaques par canaux auxiliaires [139] ou de garantir une sécurité contre les attaques en fautes [140, 141, 142, 143]. C'est donc une implémentation mature et éprouvée, avec une sécurité théorique vérifiée de nombreuses fois en pratique.

Le masquage TI repose sur les théories du *partage du secret* et du *calcul multipartite*. On considère ici un masquage TI à l'ordre d tel que décrit par ses auteurs [135]. Une variable x est séparée en n parts x^i, comme décrit précédemment. Les opérations linéaires, telles que les XOR, sont effectuées indépendamment sur chaque part. Les fonctions non-linéaires telles que les portes ET, quant à elles, ne peuvent pas être effectuées indépendamment sur chaque part en conservant l'exactitude du résultat. Une fonction non-linéaire f est donc décomposée en m sous-fonctions f^j, chacune de ces sous-fonctions pouvant combiner plusieurs parts en entrée pour générer un bit de sortie. On écrira par la suite qu'une telle fonction f est une *fonction partagée*. Cette *fonction partagée* prend les n parts de la variable x en entrée, et le nombre m de sous-fonctions détermine le nombre de *parts de sortie* de la fonction f, qui peut être différent du nombre de *parts d'entrée*. Une *fonction partagée* doit respecter les propriétés suivantes :

— **Non-complétude :** Pour un masquage à l'ordre d, n'importe quelle combinaison de d sous-fonctions doit être indépendante d'au moins une part d'entrée. Dans certaines fonctions complexes, certaines opérations doivent être séparées par un registre supplémentaire afin de garantir la non-complétude.

— **Exactitude :** La somme des parts de sortie doit être égale à la valeur de sortie de la fonction f lorsqu'elle n'est pas protégée.

— **Uniformité :** La distribution des parts de sortie est uniforme, c'est-à-dire que toutes les différentes combinaisons de parts de sortie sont produites avec la même probabilité en variant les combinaisons de parts d'entrée. Cette propriété permet de garantir l'indépendance des parts de sortie. Cette propriété ne peut pas être assurée pour chaque fonction, et il est parfois nécessaire de rajouter de nouveaux bits de masques aléatoires pour garantir l'uniformité.

D'après Bilgin *et al.* [135], le nombre de parts en entrée d'une *fonction partagée* est au minimum de $(td+1)$, où d est l'ordre du masquage et t le degré de la fonction. Ainsi, pour un masquage au premier ordre, il faut au minimum deux parts en entrée d'une fonction linéaire de degré 1 telle qu'une porte XOR, et au moins trois parts pour une porte ET, qui est une fonction de degré 2. À titre d'exemple, un masquage possible au premier ordre d'une fonction $y = f(a,b,c) = a + bc$, tiré de [134, 135] est effectué avec 3 parts d'entrée et de sortie selon l'équation (3.3) :

$$\begin{aligned} y^1 &= a^2 + b^2c^2 + b^2c^3 + b^3c^2 \\ y^2 &= a^3 + b^3c^3 + b^3c^1 + b^1c^3 \\ y^3 &= a^1 + b^1c^1 + b^1c^2 + b^2c^1 \end{aligned} \tag{3.3}$$

Le masquage de la même fonction au second ordre, selon [135], peut être effectué avec 5 parts d'entrée et 10 parts de sortie. Il est donné par l'équation (3.4) :

$$\begin{aligned} y^1 &= a^2 + b^2c^2 + b^1c^2 + b^2c^1 \\ y^2 &= a^3 + b^3c^3 + b^1c^3 + b^3c^1 \\ y^3 &= a^4 + b^4c^4 + b^1c^4 + b^4c^1 \\ y^4 &= a^1 + b^1c^1 + b^1c^5 + b^5c^1 \\ y^5 &= b^2c^3 + b^3c^2 \\ y^6 &= b^2c^4 + b^4c^2 \\ y^7 &= a^5 + b^5c^5 + b^2c^5 + b^5c^2 \\ y^8 &= b^3c^4 + b^4c^3 \\ y^9 &= b^3c^5 + b^5c^3 \\ y^{10} &= b^4c^5 + b^5c^4 \end{aligned} \tag{3.4}$$

Ce masquage au second ordre pourrait être réalisé différemment, par exemple avec 6 parts d'entrée et 7 parts de sortie ; le nombre de parts de sortie est généralement supérieur à celui de parts d'entrée lorsque l'ordre d du masquage est supérieur ou égal à 2 selon les travaux présentés dans [135]. Afin de maintenir un nombre de parts constant sur l'ensemble de l'algorithme, les auteurs proposent de réduire le nombre de parts selon la *fonction de réduction* $z = g(y)$ décrite par les équations suivantes :

$$\begin{aligned} z^i &= y^i \text{ pour } i \in \{1,\ldots,4\} \\ z^5 &= y^5 + y^6 + y^7 + y^8 + y^9 + y^{10} \end{aligned} \tag{3.5}$$

Afin de satisfaire la propriété de non-complétude et d'éviter les fuites d'information, la fonction non-linéaire partagée, décrite par l'équation (3.4), et la réduction du nombre de parts selon l'équation (3.5) doivent être séparées par un registre. On notera que le masquage de la fonction f à l'ordre 1 selon l'équation (3.3) est uniforme. Le masquage à l'ordre 2 selon l'équation (3.4) n'est pas uniforme en lui-même, mais la *fonction partagée* $h(a,b,c) = g(f(a,b,c))$ résultant de la combinaison des équations (3.4) et (3.5) est bien uniforme.

De manière générale, le coût du masquage est important, à la fois en surface silicium et en consommation. Il croît avec le nombre de parts, et dépend également du nombre de bits aléatoires nécessaires lors de l'initialisation et du fonctionnement, et de la complexité de l'implémentation des fonctions non-linéaires. De plus, selon les implémentations, le masquage peut également entraîner une augmentation du temps d'exécution du fait de l'ajout de registres supplémentaires. Par exemple, une simulation post-synthèse d'une implémentation de ce masquage au premier ordre

sur l'AES montre une augmentation de la consommation instantanée de 3,6 fois, et du nombre de cycles d'horloge de 18% [136]. Ce surcoût ne considère pas l'ajout de nouveaux bits de masques aléatoires à chaque cycle d'horloge, nécessaire dans ce cas afin de garantir l'uniformité. Un masquage à un ordre supérieur nécessiterait non seulement une augmentation du nombre de parts, mais également une augmentation de la complexité des fonctions partagées et l'ajout de registres lors de la réduction du nombre de parts après chaque opération non-linéaire.

Remarque : Nous avons présenté ici un masquage TI qui requiert au minimum $(td+1)$ parts pour masquer une fonction de degré t à l'ordre d. Reparaz *et al.* ont montré récemment que de telles fonctions peuvent être masquées avec seulement $(d+1)$ parts [130], ce qui est suffisant en théorie pour contrer une attaque d'ordre d. Réduire le nombre de parts peut a priori se montrer avantageux tant d'un point de vue du coût silicium qu'en ce qui concerne la consommation. Une implémentation et une évaluation de ce masquage à $(d+1)$ parts de l'AES a été réalisée dans [144]. Les auteurs montrent une réduction de la surface par rapport à un masquage avec $(td+1)$ parts ; cependant, l'implémentation avec $(d+1)$ parts a une plus grande complexité, et le gain en surface n'est pas proportionnel au gain en nombre de parts. En effet, l'implémentation avec $(d+1)$ parts repose sur des fonctions partagées plus complexes, avec plus de sous-fonctions, et présente plus de contraintes sur l'indépendance des différentes variables manipulées. Cette implémentation requiert également plus de bits d'aléa, et plus de cycles d'horloge pour l'exécution. La différence de consommation d'énergie entre les implémentations à $(d+1)$ et $(td+1)$ parts reste donc à évaluer. En ce qui concerne la sécurité, l'implémentation avec $(d+1)$ parts est généralement vulnérable aux attaques du $(d+1)$-ième ordre, tandis que celle avec $(td+1)$ parts n'est vulnérable qu'aux attaques du $(td+1)$-ième ordre dans certains cas. De plus, une implémentation similaire, reprenant les principes de Reparaz *et al.* [130] pour un masquage à $(d+1)$ parts, a été décrite dans [145] pour les algorithmes SIMON et PRESENT ; les auteurs y démontrent des fuites d'information importantes au second ordre avec seulement 500 traces pour SIMON et 6000 traces pour PRESENT. En attendant une évaluation complète des gains et coûts liés à l'implémentation avec $(d+1)$ parts ainsi que de sa sécurité, nous restreignons donc par la suite notre étude aux seules implémentations à $(td+1)$ parts, plus matures et plus évaluées.

3.4.3 Combinaison de plusieurs contre-mesures

Les contre-mesures présentées précédemment contre les attaques par analyse des canaux auxiliaires peuvent être implémentées à différents niveaux, des transistors au circuit intégré dans son ensemble. Elles ont des effets différents et complémentaires : tandis que le masquage vise à décorréler les données secrètes des traces de consommation mesurées, la réduction du SNR a pour but de rendre les attaques plus difficiles en générant du bruit ou en lissant la consommation. Pour une meilleure protection, les différentes contre-mesures contre les attaques par canaux auxiliaires peuvent être combinées. Cela permet à la fois d'augmenter la difficulté globale des attaques, tout en empêchant les attaques visant une contre-mesure en particulier, comme les attaques d'ordre supérieur contre le masquage ou l'élimination par des méthodes statistiques du bruit.

Par exemple, de nombreuses protections gagnent en efficacité en présence de bruit. C'est notamment le cas du masquage, car selon la section 3.4.2, le bruit est amplifié lors de l'exécution d'attaques d'ordre supérieur contre cette contre-mesure. Ainsi, Kamoun *et al.* [93] étudient la combinaison du masquage et de la génération de bruit corrélé, tandis que Moradi *et al.* [139] proposent de combiner un masquage TI à un lissage de la consommation au niveau des portes logiques. De la même manière, plusieurs auteurs [146, 117, 106, 107] préconisent de générer du bruit en complément de contre-mesures de lissage de la consommation. Dans ce cas, l'addition de bruit permet de masquer les variations de tension subsistantes et les imperfections du lissage de la consommation inhérentes à une implémentation matérielle. Le bruit peut alors être temporel comme dans [146] ou en amplitude comme dans [117, 106, 107], tandis que le lissage peut être effectué au niveau logique [146], algorithmique [117] ou au niveau du circuit intégré [106, 107]. On notera que cette addition de plusieurs contre-mesures afin d'augmenter la sécurité résulte en une addition des coûts en surface, consommation et temps d'exécution de ces contre-mesures.

3.5 Protections contre les attaques par injection de fautes

Plusieurs protections existent pour protéger un circuit intégré contre une attaque par injection de fautes. Certaines protections permettent de détecter l'utilisation de certains moyens physiques d'injection de fautes, ou d'empêcher l'injection de fautes avec ces moyens physiques. Ces protections nécessitent de modifier l'implémentation physique du circuit intégré lui-même, ou d'intégrer divers capteurs à ce circuit. D'autres protections sont appliquées au niveau algorithmique ou au niveau logique, et permettent la détection ou la correction de fautes quel que soit le moyen d'injection, une fois ces fautes injectées dans le circuit.

3.5.1 Détection et prévention de l'injection physique de fautes

La première catégorie de contre-mesures vise les moyens matériels permettant d'injecter des fautes [6, 40, 41, 42]. L'objectif de ces contre-mesures est soit d'empêcher physiquement un attaquant d'injecter des fautes, soit de détecter les moyens matériels mis en œuvre afin d'injecter ces fautes. Chacune de ces contre-mesures vise donc à protéger un circuit intégré contre un moyen d'attaque en particulier.

Plusieurs protections consistent à modifier l'implémentation physique d'un circuit intégré. Par exemple, l'utilisation d'horloges internes ou de filtres sur la tension permettent de rendre plus complexe, voire impossible, l'injection de glitchs d'horloge ou de tension. De même, des grilles métalliques ou des couches de métal réfléchissantes, placées au-dessus d'un circuit intégré et appelées *boucliers*, permettent de protéger celui-ci contre les injections de fautes avec un laser et/ou une sonde électromagnétique. Certains boucliers, dits *actifs*, permettent la détection d'une attaque s'ils sont altérés ou détruits. D'autres protections consistent à intégrer au circuit des capteurs permettant de détecter certains types d'attaques, comme des capteurs de tension, de température, de fréquence ou encore de lumière.

Un des principaux inconvénients de ces contre-mesures est leur coût. En effet, ces contre-mesures nécessitent la modification de l'implémentation matérielle de l'ensemble d'un circuit intégré, ou l'insertion d'un certain nombre de capteurs dans ce circuit. Comme chacune de ces contre-mesures vise à empêcher ou détecter un type d'injection de fautes en particulier, plusieurs de ces contre-mesures doivent être additionnées afin de couvrir divers moyens d'injections. De plus, ces contre-mesures sont vulnérables contre de futures attaques, qui utiliseraient d'autres moyens d'injection de fautes, et ne sont donc pas suffisantes pour la protection d'un circuit intégré sur le long terme selon les auteurs de [41].

3.5.2 Détection et correction des fautes

Les contre-mesures algorithmiques et logiques, quant à elles, protègent un algorithme en particulier plutôt que l'ensemble du circuit intégré, et sont donc moins coûteuses que les protections consistant à modifier l'implémentation physique d'un circuit, selon [40, 41, 42]. Leur objectif est de détecter ou de rendre inexploitables les fautes, une fois celles-ci injectées et indépendamment de la manière dont l'injection a été effectuée. Les contre-mesures algorithmiques sont appliquées au niveau RTL, et elles sont donc plus génériques et plus facilement applicables à différentes cibles (FPGA ou ASIC, dans différentes technologies) que la protections de l'implémentation physique du circuit ou l'insertion de divers capteurs dans le circuit..

De même que pour les attaques par canaux auxiliaires, un changement fréquent des clés de chiffrement peut permettre de contrer les attaques en fautes qui visent à récupérer ces clés de chiffrement et qui nécessitent un nombre important d'injections de fautes. Les coûts en énergie et en temps d'exécution d'une telle contre-mesure, ainsi que la complexité de mettre celle-ci en œuvre, ont été évoqués dans la section 3.4.

L'ajout de bruit temporel, décrit dans la section 3.4.1 pour la protection contre les attaques par canaux auxiliaires, peut complexifier la mise en œuvre d'une injection de fautes, ou rendre plus difficile l'exploitation des fautes injectées [42]. Cette protection peut empêcher un attaquant de

savoir précisément à quel moment de l'exécution d'un algorithme une faute a été injectée. Cependant, certaines attaques par injection de fautes, comme celles décrites dans la section 3.3.3 contre l'algorithme Trivium, ne nécessitent pas une grande précision temporelle, car un attaquant peut retrouver le moment précis de l'injection à partir de l'analyse de la sortie fautée de l'algorithme. De plus, il est possible, via une attaque par analyse des canaux auxiliaires, de détecter à quel moment les différentes opérations d'un algorithme sont effectuées, et d'injecter des fautes aux bons moments en fonction de ces observations [147].

La détection et la correction algorithmique de fautes repose sur diverses formes de redondance [148]. La redondance spatiale consiste à dupliquer le bloc fonctionnel à protéger, en l'intégrant au moins deux fois sur un circuit intégré. Les calculs sont effectués en parallèle par les blocs dupliqués, et les résultats sont comparés. L'utilisation de deux blocs en parallèle permet de détecter une faute injectée dans un seul des deux blocs, tandis que l'intégration de trois blocs ou plus permet également de corriger la faute en choisissant le résultat majoritaire. La redondance temporelle consiste à effectuer plusieurs fois de suite les mêmes opérations avec un même bloc fonctionnel, et à comparer les résultats des différentes exécutions. Enfin, la redondance d'information consiste à utiliser des codes de détection (ou de correction) d'erreurs calculés à partir des données à protéger. Le principal avantage de cette méthode est son coût par rapport aux deux autres types de redondance, ces codes étant généralement plus petits que l'information protégée. Par exemple, un bit de parité compte le nombre de bits à "1" dans une donnée, et vaudra "0" ou "1" selon la parité de ce nombre, quelle que soit la taille de la donnée. Ce bit de parité permet de détecter si une faute est injectée sur un nombre impair de bits dans la donnée protégée.

La redondance peut également être implémentée au niveau de chaque porte logique qui constitue le circuit intégré. Les styles de logique différentielle avec précharge présentés dans la section 3.4.1 constituent en effet une forme de redondance spatiale. L'information originale y est contenue sur l'un des deux chemins de données, et le complémentaire de cette information est contenu sur l'autre chemin de données. Pour ce type de logique, seuls les états "01" et "10" sont possibles lors de la phase d'évaluation, pour coder respectivement les bits "0" et "1", et une injection de fautes résultant en un état différent peut donc être détectée [112, 111].

Les contre-mesures à base de redondance sont toujours susceptibles à certaines attaques en fautes. La redondance spatiale est notamment vulnérable face aux injections de fautes globales par altération de la tension d'alimentation ou de l'horloge, les fautes injectées par ces moyens affectant de manière similaire les chemins de données redondants [6]. Afin d'attaquer une implémentation protégée par de la redondance temporelle, un attaquant peut injecter plusieurs fois de suite la même faute au même moment lors des différentes exécutions redondantes d'une opération [149], ce qui est réalisable avec plusieurs méthodes d'injection qui sont reproductibles avec précision, comme des glitchs d'horloge ou l'utilisation d'un laser. De plus, l'injection de fautes permanentes dans un bloc fonctionnel ne peut pas être détectée par la redondance temporelle [149]. Enfin, la couverture des fautes par la redondance d'information est très variable selon le code de détection d'erreurs utilisé, mais elle est généralement inférieure à celle de la redondance spatiale ou temporelle [40, 149]. En connaissant le type de code de détection d'erreurs utilisé, un attaquant peut choisir les fautes à injecter afin que celles-ci ne soient pas détectées par ce code [149]. Un exemple simple est l'injection d'un nombre pair de bits fautés contre un circuit utilisant des bits de parité pour détecter les fautes.

Ces contre-mesures introduisent également un coût important. Ainsi, la redondance spatiale a généralement pour effet de doubler, au minimum, la surface et la consommation instantanée, tandis que la redondance temporelle implique une réduction des performances d'au moins deux fois [42]. La nécessité de comparer les résultats et de stocker certains résultats intermédiaires peut également résulter en un coût supplémentaire en surface et en consommation pour la redondance temporelle [148]. Certaines optimisations, dont une liste non exhaustive est donnée dans [148], peuvent être réalisées afin de réduire ces coûts importants, mais elles sont dédiées à des architectures particulières, disposant par exemple déjà de blocs identiques en parallèle. Enfin, le coût de la redondance d'information varie selon le type de code de détection utilisé. Par exemple, certains

codes ont un coût en surface très faible, tandis que d'autres ont un coût plus important que la redondance spatiale [149]. Le choix de ce code résulte donc d'un compromis entre la protection fournie et le coût en surface, consommation et temps d'exécution.

Au final, les différentes méthodes de détection basées sur la redondance sont notamment coûteuses en consommation instantanée et/ou en temps d'exécution, et ont donc un impact important sur l'énergie consommée. Elles sont efficaces pour des modèles d'attaques précis correspondant à des méthodes d'injection données, mais peuvent échouer à fournir une protection adaptée contre d'autres méthodes d'injection. Un attaquant ayant connaissance du type de redondance utilisé peut donc choisir la meilleure manière d'injecter des fautes afin de contourner les protections en place. Pour offrir une meilleure protection, ces différentes protections peuvent être combinées. Par exemple, en chiffrant une donnée, puis en la déchiffrant avec un bloc fonctionnel différent et en comparant les deux données, on utilise à la fois les principes de la redondance spatiale et de la redondance temporelle, ce qui permet en théorie une plus grande robustesse [148, 149]. Toutefois, en combinant plusieurs types de redondance, leurs coûts s'additionnent également.

3.6 Protections mixtes

Dans les sections précédentes, des contre-mesures visant à protéger une implémentation soit contre les attaques par canaux auxiliaires, soit contre les attaques en fautes, ont été présentées. Or, ces deux attaques peuvent toutes deux être réalisées séparément contre un circuit, voire être combinées en une attaque plus efficace. Par exemple, les auteurs de [150] montrent qu'en fixant certaines valeurs intermédiaires à zéro grâce à une injection de fautes contre l'algorithme RSA, l'analyse visuelle d'une seule trace de consommation suffit à déterminer la valeur de la clé de chiffrement. Afin de protéger un circuit efficacement, il convient donc de s'assurer que l'implémentation de ce circuit résiste aux deux types d'attaques.

Certaines protections conçues uniquement contre un seul de ces deux type d'attaques peuvent faciliter l'autre type d'attaques. Par exemple, la redondance au niveau algorithmique, qu'elle soit spatiale, temporelle, ou en information, renforce les fuites par canaux auxiliaires [148, 111, 151]. Les auteurs de [151] évaluent expérimentalement la résistance aux attaques par canaux auxiliaires de plusieurs implémentations d'un AES comportant de la redondance spatiale; en particulier, l'une de ces implémentations contient des chemins redondants sur des données complémentaires, ce qui devrait fournir une certaine résistance contre les attaques par canaux auxiliaires selon les auteurs de cette contre-mesure [152]. Cette attaque par canaux auxiliaires menée dans [151] requiert trois fois moins de traces contre toutes les implémentations protégées avec de la redondance que contre une implémentation non protégée. Un autre exemple de contre-mesure favorisant un autre type d'attaques est fourni dès 2001 dans [153], où les auteurs montrent qu'une protection contre les attaques par canaux auxiliaires, spécifique aux algorithmes basés sur une exponentiation modulaire, introduit une nouvelle vulnérabilité pour l'injection de fautes.

D'autres contre-mesures conçues contre un seul type d'attaques ne créent a priori pas de vulnérabilité contre l'autre type d'attaques, mais ne fournissent pas non plus de sécurité particulière contre celui-ci. C'est le cas par exemple du masquage TI, qui ne facilite généralement pas, ni n'empêche, les attaques par injection de fautes. Il est possible de combiner de telles protections contre les deux types d'attaques, sous réserve que cette combinaison elle-même n'introduise pas de vulnérabilité. Cependant, les coûts des différentes contre-mesures s'additionnent alors, et peuvent devenir prohibitifs.

Certaines protections initialement conçues contre un type d'attaque en particulier présentent également une certaine résistance contre l'autre attaque. Par exemple, le bruit temporel, avec un décalage des traces ou l'ajout d'instructions inutiles, permet de complexifier la mise en pratique des deux types d'attaques, sans pour autant les empêcher totalement [42]. La logique différentielle avec précharge offre également une protection contre les attaques en fautes [112, 111], dans une certaine mesure. Cependant, une faute injectée durant la phase de précharge n'est pas forcément détectée, et peut entraîner des fuites par canaux auxiliaires. De plus, dans [154], les auteurs pro-

posent une attaque combinée par injection de fautes et par analyse des canaux auxiliaires sur ce style de logique. Ils montrent que certaines fautes injectées durant la phase d'évaluation ne créent pas d'erreur fonctionnelle dans le circuit et sont donc indétectables, mais ont pour effet de déséquilibrer la consommation dans les chemins de données différentiels à l'intérieur d'une porte logique protégée, créant ainsi des fuites exploitables par des attaques par canaux auxiliaires.

Enfin, quelques contre-mesures mixtes, spécifiquement conçues pour contrer à la fois les attaques par analyse des canaux auxiliaires et par injection de fautes, ont été publiées. Celles-ci présentent généralement une meilleure résistance contre ces deux attaques que les protections imaginées initialement pour contrer une seule de ces deux attaques. En contrepartie, leur coût en surface, consommation et/ou temps d'exécution est généralement très important. Quatre approches combinant les principes du masquage algorithmique et de la redondance ont été proposées récemment et sont détaillées ci-après.

Les auteurs de ParTI [140] proposent de combiner le masquage TI et l'utilisation de codes de détection d'erreurs. Selon les auteurs de [142, 143], les codes de détection d'erreurs utilisés dans ParTI permettent de détecter des fautes aléatoires, mais sont vulnérables face à un attaquant connaissant les codes utilisés et capable d'injecter des fautes précises en fonction de ces codes. En particulier, la détection est limitée aux fautes ne dépassant pas un certain poids de Hamming, c'est-à-dire aux fautes injectées seulement sur un nombre restreint de bits ; les fautes injectées sur un grand nombre de bits du code de détection d'erreurs sont potentiellement indétectables. Une implémentation de ParTI avec un masquage au premier ordre sur l'algorithme LED, synthétisée avec la bibliothèque de cellules UMC 180 nm, montre une surface 2,56 fois supérieure à celle d'une version simplement protégée avec un masquage TI au premier ordre. Le nombre de cycles d'horloges nécessaires pour un chiffrement avec cette implémentation est doublé par rapport à l'algorithme non protégé.

Les travaux publiés dans [141] se basent sur l'approche théorique d'Ishiai *et al.* pour construire des *circuits privés* [131, 155]. La seconde version de ces circuits, notée PC-II pour *Private Circuits II*, fournit une protection théorique à la fois contre les attaques par analyse des canaux auxiliaires et par injection de fautes, en combinant un masquage algorithmique et un encodage des données. Contrairement à la proposition initiale d'Ishiai *et al.* reposant sur des portes logiques idéales et dont la vulnérabilité face aux glitchs a été montrée dans [130], les auteurs de [141] choisissent d'utiliser un masquage TI. La protection contre les fautes est assurée par un encodage différentiel des données, où le bit "0" est représenté par l'encodage "01", et le bit "1" par "10". Cette méthode permet donc de détecter les fautes qui seraient injectées sur un seul bit à la fois. Une implémentation de celle-ci a été effectuée sur l'algorithme PRESENT dans [137], et évaluée en synthèse dans le nœud technologique 45 nm. Les auteurs évaluent la surface nécessaire pour la détection de fautes à 8,75 fois celle du PRESENT masqué. La surface du PRESENT masqué est, elle, environ égale à 2,99 fois celle de l'implémentation non protégée de l'algorithme [156]. Au final, cette implémentation de PC-II sur PRESENT fournit une protection contre les attaques par canaux auxiliaires au premier ordre combinée à une détection des fautes injectées uniquement sur un bit à la fois, pour une surface supérieure à 26 fois celle de l'algorithme non protégé. Bien que le surcoût en consommation de cette protection n'ait pas été évalué, on peut s'attendre à ce que celui-ci soit du même ordre de grandeur que le surcoût en surface, tous les calculs étant effectués en parallèle sur chaque bit masqué et encodé.

Reparaz *et al.* [142] proposent une contre-mesure mixte nommée CAPA (*combined Countermeasure Against Physical Attacks*), fondée sur l'utilisation conjointe de masquage et de codes MAC. Sa sécurité peut être démontrée de manière formelle contre les deux types d'attaques. Les données sont séparées en d parts pour une protection à l'ordre $(d-1)$ contre les attaques par canaux auxiliaires, tandis que la détection des fautes est effectuée avec un code MAC lui-même séparé en d parts et calculé avec une clé MAC renouvelée lors de chaque exécution de l'algorithme protégé. Les auteurs considèrent un modèle d'attaquant capable d'obtenir des informations par canaux auxiliaires sur d_p parts et d'injecter simultanément des fautes arbitraires sur d_f parts, avec la contrainte qu'au moins une des d parts ne doit pas être visée par l'une de ces deux attaques.

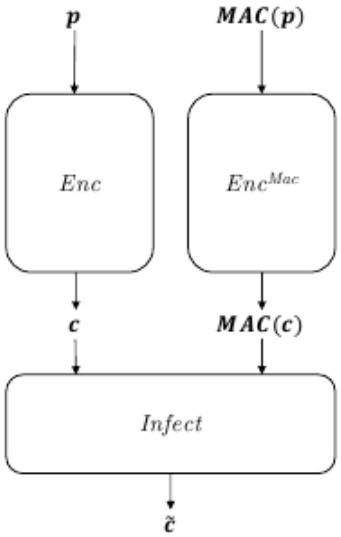


FIGURE 3.13 – Contre-mesure mixte, à base d'un masquage algorithmique des données et de codes MAC [143].

Ils proposent également un second modèle où l'attaquant peut injecter des fautes aléatoires sur n'importe quel nombre de parts avec une certaine probabilité. Pour ces deux modèles, la probabilité de détecter des fautes est égale à $1-2^{-n}$, où n est la taille de la clé MAC utilisée. Le coût de la contre-mesure augmente avec la taille de cette clé. La surface silicium d'une implémentation de CAPA sur l'algorithme KATAN dans le nœud technologique 45 nm, avec une clé MAC de 8 bits et une protection au premier ordre contre les attaques par canaux auxiliaires, est 8,33 fois plus élevée que celle d'une implémentation uniquement protégée avec un masquage au premier ordre. Afin d'obtenir la surface totale de la protection, il faudrait donc multiplier le coût du masquage, qui croît au moins linéairement avec le nombre de parts, à ce chiffre ; on peut donc s'attendre à une surface totale supérieure à 16 fois celle de l'algorithme non protégé en considérant un masquage au premier ordre avec seulement deux parts. En plus de présenter une augmentation significative de la surface, cette protection nécessite un nombre élevé de bits de masques aléatoires. Ainsi, l'implémentation de CAPA sur KATAN requiert 174 nouveaux bits aléatoires par tour de l'algorithme, soit 44196 bits aléatoires au total pour un chiffrement. Ce nombre de bits aléatoires augmente à la fois avec la taille de clé MAC et avec l'ordre de protection. Au final, cette solution est plus robuste que les autres dans la mesure où sa sécurité peut être prouvée formellement à la fois contre les attaques par canaux auxiliaires et par injection de fautes, mais le coût de son implémentation peut être prohibitif dans certains contextes, notamment dans le cas de l'IoT.

Les auteurs de M&M (*Masks and Macs*) [143] combinent l'utilisation de masques et de codes MAC, de même que pour CAPA, mais utilisent une manière plus légère de calculer ces codes MAC, dont la sécurité ne peut pas être démontrée de manière formelle. Contrairement à CAPA, le code MAC est traité en parallèle et indépendamment des données masquées, comme présenté sur la figure 3.13. De plus, au lieu de simplement détecter les attaques, ceux-ci proposent un mécanisme d'*infection*, c'est-à-dire qu'une injection de fautes entraîne la création d'un texte chiffré aléatoire, indépendant de la clé de chiffrement et donc inexploitable pour un attaquant. Le modèle d'attaquant est similaire à celui utilisé dans CAPA, de même que la probabilité de détection de fautes en fonction de la taille de la clé MAC. On notera en particulier la difficulté de détecter des fautes globales, affectant toutes les parts simultanément, pouvant par exemple être injectées en modifiant la tension d'alimentation ou l'horloge. Une évaluation de cette protection sur l'AES dans le nœud technologique 45 nm a été effectuée par les auteurs. Ceux-ci considèrent un masquage à l'ordre d avec $(d+1)$ parts, tel que décrit dans [144]. Pour un masquage au premier ordre, cette méthode requiert l'utilisation de 116 nouveaux bits aléatoires par tour de l'AES, et augmente également le temps d'exécution total. Elle présente un coût silicium de 2,53 fois celui d'une implémentation

uniquement protégée contre les attaques par canaux auxiliaires au premier ordre telle que celle de [144] ; au total, on peut s'attendre à avoir une surface et une consommation de puissance supérieures à 6 fois celles du circuit non protégé.

3.7 Conclusion

Les attaques par analyse des canaux auxiliaires et par injection de fautes visent l'implémentation matérielle de primitives cryptographiques afin d'obtenir des informations sur le fonctionnement de ces algorithmes ou de récupérer des valeurs secrètes comme les clés de chiffrement. Du fait de leur faible coût et du peu d'expertise nécessaire pour les mettre en œuvre, elles constituent une menace importante contre certains circuits intégrés pour l'IoT auxquels l'accès physique est possible par des personnes malintentionnées.

Nous montrons l'efficacité d'une attaque par analyse des canaux auxiliaires contre une primitive cryptographique légère en effectuant avec succès une telle attaque contre une implémentation de l'algorithme de chiffrement par flot Trivium, à partir de 100 traces de consommation de puissance obtenues avec des simulations post-layout. Nous évaluons également la présence de fuites d'information par canaux auxiliaires grâce au T-test de Welch, et nous montrons que de telles fuites sont observables durant l'ensemble de l'exécution de cet algorithme, même si elles ne sont exploitables avec les moyens actuels que durant les 200 premiers cycles d'horloge de la phase d'initialisation de Trivium. Trivium est également vulnérable aux attaques par injection de fautes ; cependant, du fait du biais qui serait introduit par le choix arbitraire d'un modèle de fautes en particulier, nous n'étudions pas l'exploitation de ces fautes à partir de simulations.

De nombreuses protections existent contre l'une ou l'autre de ces attaques matérielles. D'un côté, les protections contre les attaques par analyse des canaux auxiliaires consistent à générer du bruit, lisser la consommation, ou rendre cette consommation aléatoire et donc inexploitable pour un attaquant. La plupart d'entre elles sont vulnérables face à certaines attaques, et il peut donc être nécessaire de combiner plusieurs protections. D'un autre côté, les protections contre les attaques en fautes visent soit les moyens physiques mis en œuvre afin d'injecter des fautes, soit les fautes elles-mêmes une fois celles-ci injectées. Les protections algorithmiques et logiques contre les fautes elle-mêmes reposent sur diverses formes de redondance. Les différents types de redondance sont pour la plupart adaptés à un modèle d'attaques en fautes donné, mais présentent des faiblesses exploitables par un attaquant si les fautes injectées sortent de ce modèle.

Des protections mixtes contre ces deux types d'attaques doivent être envisagées afin de garantir la sécurité d'un circuit intégré. Or, de nombreuses protections conçues contre une seule de ces deux attaques peuvent faciliter l'autre attaque. La conception de contre-mesures mixtes est donc plus complexe que la simple combinaison de protections existantes contre les deux types d'attaques. Plusieurs protections mixtes ont été publiées. Cependant, elles sont toujours vulnérables à certaines injections de fautes, comme les fautes globales qui peuvent être obtenues par modification de la tension d'alimentation ou de l'horloge.

De manière générale, les protections existantes contre les attaques matérielles présentent plusieurs désavantages, parmi lesquels une consommation de puissance élevée, un temps d'exécution rallongé, ou encore un besoin important en bits aléatoires dont la génération elle-même introduit une surconsommation importante. Par conséquent, ces contre-mesures ont une consommation élevée d'énergie, ce qui peut s'avérer problématique pour des circuits intégrés pour l'IoT disposant d'un budget énergétique limité. De plus, certaines contre-mesures, telles que l'utilisation de certains styles de logique spécialisés, demandent un effort d'implémentation très important et la mise en place d'un flot de conception dédié. Au final, la conception de nouvelles contre-mesures, simples à implémenter avec un flot de conception standard, efficaces en énergie et adaptées aux cas d'usage variés de l'IoT, est donc nécessaire afin de concilier la sécurité matérielle, le coût de conception, et la durée de vie des circuits intégrés.

Au cours de nos travaux, nous étudions deux approches différentes afin de répondre à cette problématique. Dans le chapitre 4, nous proposons une nouvelle contre-mesure mixte, avec une

consommation énergétique plus faible que les protections existantes et pouvant être implémentée avec un flot de conception standard. Dans le chapitre 5, nous proposons de modifier l'implémentation de contre-mesures algorithmiques afin d'améliorer leur efficacité énergétique sur le long terme.

Chapitre 4

L'encodage dynamique, une contre-mesure mixte pour les algorithmes à base de registres à décalage

Sommaire

4.1 Encodage dynamique des registres à décalage 76
4.2 Sécurité contre les attaques par injection de fautes 79
4.2.1 Encodage 1-sur-4 80
4.2.2 Encodage 2-sur-4 81
4.2.3 Conclusion sur la détection des fautes 81
4.3 Encodage dynamique aléatoire 82
4.4 Extension à la logique combinatoire 83
4.4.1 Principe 83
4.4.2 Mise en œuvre 83
4.4.3 Encodage dynamique aléatoire de la logique combinatoire 84
4.4.4 Sécurité contre les attaques matérielles 86
4.5 Caractérisation et évaluation de l'encodage dynamique 87
4.5.1 Implémentation de l'encodage dynamique 87
4.5.2 Coût d'implémentation de l'encodage dynamique 90
4.5.3 Résistance aux attaques par canaux auxiliaires 91
4.6 Conclusion 98

Les registres à décalage sont utilisés dans les implémentations de nombreuses primitives cryptographiques conçues pour des circuits contraints en énergie et en surface. Par exemple, plusieurs algorithmes légers de chiffrement par flot, dont Trivium [29] et Grain [31], sont construits à partir de registres à décalage à rétroaction non-linéaire. C'est également le cas de certains algorithmes de chiffrement par bloc tels que KATAN [157], ou encore de fonctions de hachage comme QUARK [158]. Dans toutes ces implémentations, la surface et la consommation dues à la logique combinatoire sont négligeables par rapport à celles des bascules constituant les registres à décalage; par exemple, tandis que Trivium contient 288 bascules pour seulement 3 portes ET et 10 portes XOR, une bascule a une consommation 10,5 fois plus élevée qu'une porte ET et 6,7 plus élevée qu'une porte XOR dans le nœud technologique 0,18 µm, selon [8]. Par conséquent, les attaques matérielles par analyse des canaux auxiliaires effectuées contre ces algorithmes [159, 160, 72, 70, 71] visent uniquement la consommation des bascules, et négligent celle de la logique combinatoire. De même, les attaques en fautes contre ces algorithmes considèrent que l'injection de fautes est réalisée dans les registres eux-mêmes [85, 86, 161, 162, 87]. Or, peu de contre-mesures efficaces contre l'un ou l'autre de ces types d'attaques ont été proposés à ce jour en tenant compte des spécificités des registres à décalage.

D'un autre côté, comme nous l'avons vu dans le chapitre 3, les contre-mesures mixtes, protégeant l'implémentation d'un algorithme à la fois contre les attaques par analyse des canaux auxiliaires et par injection de fautes, sont peu nombreuses, et leur implémentation est généralement très coûteuse, autant du point de vue de la surface qu'en ce qui concerne la consommation, le temps d'exécution et l'utilisation de bits aléatoires. De plus, ces contre-mesures couvrent uniquement certains modèles de fautes, mais sont vulnérables quand les fautes sortent de ces modèles, ce qui est réalisable en pratique pour un attaquant connaissant ces protections. Par exemple, les contre-mesures mixtes fondées sur du masquage, présentées dans le chapitre 3, consistent à représenter les données protégées sur d parts, ou sur d chemins de données redondants. Ces contre-mesures ne permettent pas la détection de fautes injectées simultanément sur ces d chemins de données. Or, de telles fautes globales peuvent être injectées avec un faible coût, par exemple en altérant la tension d'alimentation ou la fréquence d'horloge.

Nous proposons donc une nouvelle contre-mesure, appelée *encodage dynamique*, permettant de contrer à la fois les attaques par injection de fautes et par analyse des canaux auxiliaires. Elle consiste en un lissage algorithmique de la consommation sans précharge, et elle peut être implémentée en utilisant un flot de conception standard. L'encodage dynamique permet une détection intrinsèque des fautes, avec une meilleure couverture des fautes que d'autres contre-mesures mixtes. Cette protection présente une surface et une consommation plus faible que les contre-mesures mixtes existantes; elle ne rallonge pas le temps d'exécution; et elle ne nécessite pas, ou très peu selon les variantes, de bits aléatoires. Elle offre une protection efficace des registres à décalage, mais également de la logique combinatoire, et est donc particulièrement adaptée pour la protection d'algorithmes légers, tels que ceux cités plus haut, dans le cadre de l'IoT.

Dans un premier temps, les principes de l'encodage dynamique sont détaillés uniquement pour la protection de registres à décalages, dans un soucis de simplicité. Une analyse de la sécurité contre les attaques en fautes est fournie, et le rajout d'aléa afin de renforcer cette contre-mesure est proposé. L'extension de cet encodage dynamique à la logique combinatoire est ensuite discutée. Pour conclure, l'implémentation matérielle de plusieurs variantes de l'encodage dynamique est décrite, et une évaluation du coût d'implémentation et de la protection apportée par cette contre-mesure est effectuée.

4.1 Encodage dynamique des registres à décalage

On considère ici un registre à décalage non protégé, avec un état interne S de k bits, notés $S_1, \ldots, S_k$. À chaque cycle d'horloge, les bits de l'état interne sont décalés d'une position vers la sortie (c'est-à-dire vers la droite dans les figures présentées dans nos travaux), le registre prend un bit x en entrée, et produit un bit z en sortie.

Une manière d'assurer une consommation constante dans ce registre à décalage est de transformer chaque bit S_i de son état interne avec une fonction $\mathscr{F}$ en un code de n bits noté $\mathscr{F}(S_i)$, avec $n > 1$. Ces codes doivent respecter les propriétés suivantes :

Propriété 1 *Chaque bit S_i de l'état interne doit être encodé avec le même poids de Hamming, noté HW pour Hamming Weight :* $\exists C \mid \forall i \in \{1,\ldots,k\}, HW(\mathscr{F}(S_i)) = C$.

Propriété 2 *Deux bits consécutifs de l'état interne doivent être encodés avec une distance de Hamming constante, notée HD pour Hamming Distance :*
$\exists C \mid \forall i \in \{1,\ldots,k-1\}, HD(\mathscr{F}(S_i),\mathscr{F}(S_{i+1})) = HW(\mathscr{F}(S_i) + \mathscr{F}(S_{i+1})) = C$.

Cet encodage est mis en œuvre en appliquant la fonction $\mathscr{F}$ au bit d'entrée x du registre à décalage à protéger, à chaque cycle d'horloge. De même, la fonction inverse $\mathscr{F}^{-1}$ est appliqué à la sortie encodée $\mathscr{F}(z)$ afin d'obtenir un bit de sortie z. Ainsi, pour un code $\mathscr{F}(x)$ de n bits, n registres à décalage en parallèle sont nécessaires pour contenir l'état interne encodé. Par la suite, nous notons *part* le contenu de chacun de ces registres à décalage parallèles, et un code $\mathscr{F}(x)$ est donc décomposé en n parts. L'encodage dynamique sur n parts du i-ième bit de l'état interne est noté $\mathscr{F}(S_i) = (S_i^1,\ldots,S_i^n)$, avec $\mathscr{F}(S_i) \in GF(2^n)$ et $S_i^j \in GF(2)$ pour $i \in \{1,\ldots,k\}$ et $j \in \{1,\ldots,n\}$. On notera que ces parts ne doivent pas être confondues avec les parts d'un masquage algorithmique tel que décrit dans la section 3.4.2. Dans les deux cas, plusieurs bits sont utilisés par les contre-mesures pour représenter un seul bit non protégé, mais la façon de calculer ces parts est différente, l'encodage dynamique ne reposant pas sur le principe de *partage de secret*.

Un exemple d'encodage fréquemment utilisé est l'encodage différentiel, où un bit "0" est encodé en "01" et un "1" en "10". Celui-ci assure un poids de Hamming constant pour chaque bit encodé, et donc pour l'ensemble des bits encodés de cette manière. Cependant, cet encodage requiert un autre mécanisme, telle qu'une précharge à "0" ou à "1", afin de garantir également une distance de Hamming constante ; c'est le principe de la logique différentielle avec précharge décrite dans le chapitre 3. Nous appellerons cette manière d'encoder les bits un *encodage statique*, car chaque bit est toujours encodé de la même manière.

Afin d'équilibrer à la fois le poids et la distance de Hamming sans nécessiter de précharge, l'encodage dynamique consiste à encoder différemment deux bits entrants x et y consécutifs. De cette manière, quels que soient x et y et même si ces deux bits sont identiques, les propriétés 1 et 2 peuvent être respectées.

Un tel encodage dynamique doit être effectué sur au minimum 4 bits, ce qui signifie que $\mathscr{F}(x) \in GF(2^n)$, avec $n \geq 4$. En effet, la propriété 2 implique que deux bits identiques consécutifs soient encodés différemment par $\mathscr{F}$; par conséquent, il existe au minimum deux codes différents pour représenter un "0", et deux autres codes pour représenter un "1". Ces quatre codes différents doivent tous avoir le même poids de Hamming d'après la propriété 1. Or, il y a au plus deux codes respectant cette propriété dans $GF(2^2)$, i.e. {10,01}, et trois dans $GF(2^3)$, ceux-ci étant soit {001,010,100}, soit {110,101,011}. Plusieurs encodages dynamiques valides existent dans $GF(2^4)$. Par exemple, avec deux manières d'encoder respectivement les bits "0" et "1" parmi {0001, 0010, 0100, 1000}, et en alternant entre ces deux manières d'encoder ces bits, l'encodage dynamique respecte les propriétés 1 et 2. On appellera cet encodage un *encodage 1-sur-4*. De la même manière, un encodage dynamique peut être réalisé en choisissant judicieusement quatre codes parmi {0011, 1100, 1001, 0110, 1010, 0101}, ce que nous appellerons un *encodage 2-sur-4*. Enfin, un *encodage 3-sur-4* est également possible, en inversant simplement les codes de l'encodage 1-sur-4 ; ces deux approches étant identiques, nous ne détaillerons pas l'encodage 3-sur-4 par la suite.

Le reste de nos travaux porte sur un encodage dynamique dans $GF(2^4)$, qui est le plus petit encodage dynamique possible, et donc le plus léger en surface et en consommation. Pour les mêmes raisons, nous considérons seulement deux manières différentes d'encoder respectivement les bits "0" et "1", soit quatre codes en tout, ce qui est suffisant pour équilibrer le poids et la distance de Hamming. Bien que l'encodage dynamique puisse être réalisé dans $GF(2^n)$ avec $n > 4$, et que l'encodage 2-sur-4 dans $GF(2^4)$ puisse être réalisé avec les six codes {0011, 1100, 1001, 0110, 1010,

TABLEAU 4.1 – Principe de l'encodage dynamique d'un bit x.

	Temps T	**Temps T+1**
x = 1	1000	0100
x = 0	0010	0001

TABLEAU 4.2 – Exemple de table d'encodage dynamique 1-sur-4 d'un bit x.

	c = 0	**c = 1**
x = 1	1000	0100
x = 0	0010	0001

0101}, ces solutions n'apportent a priori pas de gain significatif vis-à-vis de la protection contre les attaques par analyse des canaux auxiliaires.

Un exemple d'encodage dynamique 1-sur-4 est représenté dans le tableau 4.1. Chaque bit est encodé de deux manières différentes, et l'encodage dynamique consiste à alterner à chaque cycle d'horloge entre ces deux manières, donc entre les deux colonnes aux temps T et T+1 (modulo 2).

Une manière d'alterner entre des deux colonnes est l'utilisation d'un signal de contrôle c alternant entre "0" et "1" à chaque cycle d'horloge. Un tel signal peut être généré facilement avec une bascule et un inverseur. Une autre manière serait l'usage d'une machine à états finis, qui sélectionnerait l'encodage au temps T+1 en fonction de l'encodage au temps T. Nous choisissons l'utilisation d'un signal de contrôle alternant entre "0" et "1", moins complexe à implémenter qu'une machine à états finis, et donc moins coûteux en surface et en consommation. En pratique, l'exemple d'encodage donné précédemment dans le tableau 4.1 est donc réalisé en utilisant un bit c selon le tableau 4.2. De la même manière, le tableau 4.3 présente deux types d'encodage 2-sur-4 différents. Par la suite, nous appellerons *table d'encodage* un tel tableau.

La fonction d'encodage dynamique $\mathscr{F}$ prend donc deux entrées, i.e. un bit x à encoder et le signal de contrôle c, afin de générer un encodage dynamique de x. La fonction de décodage $\mathscr{F}^{-1}$, quant à elle, produit le même résultat pour les deux codes représentant le même bit x, quelle que soit la valeur de c. Il serait donc possible d'appliquer cette fonction $\mathscr{F}^{-1}$ uniquement au bit encodé $\mathscr{F}(x)$, indépendamment de c. Cependant, nous choisissons d'utiliser à la fois le bit encodé $\mathscr{F}(x)$ et le signal de contrôle c comme arguments de cette fonction afin d'assurer une meilleure détection des fautes, tel qu'expliqué dans la section 4.2. Comme c alterne entre "0" et "1", à chaque cycle d'horloge, cette fonction $\mathscr{F}^{-1}$ dépend de la taille du registre à décalage, c'est-à-dire du nombre de bascules entre $\mathscr{F}$ et $\mathscr{F}^{-1}$. Si ce nombre est pair, le signal de contrôle c a la même valeur lors de l'encodage et du décodage d'un bit donné, et le décodage est donc effectué avec $\mathscr{F}^{-1}(\mathscr{F}(x), c)$. Si ce nombre est impair, le signal c prend des valeurs complémentaires lors de l'encodage et du décodage de ce bit, et le décodage est donc effectué avec $\mathscr{F}^{-1}(\mathscr{F}(x), \overline{c})$. Par la suite, nous noterons les fonctions d'encodage et de décodage $\mathscr{F}_c$ et $\mathscr{F}_c^{-1}$, indépendamment de la taille du registre à décalage, afin d'alléger les notations. De plus, nous notons *parité* d'un code $\mathscr{F}_c(x)$ la valeur de c utilisée lors de l'encodage avec $\mathscr{F}_c$ d'un bit x. Par exemple, en reprenant l'exemple du tableau 4.2, le code "1000" a une parité de "0" pour représenter le bit "1", tandis que le code "0100" a une parité de "1".

Avec l'encodage 1-sur-4, les bits "0" et "1" sont encodés chacun de deux manières différentes parmi les quatre codes {0001, 0010, 0100, 1000} ; toutes les permutations de ces encodages sont possibles afin de garantir un poids et une distance de Hamming constants. Il existe donc 4! = 24 encodages dynamiques 1-sur-4 différents. En ce qui concerne l'encodage dynamique 2-sur-4, il n'est pas possible de choisir toutes les combinaisons de quatre codes parmi {0011, 1100, 1001, 0110, 1010, 0101} tout en respectant la propriété 2. En effet, il existe trois paires de codes complémentaires parmi les six codes possibles : {0011, 1100}, {1001, 0110} et {1010, 0101}. La distance de Hamming est de 4 entre deux codes complémentaires, et de 2 entre deux codes non complémentaires. Comme il n'existe que 6 codes possibles pour l'encodage dynamique 2-sur-4, au moins deux codes parmi les 4 sélectionnés doivent être complémentaires. Afin de s'assurer que la dis-

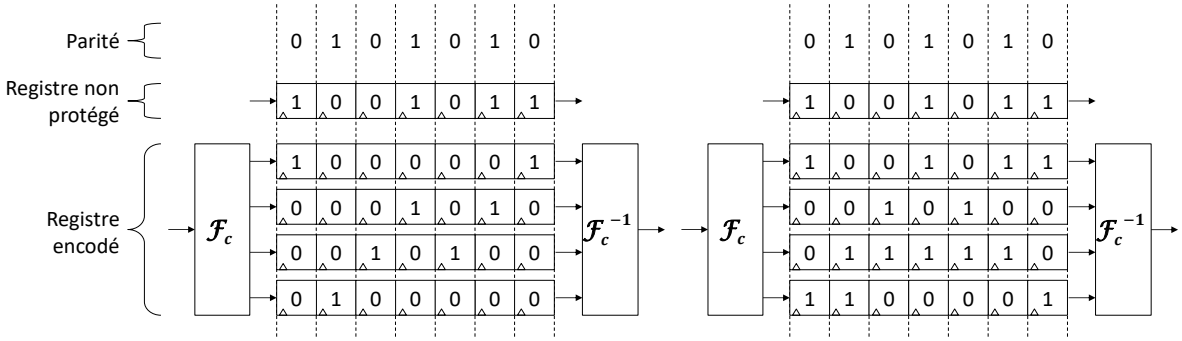
TABLEAU 4.3 – Deux exemples de tables d'encodage dynamique 2-sur-4 d'un bit x.

(a)

	c = 0	**c = 1**
x = 1	1001	1010
x = 0	0110	0011

(b)

	c = 0	**c = 1**
x = 1	0101	0011
x = 0	1010	1100



(a) Encodage 1-sur-4 (b) Encodage 2-sur-4

FIGURE 4.1 – Exemples d'un état interne encodé avec un encodage dynamique 1-sur-4 et un encodage dynamique 2-sur-4.

tance de Hamming soit toujours égale à 2, il est possible d'avoir deux codes complémentaires uniquement pour une même valeur de c, car deux bits consécutifs sont encodés avec des valeurs de c différentes. Au total, il existe 144 encodages dynamiques 2-sur-4 valides, dont deux exemples sont donnés dans les tableaux 4.3a et 4.3b.

Un exemple de registre à décalage non protégé avec un état interne arbitraire, et d'un registre à décalage avec le même état interne protégé avec un encodage dynamique 1-sur-4, est donné sur la figure 4.1. Cette figure présente également un exemple du même registre encodé avec un encodage 2-sur-4. De manière générale, un encodage dynamique 1-sur-4 d'un registre à décalage de longueur k a un poids de Hamming constant de k, et une distance de Hamming constante de $2k$. Un encodage dynamique 2-sur-4 a toujours un poids de Hamming égal à $2k$, et une distance de Hamming également égale à $2k$.

4.2 Sécurité contre les attaques par injection de fautes

La détection de fautes avec l'encodage dynamique est effectuée lors du décodage avec la fonction $\mathscr{F}_c^{-1}$ en sortie du registre protégé. Lorsqu'un code ne peut pas être décodé par la fonction $\mathscr{F}_c^{-1}$, c'est-à-dire lorsqu'il n'est pas parmi les deux codes possibles pour une parité donnée, cela signifie qu'une faute a été injectée et la fonction $\mathscr{F}_c^{-1}$ envoie un signal d'alerte. Par la suite, on considère qu'une faute est injectée avec succès sur un registre à décalage si cette faute n'est pas détectée lors du décodage.

Cette section aborde la sécurité de l'encodage dynamique d'un registre à décalage contre les attaques par injection de fautes, selon plusieurs modèles de fautes et pour les encodages 1-sur-4 et 2-sur-4. On considère plusieurs types de fautes, listés ci-dessous et dont le détail est donné dans la section 3.3 :

— Les fautes en *commutation* au cours desquelles tous les bits ciblés changent de valeur, et les fautes consistant en une *mise à "0"* ou une *mise à "1"* d'un certain nombre de bits.

— Les fautes *permanentes*, et les fautes *éphémères*. On distingue notamment les fautes *éphémères* en fonction de leur précision temporelle : celles-ci sont soit *temporellement précises*, soit *temporellement imprécises*.

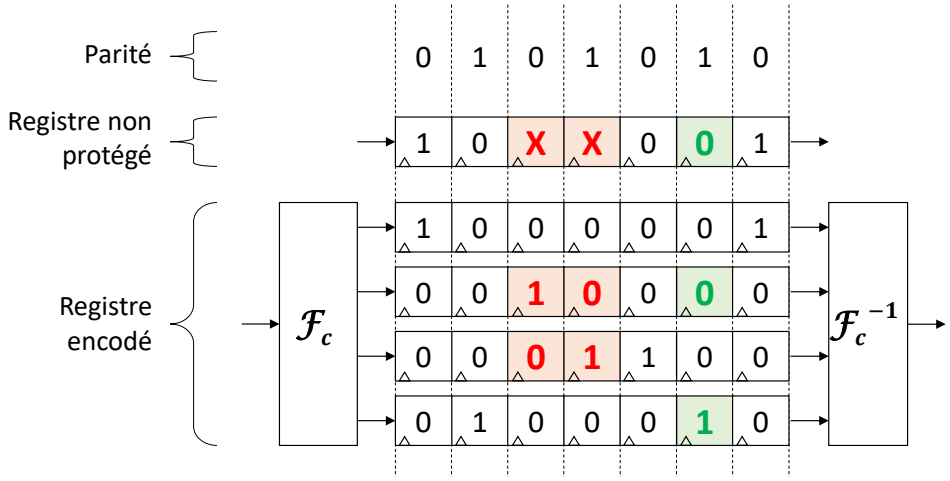


FIGURE 4.2 – Exemples de fautes injectées sur exactement deux parts parmi quatre, pour un encodage 1-sur-4.

— Les fautes sur *un seul bit* ou sur *plusieurs bits consécutifs*, au sein d'un même registre, c'est-à-dire d'une même part.

— Les fautes sur *une seule part* ou sur *plusieurs parts*, c'est-à-dire affectant simultanément soit un seul, soit plusieurs registres parmi les quatre registres en parallèle.

Dans un registre à décalage, l'état interne est décalé d'un bit à chaque cycle d'horloge. L'injection d'une faute *temporellement imprécise* sur une bascule, c'est-à-dire dont l'injection dure plusieurs cycles d'horloge, affectera donc tous les bits successivement stockés dans cette bascule durant toute la durée de l'injection. Par conséquent, une telle faute est fonctionnellement équivalente à une faute *temporellement précise* sur *plusieurs bits consécutifs* au sein d'un même registre, c'est-à-dire au sein d'une même part de l'encodage. Pour cette raison, la détection des fautes *temporellement précises* et *temporellement imprécises* par l'encodage dynamique est similaire à la détection de fautes sur *un seul bit* ou sur *plusieurs bits consécutifs* au sein d'un même registre, et nous ne décrirons donc pas la détection des fautes en fonction de la précision spatiale de l'injection de ces fautes.

Pour les deux types d'encodage, une faute injectée avec succès doit résulter en un code ayant le même poids de Hamming que le code original, non fauté. Cela signifie qu'une commutation de "0" vers "1" doit être compensée par une commutation de "1" vers "0", et inversement. Par conséquent, les fautes consistant uniquement en une *mise à "0"* ou une *mise à "1"* de certains bits sont détectées par $\mathcal{F}_c^{-1}$, et seules les fautes en *commutation* sur un nombre pair de bits peuvent être injectées avec succès.

4.2.1 Encodage 1-sur-4

Afin d'injecter avec succès une faute sur un bit encodé avec un encodage 1-sur-4, il faut faire commuter 2 parts exactement parmi les 4 parts : l'une doit commuter de "0" vers "1", et l'autre de "1" vers "0". Une telle faute est représentée en vert sur la figure 4.2. Toute faute affectant une seule part, ou trois ou quatre parts, résulterait en un état impossible à décoder par $\mathcal{F}_c^{-1}$ et serait donc détectée. De plus, l'attaquant doit savoir sur quelles parts injecter les fautes parmi les 4 parts possibles ; il doit connaître pour cela l'encodage utilisé, mais également la valeur du signal de contrôle c.

Comme c alterne entre "0" et "1" à chaque cycle d'horloge, l'injection de fautes avec succès sur deux codes consécutifs doit viser des parts différentes pour ces deux codes. Par conséquent, les fautes injectées sur deux bits consécutifs (ou plus) d'une même part, c'est-à-dire d'un même registre à décalage, sont détectées par $\mathcal{F}_c^{-1}$. Nous illustrons ceci par un exemple, en nous appuyant sur l'encodage donné dans le tableau 4.2 et le registre encodé représenté sur la figure 4.1 ; cet

exemple peut être généralisé à tous les encodages dynamiques 1-sur-4, ceux-ci étant tous équivalents. Dans cet exemple, un bit "1" peut être encodé par "1000" ou "0100" ; si la fonction $\mathscr{F}_c^{-1}$ ne prenait pas en compte le signal de contrôle c, ces codes seraient équivalents et interchangeables, et seraient donc décodés de la même manière. Dans ce cas, une faute injectée sur les troisième et quatrième bits de la seconde et de la troisième part, telle que représentée en rouge sur la figure 4.2, résulterait en un encodage valide. Or, si la fonction $\mathscr{F}_c^{-1}$ prend bien c en argument, le code "0100" n'est pas permis lorsque c vaut "0". Cet état est donc détecté comme étant une faute, de même que le code "0010" lorsque c vaut "1".

4.2.2 Encodage 2-sur-4

Nous avons souligné précédemment que deux des quatre codes d'un encodage dynamique 2-sur-4 sont forcément complémentaires, et que ces deux codes complémentaires doivent encoder respectivement un "0" et un "1" pour une même parité, c'est-à-dire une même valeur de c. Deux cas se présentent alors pour les deux autres codes, pour l'autre valeur de c : ceux-ci peuvent être complémentaires également, ce qui correspond au cas du tableau 4.3b, ou non complémentaires comme dans le tableau 4.3a. L'injection de fautes dépend de ces deux cas, qui sont détaillés ci-dessous.

Cas 1 : la table d'encodage contient deux paires de codes complémentaires, pour les deux parités. Quelle que soit la parité, les codes représentant un "0" et un "1" sont complémentaires. Par conséquent, une injection de fautes avec succès revient à inverser tous les bits d'un ou plusieurs codes. Les fautes sur plusieurs bits consécutifs sont donc indétectables, du moment qu'elles affectent le même nombre de bits sur les 4 parts simultanément. La détection de fautes dans ce cas est moins robuste que celle décrite précédemment pour l'encodage 1-sur-4, et ce choix de codes pour un encodage 2-sur-4 doit donc être évité.

Cas 2 : la table d'encodage contient seulement une paire de codes complémentaires, pour une parité donnée. Dans ce cas, l'injection de fautes avec succès sur les deux codes complémentaires doit se faire sur les quatre parts de ces codes. L'injection de fautes sur les deux autres codes, non complémentaires, doit se faire sur exactement deux parts parmi les quatre. L'encodage étant effectué alternativement avec ces deux paires de codes, une faute injectée avec succès sur deux bits encodés consécutifs devrait viser alternativement deux et quatre parts. Par conséquent, les fautes injectées sur plusieurs bits consécutifs, sur un nombre de parts fixé, sont détectées par $\mathscr{F}_c^{-1}$.

4.2.3 Conclusion sur la détection des fautes

Seules les fautes *temporellement précises*, en *commutation*, sur *un seul bit encodé consécutif* et sur exactement *deux parts* de ce bit encodé, ne sont pas détectées par un encodage dynamique 1-sur-4. De telles injections de fautes requièrent un matériel coûteux, une connaissance précise de l'implémentation matérielle de l'algorithme ciblé, et une connaissance de l'état de l'encodage à un instant donné. Le complexité d'une telle attaque est très élevée, et celle-ci est difficilement réalisable en pratique.

Avec un bon choix de l'encodage dynamique 2-sur-4, les fautes non détectables sont les fautes *temporellement précises*, en *commutation*, sur *un seul bit encodé consécutif* et sur exactement *deux parts* ou *quatre parts* selon la parité de ce bit encodé. Avec la possibilité d'injecter des fautes soit sur exactement 2 parts parmi 4, soit sur l'ensemble des 4 parts, le modèle d'attaque est légèrement moins restrictif que pour l'encodage 1-sur-4, mais la complexité d'une telle attaque reste très élevée.

D'après le chapitre 3, les contre-mesures mixtes fondées sur du masquage sont vulnérables face aux injections de fautes globales, qui visent un nombre élevé de bits, et qui peuvent être mises en œuvre à bas coût, par exemple avec des glitchs d'horloge ou de tension. En effet, l'implémentation de PC-II [141] est fondée sur un encodage différentiel statique, et l'inversion des deux bits de cet encodage n'est pas détectable ; ParT [140] permet de détecter les fautes injectées sur un code de détection d'erreurs uniquement si ces fautes ne portent que sur un nombre restreint de bits ; et

CAPA [142] et M&M [143] ne sont capables de détecter que les fautes injectées sur au plus $(d-1)$ parts parmi d.

De manière générale, l'encodage différentiel statique sur deux bits, sur lequel reposent les styles de logique différentielle avec précharge, est vulnérable face à l'injection de fautes sur les deux chemins de données de l'encodage différentiel. De la même manière, la redondance spatiale ne permet pas de détecter une même faute injectée dans tous les blocs logiques redondants ; cette faute peut être sur *un ou plusieurs bits*, et être temporellement *précise* ou *imprécise*. La redondance temporelle est vulnérable face aux fautes identiques injectées à deux instants différents, et aux fautes *permanentes* ; ces fautes peuvent également être injectées *sur un seul ou sur plusieurs bits consécutifs*. Enfin, la couverture des fautes par la redondance d'information dépend du code de détection d'erreurs utilisé mais est généralement inférieure à celle des autres formes de redondance.

Nous en concluons que, compte tenu des moyens d'injection actuels (détaillés dans le chapitre 3), l'injection de fautes non détectables contre l'encodage dynamique est plus complexe à mettre en œuvre que contre les autres protections évoquées ci-dessus, notamment en ce qui concerne les protections mixtes. Cette conclusion est valable pour l'encodage 1-sur-4, mais également avec un bon choix de codes pour l'encodage 2-sur-4.

Enfin, les attaques *safe-error*, exploitant le fait d'observer si le résultat d'un algorithme est modifié ou non après une injection de fautes, n'ont pas été détaillées ici. Par exemple, la mise à "0" de certains bits déjà à "0" dans l'une des parts permettrait à un attaquant d'obtenir des informations utiles sur l'état de l'encodage. En pratique, l'exécution d'une telle attaque contre l'encodage dynamique est complexe, et nécessite qu'un attaquant connaisse précisément l'implémentation physique du circuit, mais également que cet attaquant puisse injecter des fautes totalement maîtrisées sur un nombre pré-déterminé de bits.

4.3 Encodage dynamique aléatoire

L'encodage dynamique décrit jusqu'à présent consiste à assurer un poids et une distance de Hamming constants pour chaque bit encodé, et donc également pour l'ensemble du registre. Cela permet donc d'avoir une consommation théorique identique à chaque cycle d'horloge. Or, dans une implémentation réelle sur silicium, les temps de propagation des signaux et les variations de la consommation ne sont pas strictement identiques pour toutes les parts. On peut donc s'attendre à observer des déséquilibres entre ces parts, potentiellement exploitables par un attaquant de la même manière que pour le lissage de la consommation avec divers styles de logique différentielle avec précharge [110, 111].

Pour compenser ces effets, nous proposons de rendre aléatoires ces fuites d'information potentielles, en réalisant un encodage dynamique *aléatoire*, différent lors de chaque exécution. Pour cela, les différentes combinaisons de codes possibles pour l'encodage dynamique sont exploitées. En effet, selon la section 4.1, il existe 24 tables d'encodage 1-sur-4 différentes, et 144 manières de réaliser un encodage 2-sur-4. Pour chaque type d'encodage, le choix des quatre codes peut être effectué aléatoirement à chaque initialisation du registre à décalage. Ce choix nécessite seulement 5 bits aléatoires pour un encodage 1-sur-4, et 8 bits pour un encodage 2-sur-4. Il serait également possible de varier entre ces deux types d'encodage à chaque initialisation, pour un nombre total de 168 combinaisons possibles. Cette dernière solution requiert également 8 bits aléatoires à chaque exécution.

Par exemple, pour un encodage dynamique 1-sur-4 aléatoire, les codes "1000" et "0100" pourraient, selon les exécutions, encoder chacun soit un "0", soit un "1". Un attaquant capable de détecter la commutation entre ces deux codes, en exploitant les déséquilibres de l'implémentation physique, ne pourrait pas exploiter cette information afin de retrouver la valeur originale des bits non encodés sans avoir connaissance par ailleurs de la table d'encodage générée lors de chaque exécution.

Cet encodage dynamique aléatoire permet également de renforcer la sécurité contre les at-

taques par injection de fautes. L'attaquant doit connaître l'encodage utilisé afin de sélectionner les parts sur lesquelles injecter les fautes ; si cet encodage n'est pas fixé, mais est généré aléatoirement à chaque nouvelle exécution, l'attaquant doit disposer de moyens supplémentaires de connaître la valeur de l'encodage, ou choisir les parts de manière aléatoire. De plus, les attaques en fautes ne sont alors pas reproductibles d'une exécution sur l'autre. Enfin, l'alternance aléatoire entre un encodage 1-sur-4 et un encodage 2-sur-4 nécessiterait que l'attaquant soit à la fois capable d'injecter des fautes sur exactement deux parts parmi les quatre, et sur les quatre parts.

Cet encodage dynamique aléatoire peut être mis en œuvre avec une complexité raisonnable. Le choix de la table d'encodage à utiliser pour chaque exécution est effectué une seule fois lors de l'initialisation de l'algorithme protégé. Cette table d'encodage peut être générée à chaque initialisation, ou choisie parmi un certain nombre de tables préalablement stockées dans le circuit. La première méthode requiert une surface bien plus faible que le stockage de 24, 144 ou 168 tables (selon le type d'encodage implémenté de manière aléatoire), et c'est donc la solution retenue. Ensuite, l'encodage avec $\mathscr{F}_c$ et le décodage avec $\mathscr{F}_c^{-1}$, respectivement en entrée et en sortie du registre protégé, sont effectués en fonction des quatre codes contenus dans cette table, plutôt qu'avec des codes constants à chaque exécution. Les quatre registres en parallèle dans lesquels sont contenues les quatre parts de l'encodage dynamique ne sont pas modifiés pour l'implémentation de l'encodage aléatoire. Au final, le surcoût en complexité, surface et consommation lié à l'encodage dynamique aléatoire est principalement lié à la génération d'une nouvelle table d'encodage à chaque exécution.

4.4 Extension à la logique combinatoire

4.4.1 Principe

L'encodage dynamique a été présenté pour la protection des registres à décalage. Nous détaillons ici comment appliquer cet encodage dynamique à la logique combinatoire, afin de protéger celle-ci à la fois contre les attaques par analyse des canaux auxiliaires et par injection de fautes. Dans le cas où l'encodage dynamique est appliqué à la fois à la logique séquentielle et à la logique combinatoire, les calculs sont effectués directement sur les signaux encodés, et l'ensemble de l'implémentation d'un algorithme peut donc être protégé sans nécessiter l'utilisation des fonctions $\mathscr{F}_c$ et $\mathscr{F}_c^{-1}$ pour encoder et décoder les données en entrée et sortie des registres.

On définit un bloc combinatoire, ou fonction combinatoire, comme une combinaison de plusieurs portes logiques élémentaires. Avec des opérations effectuées sur des bits dans le corps de Galois GF(2), toutes les fonctions combinatoires peuvent être décrites en utilisant les portes logiques élémentaires ET, XOR, et INV, correspondant respectivement à la multiplication, l'addition, et l'inversion.

On considère qu'un bloc combinatoire protégé par l'encodage dynamique traite des signaux d'entrée encodés dynamiquement ; ceux-ci proviennent de logique séquentielle encodée, ou sont encodés préalablement avec la fonction $\mathscr{F}_c$. Ce bloc combinatoire encodé produit des signaux de sortie encodés de la même manière que les signaux d'entrée, ayant un poids et une distance de Hamming constants quels que soient les signaux d'entrée et la parité de ces signaux. Pour cela, chaque porte logique élémentaire constituant le bloc combinatoire est modifiée afin de traiter des entrées encodées sur 4 bits, et de produire des sorties encodées de la même manière. Dans le but de permettre une intégration aisée de cette contre-mesure dans un flot de conception classique, l'encodage des portes logiques élémentaires est réalisé au niveau algorithmique et peut être décrit par des équations utilisant elles-mêmes exclusivement une combinaison de portes ET et XOR.

4.4.2 Mise en œuvre

L'encodage dynamique de la logique combinatoire est spécifique à l'encodage choisi, c'est-à-dire à une table d'encodage donnée. Nous détaillons ici la façon d'encoder la logique combinatoire avec la table d'encodage 4.2, correspondant à un encodage dynamique 1-sur-4. On considère des

portes logiques prenant des entrées x et y, et produisant une sortie z. Ces signaux sont encodés sur quatre parts notées (w^1, w^2, w^3, w^4) pour un code quelconque $\mathscr{F}_c(w)$. Les équations des portes logiques encodées sont données ci-dessous.

— **Porte ET :**

$$\begin{aligned} z^1 &= x^1(y^1 + y^2) \\ z^2 &= x^2(y^1 + y^2) \\ z^3 &= x^3 + x^1(y^3 + y^4) \\ z^4 &= x^4 + x^2(y^3 + y^4) \end{aligned} \tag{4.1}$$

— **Porte XOR :**

$$\begin{aligned} z^1 &= x^1(y^3 + y^4) + x^3(y^1 + y^2) \\ z^2 &= x^2(y^3 + y^4) + x^4(y^1 + y^2) \\ z^3 &= x^1(y^1 + y^2) + x^3(y^3 + y^4) \\ z^4 &= x^2(y^1 + y^2) + x^4(y^3 + y^4) \end{aligned} \tag{4.2}$$

— **Porte INV :**

$$\begin{aligned} z^1 &= x^3 \\ z^2 &= x^4 \\ z^3 &= x^1 \\ z^4 &= x^2 \end{aligned} \tag{4.3}$$

Les tables de vérité correspondant à chacune de ces équations sont données sur la figure 4.3. Deux codes $\mathscr{F}_c(x)$ (respectivement, $\mathscr{F}_c(y)$) permettent d'encoder chaque valeur de x (resp. y) en fonction de la valeur du signal de contrôle c. Un signal de sortie $\mathscr{F}_c(z)$ est calculé pour chaque paire $(\mathscr{F}_c(x), \mathscr{F}_c(y))$ pour les portes ET et XOR, et pour chaque valeur de $\mathscr{F}_c(x)$ pour la porte INV. On constate que la fonctionnalité de ces portes logiques est préservée : la sortie z est correcte, quels que soient x et y. On remarque également qu'avec l'implémentation choisie pour l'encodage de la logique combinatoire, la parité de $\mathscr{F}_c(z)$ est imposée par celle de $\mathscr{F}_c(x)$. Autrement dit, $\mathscr{F}_c(z)$ a toujours la même parité que $\mathscr{F}_c(x)$, indépendamment de celle de $\mathscr{F}_c(y)$. Comme $\mathscr{F}_c(x)$ est encodé dynamiquement par hypothèse, $\mathscr{F}_c(z)$ l'est également, et deux valeurs consécutives de $\mathscr{F}_c(z)$ ont donc une distance et un poids de Hamming constants. Ainsi, n'importe quelle fonction combinatoire, construite comme une combinaison de ces trois portes logiques, présentera une sortie encodée dynamiquement si ses entrées sont encodées dynamiquement, quelles que soient les parités de ces entrées.

On note *parité relative* de deux signaux encodés $\mathscr{F}_c(x)$ et $\mathscr{F}_c(y)$ la différence de parité de ces deux codes : ceux-ci ont soit la même parité, soit une parité opposée. Les équations (4.1) et (4.2), décrivant l'encodage dynamique des portes ET et XOR, sont valables quelle que soit la parité relative de $\mathscr{F}_c(x)$ et $\mathscr{F}_c(y)$, c'est-à-dire qu'elles décrivent à la fois le cas où les parités de ces signaux sont égales, et le cas où elles sont opposées. Or, dans un circuit donné, la parité relative de deux signaux encodés est généralement constante : les parités de ces deux signaux changent simultanément, à chaque cycle d'horloge. Certains termes de ces équations sont donc superflus, et peuvent être éliminés en tenant compte de la parité relative de $\mathscr{F}_c(x)$ et $\mathscr{F}_c(y)$. Par exemple, si $\mathscr{F}_c(x)$ et $\mathscr{F}_c(y)$ ont la même parité, les termes x^1y^2 et x^2y^1 peuvent être éliminés, car ils correspondent à des opérations effectuées sur des signaux de parités différentes. La version simplifiée de ces équations, en fonction des parités respectives des signaux d'entrée, est donnée dans le tableau 4.4.

4.4.3 Encodage dynamique aléatoire de la logique combinatoire

Les équations (4.1) à (4.3) correspondent à un encodage dynamique de la logique combinatoire avec une table d'encodage donnée ; elles doivent être adaptées si une table d'encodage différente est choisie. Un encodage dynamique aléatoire de la logique combinatoire nécessiterait donc

Porte ET					
x	$\mathscr{F}_c(x)$	**y**	$\mathscr{F}_c(y)$	$\mathscr{F}_c(z)$	**z**
0	0001	0	0001	0001	0
			0010	0001	
		1	0100	0001	0
			1000	0001	
	0010	0	0001	0010	0
			0010	0010	
		1	0100	0010	0
			1000	0010	
1	0100	0	0001	0001	0
			0010	0001	
		1	0100	0100	1
			1000	0100	
	1000	0	0001	0010	0
			0010	0010	
		1	0100	1000	1
			1000	1000	

(a)

Porte XOR					
x	$\mathscr{F}_c(x)$	**y**	$\mathscr{F}_c(y)$	$\mathscr{F}_c(z)$	**z**
0	0001	0	0001	0001	0
			0010	0001	
		1	0100	0100	1
			1000	0100	
	0010	0	0001	0010	0
			0010	0010	
		1	0100	1000	1
			1000	1000	
1	0100	0	0001	0100	1
			0010	0100	
		1	0100	0001	0
			1000	0001	
	1000	0	0001	1000	1
			0010	1000	
		1	0100	0010	0
			1000	0010	

(b)

Porte INV			
x	$\mathscr{F}_c(x)$	$\mathscr{F}_c(z)$	**z**
0	0001	0100	1
	0010	1000	
1	0100	0001	0
	1000	0010	

(c)

FIGURE 4.3 – Tables de vérités pour les fonctions ET, XOR et INV encodées.

l'implémentation en parallèle de plusieurs versions d'une même fonction combinatoire, encodée avec plusieurs tables d'encodage différentes. Cela signifie que la mise en place d'un tel encodage aléatoire de la logique combinatoire introduirait un surcoût important en surface par rapport à la logique non protégée. Toutefois, ce surcoût doit être mis en perspective par rapport à la proportion de logique combinatoire dans le circuit. Ainsi, nous avons expliqué dans l'introduction de ce chapitre que pour de nombreuses primitives cryptographiques légères, la surface et la consommation dues à la logique combinatoire sont négligeables par rapport à celles des registres à décalage. Rapporté à l'ensemble de l'implémentation de l'algorithme protégé, le coût en surface de l'encodage dynamique aléatoire de la logique combinatoire pourrait donc être également négligeable, ou du moins acceptable.

Les équations données dans cette section ont été établies manuellement; une permutation des parts pourrait permettre d'implémenter d'autres encodages 1-sur-4, mais il n'existe à ce jour pas de méthode permettant de générer automatiquement ces équations pour n'importe quelle table d'encodage. Cela augmente la complexité de l'implémentation d'un encodage dynamique aléatoire de la logique combinatoire, les équations pour chaque version de la fonction protégée devant préalablement être établies.

Il serait également possible d'implémenter l'ensemble de la logique combinatoire encodée avec des multiplexeurs dépendant de la table d'encodage choisie, ce qui faciliterait l'implémentation de l'encodage dynamique aléatoire de la logique combinatoire. Cette solution, qui aurait également un coût élevé en surface, se rapproche du concept de logique programmable et de l'architecture des FPGA. Enfin, sur FPGA, on peut envisager de tirer partie de la reconfiguration

TABLEAU 4.4 – Équations simplifiées pour les portes ET et XOR, en fonction de la parité relative des entrées $\mathscr{F}_c(x)$ et $\mathscr{F}_c(y)$.

	Porte ET	**Porte XOR**
Parités identiques	$z^1 = x^1 y^1$ $z^2 = x^2 y^2$ $z^3 = x^3 + x^1 y^3$ $z^4 = x^4 + x^2 y^4$	$z^1 = x^1 y^3 + x^3 y^1$ $z^2 = x^2 y^4 + x^4 y^2$ $z^3 = x^1 y^1 + x^3 y^3$ $z^4 = x^2 y^2 + x^4 y^4$
Parités différentes	$z^1 = x^1 y^2$ $z^2 = x^2 y^1$ $z^3 = x^3 + x^1 y^4$ $z^4 = x^4 + x^2 y^3$	$z^1 = x^1 y^4 + x^3 y^2$ $z^2 = x^2 y^3 + x^4 y^1$ $z^3 = x^1 y^2 + x^3 y^4$ $z^4 = x^2 y^1 + x^4 y^3$

partielle afin d'implémenter efficacement un encodage dynamique aléatoire à la fois de la logique séquentielle et combinatoire, ce qui sort du cadre de notre étude.

4.4.4 Sécurité contre les attaques matérielles

Sécurité contre les attaques par analyse des canaux auxiliaires

Les entrées et les sorties du bloc combinatoire ont à la fois un poids et une distance de Hamming constants, ce qui garantit un lissage de la consommation théorique de ces signaux. Cependant, dans une implémentation sur silicium, les différents chemins de données dans chaque bloc combinatoire sont déséquilibrés, comme cela a été décrit dans la section 4.3 pour la protection de la logique séquentielle. L'encodage de la logique combinatoire est a priori également sensible aux glitchs et à l'évaluation prématurée des signaux, qui sont des phénomènes déjà utilisés pour attaquer certains styles de logique tels que le WDDL. L'implémentation d'un encodage dynamique aléatoire de la logique combinatoire permettrait de compenser ces déséquilibres et d'empêcher leur exploitation par un attaquant, au prix d'une augmentation du coût silicium et des efforts fournis lors de la conception.

Sécurité contre les attaques par injection de fautes

Le modèle d'injection de fautes est beaucoup plus complexe en ce qui concerne la logique combinatoire encodée que pour la logique séquentielle. On obtient des effets différents selon que les fautes sont injectées sur les signaux en entrée des portes logiques encodées ou à l'intérieur de celles-ci. De plus, ces fautes peuvent être injectées sur un nombre quelconque de parts, sur un ou plusieurs signaux encodés. Enfin, ces fautes dépendent également de la table d'encodage utilisée, et de la combinaison particulière de plusieurs portes logiques élémentaires encodées permettant de réaliser un bloc combinatoire donné. L'étude exhaustive de l'ensemble des fautes pouvant être injectées sur de la logique combinatoire encodée est hors du champs de nos travaux.

Dans le cas particulier présenté ci-dessus, avec les équations du tableau 4.4, l'injection de fautes non détectables doit être effectuée sur plusieurs bits particuliers d'un ou des deux signaux d'entrée, mais pas sur l'ensemble de ces bits. Selon les cas, certaines fautes peuvent se propager et s'amplifier, tandis que d'autres sont annulées au sein des portes logiques encodées. Par exemple, les fautes modifiant la parité de $\mathscr{F}_c(y)$ n'ont aucun effet sur celle de $\mathscr{F}_c(z)$. D'une manière générale, seule une partie des fautes sont transmises à la sortie du bloc combinatoire, et parmi celles-ci, seulement quelques-unes ne peuvent pas être détectées. En effet, avec un encodage sur 4 bits, seul un état sur les 16 possibles en sortie du bloc combinatoire correspond à une faute valide. Afin d'injecter une faute avec succès, l'attaquant devrait non seulement connaître l'implémentation exacte, mais également la parité et la valeur des différents signaux ; de plus, il lui faudrait maîtriser

parfaitement l'emplacement et la taille des fautes injectées. L'injection de fautes avec un tel niveau de précision est très complexe à mettre en œuvre, et la meilleure stratégie reste probablement l'injection aléatoire, en espérant que certaines fautes ne soient pas détectées.

4.5 Caractérisation et évaluation de l'encodage dynamique

Dans cette section, nous évaluons le coût d'implémentation et la résistance aux attaques par canaux auxiliaires de l'encodage dynamique à travers deux cas d'étude. Tout d'abord, l'encodage dynamique est implémenté sur un simple registre à décalage, afin d'étudier l'efficacité de cette protection lorsqu'elle est appliquée uniquement à de la logique séquentielle. Dans un second temps, nous étudions un cas plus représentatif d'un circuit réel, contenant à la fois de la logique séquentielle et combinatoire ; pour cela, nous détaillons l'application de l'encodage dynamique à l'algorithme de chiffrement par flot Trivium, qui est un exemple typique de primitive cryptographique légère, adaptée pour une utilisation dans l'IoT.

Cette étude du coût et de l'efficacité de l'encodage dynamique est effectuée de la même manière que l'étude sur le Trivium non protégé décrite dans la section 3.2. Elle repose sur des résultats de synthèse et des simulations post-layout pour une implémentation matérielle sur ASIC 28 nm FDSOI.

4.5.1 Implémentation de l'encodage dynamique

Pour toutes les implémentations de l'encodage dynamique décrites dans cette section, un signal de contrôle c alternant entre "0" et "1" à chaque cycle d'horloge est utilisé pour encoder différemment deux bits consécutifs, comme expliqué dans la section 4.1. Ce signal de contrôle varie périodiquement, de manière toujours identique et prévisible, et il ne fait pas partie des fuites d'informations utiles par canaux auxiliaires, c'est-à-dire des fuites exploitables par un attaquant pour récupérer une clé de chiffrement par exemple. La génération de ce signal crée donc du bruit qui pourrait masquer partiellement les fuites exploitables. Par conséquent, la bascule et l'inverseur permettant de le générer ont été exclues des implémentations étudiées, et le signal de contrôle c est généré dans le banc de test utilisé lors des simulations post-layout de ces implémentations.

Premier cas d'étude : registre à décalage

Nous détaillons ici l'implémentation de l'encodage dynamique sur un simple registre à décalage de 128 bits, afin d'étudier uniquement l'encodage de la logique séquentielle. Celui-ci ne comporte pas de rétroaction et se compose de 128 bascules chaînées. Afin de reprendre la terminologie utilisée dans le cas de Trivium, on considère que lors de l'initialisation du registre à décalage préalable à une simulation de son comportement, un vecteur d'initialisation de 128 bits, ou IV pour *Initialization Vector*, est chargé dans ce registre.

Dans le but d'éliminer le bruit et les signaux liés à la logique combinatoire, l'ensemble de la logique de contrôle, y compris l'encodage avec $\mathscr{F}_c$ et le décodage avec $\mathscr{F}_c^{-1}$ en entrée et en sortie du registre encodé, est exclue de cette étude. Par conséquent, les signaux de contrôle et d'initialisation, ainsi que les entrées encodées du registre à décalage protégé, sont générés au sein du banc de test qui est utilisé lors des simulations post-layout. L'implémentation physique du registre à décalage encodé consiste uniquement en quatre registre à décalage en parallèle, dont les entrées encodées sont fournies par le banc de test. La table d'encodage utilisée est définie dans le banc de test, et on peut choisir d'utiliser n'importe quel type d'encodage pour chaque simulation ; en particulier, l'encodage peut être identique pour toutes les simulations, ou généré de manière aléatoire.

On rappelle ici que lors de la conception d'un circuit intégré, les outils de conception peuvent choisir d'implémenter une cellule standard parmi plusieurs cellules standard fonctionnellement équivalentes afin de réaliser une tâche donnée. Cela permet d'optimiser divers paramètres globaux tels que la fréquence maximale, la surface, ou la consommation. En particulier, plusieurs

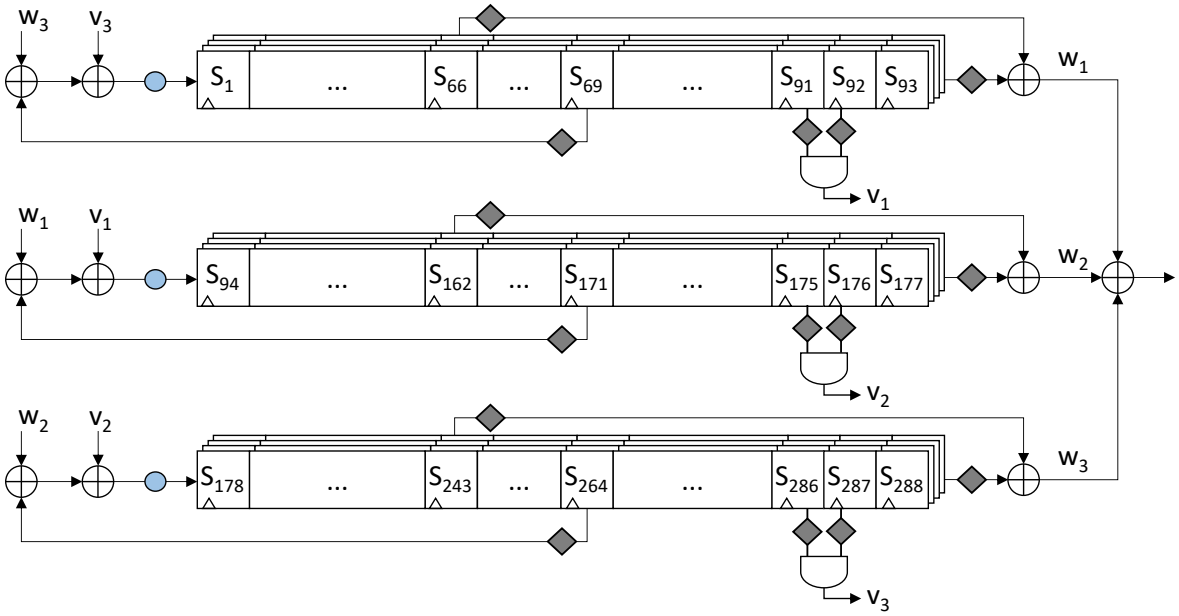


FIGURE 4.4 – Application de l'encodage dynamique à la protection des registres à décalage de Trivium.

types de bascules peuvent être sélectionnées. Il est alors possible que les bascules de deux registres soient différentes, mais également qu'un registre contienne plusieurs types de bascules différentes. Chaque bascule différente a des caractéristiques qui lui sont propres, telles que le temps de propagation dans les transistors ou la consommation. Du point de vue de l'analyse des canaux auxiliaires, ces déséquilibres de temps de propagation et de consommation entre les différentes bascules peuvent créer des fuites d'information, à condition qu'un attaquant dispose d'outils de mesure assez précis pour les quantifier. Afin d'étudier l'impact de ces déséquilibres, deux variantes du registre à décalage encodé sont implémentées.

La première variante suit un flot de conception standard, identique à celui décrit dans la section 3.2 pour le Trivium non protégé, c'est-à-dire avec les mêmes contraintes et paramètres pour les outils de synthèse, layout et simulations. À l'issue de ce flot de conception, plusieurs bascules différentes sont présentes dans les quatre registres à décalage parallèles. Cette variante est notée REG_ENC_DIFF, et un encodage dynamique 1-sur-4 fixé lui est appliqué dans toutes les simulations décrites par la suite.

Pour la seconde variante, les outils de conception sont contraints d'utiliser un seul type de bascules. On observe dans notre cas que toutes les bascules sont identiques dans la netlist obtenue en sortie de synthèse, et que c'est l'outil de placement et de routage qui remplace ensuite certaines bascules par d'autres bascules fonctionnellement équivalentes. Par conséquent, nous forçons cet outil à conserver les bascules présentes dans la netlist issue de la synthèse. Cette modification est très facile à effectuer et peut aisément s'intégrer dans un flot de conception standard. Comme nous le verrons par la suite, elle permet un équilibrage efficace de la consommation, au prix d'une très légère sous-optimisation de la consommation totale. Cette variante est notée REG_ENC_EQU.

Enfin, le registre à décalage de référence, non protégé, est noté REGISTRE. De même que la variante REG_ENC_DIFF, ce registre est implémenté avec un flot de conception standard auquel aucune contrainte spécifique n'a été rajoutée.

Second cas d'étude : Trivium

Nous détaillons maintenant l'application de l'encodage dynamique à la protection de l'algorithme de chiffrement par flot Trivium. Cet algorithme, décrit dans la section 1.3.2, a une structure très simple constituée de trois registres à décalage et de quelques portes logiques élémentaires pour la mise à jour de son état interne. Par la suite, nous présentons plusieurs variantes de l'encodage dynamique de Trivium. Toutes ces variantes sont implémentées selon un flot de conception

standard, strictement identique à celui décrit dans la section 3.2 pour le Trivium non protégé. La surface, le coût, et les fuites d'information par canaux auxiliaire des différentes variantes de l'encodage dynamique peuvent donc être comparées équitablement à celles du Trivium de référence. En particulier, on notera qu'aucun effort n'a été mis en œuvre pour équilibrer les différents chemins de données et diminuer ainsi les fuites par canaux auxiliaires, contrairement à ce qui a été fait pour l'une des variantes du registre à décalage encodé de manière dynamique, REG_ENC_EQU.

Dans un premier temps, nous implémentons l'encodage dynamique uniquement sur les registres à décalage qui constituent Trivium. Cette solution est représentée sur la figure 4.4, où les losanges gris représentent le décodage des sorties des registres avec $\mathscr{F}_c^{-1}$, et les ronds bleus montrent l'encodage en entrée des registres avec $\mathscr{F}_c$. On constate en particulier que le décodage avec $\mathscr{F}_c^{-1}$ doit être effectué de nombreuses fois en parallèle, pour chaque sortie des registres encodés. Cet encodage dynamique de la logique séquentielle peut être mis en œuvre avec une table d'encodage prédéfinie, ou générée aléatoirement à chaque exécution. Plusieurs variantes sont implémentées en fonction de la valeur de cette table d'encodage. Les variantes TRI_ENC_1 et TRI_ENC_2 contiennent toutes deux une table d'encodage fixée, correspondant, respectivement, à un encodage 1-sur-4 et à un encodage 2-sur-4. Une autre variante, notée TRI_ENC_ALEA, est réalisée avec un encodage dynamique aléatoire des registres à décalage ; celle-ci requiert 5 bits aléatoires à chaque nouveau chiffrement afin de générer une nouvelle table d'encodage 1-sur-4, tel que décrit dans la section 4.3. La génération et le stockage de cette table sont pris en compte dans l'évaluation de la surface et de la consommation.

Dans un second temps, nous implémentons l'encodage dynamique à la fois sur la logique séquentielle et la logique combinatoire ; en effet, les attaques matérielles étant actuellement effectuées contre la logique séquentielle, protéger uniquement la logique combinatoire aurait peu de sens. Au sein de la logique combinatoire, les portes logiques sont remplacées par des portes encodées, en utilisant les équations tirées du tableau 4.4. L'encodage dynamique des données avec $\mathscr{F}_c$ est alors réalisé une seule fois, lors du chargement des données dans les registres de Trivium ; ensuite, les calculs sont effectués directement sur les données encodées. Bien que le décodage en sortie des registres avec $\mathscr{F}_c^{-1}$ ne soit plus nécessaires dans ce cas, nous conservons une version allégée de cette fonction, dont le seul but est la détection des fautes. L'encodage dynamique de la logique combinatoire et de la logique séquentielle est effectué avec l'encodage 1-sur-4 donné dans le tableau 4.2 et les équations associées fournies dans la section 4.4.2. Cette variante est notée TRI_ENC_COMB.

Un point important à considérer pour l'encodage aléatoire de Trivium est l'initialisation des registres à décalage. Pour rappel, la clé secrète, l'IV et une constante sont chargés dans l'état interne de Trivium avant la phase d'initialisation. Ces données doivent être encodées lors de leur chargement dans l'état interne de Trivium, ce qui peut être réalisé de deux manières. D'un côté, il est possible d'initialiser tous les registres à "0", et de charger les données en série, bit par bit. Les données sont alors encodées au fur et à mesure, et le poids et la distance de Hamming totaux augmentent linéairement, jusqu'à atteindre une valeur constante, sans que des fuites par canaux auxiliaires ne soient observables. Cette solution est la plus simple à implémenter. D'un autre côté, il est possible de charger directement les données encodées dans les registres, en parallèle et sur un seul cycle d'horloge. Ce chargement en parallèle, plus rapide, est également recommandé par Gierlichs *et al.* [75] dans le cas où aucune protection n'est appliquée à Trivium afin d'empêcher un attaquant d'observer par canaux auxiliaires le chargement bit par bit de la clé. C'est cette dernière solution qui est retenue dans nos travaux.

L'état interne initial de Trivium doit donc être préalablement encodé avant d'être chargé dans les registres encodés. Pour cela, un bit sur deux de l'état interne est encodé avec une parité de "1", les autres bits ayant une parité de "0". Cette solution peut être implémentée avec l'ajout d'un multiplexeur pour chacun des 288 bits à encoder, dont la sortie dépend à la fois du bit à encoder, de sa parité, et de la table d'encodage. L'encodage dynamique aléatoire TRI_ENC_ALEA est mis en œuvre de cette manière. Une autre possibilité, plus légère, est une initialisation spécifique à une table d'encodage donnée, qui est donc possible uniquement si la table d'encodage est fixée. Celle-

TABLEAU 4.5 – Liste des implémentations étudiées.

Implémentation	Description
REGISTRE	Registre à décalage de référence, non protégé
REG_ENC_DIFF	Registre à décalage encodé, contenant plusieurs types de bascules différentes
REG_ENC_EQU	Registre à décalage encodé, contenant un seul type de bascule
TRIVIUM	Trivium de référence, non protégé
TRI_ENC_1	Encodage dynamique 1-sur-4 des registres à décalage de Trivium
TRI_ENC_2	Encodage dynamique 2-sur-4 des registres à décalage de Trivium
TRI_ENC_ALEA	Encodage dynamique aléatoire des registres à décalage de Trivium
TRI_ENC_COMB	Encodage dynamique 1-sur-4 des registres et de la logique combinatoire de Trivium

ci est décrite ci-dessous pour l'exemple d'encodage dynamique 1-sur-4 du tableau 4.2. Dans cet exemple, nous montrons comment initialiser les quatre chemins de données uniquement pour le chargement de l'IV encodé, l'ensemble de l'état interne de Trivium étant initialisé de la même manière :

$$\begin{aligned} S^1 &= IV \cdot \texttt{0x55555555555555555555} \\ S^2 &= IV \cdot \texttt{0xAAAAAAAAAAAAAAAAAAAA} \\ S^3 &= \overline{IV} \cdot \texttt{0x55555555555555555555} \\ S^4 &= \overline{IV} \cdot \texttt{0xAAAAAAAAAAAAAAAAAAAA} \end{aligned} \tag{4.4}$$

De cette manière, les bits de l'IV valant "1" sont écrits alternativement sur les parts 1 et 2, tandis que pour les bits valant "0", un "1" est écrit alternativement sur les parts 3 et 4.

4.5.2 Coût d'implémentation de l'encodage dynamique

Dans cette section, nous détaillons la surface et la consommation moyenne de puissance des différentes variantes décrites plus haut. La comparaison de la surface et de la consommation des différentes implémentations protégées à celles des implémentations non protégées de référence permet d'évaluer le coût d'implémentation de l'encodage dynamique. Par la suite, nous notons TRIVIUM l'implémentation du Trivium non protégé de référence, décrit dans la section 3.2. Un récapitulatif de toutes les implémentations étudiées est fourni dans le tableau 4.5.

Surface

Les surfaces des différentes variantes du registre à décalage, avec et sans encodage, sont données en GE (*Gate Equivalent*) dans le tableau 4.6. Les surfaces des variantes de Trivium sont données dans le tableau 4.7. Ces tableaux donnent, pour chaque implémentation, la surface de la logique combinatoire et séquentielle, la surface totale, ainsi que la surface normalisée des contre-mesures par rapport à l'implémentation de référence correspondante (REGISTRE et TRIVIUM, respectivement). Pour rappel, les surfaces en μm^2 peuvent être obtenues en multipliant par 0,4896 les surfaces fournies en GE.

La surface totale pour l'encodage dynamique 1-sur-4 et 2-sur-4, mais également pour l'implémentation TRI_ENC_COMB où la logique combinatoire est aussi encodée, est légèrement inférieure à quatre fois celle d'une implémentation non protégée. En particulier, on remarque que l'encodage de l'ensemble de la logique combinatoire de Trivium (dans TRI_ENC_COMB) rajoute un coût très faible en surface par rapport à l'encodage 1-sur-4 des registres uniquement (dans TRI_ENC_1). Cela s'explique par le fait que l'encodage avec $\mathscr{F}_c$ des bits entrants dans les registres et le décodage avec $\mathscr{F}_c^{-1}$ des bits sortants ne sont pas nécessaires dans ce cas, les calculs étant effectués directement sur la représentation encodée des bits. La complexité accrue de la logique

TABLEAU 4.6 – Surface après synthèse (GE) pour différentes variantes du registre à décalage, avec et sans encodage dynamique.

Implémentation	Surface séquentielle	Surface combinatoire	Surface totale	Surface normalisée
REGISTRE	986	0	986	1
REG_ENC_DIFF	3930	0	3930	3,99
REG_ENC_EQU	3930	0	3930	3,99

TABLEAU 4.7 – Surface après synthèse (GE) pour différentes variantes de Trivium, avec et sans encodage dynamique.

Implémentation	Surface séquentielle	Surface combinatoire	Surface totale	Surface normalisée
TRIVIUM	2212	595	2808	1
TRI_ENC_1	8836	2148	10985	3,91
TRI_ENC_2	8836	2352	11188	3,98
TRI_ENC_ALEA	8859	3905	12764	4,55
TRI_ENC_COMB	8836	2344	11181	3,98

combinatoire encodée est donc compensée par cette absence d'encodage et de décodage, qui se limite à la détection de fautes.

L'encodage dynamique aléatoire (dans TRI_ENC_ADAPT) rajoute un léger surcoût en surface, principalement dans la logique combinatoire. Cela provient de la logique supplémentaire nécessaire pour générer une table d'encodage à partir d'un nombre aléatoire de 5 bits.

Consommation

La consommation moyenne de puissance est donnée dans le tableau 4.8 pour les variantes du registre à décalage, et dans le tableau 4.9 pour les variantes de Trivium. De même que précédemment pour la surface, une consommation moyenne normalisée par rapport à l'implémentation de référence correspondante (REGISTRE et TRIVIUM, respectivement) est donnée pour chaque implémentation.

On constate que les implémentations protégées par l'encodage dynamique ont une consommation environ quatre fois supérieure à celle des implémentations non protégées. Le surcoût le plus important est à nouveau observable pour l'implémentation de Trivium protégée avec l'encodage dynamique aléatoire, pour laquelle la consommation normalisée par rapport au Trivium de référence est de 4,25.

4.5.3 Résistance aux attaques par canaux auxiliaires

Nous détaillons ici l'évaluation de la résistance contre les attaques par canaux auxiliaires des différentes implémentations étudiées. Plusieurs méthodes sont mises en œuvre dans ce but, à partir de traces de consommation de puissance obtenues avec des simulations post-layout.

On notera que, en dehors de la contrainte d'utiliser un seul type de bascules pour la variante REG_ENC_EQU du registre à décalage, aucune contrainte spécifique n'a été spécifiée lors des différentes étapes de conception dans le but d'équilibrer les fuites d'information sur les différents chemins de données. En particulier, le placement des cellules standard, ainsi que le routage entre celles-ci et avec les sources d'alimentation, n'ont pas été définis explicitement, mais effectués automatiquement et par défaut par l'outil de placement-routage. Du fait de routages a priori différents sur ces chemins de données, et de placements différents par rapport aux sources d'alimentation, tous les chemins de données d'une même implémentation présentent de légers déséquilibres de temps de propagation et de consommation. Ces déséquilibres devraient être d'autant plus vi-

TABLEAU 4.8 – Consommation moyenne simulée (µW) pour différentes variantes du registre à décalage, avec et sans encodage dynamique.

Implémentation	Consommation moyenne	Consommation normalisée
REGISTRE	103	1
REG_ENC_DIFF	406	3,94
REG_ENC_EQU	414	4,02

TABLEAU 4.9 – Consommation moyenne simulée (µW) pour différentes variantes de Trivium, avec et sans encodage dynamique.

Implémentation	Consommation moyenne	Consommation normalisée
TRIVIUM	241	1
TRI_ENC_1	987	4,1
TRI_ENC_2	997	4,14
TRI_ENC_ALEA	1024	4,25
TRI_ENC_COMB	985	4,09

sibles sur les implémentations qui contiennent plusieurs types de bascules différentes. Ces déséquilibres créent a priori des fuites d'information éventuellement observables par un attaquant, de la même manière que pour le lissage de la consommation avec divers styles de logique différentielle avec précharge [110, 111]. Ce choix de conserver des chemins de données déséquilibrés reflète un flot de conception standard où un effort minimal est fourni dans le but d'éliminer les fuites d'information par canaux auxiliaires. Ce choix permet d'évaluer l'efficacité de l'encodage dynamique même en présence de déséquilibres, et en particulier les gains apportés par l'encodage dynamique aléatoire afin de diminuer les fuites dues à ces déséquilibres.

Évaluation de l'équilibrage de la consommation

Tout d'abord, nous étudions l'efficacité de l'équilibrage de la consommation grâce à l'encodage dynamique. Cette évaluation repose sur l'étude de l'énergie par cycle d'horloge, obtenue en intégrant la consommation instantanée sur chaque cycle d'horloge. Au total, 1000 traces de 100 cycles d'horloge sont simulées pour chaque implémentation avec des paramètres d'initialisation différents.

La consommation d'énergie par cycle d'horloge est étudiée avec deux métriques : la variation maximale d'énergie normalisée, et la variance normalisée. La variation maximale d'énergie normalisée, appelée par la suite NED pour *Normalized Energy Deviation*, est définie comme la différence entre l'énergie par cycle maximale et l'énergie par cycle minimale, cette différence étant ensuite divisée par la valeur maximale. La variance normalisée, notée NSD pour *Normalized Standard Deviation*, est la variance divisée par la moyenne. Ces deux métriques sont comprises entre 0 et 1 ; plus elles sont proches de 0, plus le lissage de la consommation est efficace. Elles sont souvent utilisées pour évaluer l'efficacité des contre-mesures de lissage de la consommation [108, 109, 114, 115, 116].

L'énergie par cycle d'horloge pour cinq traces d'exécution différentes de l'implémentation TRI_ENC_COMB, dont la logique séquentielle et la logique combinatoires sont encodées dynamiquement, est représentée sur la figure 4.5 pour les 20 premiers cycles d'horloge. On constate que ces traces ont une consommation qui oscille à chaque cycle d'horloge. Cet effet est dû à l'utilisation d'un signal cyclique de contrôle c afin de réaliser l'encodage dynamique. À chaque cycle d'horloge, une partie des portes logiques de l'implémentation protégée commute, afin d'encoder ou de décoder les bits actuels différemment des bits précédents. Or, ce signal de contrôle ne donne pas d'information exploitable à un attaquant souhaitant réaliser une attaque CPA. L'analyse de l'équilibrage de la consommation porte uniquement sur les données exploitables par un attaquant, et ces variations périodiques sont donc exclues de celle-ci. Pour cela, le NED et le NSD

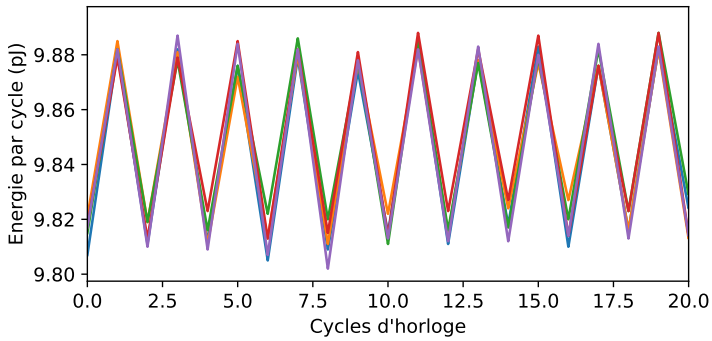


FIGURE 4.5 – Cinq exemples de traces de consommation, en énergie par cycle d'horloge, pour l'implémentation TRI_ENC_COMB.

TABLEAU 4.10 – NED et NSD maximaux.

Implémentation	NED	NSD
TRIVIUM	0,1705	0,0271
TRI_ENC_1	0,0181	0,0026
TRI_ENC_2	0,0195	0,0029
TRI_ENC_ALEA	0,0203	0,0032
TRI_ENC_COMB	0,0043	0,0006
REGISTRE	0,2272	0,0381
REG_ENC_DIFF	0,0012	0,0002
REG_ENC_EQU	0,0	0,0

sont calculés indépendamment pour chacun des cycles d'horloge ; on a donc 100 NED et 100 NSD calculés chacun avec 1000 valeurs d'énergie par cycle. Le maximum sur les 100 cycles d'horloge de chacune de ces métriques est utilisé pour estimer le lissage de la consommation au pire cas.

Les NED et NSD maximaux sont donnés dans le tableau 4.10 pour chacune des implémentations, et montrent un lissage efficace de la consommation. Lorsque seule la logique séquentielle de Trivium est encodée, c'est-à-dire pour les variantes TRI_ENC_1, TRI_ENC_2 et TRI_ENC_ALEA, le NED et le NSD sont réduits d'un facteur compris entre 8,4 et 10,4 par rapport au Trivium de référence. L'essentiel des variations de consommation d'un cycle d'horloge à l'autre vient alors de la logique combinatoire non protégée, et l'implémentation ayant le NED et le NSD les plus élevés est l'encodage aléatoire, car la part de logique combinatoire est plus importante dans celle-ci. Lorsque la logique combinatoire est également encodée dans Trivium, le NED et le NSD sont réduits respectivement de 40 et 44 fois. Enfin, l'encodage dynamique appliqué à un registre à décalage présente une consommation d'énergie par cycle d'horloge constante lorsque les bascules sont identiques, et un NED et un NSD divisés par environ 190 lorsque les bascules sont différentes.

Évaluation des fuites d'information par canaux auxiliaires

Dans la partie précédente, nous avons montré que l'encodage dynamique permet d'équilibrer la consommation énergétique de chaque cycle d'horloge de manière efficace. Cependant, de faibles variations de cette consommation énergétique sont toujours observables, et pourraient être exploitées par un attaquant. De plus, une attaque par analyse des canaux auxiliaires n'est généralement pas menée sur la consommation moyenne par cycle d'horloge, mais en observant les variations de la consommation instantanée. Des courtes variations et des pics brefs de consommation pourraient être exploités pour mener à bien une attaque CPA, tandis que ces variations éphémères sont partiellement masquées lors de l'intégration de la puissance sur la durée d'un cycle d'horloge, comme cela est effectué pour le calcul du NED et du NSD. Par conséquent, les résultats présentés dans la partie précédente, s'ils traduisent un équilibrage efficace de la consommation et une difficulté accrue de mener à bien une attaque par canaux auxiliaire, ne permettent

pas de conclure sur la possibilité ou non d'effectuer une telle attaque.

Nous évaluons ici la présence de fuites d'information par canaux auxiliaires avec le calcul de T-tests non-spécifiques à partir de traces de consommation de puissance instantanée. La clé de chiffrement est identique pour toutes les traces simulées pour les variantes de Trivium. Pour chaque implémentation évaluée, un T-test non-spécifique est réalisé avec 100.000 traces par lot. Dans l'un de ces lots, les traces sont générées avec des IV aléatoires. Dans l'autre lot, l'IV est fixé pour toutes les traces ; un IV fixé de 80 bits est identique pour toutes les variantes de Trivium, et un autre IV fixé de 128 bits est identique pour toutes les variantes du registre à décalage, ce qui permet une comparaison équitable de toutes les implémentations.

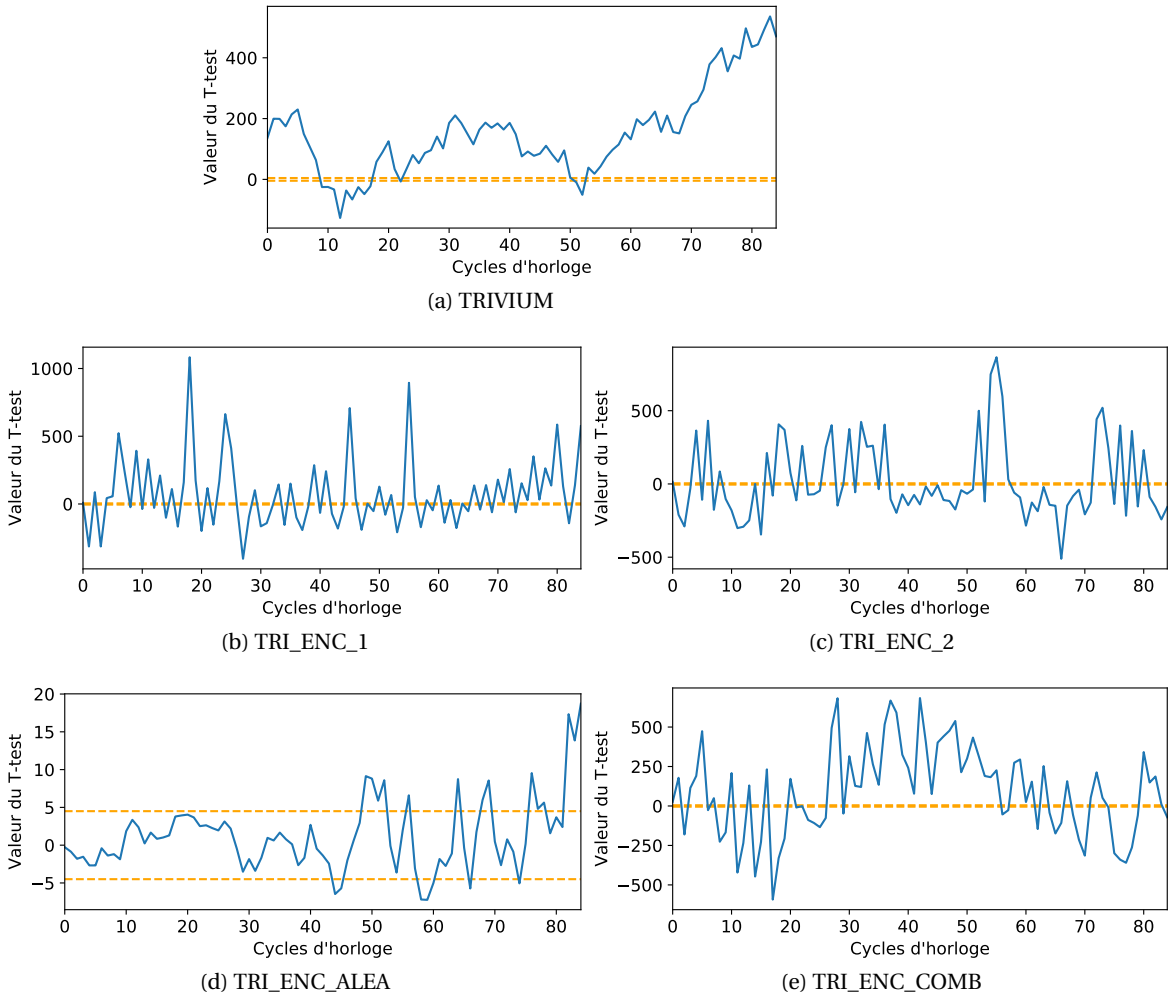
Toutes les traces de consommation sont simulées sur une durée de 85 cycles d'horloge. Cette durée est suffisante pour étudier la possibilité d'effectuer une attaque CPA contre Trivium, les attaques existantes nécessitant entre 76 et 81 cycles d'horloge [70, 71, 72, 73, 74]. De plus, toutes les implémentations étudiées ont un comportement identique au cours du temps, qui se répète à chaque cycle d'horloge. Ainsi, et d'après les résultats présentés dans la section 3.2.3 sur 1500 cycles pour Trivium, on peut s'attendre à ce que l'absence (ou la présence) de fuites pendant ces 85 premiers cycles d'horloge implique une absence (ou une présence) de fuites durant l'ensemble de la durée de fonctionnement de ces implémentations.

De même que pour les traces décrites dans la section 3.2, les traces simulées ici sont générées avec un intervalle de 10 ps entre deux points, et la fréquence utilisée pour les simulations est de 100 MHz. Chacune des 100.000 traces de chaque lot comporte donc 85.000 points, c'est-à-dire 1000 points par cycle d'horloge ; plusieurs jours sont nécessaires pour obtenir chaque lot de traces, cette durée variant notamment selon la taille de chaque implémentation. Chaque trace est ensuite compressée, afin de gagner à la fois en mémoire et en temps de traitement, en conservant uniquement la valeur maximale de consommation de chaque cycle d'horloge.

La figure 4.6 montre les variations des T-tests pour le Trivium non protégé et pour toutes les variantes de l'encodage dynamique appliqué à Trivium. On rappelle ici que le dépassement de la valeur de seuil de 4,5 matérialisée en pointillés oranges indique la présence de fuites d'information par canaux auxiliaires. Selon cette figure, il y aurait des fuites d'information importantes pour les différentes versions de l'encodage dynamique, lorsque cet encodage n'est pas généré aléatoirement. La figure 4.6d montre que lorsque la table d'encodage est générée aléatoirement, et même si la *fonction de rétroaction* de Trivium n'est pas protégée, il n'y a pas de fuites pendant les 40 premiers cycles d'exécution de Trivium. Les faibles fuites visibles après ces 40 cycles peuvent s'expliquer, du moins partiellement, par le fait que la logique combinatoire n'est pas protégée.

Deux T-tests sont effectués sur le registre protégé dont toutes les bascules sont identiques, REG_ENC_EQU : l'un avec avec un encodage 1-sur-4 fixe pour toutes les traces, et l'autre avec un encodage 1-sur-4 généré aléatoirement pour chaque trace. La figure 4.7, qui montre les résultats du T-test appliqué aux différentes versions du registre à décalage, valide l'efficacité de l'encodage dynamique aléatoire : lorsque la table d'encodage est générée aléatoirement et en l'absence de logique combinatoire non protégée, il n'y a pas de fuites par canaux auxiliaires, et ce même en considérant que les différents chemins de données présentent de légers déséquilibres.

On notera que ces simulations ne prennent en compte aucune sorte de bruit. Cette absence de bruit permet une étude des fuites d'information avec une grande précision, mais introduit également un biais lors du calcul du T-test non-spécifique pour certaines implémentations. En effet, l'un des deux lots est généré avec à la fois une clé et un IV fixe. Si des données aléatoires ne sont pas rajoutées par ailleurs, donc dans le cas du Trivium non protégé et de l'encodage dynamique non aléatoire, ce lot contient en réalité une unique trace, répétée 100.000 fois à l'identique. En particulier, pour chaque point de ces traces, la variance est nulle, et la moyenne est égale à la valeur de ce point. Les valeurs du T-test non-spécifique pour ces implémentations sont donc probablement sur-évaluées, et une mesure expérimentale de traces bruitées permettrait de fournir une meilleure évaluation des fuites par canaux auxiliaires. Dans la section 6.2, nous détaillons l'analyse des fuites par canaux auxiliaires sur FPGA pour un Trivium de référence non protégé, ce qui permet de valider la présence de fuites d'information pour cette implémentation, déjà observées



(a) TRIVIUM

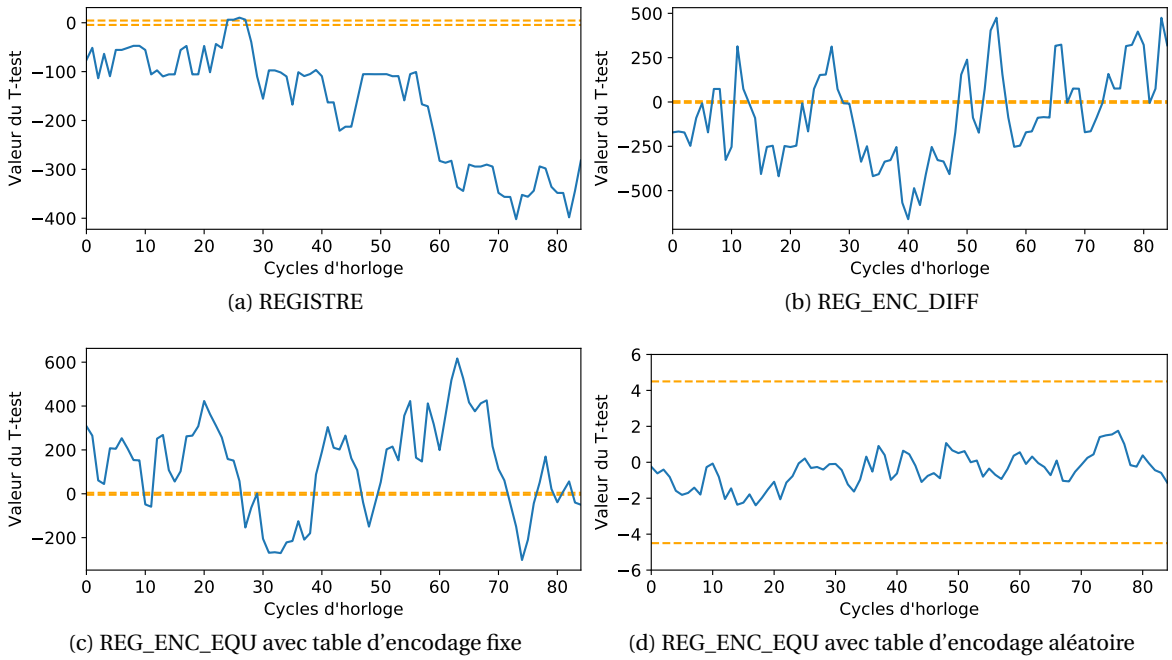
(b) TRI_ENC_1

(c) TRI_ENC_2

(d) TRI_ENC_ALEA

(e) TRI_ENC_COMB

FIGURE 4.6 – Valeurs du T-test pour les différentes versions de Trivium, avec et sans encodage dynamique.



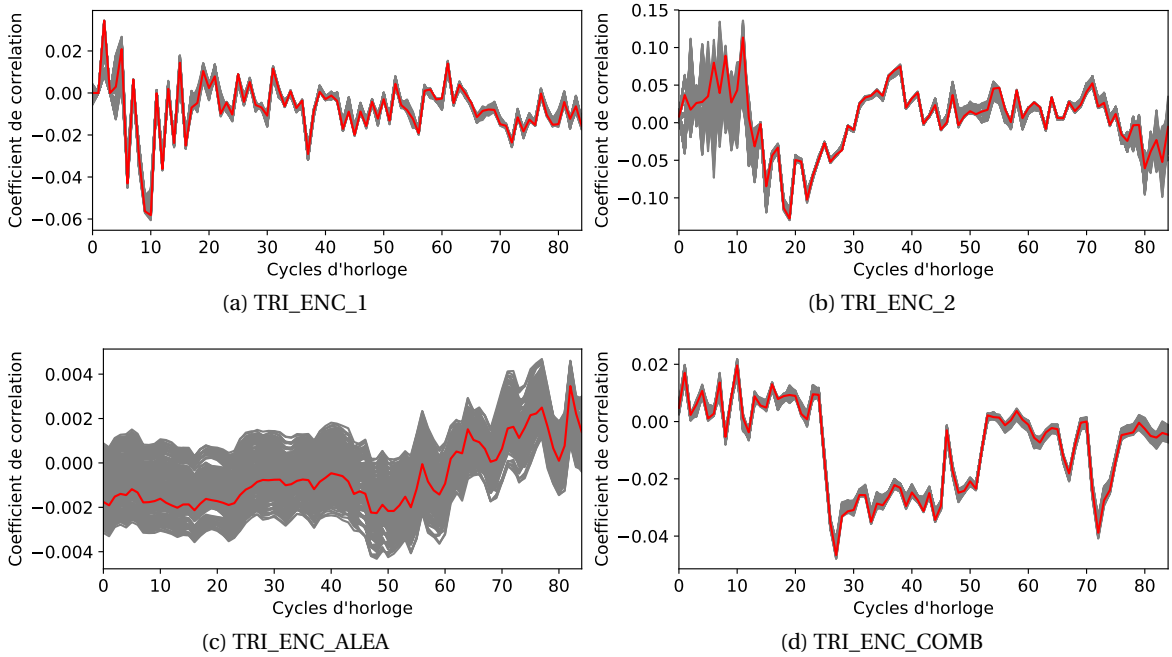
(a) REGISTRE

(b) REG_ENC_DIFF

(c) REG_ENC_EQU avec table d'encodage fixe

(d) REG_ENC_EQU avec table d'encodage aléatoire

FIGURE 4.7 – Valeurs du T-test pour les différentes versions du registre à décalage, avec et sans encodage dynamique.



(a) TRI_ENC_1 (b) TRI_ENC_2

(c) TRI_ENC_ALEA (d) TRI_ENC_COMB

FIGURE 4.8 – Évolution du coefficient de corrélation pour les différentes implémentations de Trivium protégées par l'encodage dynamique.

TABLEAU 4.11 – Maximum du coefficient de corrélation pour les implémentations de référence et les implémentations protégées par l'encodage dynamique.

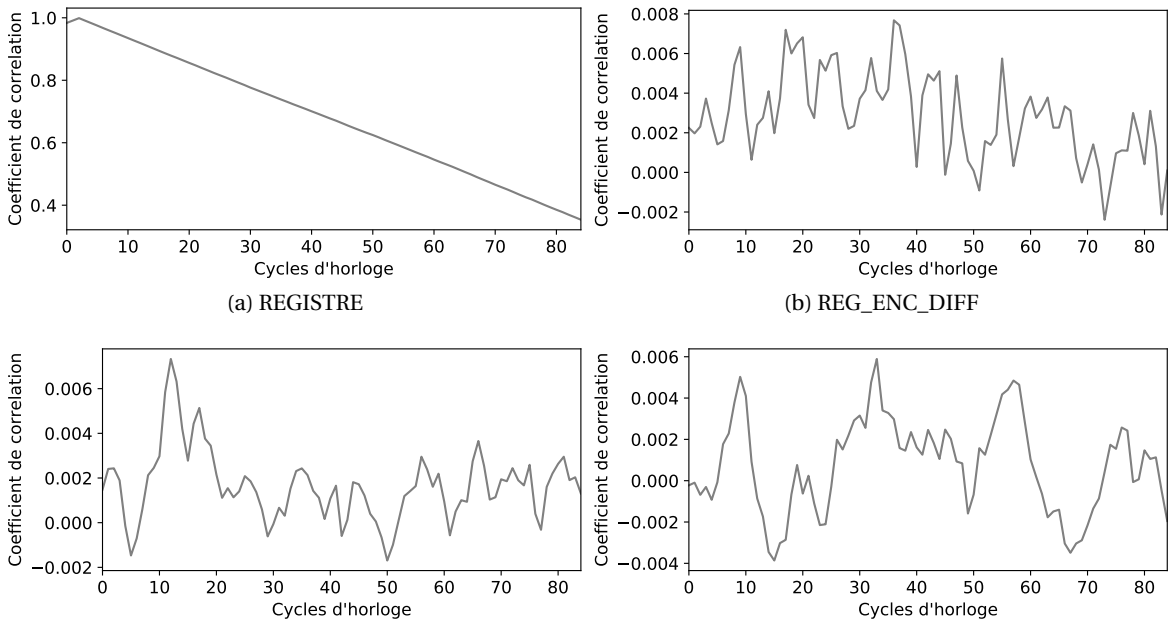
Implémentation	**Corrélation (valeur absolue)**
TRIVIUM	0,8950
TRI_ENC_1	0,0582
TRI_ENC_2	0,1274
TRI_ENC_ALEA	0,0035
TRI_ENC_COMB	0,0466
REGISTRE	0,9989
REG_ENC_DIFF	0,0077
REG_ENC_EQU	0,0073

dans la section 3.2.3. De même, une évaluation des fuites par canaux auxiliaires de l'encodage dynamique devrait être menée expérimentalement sur ASIC, afin de valider ou d'invalider les résultats présentés ici. Bien que cette évaluation sur silicium n'ait pas encore été menée à ce jour, la section 6.1 décrit un circuit réalisé avec cette protection et envoyé en fonderie, qui sera testé dans le futur.

Exploitation des fuites d'information par canaux auxiliaires

Nous avons montré précédemment l'existence de fuites d'information potentielles pour la plupart des implémentations protégées par l'encodage dynamique, avec des T-tests non-spécifiques effectués sur des traces non bruitées. Nous évaluons ici la possibilité d'exploiter ces fuites dans une attaque CPA similaire à celles publiées à l'heure actuelle contre Trivium. Pour cela, nous évaluons, pour toutes les implémentations étudiées, les variations temporelles de la corrélation entre une consommation hypothétique et la consommation simulée. Cette évaluation suit la méthodologie décrite dans la section 3.2.2. Pour chaque implémentation, les 100.000 traces utilisées correspondent au lot de traces avec des IV aléatoires du T-test non-spécifique.

Pour chaque variante de Trivium, le coefficient de corrélation est étudié pour les 8 premières



(a) REGISTRE

(b) REG_ENC_DIFF

(c) REG_ENC_EQU avec table d'encodage fixe

(d) REG_ENC_EQU avec table d'encodage aléatoire

FIGURE 4.9 – Évolution du coefficient de corrélation pour les différentes versions du registre à décalage, avec et sans encodage dynamique.

valeurs intermédiaires σ_i, ce qui correspond à un octet de la clé. Pour chacune des 256 hypothèses possibles sur cet octet de clé, la corrélation entre la consommation hypothétique correspondante et la consommation réelle est calculée. La figure 4.8 montre l'évolution du coefficient de corrélation pour ces 256 hypothèses, pour chaque implémentation de l'encodage dynamique à Trivium ; la corrélation pour la bonne hypothèse de clé y est affichée en rouge. On constate que pour toutes les implémentations, la bonne hypothèse de clé n'est pas distinguable des 255 mauvaises hypothèses, ce qui indique la difficulté d'exploiter ces résultats afin d'effectuer une CPA. Cette observation est validée par la mise en œuvre d'une CPA itérative visant à retrouver bit par bit les 81 σ_i, telle que décrite dans la section 3.2. Cette CPA, effectuée sur toutes les variantes de Trivium protégées, ne permet pas de retrouver la clé de chiffrement avec 100.000 traces de consommation. Pour rappel, moins de 100 traces sont nécessaires pour retrouver cette clé de chiffrement dans le cas du Trivium non protégé.

Les différentes variantes du registre à décalage sont initialisées avec un IV différent pour chaque trace, mais aucune clé de chiffrement n'est manipulée. En l'absence de clé, une seule consommation hypothétique est calculée pour chaque trace, cette consommation dépendant directement de l'IV utilisé. La corrélation est donc calculée entre cette unique consommation hypothétique et la consommation obtenue avec les simulations. De même que pour le T-test, le coefficient de corrélation est calculé avec deux jeux de traces différents pour l'implémentation REG_ENC_EQU : l'un avec avec un encodage 1-sur-4 fixe pour toutes les traces, et l'autre avec un encodage 1-sur-4 généré aléatoirement pour chaque trace. La figure 4.9 montre l'évolution du coefficient de corrélation pour chaque variante du registre à décalage.

Le tableau 4.11 donne le maximum du coefficient de corrélation pour chaque implémentation étudiée, et rappelle la valeur maximale du coefficient de corrélation obtenue pour le Trivium de référence dans la section 3.2.2. On constate que les valeurs du coefficient de corrélation sont très élevées pour le Trivium et le registre à décalage non protégé, ce qui indique que le modèle théorique de consommation correspond bien à la consommation simulée de ces implémentations. Cette corrélation est fortement réduite pour toutes les implémentations protégées. Le coefficient de corrélation est particulièrement faible pour les deux versions du registre à décalage encodé, et pour le Trivium protégé par un encodage dynamique aléatoire. La faible valeur de ce coefficient indique une absence de corrélation entre les consommations hypothétiques et la consommation

simulée.

Ces observations permettent de conclure que le modèle de consommation utilisé pour les attaques CPA contre Trivium, qui considère uniquement les commutations dans les bascules, ne peut pas être utilisé pour effectuer une attaque avec succès sur un circuit protégé par l'encodage dynamique. Bien que des fuites soient observables avec un T-test non-spécifique, il n'existe pas à l'heure actuelle de moyen d'exploiter ces fuites pour retrouver des valeurs secrètes. Ces fuites sont probablement dues, au moins partiellement, aux déséquilibres de placement et de routage présents sur toutes les implémentations. La prise en compte de ces déséquilibres dans un modèle de consommation précis demanderait à un attaquant une excellente connaissance du layout du circuit visé. De plus, du fait de l'implémentation physique différente de chaque circuit, mais également des variations de processus lors de la fabrication des circuits intégrés, ces déséquilibres seraient différents pour tous les circuits, ce qui limiterait l'efficacité d'une attaque exploitant les fuites d'information observables dans le but de récupérer une clé de chiffrement.

4.6 Conclusion

Dans ce chapitre, nous proposons une nouvelle contre-mesure mixte contre les attaques par analyse des canaux auxiliaires et par injection de fautes, appelée *encodage dynamique*. Cette protection permet de lisser la consommation d'un algorithme, sans nécessiter de phase de précharge. L'encodage dynamique consiste à encoder différemment deux bits consécutifs, de sorte à maintenir un poids et une distance de Hamming constants dans le circuit protégé. Pour cela, une table d'encodage contient les codes permettant de représenter chaque bit, et un signal de contrôle qui alterne entre "0" et "1" à chaque cycle d'horloge permet de sélectionner le code adéquat. Il y a plusieurs manières d'implémenter l'encodage dynamique. Dans ce chapitre, nous présentons uniquement l'encodage 1-sur-4 et l'encodage 2-sur-4, avec des codes dans GF(2^4).

Plusieurs variantes de l'encodage dynamique sont présentées. Nous détaillons comment protéger les registres à décalage et la logique combinatoire, et discutons de la sécurité fournie dans les deux cas. Appliqué à la logique séquentielle, l'encodage dynamique permet une meilleure détection des fautes que les contre-mesures mixtes existantes, au vu des techniques actuelles d'injection de fautes. Cette contre-mesure permet également d'augmenter considérablement la complexité des attaques en fautes contre la logique combinatoire, bien qu'un modèle de fautes exhaustif dans ce cas n'ait pas été établi.

Une évaluation de la protection fournie par l'encodage dynamique contre les attaques par canaux auxiliaires est réalisée à partir de simulations post-layout de plusieurs implémentations d'un registre à décalage et de l'algorithme de chiffrement par flot Trivium. Ces simulations montrent un lissage efficace de la consommation d'énergie par cycle d'horloge, cette consommation d'énergie étant constante pour l'une des implémentations protégées du registre à décalage. Bien que des fuites d'information par canaux auxiliaires soient observables avec un T-test non-spécifique, ces fuites ne sont pas exploitables avec les attaques publiées à l'heure actuelle contre Trivium, et proviennent probablement de légers déséquilibres de temps de propagation et de consommation sur les différents chemins de données. Ces déséquilibres diffèrent d'un circuit à l'autre, et l'exploitation de ces déséquilibres afin de retrouver une clé de chiffrement grâce à une attaque par analyse des canaux auxiliaires serait très complexe.

L'encodage dynamique aléatoire, qui consiste à générer une nouvelle table d'encodage lors de chaque nouvelle exécution de l'algorithme protégé, semble particulièrement efficace à la fois contre les attaques par canaux auxiliaires et par injection de fautes. Cet encodage aléatoire peut être implémenté pour la protection de registres à décalage avec un surcoût et une complexité faibles. Des simulations post-layout de cet encodage dynamique aléatoire des registres à décalage montrent une absence de fuites par canaux auxiliaires. L'application de cet encodage dynamique aléatoire à la logique combinatoire est plus complexe. Pour cela, il est nécessaire de générer des équations représentant l'encodage dynamique des différentes portes logiques élémentaires pour chaque table d'encodage. Il n'existe actuellement pas de méthode pour générer automa-

tiquement ces équations pour n'importe quelle table d'encodage. Une telle méthode faciliterait grandement l'implémentation de l'encodage dynamique aléatoire de la logique combinatoire, et cela peut constituer une piste de recherche pour de futurs travaux.

L'encodage dynamique a une consommation de puissance environ quatre fois plus élevée qu'une implémentation non protégée, n'augmente pas le temps d'exécution de l'algorithme protégé, et nécessite au maximum 8 bits aléatoires à chaque exécution de cet algorithme si la table d'encodage est générée aléatoirement. Le coût énergétique total de cette contre-mesure est donc bien plus faible que celui des autres protections mixtes existantes.

L'encodage dynamique a fait l'objet d'un dépôt de brevet en 2018.

Chapitre 5

Concept et étude des contre-mesures adaptatives

Sommaire

5.1 Masquage TI de Trivium 103
5.2 Masquage semi-adaptatif, énergétiquement efficace 105
5.3 Masquage adaptatif 106
5.3.1 Contrôle du niveau de sécurité 107
5.3.2 Réduction du niveau de sécurité 107
5.3.3 Augmentation du niveau de sécurité 108
5.4 Encodage dynamique adaptatif 109
5.5 Caractérisation et évaluation des contre-mesures 110
5.5.1 Surface 110
5.5.2 Consommation 111
5.5.3 Fuites d'information par canaux auxiliaires 113
5.5.4 Corrélation et CPA 115
5.6 Implémentation de sécurité adaptative pour divers cas d'usage 117
5.6.1 Choix d'implémentation 117
5.6.2 Extension à d'autres algorithmes 118
5.7 Conclusion 118

Dans le chapitre précédent, nous avons présenté une nouvelle contre-mesure ayant une faible consommation énergétique et permettant de protéger l'implémentation matérielle d'une primitive cryptographique contre les attaques par analyse des canaux auxiliaires et par injection de fautes. Cependant, nous avons expliqué dans le chapitre 1 qu'il est possible d'aller plus loin dans l'optimisation du compromis entre consommation d'énergie et sécurité matérielle, avec la mise en place de contre-mesures matérielles adaptatives. Ces contre-mesures configurables permettent notamment de répondre aux attaques matérielles lorsqu'elles sont détectées tout en conservant un niveau de sécurité faible et donc une consommation minimale le reste du temps, ou encore d'adapter le comportement d'un objet connecté à l'évolution des besoins en sécurité et en consommation au cours de sa vie.

Seuls quelques travaux portent actuellement sur des contre-mesures matérielles configurables. Parmi ceux-ci, plusieurs sont adaptés à des implémentations reconfigurables, qu'elles soient logicielles ou sur FPGA. Nous nous intéressons ici exclusivement aux protections adaptatives pouvant être implémentées sur ASIC, qui sont présentées par la suite.

Les protections adaptatives existantes contre les attaques par canaux auxiliaires consistent toutes à générer du bruit sur demande, afin de couvrir les fuites par canaux auxiliaires lorsque c'est nécessaire. Ainsi, Robisson *et al.* [89] proposent d'activer des générateurs de nombres aléatoires, et Canto *et al.* [163] d'effectuer les opérations nécessaires au chiffrement à la fois sur les données réelles et sur des données aléatoires. La proposition de Shan *et al.* [91] se rapproche des implémentations sur FPGA, dans la mesure où une grille d'éléments reconfigurables est utilisée afin de mettre en œuvre diverses protections, telles qu'une exécution des opérations dans un ordre aléatoire et sur des chemins de données différents d'une exécution à l'autre, ou la génération de bruit en amplitude. Or, nous avons expliqué dans le chapitre 3 que la génération de bruit, qu'il soit temporel ou en amplitude, n'est généralement pas suffisante pour contrer les attaques par canaux auxiliaires lorsque celles-ci sont effectuées avec suffisamment de traces et un traitement adapté de ces traces.

En ce qui concerne la protection contre les attaques en fautes, une contre-mesure adaptative consisterait par exemple à activer ou non de la redondance, qu'elle soit spatiale, temporelle, ou en information. Cette solution a été proposée en logiciel par Robisson *et al.* [89] et sur FPGA par Gogniat *et al.* [164], mais pourrait être adaptée sur ASIC. Cependant, comme expliqué dans le chapitre 3, un attaquant connaissant le type de redondance utilisée peut théoriquement créer des fautes en conséquence, difficiles à détecter. De plus, la redondance est coûteuse en énergie, particulièrement si elle doit être combinée à une protection contre l'analyse des canaux auxiliaires.

Afin d'offrir une meilleure protection contre les attaques matérielles, nous proposons de nouvelles contre-mesures matérielles adaptatives, implémentées au niveau algorithmique. Nos contre-mesures adaptatives sont appliquée à la protection de l'algorithme de chiffrement par flot Trivium. L'implémentation de cette sécurité matérielle adaptative est étudiée pour deux types de contre-mesures. D'un côté, le masquage en *Threshold Implementation*, ou masquage TI, est une protection mature et largement étudiée, et l'implémentation adaptative de cette contre-mesure bénéficie donc de la sécurité théoriquement prouvée et expérimentalement éprouvée de celle-ci. D'un autre côté, le masquage TI ne permettant pas de détecter les fautes, nous appliquons le concept de sécurité adaptative à l'encodage dynamique décrit dans le chapitre 4. On notera que le masquage est plus efficace en présence de bruit, notamment aux ordres supérieurs comme expliqué dans le chapitre 3. De même, le bruit permet de couvrir les petites variations de consommation subsistant avec l'encodage dynamique du fait des divers déséquilibres discutés dans le chapitre 4. Nos propositions de contre-mesures adaptatives sont donc complémentaires à la génération sur demande de bruit proposée dans les travaux existants [89, 163, 91].

Après avoir détaillé un masquage TI classique de Trivium aux premier et second ordres, nous décrivons un masquage semi-adaptatif énergétiquement efficace, puis un masquage adaptatif permettant de protéger cet algorithme au premier ou au second ordre, ou de désactiver le masquage afin d'économiser de l'énergie. Nous proposons ensuite de modifier l'encodage dynamique présenté dans le chapitre 4 afin de le rendre adaptatif. Dans nos deux protections adaptatives,

le niveau de sécurité peut être modifié à la demande durant le fonctionnement de Trivium, et ce changement de niveau de sécurité n'a pas d'impact sur le temps d'exécution de l'algorithme. Nous décrivons ensuite l'implémentation matérielle et l'évaluation des différentes contre-mesures grâce à des simulations post-layout. Enfin, nous discutons divers aspects de cette sécurité adaptative, comme l'extension à d'autres algorithmes ou des variantes d'implémentation.

Remarque : Nos travaux portent uniquement sur les contre-mesures elles-mêmes, et non sur le mécanisme effectuant le choix du niveau de protection à appliquer. Ce mécanisme a déjà été étudié dans plusieurs travaux, notamment par les auteurs de [89] et de [164], qui pourraient être appliqués au contrôle de nos contre-mesures adaptatives. Dans les deux cas, divers capteurs permettent de détecter certaines attaques effectuées contre un circuit ; ainsi, les auteurs de [89] proposent la mesure embarquée de lumière, de température et de tension afin de détecter des injections de fautes, tandis que les auteurs de [164] proposent de surveiller plusieurs aspects du fonctionnement global d'un système sur puce, comme les transferts de données ou la consommation, afin de déterminer si ce fonctionnement dévie d'un comportement type, ce qui indiquerait une attaque potentielle. L'intégration d'autres capteurs, pour détecter par exemple la proximité d'une sonde électromagnétique [46] ou des fluctuations de la tension d'alimentation traduisant une attaque par analyse de cette tension [47, 48], peut également être envisagée dans le but de compléter ces travaux. Les mesures issues de ces capteurs sont ensuite traitées en fonction d'une politique de sécurité donnée, afin de déterminer le niveau de protection à appliquer ; par exemple, dans [89], une politique de sécurité fondée sur de la logique floue permet de déterminer plusieurs niveaux de réponse appropriés. Le mécanisme de contrôle des contre-mesures est généralement implémenté en logiciel ou sur FPGA, ce qui permet de mettre à jour les politiques de sécurité au cours de la vie du circuit intégré, tel qu'expliqué dans [9].

5.1 Masquage TI de Trivium

Nous présentons ici la protection de Trivium avec un masquage TI. Un masquage TI au premier ordre de Trivium est décrit dans [138] au niveau algorithmique, mais aucune implémentation matérielle n'est réalisée par les auteurs afin d'évaluer la surface, la consommation et la résistance aux attaques par canaux auxiliaires de cette protection. Nous nous appuyons sur ces travaux pour le masquage au premier ordre de Trivium, et décrivons celui-ci ci-après. Nous proposons et décrivons également un masquage au second ordre. À partir du second ordre, les modifications de l'implémentation et les optimisations présentées par la suite sont similaires pour tous les ordres, la principale différence étant le nombre de parts. Ces travaux sur le masquage au second ordre pourraient donc être étendus à des ordres supérieurs afin de répondre à des besoins divers en sécurité. Par la suite, nous considérons uniquement des implémentations du masquage TI avec $(td+1)$ parts, d étant l'ordre du masquage, et t le degré de la fonction à protéger, pour les raisons données dans la section 3.4.2.

Pour un masquage avec n parts, les trois registres à décalage qui constituent Trivium sont répliqués n fois en parallèle afin de contenir ces parts. Nous noterons donc par la suite S^i la $i^{ème}$ part, contenue dans le $i^{ème}$ registre en parallèle, et S^i_j le $j^{ème}$ bit, sur 288, de cette $i^{ème}$ part. Les n registres en parallèle, quant à eux, sont appelés *chemins de données*. Cette distinction entre les parts et les chemins de données, c'est-à-dire entre le contenu et le contenant, sera particulièrement utile pour la compréhension des contre-mesures adaptatives, qui consistent à modifier le nombre de parts afin d'augmenter ou de diminuer le niveau de sécurité.

La fonction combinatoire permettant de mettre à jour l'état interne de Trivium, appelée *fonction de rétroaction*, est transformée en une *fonction partagée* avec plusieurs parts d'entrée et de sortie, telle que décrite dans la section 3.4.2. Dans ce chapitre, afin de simplifier les notations, nous détaillons uniquement le masquage d'une partie de cette fonction, pour le calcul du premier bit du second registre à décalage, noté S_{94}. Ce bit est la valeur intermédiaire ciblée par les attaques par analyse des canaux auxiliaires publiées contre Trivium. Le masquage du reste de la *fonction de rétroaction*, pour le calcul du bit d'entrée des deux autres registres de Trivium, est identique. Par

la suite, nous noterons donc *fonction non protégée* la formule suivante :

$$S_{94} = S_{66} + S_{93} + S_{171} + S_{91}S_{92} \tag{5.1}$$

Cette fonction contient uniquement une porte ET des portes XOR. En appliquant l'équation (3.3) à cette *fonction non protégée*, on obtient un masquage TI au premier ordre de Trivium tel que décrit par les auteurs de [138]. La *fonction partagée* au premier ordre pour la mise à jour du 94-ième bit de l'état interne est donc :

$$\begin{aligned}
S^1_{94} &= S^2_{66} + S^2_{93} + S^2_{171} + S^2_{91}S^2_{92} + S^2_{91}S^3_{92} + S^3_{91}S^2_{92} \\
S^2_{94} &= S^3_{66} + S^3_{93} + S^3_{171} + S^3_{91}S^3_{92} + S^3_{91}S^1_{92} + S^1_{91}S^3_{92} \\
S^3_{94} &= S^1_{66} + S^1_{93} + S^1_{171} + S^1_{91}S^1_{92} + S^1_{91}S^2_{92} + S^2_{91}S^1_{92}
\end{aligned} \tag{5.2}$$

Le masquage au second ordre de cette fonction est effectué avec 5 parts d'entrée et 10 parts de sortie. Un exemple d'un tel masquage est décrit par l'équation (3.4). Cependant, nous choisissons de modifier cette équation en permutant les parts d'entrée, de sorte à obtenir l'équation suivante dans le cas de Trivium :

$$\begin{aligned}
S^1_{94} &= S^2_{66} + S^2_{93} + S^2_{171} + S^2_{91}S^2_{92} + S^2_{91}S^3_{92} + S^3_{91}S^2_{92} \\
S^2_{94} &= S^3_{66} + S^3_{93} + S^3_{171} + S^3_{91}S^3_{92} + S^3_{91}S^1_{92} + S^1_{91}S^3_{92} \\
S^3_{94} &= S^1_{66} + S^1_{93} + S^1_{171} + S^1_{91}S^1_{92} + S^1_{91}S^2_{92} + S^2_{91}S^1_{92} \\
S^4_{94} &= S^5_{66} + S^5_{93} + S^5_{171} + S^5_{91}S^5_{92} + S^5_{91}S^2_{92} + S^2_{91}S^5_{92} \\
S^5_{94} &= S^4_{66} + S^4_{93} + S^4_{171} + S^4_{91}S^4_{92} + S^4_{91}S^2_{92} + S^2_{91}S^4_{92} \\
S^6_{94} &= S^4_{91}S^3_{92} + S^3_{91}S^4_{92} \\
S^7_{94} &= S^5_{91}S^3_{92} + S^3_{91}S^5_{92} \\
S^8_{94} &= S^5_{91}S^1_{92} + S^1_{91}S^5_{92} \\
S^9_{94} &= S^5_{91}S^4_{92} + S^4_{91}S^5_{92} \\
S^{10}_{94} &= S^4_{91}S^1_{92} + S^1_{91}S^4_{92}
\end{aligned} \tag{5.3}$$

Cette permutation des parts d'entrée permet de réaliser une autre implémentation du masquage TI au second ordre. On remarque que les trois premières égalités de l'équation (5.3) correspondent au masquage TI de Trivium au premier ordre donné par l'équation (5.2). Cette propriété sera exploitée par la suite pour la réalisation du masquage adaptatif.

Les 10 parts de sortie de cette fonction sont ensuite réduites à 5 parts, afin de maintenir un nombre de parts constant. Pour cela, la *fonction de réduction* décrite par l'équation (3.5) pourrait être utilisée. Comme expliqué dans la section 3.4.2, la *fonction partagée* et la *fonction de réduction* doivent être séparées par un registre, afin de ne pas exécuter celles-ci lors du même cycle d'horloge, et donc de garantir la non-complétude du masquage. Pour éviter une augmentation du temps d'exécution, nous tirons ici partie de la structure de Trivium à base de registres à décalages. La première bascule du registre à décalage après la *fonction partagée* (5.3) est utilisée pour séparer cette opération de la *fonction de réduction*. Ainsi, le premier bit d'état interne après la *fonction partagée* est masqué sur 10 parts, et la *fonction de réduction* est appliquée après celui-ci. Plutôt que d'utiliser la *fonction de réduction* de l'équation (3.5), nous proposons une autre manière de réduire les parts, plus équilibrée et adaptée pour être utilisée en sortie de l'équation (5.3). Celle-ci est donnée ci-dessous :

$$\begin{aligned}
S^1_{95} &= S^1_{94} + S^6_{94} & \qquad S^2_{95} &= S^2_{94} + S^7_{94} \\
S^3_{95} &= S^3_{94} + S^8_{94} & \qquad S^4_{95} &= S^4_{94} + S^9_{94} \\
S^5_{95} &= S^5_{94} + S^{10}_{94}
\end{aligned} \tag{5.4}$$

Le masquage proposé au premier ordre, directement tiré de l'équation (3.3), respecte toutes les propriétés du masquage TI, y compris l'uniformité. De même, les équations (5.3) et (5.4) pour le

masquage au second ordre respectent toutes deux les propriétés d'exactitude et de non-complétude à l'ordre 2. De même que pour les équations présentées dans la section 3.4.2 pour un masquage TI au second ordre, la combinaison de ces deux équations est uniforme. Au final, le masquage TI de Trivium, au premier comme au second ordre, respecte à la fois les propriétés de non-complétude et d'uniformité, et l'ajout de nouveaux bits aléatoires de masques durant le fonctionnement n'est donc pas nécessaire.

5.2 Masquage semi-adaptatif, énergétiquement efficace

L'ensemble des attaques par analyse des canaux auxiliaires contre Trivium publiées à ce jour [70, 71, 72, 73, 74, 160] sont effectuées exclusivement au début de la phase d'initialisation de Trivium. Nous avons montré dans la section 3.2.3 que ces attaques, qui visent certaines valeurs intermédiaires en particulier, ne sont possibles que durant cette phase d'initialisation. De plus, aucune évaluation de la sécurité du masquage appliqué à Trivium n'a été effectuée jusqu'à présent; en particulier, il n'existe pas à ce jour d'attaque d'ordre supérieur réalisée avec succès contre Trivium. En se basant sur ces observations, nous proposons dans un premier temps un masquage énergétiquement efficace de Trivium, qui consiste à activer un masquage au premier ordre lors de l'initialisation de Trivium, et à désactiver ce masquage lors du chiffrement. Cette solution permet de protéger Trivium contre les attaques existantes, tout en réduisant le niveau de sécurité, et donc la consommation, lorsque le masquage n'est pas nécessaire. Cette version est semi-adaptative, dans la mesure où l'état du masquage change au cours de l'exécution de Trivium, mais que ce changement s'effectue une seule fois par exécution, dans un seul sens et à partir de paramètres pré-déterminés.

Le masquage énergétiquement efficace de Trivium est réalisé de manière similaire à l'implémentation du masquage TI au premier ordre décrite précédemment, à laquelle est rajoutée de la logique de contrôle permettant de désactiver le masquage à la fin de la phase d'initialisation de Trivium. De plus, dans l'implémentation du masquage au premier ordre de Trivium décrite dans [138], l'ensemble de l'état interne de Trivium est masqué, ce qui nécessite 576 bits aléatoires. Or, sur les 288 bits d'état interne, 160 sont occupés par la clé et l'IV lors du chargement des données dans Trivium, tandis que 128 bits sont identiques à toutes les exécutions et indépendants de la clé et de l'IV. Ces 128 bits constants ne donnent pas d'information utile lors d'une attaque par canaux auxiliaire, et le masquage de ces bits est donc a priori inutile. Afin d'économiser de la surface, de l'énergie et des bits aléatoires, seuls les 160 bits de clé et d'IV sont masqués lors du chargement des données dans l'implémentation du masquage énergétiquement efficace de Trivium. Ainsi, 244 bits d'aléa sont économisés par rapport au masquage présenté dans [138].

Le masquage TI simple est représenté avec des traits pleins noirs sur la figure 5.1, et les modifications apportées pour le masquage énergétiquement efficace sont représentées en traits pointillés rouges. Durant l'initialisation, les trois chemins de données, c'est-à-dire les trois registres à décalage en parallèle, contiennent les parts S^1 à S^3, dont la somme avec des portes XOR est égale à l'état interne S d'un Trivium non masqué. Le principe du masquage énergétiquement efficace est d'additionner ces trois parts dans un seul chemin de données une fois la phase d'initialisation terminée, et d'effectuer ensuite l'ensemble des calculs sur l'état interne non masqué contenu dans ce chemin de données, les deux autres étant désactivés.

Pour cela, deux blocs fonctionnels sont rajoutés par rapport au masquage simple. Le *Démasquage* consiste à retirer les masques en additionnant avec un XOR bit à bit le contenu des trois parts. En sortie de celui-ci, la *fonction non protégée* est implémentée suivant l'équation (5.1). Un multiplexeur permet de sélectionner soit la sortie masquée de la *fonction partagée*, soit la sortie non masquée de la *fonction non protégée*, afin de mettre à jour l'état interne de Trivium. Le *Démasquage* et le multiplexeur sont contrôlés par un signal *M_ctrl*, qui vaut "1" durant l'initialisation et "0" durant le chiffrement, lui-même contrôlé par le compteur qui détecte la fin de la phase d'initialisation. Le contrôle avec le signal *M_ctrl* du *Démasquage* permet d'additionner les parts uniquement lorsque *M_ctrl* vaut "0", et d'éviter ainsi les fuites par canaux auxiliaires lorsque le

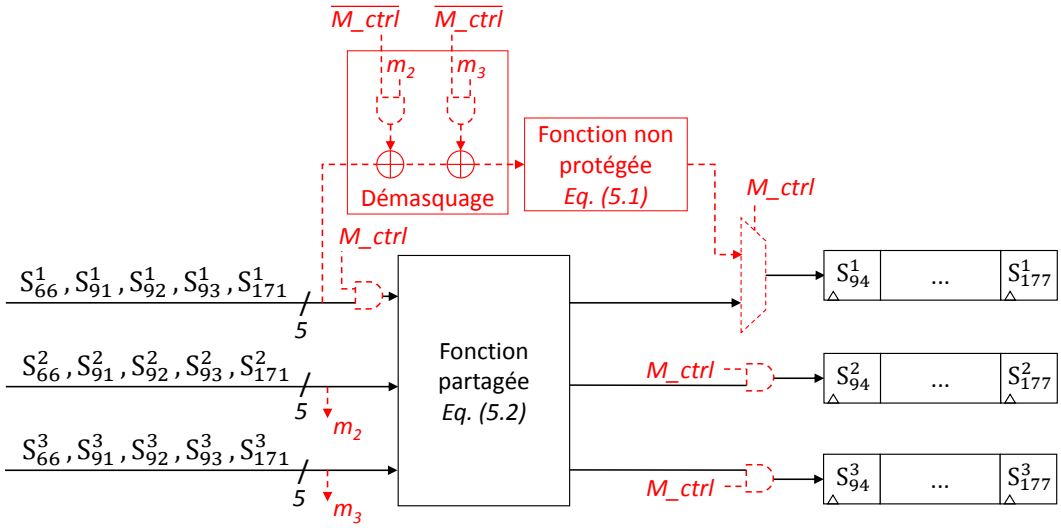


FIGURE 5.1 – Masquage TI au premier ordre de trivium (en noir), et modifications apportées pour le masquage énergétiquement efficace (en rouge).

masquage est actif.

Ce signal de contrôle permet également de mettre à zéro les chemins de données correspondant au masquage, afin de diminuer la consommation. Trois portes ET sont utilisées dans ce but. Deux d'entre elles permettent de mettre à zéro les entrées des deux chemins de données inutilisés après l'initialisation. Cette opération étant effectuée bit par bit et le plus long des registres constituant Trivium contenant 111 bascules, il faut 111 cycles d'horloge pour retirer progressivement les masques. À l'issue de ces 111 cycles, les deux chemins de données inutilisés contiennent uniquement des zéros, et du *clock gating* est appliqué à ceux-ci. Les parts S^2 et S^3 sont donc nulles, et la troisième porte ET permet de mettre à zéro la dernière entrée non nulle de la *fonction partagée*, afin d'éviter les commutations inutiles au sein de celle-ci et d'économiser ainsi plus d'énergie.

5.3 Masquage adaptatif

Le masquage énergétiquement efficace présenté dans la section précédente permet de protéger l'implémentation de Trivium contre les attaques par canaux auxiliaires publiées à ce jour contre l'initialisation de cet algorithme, avec un coût faible en énergie. Cependant, cette protection n'est pas adaptative, dans la mesure où il n'est pas possible de choisir le niveau de sécurité fourni. Un objet connecté pouvant avoir une durée de vie de plusieurs années, les attaques peuvent évoluer, et il est possible que de futures attaques puissent également être effectuées après la phase d'initialisation, en exploitant les fuites d'information par canaux auxiliaires montrées dans la section 3.2.3 lors de la phase de chiffrement. De plus, cette protection ne protège pas des attaques d'ordre supérieur contre le masquage. Les auteurs de [135] montrent des fuites par canaux auxiliaires aux ordres supérieurs sur une version masquée de l'algorithme de chiffrement par bloc KATAN, dont la structure interne s'inspire directement de celle de Trivium, et on peut donc s'attendre à retrouver les mêmes fuites avec Trivium. Par conséquent, il est nécessaire de protéger Trivium avec un masquage adaptatif, capable de protéger son implémentation à la fois pendant l'initialisation et le chiffrement, contre des attaques au premier ordre ou à des ordres supérieurs. Enfin, dans le cas où des économies importantes d'énergie doivent être réalisées et où le niveau de sécurité peut être faible, ce masquage adaptatif devrait permettre d'obtenir des gains en énergie plus élevés que le masquage énergétiquement efficace, grâce à la possibilité de désactiver le masquage également pendant l'initialisation.

Nous proposons donc un masquage adaptatif de Trivium, pouvant être activé ou désactivé à la demande sans impacter le temps d'exécution de Trivium, avec trois niveaux de sécurité. Au premier niveau, le masquage est désactivé, tandis que les deux niveaux supérieurs consistent res-

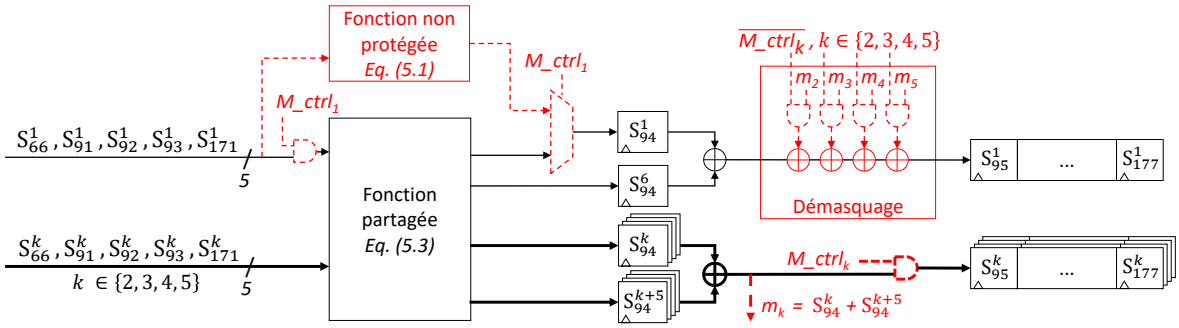


FIGURE 5.2 – Masquage TI au second ordre de trivium (en noir), et modifications apportées pour le masquage adaptatif (en rouge).

pectivement en un masquage au premier et au second ordre. Les mécanismes décrits pour ce masquage adaptatif peuvent être étendus de manière similaire pour la protection adaptative aux ordres supérieurs, et notre solution peut donc être adaptée selon les besoins pour fournir plusieurs niveaux de sécurité supplémentaires différents.

La *fonction partagée* au second ordre suit l'équation (5.3) ; en particulier, les trois premières égalités de celle-ci correspondent à un masquage au premier ordre. Le principe du masquage adaptatif est donc d'activer ou de désactiver un certain nombre de parts, en fonction du niveau de sécurité requis. Seule la part 1 est activée en l'absence de masquage, tandis que les parts 1 à 3 sont utilisées pour le masquage au premier ordre, et l'activation de toutes les parts permet d'effectuer un masquage au second ordre. Lors de l'activation ou de la désactivation de parts durant l'exécution de Trivium, il est nécessaire de s'assurer que l'exactitude de l'état interne est préservée, c'est-à-dire que la somme des parts est toujours égale à l'état interne non masqué. Les mécanismes permettant d'effectuer un tel masquage adaptatif de Trivium sont détaillés par la suite.

5.3.1 Contrôle du niveau de sécurité

Le choix du niveau de sécurité est effectué de manière extérieure, et transmis à une machine à états finis qui génère deux signaux de contrôle : *M_ctrl* contrôle le masquage adaptatif lors de la diminution du niveau de sécurité, et *M_init* gère le chargement de nouveaux masques lors de l'augmentation du niveau de sécurité. Cette machine à états finis gère également l'application de *clock gating* sur les chemins de données inutilisés.

Le signal *M_init* est sur 1 bit, tandis que *M_ctrl* est sur 5 bits, soit un bit de contrôle par part du masquage. Dans notre proposition de masquage adaptatif, les parts 2 et 3 servent à effectuer le masquage au premier ordre, et les parts 4 et 5 sont utilisées pour le masquage au second ordre. Les parts 2 et 3 sont donc contrôlées de la même manière, tout comme les parts 4 et 5, et un signal *M_ctrl* sur 3 bits, avec un bit permettant de contrôler chaque niveau de protection, serait donc suffisant. Cependant, nous conservons un signal *M_ctrl* de 5 bits pour la clarté des explications et des figures.

5.3.2 Réduction du niveau de sécurité

Le masquage TI au second ordre de Trivium est représenté avec des traits pleins noirs sur la figure 5.2. Les modifications nécessaires pour la désactivation du masquage adaptatif sont représentées en rouge sur cette figure.

De même que pour le masquage énergétiquement efficace, des portes ET sont rajoutées sur chaque chemin de données excepté le premier, afin de mettre progressivement à zéro les parts non utilisées lors de la réduction du niveau de sécurité. Ces portes ET sont contrôlées séparément pour chaque part, et il est donc possible de mettre à zéro uniquement les parts 4 et 5 pour passer d'un masquage au second ordre à un masquage au premier ordre, ou les parts 2 à 5 pour désactiver totalement le masquage. En parallèle, les bits mis à zéro sur les parts 2 à 5 (ou 4 et 5) sont addition-

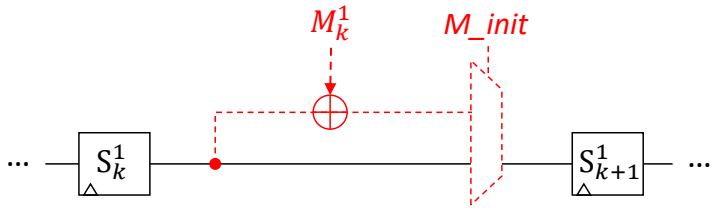


FIGURE 5.3 – Mécanisme permettant d'additionner un masque M^1 au contenu du premier chemin de données, entre les k-ième et $(k+1)$-ième bascules.

nés à la part 1 dans le bloc de *Démasquage*, afin de garantir l'exactitude du résultat. On notera que, contrairement au masquage énergétiquement efficace, le *Démasquage* est effectué sur le premier chemin de données simultanément à la mise à zéro avec les portes ET sur les autres chemins de données, c'est-à-dire après la *fonction partagée*, entre les bits d'état interne S_{94} et S_{95}. Cela permet non seulement de garantir l'exactitude du résultat dans ce cas, mais également d'optimiser la surface et la consommation de cette opération. Les quatre derniers bits du signal de contrôle *M_ctrl*, notés M_ctrl_2 à M_ctrl_5, permettent de contrôler à la fois la mise à zéro sur les chemins de données inutilisés et le démasquage sur le premier chemin de données.

Le premier bit de ce signal de contrôle, noté M_ctrl_1, permet de sélectionner soit la sortie de la *fonction partagée* si le masquage est activé (au premier ou au second ordre), soit celle de la *fonction non protégée* dans le cas contraire. Comme la désactivation du masquage s'effectue bit par bit sur une durée de 111 cycles, la somme de toutes les parts n'est ramenée sur le premier chemin de données qu'après ce délai. Ce n'est donc qu'après ces 111 cycles que la *fonction non protégée* peut être utilisée tout en conservant l'exactitude du résultat. Par conséquent, le premier bit de *M_ctrl* n'est mis à zéro qu'après ces 111 cycles; il vaut "1" lorsque le masquage est actif à l'ordre 1 ou 2 ainsi que pendant les 111 cycles d'horloge nécessaires à la désactivation du masquage. Enfin, au terme de ces 111 cycles, du *clock gating* est appliqué aux chemins de données non utilisés, et la porte ET sur le premier chemin de données avant la *fonction partagée* permet de mettre à zéro l'unique entrée non nulle de cette fonction dans le cas où le masquage est désactivé, afin d'éviter les commutations au sein de celle-ci.

On notera que d'après l'équation (5.3), les parts de sortie 6 à 10 de la *fonction partagée* dépendent des parts 4 et 5, et elles sont donc nulles en l'absence de masquage au second ordre. Par conséquent, il est inutile de rajouter de la logique de contrôle ou de mise à zéro sur les bascules contenant ces parts 6 à 10, hormis le *clock gating* permettant de les désactiver.

5.3.3 Augmentation du niveau de sécurité

La réduction du niveau de protection présentée précédemment s'effectue progressivement et dure 111 cycles d'horloge. L'augmentation du niveau de protection, quant à elle, peut répondre à des contraintes urgentes en sécurité, telles que la détection d'une attaque en cours grâce à divers capteurs. Par conséquent, nous proposons de réaliser l'augmentation de ce niveau de sécurité sur un seul cycle d'horloge.

Pour cela, de nouveaux masques aléatoires sont chargés simultanément dans deux ou quatre des chemins de données non utilisés, en fonction des niveaux de sécurité respectifs avant et après l'augmentation de ce niveau. Au cycle d'horloge suivant, lors du décalage de l'état interne dans les registres, l'un de ces masques est additionné avec des portes XOR à l'ensemble de l'état contenu dans le premier chemin de données, afin de protéger les données contenues dans celui-ci et de garantir l'exactitude de l'état interne. En passant d'un masquage désactivé, où l'ensemble de l'information est contenue sur S^1, à un masquage à l'ordre 2 sur les cinq parts, quatre masques aléatoires

de 288 bits sont nécessaires, notés M^i, $i \in \{1,\ldots,4\}$, et l'équation suivante est utilisée :

$$\begin{aligned} S^1 &= S^1 + M^1 & S^2 &= M^1 + M^2 \\ S^3 &= M^2 + M^3 & S^4 &= M^3 + M^4 \\ S^5 &= M^4 \end{aligned} \tag{5.5}$$

Lors de l'activation d'un masquage au premier ordre, on rajoute seulement deux masques aléatoires M^1 et M^2 afin de créer trois parts :

$$\begin{aligned} S^1 &= S^1 + M^1 & S^2 &= M^1 + M^2 \\ S^3 &= M^2 & S^4 &= 0 \\ S^5 &= 0 \end{aligned} \tag{5.6}$$

Enfin, lorsque le niveau de protection est augmenté du premier au second ordre, on rajoute également deux masques aléatoires M^3 et M^4, afin d'obtenir un masquage sur cinq parts :

$$\begin{aligned} S^1 &= S^1 + M^3 & S^2 &= S^2 \\ S^3 &= S^3 & S^4 &= M^3 + M^4 \\ S^5 &= M^4 \end{aligned} \tag{5.7}$$

Cette approche requiert l'ajout de portes XOR et de multiplexeurs avant chacune des 288 bascules sur le premier chemin de données. Ces multiplexeurs sont contrôlés par le signal *M_init*. Ce mécanisme est représenté sur la figure 5.3. En augmentant le niveau de sécurité de cette manière, il n'est pas nécessaire de modifier la logique combinatoire et la *fonction de rétroaction* de Trivium.

On notera qu'il est inutile de charger des bits de masques dans les parts 6 à 10 lors de l'augmentation du niveau de sécurité. Par conséquent, de même que pour le mécanisme de diminution du niveau de sécurité, il n'est pas nécessaire de rajouter de logique de contrôle sur ces parts.

5.4 Encodage dynamique adaptatif

Le masquage algorithmique protège uniquement l'implémentation d'un algorithme contre les attaques par analyse des canaux auxiliaires. Cette contre-mesure pourrait être combinée à de la redondance afin de détecter les fautes, mais le coût d'une telle combinaison de contre-mesures serait très important ; par exemple, dans le cas de la redondance spatiale, la duplication des cinq chemins de données du masquage au second ordre aboutirait à une implémentation ayant une consommation et une surface supérieures à 10 fois celles du Trivium non protégé. Nous proposons donc d'étendre le principe de sécurité adaptative à l'encodage dynamique, présenté dans le chapitre 4. Dans cette section, nous détaillons les modifications à apporter à l'encodage dynamique 1-sur-4 des registres de Trivium, noté TRI_ENC_1 dans le chapitre 4, afin de rendre adaptative cette implémentation. Contrairement au masquage adaptatif, seuls deux niveaux de sécurité sont fournis par l'encodage adaptatif : cette contre-mesure est soit activée, soit désactivée. Par conséquent, cette implémentation est plus simple que celle du masquage adaptatif, notamment en ce qui concerne la logique combinatoire. Un unique signal de contrôle sur un bit, généré de manière externe et noté *E_ctrl*, est nécessaire pour gérer l'encodage dynamique adaptatif.

De même que pour le masquage adaptatif, on considère un cas d'usage où la contre-mesure peut être désactivée progressivement, mais où sa réactivation doit être effectuée sur un seul cycle d'horloge. Les fonctions d'encodage $\mathscr{F}_c$ et de décodage $\mathscr{F}_c^{-1}$, appliquées respectivement sur les entrées et sorties des registres encodés, sont modifiées afin de rendre cette protection adaptative. Lorsque la contre-mesure est activée, ces fonctions effectuent respectivement l'encodage et le décodage de la manière décrite dans la section 4.1. Lorsqu'elle est désactivée, les entrées sont écrites sur le premier chemin de données uniquement, qui contient alors toute l'information utile, tandis que les trois autres chemins de données sont mis à zéro et désactivés.

La désactivation de l'encodage dynamique se fait progressivement, grâce à la fonction $\mathscr{F}_c$ modifiée qui écrit les données non encodées dans le premier chemin de données. Au cours de ce processus, qui dure 111 cycles, les fonctions de décodage $\mathscr{F}_c^{-1}$ pour les différentes sorties sont donc également désactivées progressivement : pour une sortie après k bascules dans un registre à décalage, le décodage se fait pendant k cycles sur des bits encodés, avant d'être effectué sur des bits non encodés.

L'activation de la contre-mesure est effectuée en un seul cycle d'horloge, et ces deux fonctions changent alors instantanément de mode de fonctionnement afin d'écrire et de traiter des données encodées dans les quatre chemins de données. De plus, l'information non encodée contenue sur la seule part S^1 doit être encodée sur les quatre parts S^1 à S^4. Afin de charger des données encodées dans ces quatre chemins de données, un mécanisme similaire à celui permettant l'initialisation de Trivium avec un état interne préalablement encodé, détaillé dans la section 4.5.1, est utilisé. Cependant, au lieu d'appliquer l'encodage à un vecteur externe de 288 bits comme pour l'initialisation, cet encodage est appliqué à l'état interne contenu dans le premier chemin de données, c'est-à-dire à S^1. Ainsi, l'activation de l'encodage dynamique durant l'exécution de Trivium se fait selon la formule suivante :

$$\begin{aligned} S^1 &= S^1 \cdot \texttt{0x5555555555...5555555555} \\ S^2 &= S^1 \cdot \texttt{0xAAAAAAAAAA...AAAAAAAAAA} \\ S^3 &= \overline{S^1} \cdot \texttt{0x5555555555...5555555555} \\ S^4 &= \overline{S^1} \cdot \texttt{0xAAAAAAAAAA...AAAAAAAAAA} \end{aligned} \tag{5.8}$$

Plusieurs versions de l'encodage dynamique ont été présentées dans le chapitre 4, comme la génération de tables d'encodage aléatoires à chaque nouvelle initialisation, ou l'encodage de la logique combinatoire et séquentielle. L'encodage dynamique aléatoire est compatible avec les mécanismes présentés plus haut, et peut être implémenté de manière adaptative ; les modifications à effectuer dans ce cas sont identiques à celles décrites dans la section 4.3 pour passer d'un encodage dynamique fixé à un encodage dynamique aléatoire. L'implémentation adaptative de l'encodage dynamique de la logique combinatoire nécessiterait d'intégrer une *fonction non protégée* et une fonction encodée en parallèle, avec un multiplexeur permettant de choisir les signaux appropriés, de la même manière que pour le masquage adaptatif.

Enfin, on notera que l'encodage dynamique aurait aussi pu être implémenté de manière semi-adaptative, énergétiquement efficace, de même que l'implémentation de masquage détaillée dans la section 5.2. Cet encodage dynamique semi-adaptatif aurait une architecture quasiment identique à celle présentée ici pour l'encodage adaptatif, et nécessiterait uniquement la modification des fonctions $\mathscr{F}_c$ et $\mathscr{F}_c^{-1}$.

5.5 Caractérisation et évaluation des contre-mesures

Toutes les contre-mesures matérielles décrites dans ce chapitre sont implémentées et simulées dans le nœud technologique 28 nm FDSOI, de la même manière que le Trivium non protégé décrit dans la section 3.2, c'est-à-dire avec le même flot de conception et les mêmes contraintes et paramètres pour les outils de synthèse, layout et simulations. La surface et la consommation de puissance sont également évaluées de la manière décrite dans la section 3.2, afin d'estimer le coût des contre-mesures adaptatives, ainsi que les potentiels gains en énergie apportés par l'approche adaptative. Enfin, l'efficacité des différentes contre-mesures contre les attaques par canaux auxiliaires est également analysée selon la méthodologie décrite précédemment.

Par la suite, nous notons TI_1 et TI_2 les implémentations du masquage TI sur Trivium respectivement au premier et au second ordre, TI_EFF le masquage énergétiquement efficace au premier ordre décrit dans la section 5.2, et TI_ADAPT le masquage adaptatif au second ordre décrit dans la section 5.3. L'encodage adaptatif détaillé dans la section 5.4 est noté TRI_ENC_ADAPT, et le Trivium non protégé de référence est noté TRIVIUM.

TABLEAU 5.1 – Surface après synthèse (GE) pour les contre-mesures adaptatives.

Implémentation	Surface séquentielle	Surface combinatoire	Surface totale	Surface normalisée
TRIVIUM	2212	595	2808	1
TI_1	6628	3149	9778	3,48
TI_2	11159	6032	17191	6,12
TI_EFF	6732	2602	9334	3,32
TI_ADAPT	11252	8253	19506	6,95
TRI_ENC_ADAPT	9073	3152	12224	4,35

TABLEAU 5.2 – Consommation moyenne simulée (µW) pour les contre-mesures adaptatives.

Implémentation	Niveau de protection	Consommation moyenne	Consommation normalisée
TRIVIUM	-	241	1
TI_1	-	727	3,02
TI_2	-	1230	5,1
TI_EFF	$Aucun$	256	1,06
	$1^{er} ordre$	741	3,07
TI_ADAPT	$Aucun$	293	1,22
	$1^{er} ordre$	791	3,28
	$2^{nd} ordre$	1287	5,34
TRI_ENC_ADAPT	$Aucun$	282	1,17
	$Protégé$	1032	4,28

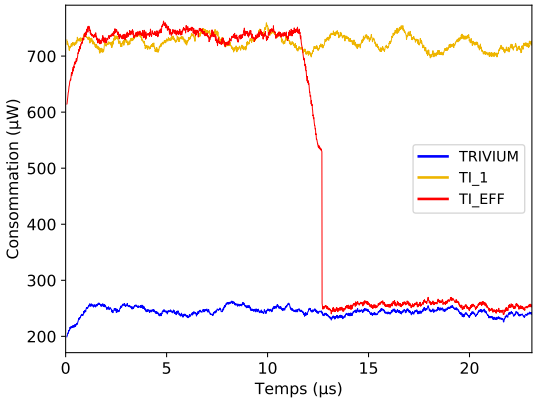
5.5.1 Surface

Les surfaces des différentes contre-mesures décrites dans ce chapitre, ainsi que celle du Trivium non protégé de référence, sont données en GE (*Gate Equivalent*) dans le tableau 5.1. Ce tableau donne, pour chaque circuit, la surface de la logique combinatoire et séquentielle, la surface totale, ainsi que la surface normalisée des contre-mesures par rapport au Trivium de référence. Pour rappel, les surfaces en μm^2 peuvent être obtenues en multipliant par 0,4896 les surfaces fournies en GE.

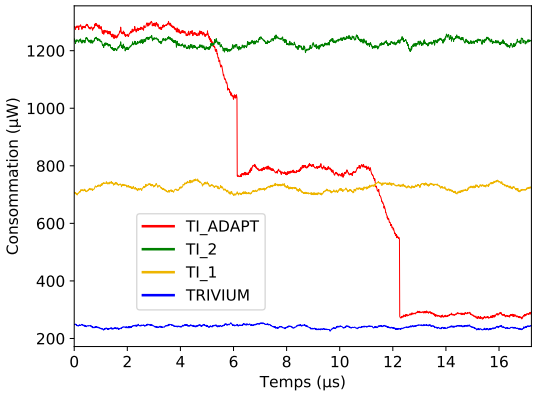
Les contre-mesures adaptatives ont globalement un coût silicium plus élevé que les contre-mesures classiques, c'est-à-dire non adaptatives. On observe en particulier que ces contre-mesures adaptatives ont une surface de logique combinatoire significativement plus élevée que les contre-mesures classiques. Ce surcoût provient d'une part de la logique supplémentaire de contrôle, comme les machines à états finis et les compteurs, et d'autre part des modifications effectuées sur les chemins de données, permettant par exemple une augmentation du niveau de sécurité en un seul cycle d'horloge. Ainsi, la surface totale du masquage adaptatif est supérieure de 13,5 % à celle d'un masquage TI classique au second ordre, tandis que la surface de l'encodage dynamique adaptatif est supérieure de 11,3 % à celle d'un encodage dynamique simple (donnée dans la section 4.5.2 et égale à 10985 GE). L'exception est le masquage au premier ordre énergétiquement efficace, TI_EFF, qui présente une surface totale plus faible de 4,5 % par rapport au masquage de référence au premier ordre TI_1 initialement proposé dans [138], grâce à la réduction de la taille des masques dans nos travaux par rapport à ce masquage de référence.

5.5.2 Consommation

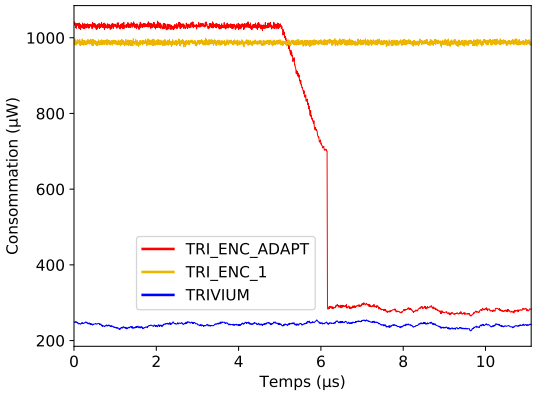
La consommation moyenne de puissance pour chaque implémentation est donnée dans le tableau 5.2. La consommation moyenne est calculée pour chaque niveau de protection possible pour le masquage énergétiquement efficace TI_EFF, le masquage adaptatif TI_ADAPT, et l'enco-



(a) Masquage énergétiquement efficace



(b) Masquage adaptatif



(c) Encodage adaptatif

FIGURE 5.4 – Évolution de la consommation des différentes contre-mesures adaptatives par rapport à d'autres contre-mesures ayant un niveau de protection similaire.

dage dynamique adaptatif TRI_ENC_ADAPT. De même que précédemment pour la surface, une consommation moyenne normalisée par rapport au Trivium de référence est donnée pour chaque contre-mesure et chaque niveau de protection.

On constate que les contre-mesures adaptatives présentent un faible surcoût en consommation par rapport à des contre-mesures équivalentes et non adaptatives, si on considère un niveau de sécurité constant tout au long de la durée de vie d'un circuit intégré. Par exemple, l'encodage dynamique adaptatif TRI_ENC_ADAPT, lorsqu'il est activé, consomme 4,6 % plus de puissance que l'encodage dynamique simple TRI_ENC_1 (dont la consommation, égale à 987 µW, est donnée dans la section 4.5.2), tandis que le masquage adaptatif au second ordre TI_ADAPT consomme également 4,6 % plus que le masquage TI au second ordre. Cependant, ces contre-mesures adaptatives présentent un réel intérêt lorsque le niveau de sécurité à appliquer varie ; par exemple, lorsque le masquage adaptatif est désactivé, 60 % de la consommation d'un masquage TI au premier ordre, ou 76 % de celle d'un masquage TI au second ordre, est économisée.

La figure 5.4 permet d'illustrer le gain en consommation apporté par la réduction du niveau de sécurité, et donc l'intérêt des contre-mesures adaptatives. La figure 5.4a montre un exemple de l'évolution de la consommation moyenne, par cycle d'horloge, d'un Trivium non protégé, d'un Trivium protégé au premier ordre, et d'un autre protégé avec notre masquage énergétiquement efficace, sur les $2 \times 1152 = 2304$ premiers cycles d'horloge d'exécution de Trivium. On constate qu'après la phase d'initialisation de Trivium, soit 1152 cycles d'horloge, la consommation du circuit TI_EFF décroît sur 111 cycles d'horloge, avant de chuter brutalement lors de l'application de *clock gating* aux chemins de données dédiés au masquage. De la même manière, la figure 5.4b présente la consommation d'un Trivium non protégé, et protégé au premier et au second ordre par des masques TI classiques. La courbe en rouge montre un Trivium protégé avec notre mas-

quage adaptatif, successivement au second et au premier ordre, puis avec le masquage désactivé. Enfin, la figure 5.4c présente la consommation d'un Trivium non protégé et protégé avec un encodage dynamique, ainsi que l'encodage dynamique adaptatif qui est successivement activé puis désactivé. Sur l'ensemble de la durée de vie d'un circuit, ces réductions de la consommation de puissance instantanée peuvent entraîner une réduction significative de la consommation énergétique.

5.5.3 Fuites d'information par canaux auxiliaires

Dans cette section, nous analysons les fuites par canaux auxiliaires des différentes implémentations avec des T-tests non-spécifiques. La méthodologie suivie jusqu'à présent permet de détecter des fuites d'information au premier ordre, c'est-à-dire sur des implémentations non masquées. Nous avons décrit dans la section 3.4.2 les attaques d'ordre supérieur, qui permettent à un attaquant de récupérer des informations sur les données traitées dans les différentes parts du masquage. De même que pour les attaques CPA, il est possible de calculer des T-tests d'ordre supérieur, permettant de détecter ces fuites d'ordre supérieur en présence de masquage algorithmique.

Un T-test d'ordre supérieur est obtenu en pré-traitant les traces avant de calculer le T-test, de la même manière que pour effectuer une attaque CPA d'ordre supérieur. Un T-test *univarié* d'ordre d est obtenu en mettant les traces à la puissance d afin d'exhiber leur d-ième moment statistique, et un T-test *multivarié* d'ordre d peut être calculé en combinant de manière non-linéaire d mesures, effectuées à plusieurs instants différents, sur chaque trace. Par exemple, pour un T-test univarié d'ordre 2, chaque trace est mise au carré avant d'effectuer les calculs. Lors d'un T-test multivarié d'ordre 2, également appelé T-test bivarié, les points de chaque trace sont multipliés deux à deux, afin de détecter les informations sensibles manipulées à deux instants différents.

Des fuites univariées d'ordre supérieur sont généralement observées pour des implémentations matérielles du masquage telles que celles décrites dans ce chapitre, où le masque est traité en parallèle des données masquées. Bien que les fuites multivariées soient en général plutôt visibles pour des implémentations logicielles, où les données masquées et les masques sont traités à des instants différents [43], certains travaux ont montré l'existence de telles fuites sur des implémentations matérielles de masquage TI d'ordre supérieur [130, 165, 78], et nous étudions donc également ces fuites multivariées par la suite. L'étude des fuites d'ordre supérieur n'est pertinente que pour des implémentations du masquage algorithmique, et nous ne calculons donc pas de T-test d'ordre supérieur pour le Trivium non protégé ni pour celui protégé avec l'encodage dynamique adaptatif.

Les lots de traces sont constitués de la même manière que pour les T-tests calculés pour l'encodage dynamique et décrits dans la section 4.5.3. Chaque lot contient 100.000 traces, chacune de ces traces couvrant les 85 premiers cycles d'horloge de l'exécution de Trivium. Les masques sont générés aléatoirement pour chaque trace.

T-test univarié sur les différentes implémentations

La figure 5.5 représente les variations du T-test non-spécifique pour le Trivium de référence, et pour les deux contre-mesures adaptatives lorsque les protections sont désactivées. Des fuites importantes sont observées pour ces trois circuits. La similarité des courbes dans ces trois cas s'explique par le fait que les clés et IV sont identiques pour les lots fixes des trois T-tests.

La figure 5.6 montre les variations des T-tests pour toutes les contre-mesures adaptatives et semi-adaptatives appliquées à Trivium, aux différents ordres de protection pour le masquage adaptatif. Des T-tests d'ordres supérieurs permettent d'évaluer les fuites d'information d'ordre supérieur des implémentations masquées, sur les figures 5.6a à 5.6e. Le masquage TI au second ordre étant implémenté avec cinq parts contenues dans cinq chemins de données en parallèle, une combinaison des données contenues dans ces cinq parts serait nécessaire afin d'extraire de l'information par canaux auxiliaires, ce qui correspond à des fuites univariées d'ordre 5. Par conséquent, nous étudions les fuites d'information jusqu'à l'ordre 5. Ces figurent montrent qu'avec 100.000

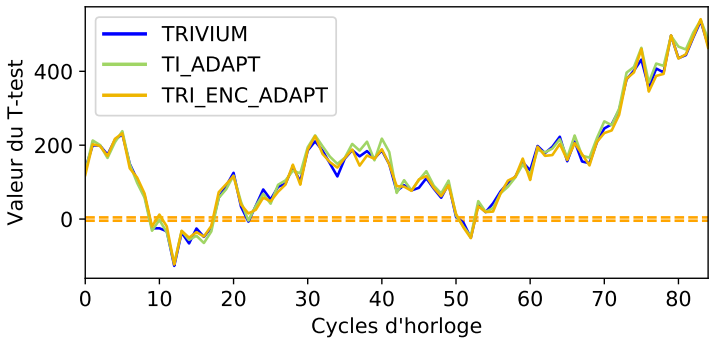
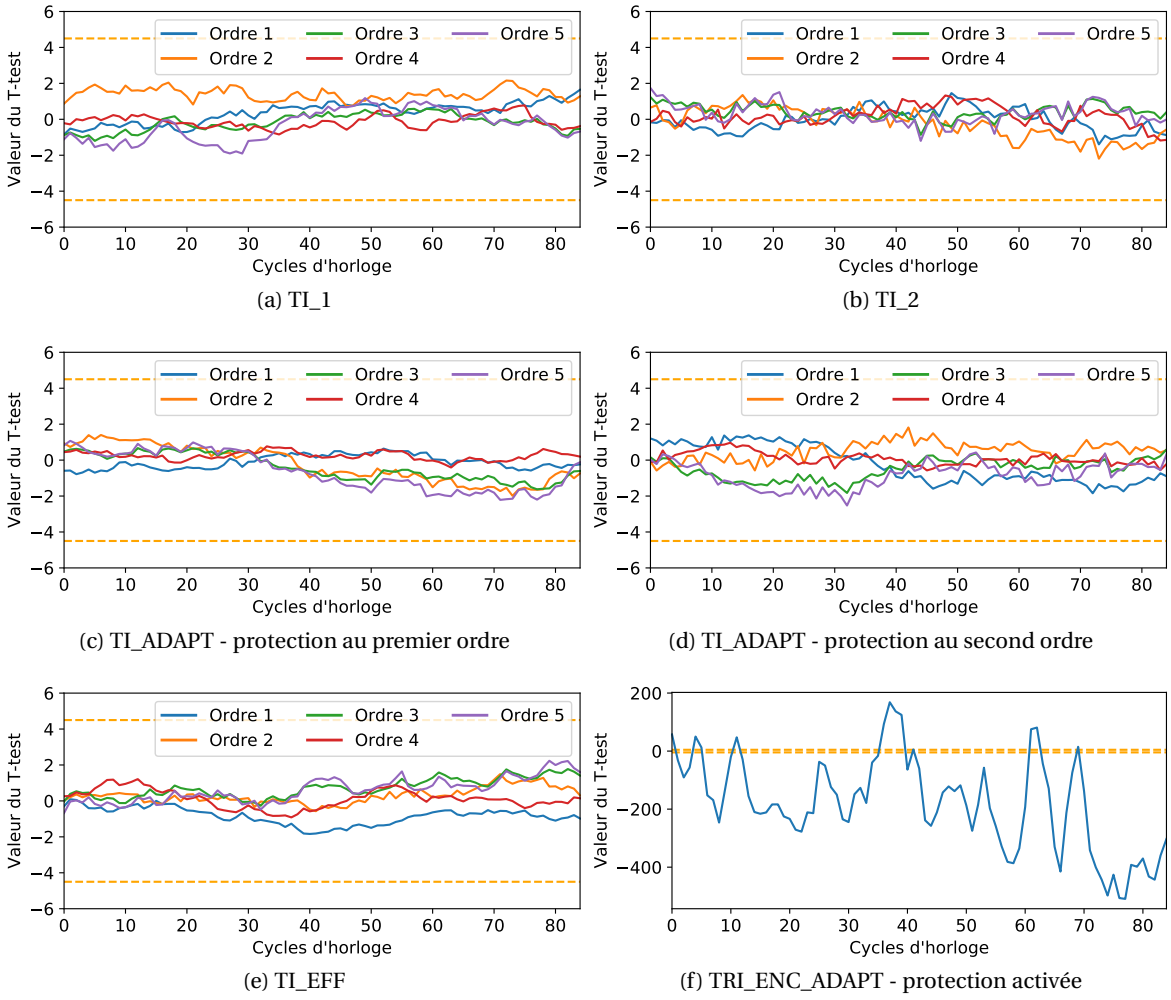


FIGURE 5.5 – Valeurs du T-test pour le Trivium de référence, et lorsque les protections adaptatives sont désactivées.



(a) TI_1

(b) TI_2

(c) TI_ADAPT - protection au premier ordre

(d) TI_ADAPT - protection au second ordre

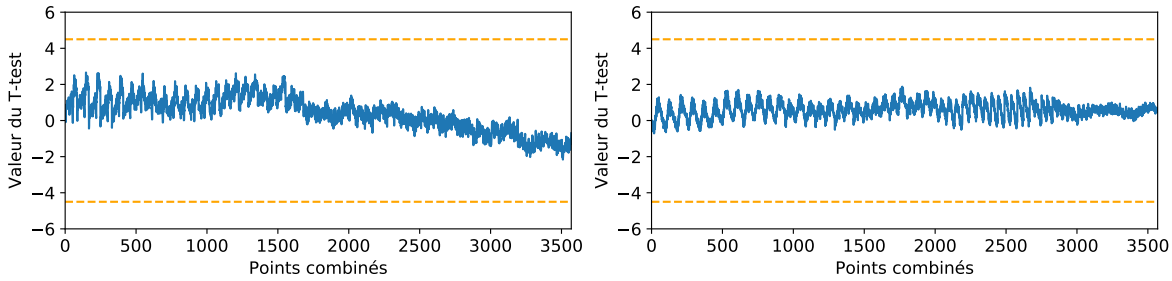
(e) TI_EFF

(f) TRI_ENC_ADAPT - protection activée

FIGURE 5.6 – Valeurs du T-test pour les différentes contre-mesures adaptatives appliquées à Trivium.

traces, il n'y a aucune fuite d'information pour toutes les contre-mesures à base de masquage. Un nombre de traces plus élevé serait sans doute nécessaire pour observer des fuites par canaux auxiliaires, ce qui nécessiterait des temps de simulation très importants avec les outils utilisés, allant jusqu'au plusieurs mois pour obtenir plusieurs millions de traces. En revanche, la figure 5.6f semble indiquer la présence de fuites d'information importantes lorsque Trivium est protégé par l'encodage dynamique adaptatif. Cette dernière observation concorde avec les résultats présentés dans le chapitre 4 à propos de l'encodage dynamique, et la génération aléatoire de tables d'encodage à chaque exécution de Trivium devrait permettre de réduire ces fuites d'information.

Nous rappelons ici la remarque effectuée dans la section 4.5.3 : sans un ajout d'aléa, c'est-à-



(a) TI_2 (b) TI_ADAPT

FIGURE 5.7 – T-test bivarié sur le masquage au second ordre de Trivium.

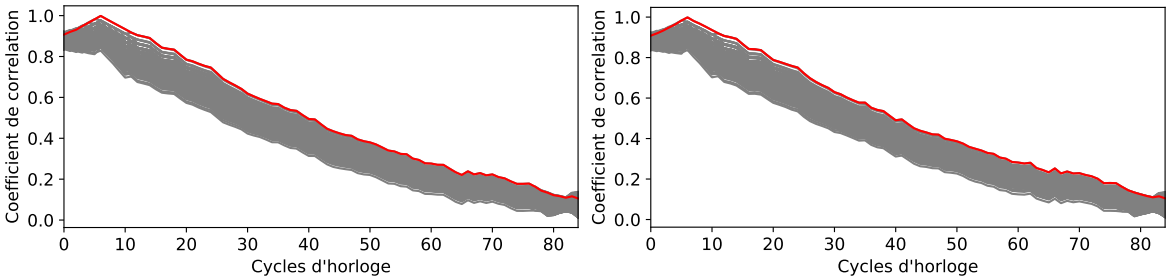
dire pour les implémentations qui ne sont pas protégés par le masquage algorithmique ou l'encodage aléatoire, les fuites sont probablement sur-évaluées du fait de l'absence de bruit dans les simulations. Ceci est donc valable à la fois pour le Trivium de référence et les contre-mesures adaptatives lorsqu'elles sont désactivées, mais également pour l'encodage dynamique adaptatif. Dans ces cas-là, une mesure expérimentale de traces bruitées permettrait de mieux évaluer les fuites par canaux auxiliaires. Dans le chapitre 6, nous validons grâce à des mesures effectuées sur FPGA les résultats présentés ici sur le Trivium non protégé et sur le masquage adaptatif lorsque la protection est désactivée. De futures mesures sur ASIC, sur un circuit également décrit dans le chapitre 6, permettront d'affiner les résultats pour l'encodage dynamique.

T-test multivarié sur le masquage au second ordre

Les auteurs de [130, 165] évoquent la possibilité d'observer des fuites multivariées par canaux auxiliaires en présence de masquage TI d'ordre supérieur. La présence de ces fuites multivariées a été démontrée expérimentalement dans [78]. Dans ces travaux, des fuites bivariées sont observées pour un NLFSR (*Non-Linear Feedback Shift Register*) de 4 bits implémenté sur FPGA avec un masquage TI au second ordre.

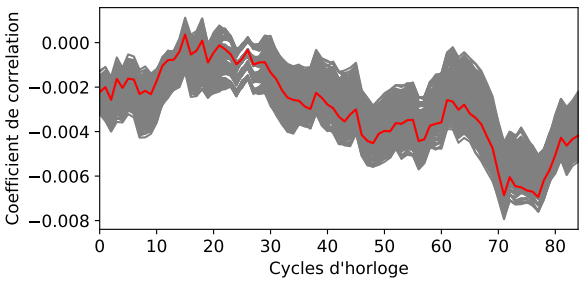
Nous analysons donc de la même manière les fuites bivariées sur le masquage TI au second ordre de Trivium, en utilisant les lots de traces préalablement obtenues pour le T-test univarié. Une analyse exhaustive est réalisée en considérant toutes les combinaisons possibles de deux points de chaque trace, soit $\binom{85}{2} = 3570$ combinaisons pour des traces originales de 85 cycles d'horloge. Pour cela, de nouvelles traces sont créées, chacune étant constituée des 3570 multiplications possibles de deux points d'une des traces originales. Un T-test non-spécifique est alors calculé avec ces nouvelles traces. Les figures 5.7a et 5.7b représentent respectivement les résultats du T-test bivarié pour un masquage TI simple de Trivium au second ordre, et pour le masquage adaptatif au second ordre. Celles-ci montrent l'absence de fuites bivariées avec 100.000 traces par lot. Compte tenu de la complexité de la *fonction de rétroaction* et de la longueur des registres à décalage de Trivium par rapport au NLFSR de 4 bits considéré dans [78], on peut s'attendre à ce que ces fuites bivariées, si elles existent pour Trivium, soient visibles avec plus de traces, et après une durée bien plus longue que 85 cycles.

Les fuites multivariées évoquées dans [130, 165, 78] proviendraient a priori de la *fonction de réduction*, qui consiste à réduire le nombre de parts en sortie de la *fonction partagée*. Dans tous ces travaux, la *fonction de réduction* utilisée est celle décrite par l'équation (3.5). Or, cette fonction a été implémentée différemment pour le masquage de Trivium, de manière plus équilibrée, avec l'équation (5.4) : au lieu d'additionner 6 des 10 parts sur une seule part de sortie et de laisser les 4 autres parts inchangées, toutes les parts sont additionnées deux à deux. Cette *fonction de réduction* différente pourrait donc également expliquer l'absence de fuites bivariées, bien que d'autres expériences doivent être menées afin de conclure à ce sujet.

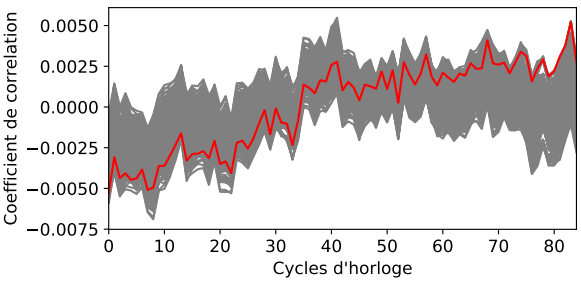


(a) TI_ADAPT - protection désactivée

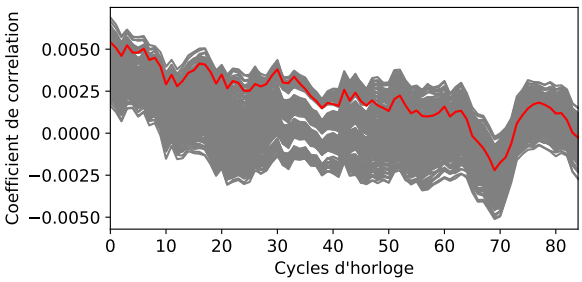
(b) TRI_ENC_ADAPT - protection désactivée



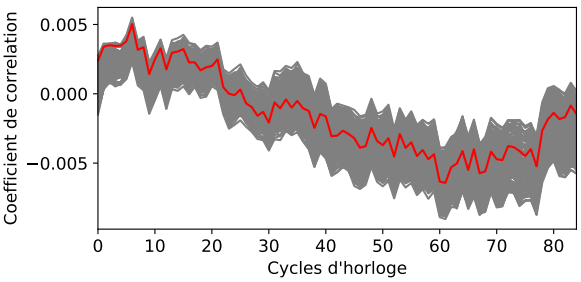
(c) TI_1



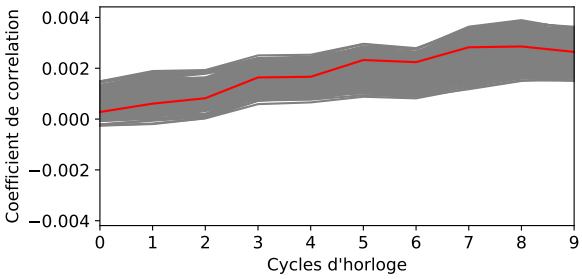
(d) TI_2



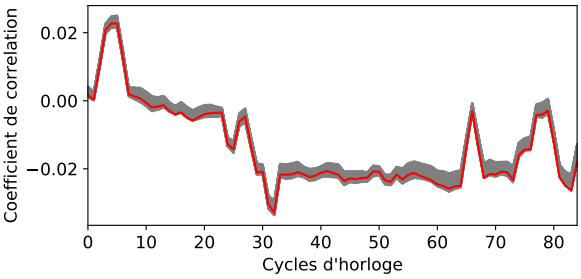
(e) TI_ADAPT - protection au premier ordre



(f) TI_ADAPT - protection au second ordre



(g) TI_EFF



(h) TRI_ENC_ADAPT - protection activée

FIGURE 5.8 – Évolution du coefficient de corrélation pour les contre-mesures adaptatives.

5.5.4 Corrélation et CPA

Dans cette section, nous évaluons les variations du coefficient de corrélation pour les différentes contre-mesures étudiées, aux différents niveaux de protection. La méthodologie adoptée est identique à celle décrite dans la section 3.2.2. En particulier, cette étude du coefficient de corrélation permet d'évaluer la possibilité d'exploiter les fuites d'information visibles sur la figure 5.6f pour l'encodage dynamique adaptatif. Bien qu'aucune fuite d'information n'ait été observée avec le T-test pour les implémentations masquées, nous étudions tout de même les variations du coefficient de corrélation pour ces implémentations, afin de souligner l'impossibilité d'effectuer une attaque CPA contre celles-ci.

La figure 5.8 montre l'évolution du coefficient de corrélation pour les 256 hypothèses effectuées sur l'octet de clé, pour chaque contre-mesure; la corrélation pour la bonne hypothèse de

TABLEAU 5.3 – Maximum du coefficient de corrélation pour les contre-mesures adaptatives.

Implémentation	Niveau de protection	Corrélation (valeur absolue)
TRIVIUM	-	0,8950
TI_1	-	0,0069
TI_2	-	0,0053
TI_EFF	-	0,0031
TI_ADAPT	$Aucun$	0,9986
	$1^{er} ordre$	0,0054
	$2^{nd} ordre$	0,0064
TRI_ENC_ADAPT	$Aucun$	0,9985
	$Protégé$	0,0329

clé y est affichée en rouge. On constate que pour toutes les implémentations protégées, la bonne hypothèse de clé n'est pas distinguable des 255 mauvaises hypothèses, ce qui indique la difficulté d'exploiter ces résultats afin d'effectuer une CPA. Les courbes de corrélation obtenues dans les cas où les contre-mesures adaptatives sont désactivées sont très similaires à celle obtenue dans la section 3.2.2 pour le Trivium de référence, avec un pic de corrélation permettant de distinguer la bonne hypothèse de clé.

La valeur maximale du coefficient de corrélation sur les 85 cycles d'horloge, en valeur absolue, est donnée dans le tableau 5.3 pour toutes les implémentations et les différents niveaux de protection ; la valeur maximale de ce coefficient pour le Trivium de référence, présentée dans la section 3.2.2, y figure également à titre de comparaison. Par rapport au Trivium non protégé, la valeur maximale du coefficient de corrélation est réduite par un facteur 129 à 288 pour les différentes implémentations du masquage. Cette valeur est réduite par un facteur 27 lorsque l'encodage dynamique adaptatif est activé. Dans tous les cas, on obtient des valeurs très faibles du coefficient de corrélation, ce qui dénote une absence de corrélation entre la consommation réelle et la consommation hypothétique pour la bonne hypothèse de clé, et donc l'impossibilité de mettre en œuvre une CPA, telle qu'elle existe actuellement, contre Trivium.

Afin de valider ces conclusions, une CPA itérative visant à retrouver bit par bit les 81 σ_i, telle que décrite dans la section 3.2, est effectuée pour chaque implémentation, et avec chaque niveau de sécurité possible. Cette CPA permet de retrouver la clé avec moins de 100 traces de consommation lorsque les protections adaptatives sont désactivées, tandis qu'elle échoue avec 100.000 traces pour toutes les versions protégées.

5.6 Implémentation de sécurité adaptative pour divers cas d'usage

5.6.1 Choix d'implémentation

Dans nos propositions de contre-mesures adaptatives et semi-adaptatives, nous avons fait le choix de désactiver les protections progressivement, sur 111 cycles d'horloge, et de les activer instantanément, en un seul cycle d'horloge. Ainsi, deux mécanismes différents sont mis en œuvre, ce qui augmente à la fois la surface et la consommation. Ce choix, arbitraire, dépend en réalité des cas d'usage, et certaines situations peuvent justifier une implémentation différente de ces contre-mesures adaptatives.

Il peut être envisageable de diminuer le niveau de sécurité en un seul cycle d'horloge, par exemple pour une réduction plus rapide de la consommation. Pour cela, le mécanisme déjà en place pour l'augmentation instantanée du niveau de sécurité peut être ré-utilisé dans le cas du masquage adaptatif. Les parts à désactiver peuvent être additionnées à la première part sur le premier chemin de données de la même manière que lors du rajout des masques présenté sur la figure 5.3 ; on tire ici partie du fait qu'une même addition avec une porte XOR permet à la fois d'ajouter et de retirer un masquage booléen. En parallèle, les chemins de données non utilisées

seraient mis à zéro. La désactivation sur un seul cycle d'horloge de l'encodage adaptatif serait plus complexe et plus coûteuse à implémenter, et nécessiterait le rajout d'un mécanisme de décodage de l'ensemble de l'état interne.

À l'inverse, l'augmentation du niveau de sécurité, si elle n'est pas effectuée en réponse à un besoin urgent comme la détection d'une attaque, mais par exemple suite à un changement dans la politique de sécurité du circuit, pourrait être réalisée de manière progressive sur 111 cycles d'horloge. Dans ce cas, il n'est pas nécessaire de rajouter de la logique de contrôle sur chaque bascule des registres à décalage du premier chemin de données. L'ensemble des modifications pour la mise en place d'une sécurité adaptative seraient alors effectuées au niveau de la *fonction de rétro-action* de Trivium et des entrées et sorties des registres à décalage, avec des mécanismes similaires et mutualisés pour la réduction et l'augmentation du niveau de sécurité.

Au final, des mécanismes semblables pour l'augmentation et la diminution du niveau de sécurité pourraient être utilisés en fonction des besoins. On peut alors s'attendre à une diminution de la logique de contrôle nécessaire pour implémenter la sécurité matérielle adaptative, particulièrement dans le cas où la modification du niveau de sécurité est effectuée de manière progressive. Cette diminution de la logique de contrôle s'accompagnerait d'une diminution des coûts en surface et en consommation des contre-mesures adaptatives par rapport aux contre-mesures classiques, c'est-à-dire non adaptatives.

5.6.2 Extension à d'autres algorithmes

Les algorithmes de chiffrement par flot peuvent générer des flots de clé très longs sans être ré-initialisés, tandis que leur phase d'initialisation peut avoir une durée importante, à l'image des 1152 cycles d'horloge de celle de Trivium. Les contre-mesures adaptatives présentées dans ce chapitre sont donc particulièrement intéressantes pour protéger de manière configurable l'implémentation de ces algorithmes de chiffrement par flot, avec la possibilité d'adapter le niveau de sécurité au cours du chiffrement sans nécessiter de ré-initialisation de ceux-ci. Une telle sécurité adaptative ou semi-adaptative peut également être intéressante pour la protection d'autres primitives cryptographiques, comme le chiffrement par bloc. En effet, l'exécution de certains algorithmes légers nécessite un nombre de *tours* important, des opérations identiques étant effectuées lors de chaque tour sur les valeurs calculées au tour précédent. Par exemple, l'algorithme KATAN nécessite 254 tours pour un seul chiffrement. La protection uniquement des premiers ou des derniers tours permettrait de contrer au moins une partie des attaques pour un coût énergétique raisonnable.

Dans les section 5.2 à 5.4, nous avons détaillé l'implémentation de contre-mesures adaptatives pour l'un des trois NLFSR qui constituent Trivium. Ces contre-mesures peuvent être appliquées de la même manière à d'autres primitives cryptographiques constituées d'un ou plusieurs LFSR ou NLFSR. Par exemple, Grain est un autre algorithme de chiffrement par flot construit à partir de NLFSR et de LFSR ; l'algorithme de chiffrement par bloc KATAN a été inspiré de Trivium et a donc une structure très similaire ; et la fonction de hachage QUARK reprend à la fois les principes architecturaux de KATAN et de Grain.

Les principes présentés pour la protection adaptative d'un NLFSR pourraient être étendus à d'autres structures, avec un effort plus important. Par exemple, la logique combinatoire est plus complexe dans le cas de l'AES, et le masquage de cette logique combinatoire requiert en général le rajout de nouveaux bits aléatoires à chaque tour de l'algorithme. La génération de bits aléatoires est coûteuse en énergie, ce qui justifie d'autant plus la nécessité d'une sécurité adaptative avec la possibilité de désactiver ce masquage lorsqu'il n'est pas nécessaire.

5.7 Conclusion

Dans ce chapitre, nous présentons plusieurs contre-mesures semi-adaptatives et adaptatives, dont le principe est de modifier en fonction des besoins le niveau de protection de l'implémenta-

tion matérielle d'un algorithme de chiffrement, afin d'optimiser la consommation totale d'énergie sur l'ensemble de la durée de vie d'un circuit intégré.

Dans un premier temps, nous présentons un masquage semi-adaptatif, énergétiquement efficace. En se basant sur le constat que les attaques par canaux auxiliaires existantes contre Trivium sont toutes effectuées durant la phase d'initialisation de cet algorithme, qui dure 1152 cycles d'horloge, cette protection consiste à activer un masquage TI au premier ordre durant cette phase, et à désactiver ensuite ce masquage. Ainsi, cette protection protège efficacement Trivium contre les attaques existantes, pour une consommation durant la phase de chiffrement presque trois fois plus faible que celle d'une implémentation de référence du masquage au premier ordre. Cette contre-mesure est semi-adaptative, car il n'est pas possible d'activer le niveau de sécurité à la demande.

Dans un second temps, nous présentons des contre-mesures matérielles adaptatives qui permettent de choisir le niveau de protection à appliquer, celui-ci pouvant être modifié à volonté durant l'exécution de Trivium. L'augmentation de ce niveau de protection peut répondre à divers cas d'usage tels que la détection d'une attaque, tandis que la désactivation de ces contre-mesures lorsqu'elles ne sont pas nécessaires permet d'économiser de l'énergie. Cette sécurité adaptative permet également d'augmenter la durée d'utilisation d'un circuit intégré avant que celui-ci ne soit obsolète, avec la possibilité d'augmenter le niveau de sécurité pour répondre à de futures attaques. Enfin, elle permet d'utiliser un même circuit intégré pour plusieurs applications différentes.

Nous proposons un masquage adaptatif de Trivium, fournissant la possibilité de choisir pendant le fonctionnement de Trivium parmi un masquage au premier ou au second ordre, ou la désactivation du masquage. La désactivation totale du masquage lorsque aucune protection n'est nécessaire permet d'économiser un maximum d'énergie. D'un autre côté, le masquage au premier ou au second ordre peut être activé à n'importe quel moment, durant l'exécution de Trivium, afin de répondre par exemple à la détection d'une attaque ou à la mise à jour de la politique de sécurité du circuit intégré. Cette protection est, parmi celles présentées dans ce chapitre, celle qui offre le plus de flexibilité grâce aux trois niveaux de protection possibles. De plus, les mécanismes présentés peuvent être appliqués de manière similaire à un masquage adaptatif d'un ordre plus élevé. Les différentes implémentations du masquage de Trivium ne montrent aucune fuite d'information par canaux auxiliaires avec un total de 200.000 traces de consommation obtenues en simulations post-layout.

Le masquage permet seulement la protection contre les attaques par canaux auxiliaires. Nous étendons donc le concept de contre-mesure adaptative à l'encodage dynamique présenté dans le chapitre 4, afin de couvrir également la détection de fautes. Un encodage dynamique adaptatif 1-sur-4 est proposé pour la protection des registres à décalage de Trivium. Cette contre-mesure fournit seulement deux niveaux de sécurité : la protection est soit activée, soit désactivée. Les mêmes principes pourraient être appliqués à d'autres variantes de l'encodage dynamique, telles que l'encodage aléatoire des registres à décalage et/ou l'encodage de la logique combinatoire. Une évaluation de la sécurité de l'encodage dynamique adaptatif, réalisée sur des traces de simulation post-layout, montre des résultats similaires à l'analyse effectuée dans le chapitre 4 sur l'encodage dynamique non adaptatif : cette contre-mesure permet de contrer efficacement les attaques existantes, mais des fuites d'observation peuvent être observées avec un T-test non-spécifique.

L'implémentation adaptative des contre-mesures introduit un léger surcoût en consommation de puissance par rapport à des contre-mesures équivalentes et non adaptatives, en considérant uniquement un niveau de sécurité fixé ; par exemple, l'encodage dynamique adaptatif et le masquage adaptatif au second ordre présentent tous deux une consommation plus élevée de 4,6 % par rapport aux contre-mesures non adaptatives équivalentes. Cependant, elles permettent de réaliser d'importants gains en énergie sur l'ensemble de la durée de vie du circuit intégré si l'on considère que le niveau de protection peut varier ; par exemple, lorsque le masquage adaptatif est désactivé, 60 % de la consommation d'un masquage TI au premier ordre, ou 76 % de celle d'un masquage TI au second ordre, est économisée.

Les contre-mesures présentées dans ce chapitre peuvent être implémentées de diverses ma-

nières, en fonction de besoins spécifiques. Ainsi, outre les différentes variantes de l'encodage dynamique et le masquage à des ordres plus élevés, d'autres façons d'implémenter ces contre-mesures sont discutées dans ce chapitre. En effet, les mécanismes d'augmentation et de réduction du niveau de sécurité peuvent être modifiés, afin que ces processus soient effectués plus rapidement ou au contraire plus lentement, ce qui peut permettre de réduire la complexité et donc la surface de l'implémentation dans certains cas. De plus, ces contre-mesures peuvent être appliquées à d'autres algorithmes ayant la même structure.

D'autres contre-mesures pourraient être implémentées de manière adaptative, afin de fournir une sécurité contre plusieurs types d'attaques, avec plusieurs optimisations possibles de la sécurité, de la consommation et du temps d'exécution. Cependant, lors de l'implémentation d'une nouvelle contre-mesure adaptative, un compromis doit être trouvé entre l'efficacité de celle-ci et son coût. Par exemple, les contre-mesures au niveau du circuit intégré, telles que l'application de DVFS, peuvent être activées ou désactivées avec un coût très faible. Cependant, ces protection peuvent être contournées, comme expliqué dans la section 3.4.1. D'un autre côté, les contre-mesures au niveau des portes logiques ou des transistors, comme certains styles de logique différentielle avec précharge, sont généralement efficaces du point de vue de la sécurité, mais la mise en place d'une sécurité adaptative au sein de chaque porte logique protégée serait extrêmement complexe et coûteuse. Au final, les contre-mesures adaptatives implémentées au niveau algorithmique, telles que celles présentées dans ce chapitre, semblent présenter le meilleur compromis entre sécurité et coût d'implémentation.

Dans ce chapitre, nous avons détaillé la protection des données traitées par un algorithme de chiffrement, mais nous n'avons pas étudié la protection du contrôle des contre-mesures adaptatives. Ce mécanisme de contrôle ne fuite pas d'informations par canaux auxiliaires, mais il pourrait être attaqué avec des injections de fautes visant à réduire artificiellement le niveau de protection des contre-mesures. On considère que le choix du niveau de sécurité est effectué de manière externe, comme proposé par exemple dans [89, 164], et la protection du mécanisme effectuant ce choix est donc hors du cadre de nos travaux. Cependant, certains signaux de contrôle sont générés au sein des différentes contre-mesures proposées dans ce chapitre. Par exemple, des machines à états finis et des compteurs permettent d'appliquer les signaux de contrôle pour la mise en œuvre de la sécurité adaptative. Une simple redondance spatiale (ou une redondance d'information) de tous les chemins de données dédiés au contrôle permettrait de protéger ceux-ci contre certaines attaques en fautes, avec un surcoût raisonnable en surface et en consommation et sans impacter le temps d'exécution de l'algorithme protégé.

Le masquage énergétiquement efficace de Trivium a été présenté lors de la conférence ICECS 2018 à Bordeaux, et figure dans les actes de cette conférence. Le masquage adaptatif a été présenté lors de la conférence ICCD 2019 à Abu Dhabi, et figure également dans les actes de cette conférence.

Chapitre 6

Évaluation des contre-mesures matérielles sur cibles physiques

Sommaire

6.1 Intégration sur ASIC 123
6.1.1 Contrôle de Trivium 123
6.1.2 Implémentation de Trivium sur WARRIOR 124
6.1.3 Implémentation de Trivium sur SamurAI 124
6.2 Implémentation et évaluation sur FPGA 125
6.2.1 Plateforme d'évaluation 125
6.2.2 Fuites d'information par canaux auxiliaires en l'absence de protections . . . 127
6.2.3 T-test non-spécifique sur les différentes versions de Trivium 128
6.2.4 Corrélation et CPA 130
6.3 Conclusion 132

Dans les chapitres 3 à 5, nous avons évalué grâce à des simulations post-layout plusieurs implémentations matérielles dans le nœud technologique 28 nm FDSOI de l'algorithme de chiffrement par flot Trivium et d'un registre à décalage. Nous avons notamment estimé la protection apportée par différentes contre-mesures face aux attaques par analyse des canaux auxiliaires. L'évaluation des fuites par canaux auxiliaires grâce à des simulations post-layout présente plusieurs avantages. D'une part, elles permettent d'évaluer la consommation de puissance de seulement certaines parties d'un circuit, tandis que des mesures sur circuits réels ne permettent parfois que d'obtenir la consommation totale de ces circuits. D'autre part, elles s'affranchissent du bruit inhérent aux mesures physiques, ce qui permet d'observer de très faibles variations de la consommation qui seraient totalement ou partiellement masquées par le bruit au cours de mesures sur circuits réels.

Cependant, cette absence de bruit peut également être perçue comme un inconvénient, dans la mesure où elle ne reflète pas une implémentation physique réelle. Comme nous l'avons expliqué dans la section 4.5.3, cette absence de bruit dans les simulations peut biaiser les résultats obtenus au T-test non-spécifique dans certains cas. L'objectif de nos travaux étant la sécurisation matérielle de circuits intégrés pour l'IoT dans des conditions de fonctionnement réelles, il est nécessaire de compléter les simulations effectuées par une évaluation des fuites par canaux auxiliaires sur cibles physiques.

De plus, dans le chapitre 4, nous avons uniquement mené une étude théorique des fautes pouvant être détectées dans les registres à décalage protégés par l'encodage dynamique, et nous n'avons pas vérifié en pratique la difficulté d'injecter ces fautes. Une implémentation physique sur ASIC de cette contre-mesure permettrait de valider ou d'invalider la détection effective de fautes avec plusieurs moyens d'injection.

Une version non protégée de Trivium, ainsi qu'une version protégée par l'encodage dynamique et deux versions protégées respectivement par un masquage TI au second ordre et un masquage adaptatif, ont été intégrées physiquement sur ASIC dans le nœud technologique 28 nm FDSOI. Cette intégration sur ASIC est décrite dans la section 6.1. L'évaluation des fuites par canaux auxiliaires grâce à des mesures physiques, et l'injection de fautes par divers moyens matériels, permettront de valider en conditions réelles l'efficacité et la pertinence des contre-mesures matérielles que nous avons proposées dans nos travaux. Du fait du temps de fabrication important pour un circuit intégré, ces attaques n'ont pas encore été réalisées.

Une évaluation des fuites par canaux auxiliaires de diverses versions de Trivium, avec et sans protections, a également été menée grâce à des implémentations sur FPGA. Pour cela, les émissions électromagnétiques émises par le circuit intégré sont mesurées durant le fonctionnement de Trivium. Cette évaluation sur FPGA permet, d'une part, d'obtenir des premières mesures sur une implémentation physique, en présence de bruit, afin d'anticiper les futures mesures effectuées sur ASIC. D'autre part, elle permet d'étudier la vulnérabilité aux attaques par canaux auxiliaires des différentes versions de Trivium sur une autre cible physique et avec d'autres canaux auxiliaires que la consommation instantanée sur ASIC. Enfin, les mesures sur FPGA étant plus rapides que les simulations post-layout décrites dans les chapitres précédents, cette étude peut être effectuée avec plus de traces qu'en simulations.

On rappelle ici que la structure interne d'une implémentation sur FPGA est très différente de celle d'une implémentation sur ASIC : tandis qu'un ASIC est constitué uniquement des cellules standard nécessaires pour une application donnée, un FPGA est constitué d'un nombre important de cellules programmables identiques. Chacune de ces cellules programmables contient généralement des tables de correspondances ou des multiplexeurs, qui sont programmés de sorte à effectuer des opérations logiques élémentaires. Cette implémentation différente sur ASIC et sur FPGA peut impacter l'efficacité des contre-mesures matérielles. D'un côté, le masquage algorithmique repose sur l'hypothèse qu'un attaquant est capable d'obtenir par une mesure des canaux auxiliaires les données manipulées, ces données étant rendues indépendantes des valeurs intermédiaires réelles grâce à un ajout d'aléa. Les principes du masquage algorithmique sont donc valables quelle que soit l'implémentation matérielle du masquage, sur ASIC ou sur FPGA. D'un

autre côté, les contre-mesures qui consistent en un équilibrage de la consommation, dont l'encodage dynamique fait partie, visent justement à empêcher l'obtention par analyse des canaux auxiliaires des données manipulées, en supprimant la dépendance entre les grandeurs physiques observables par canaux auxiliaires et ces données. Ces contre-mesures sont donc extrêmement dépendantes de la façon dont l'implémentation physique est réalisée, et ne peuvent être mises en œuvre de la même manière sur ASIC et sur FPGA. L'adaptation sur FPGA d'une protection fondée sur un équilibrage de la consommation et initialement conçue pour un ASIC se révèle souvent complexe [113, 166]. Un état de l'art des tentatives réalisées en ce sens est donné par Wild *et al.* [166], et montre que des fuites d'information par canaux auxiliaires sont généralement observables lors de l'implémentation de telles contre-mesures sur FPGA. Pour cette raison, l'encodage dynamique n'est pas implémenté et évalué sur FPGA dans nos travaux, et nous limitons notre étude à une implémentation non protégée de Trivium et à deux implémentations du masquage algorithmique. Les mesures effectuées sur FPGA et leur exploitation sont décrites dans la section 6.2.

6.1 Intégration sur ASIC

Plusieurs variantes de Trivium ont été intégrées sur deux ASIC différents, dans le nœud technologique 28 nm FDSOI. Le circuit WARRIOR contient une implémentation non protégée de Trivium (notée TRIVIUM), une autre protégée avec un masquage TI à l'ordre 2 (TI_2), et une troisième protégée avec notre masquage adaptatif (TI_ADAPT). Le circuit SamurAI contient uniquement une version de Trivium, protégée un encodage dynamique 1-sur-4 des registres à décalage (TRI_ENC_1). L'implémentation de ces variantes de Trivium et de l'interface permettant de contrôler celles-ci est décrite ici.

6.1.1 Contrôle de Trivium

Le circuit WARRIOR a été présenté brièvement dans la section 2.6, et son architecture est donnée sur la figure 2.4. Il intègre de nombreuses contributions originales ; en particulier, l'un des blocs fonctionnels, parmi les *Security IPs* de la figure 2.4, contient les implémentations de Trivium. Ce bloc fonctionnel est par la suite appelé IP, pour *Intellectual Property*. L'interface entre l'IP contenant Trivium et le reste du circuit est identique pour le circuit SamurAI, seule l'implémentation de l'IP elle-même étant différente dans ce circuit. Par conséquent, le test de cette IP et l'évaluation des différentes versions de Trivium sont réalisés de la même manière sur ces deux circuits.

L'IP contenant Trivium est contrôlée par un programme en C exécuté sur le processeur RISC-V. Ce programme permet de spécifier les données d'initialisation de Trivium, de choisir la version de Trivium à exécuter et le niveau de protection pour le masquage adaptatif, et de lancer le chiffrement. Le processeur communique avec l'IP grâce à un bus APB (*Advanced Peripheral Bus*) faisant partie des spécifications AMBA (*Advanced Microcontroller Bus Architecture*) établies par ARM [167]. Les données nécessaires à l'initialisation de Trivium et au contrôle de l'IP sont écrites via ce bus APB dans un registre de configuration intégré dans l'IP, qui sert d'interface entre celle-ci et le reste du circuit. Le flot de clé généré par Trivium est écrit dans ce même registre de configuration, et peut être lu par le processeur ; la lecture de ces données permet notamment de valider le fonctionnement de l'IP lors du test du circuit, en comparant le flot de clé généré à un flot de clé obtenu en simulations avec les mêmes paramètres d'initialisation. La récupération de ce flot de clé est également nécessaire lors d'une attaque par injection de fautes, à la fois pour vérifier le succès de l'injection de fautes et pour exploiter les informations obtenues avec le flot de clé fauté.

Lors de chaque chiffrement, un signal d'interruption est généré par l'IP contenant Trivium. Ce signal d'interruption est transmis au processeur, et sur l'une des interfaces de sortie du circuit. Il permet notamment de synchroniser le début du chiffrement avec un équipement extérieur au circuit, tel qu'un oscilloscope ou un banc d'injection de fautes, de sorte à effectuer des attaques

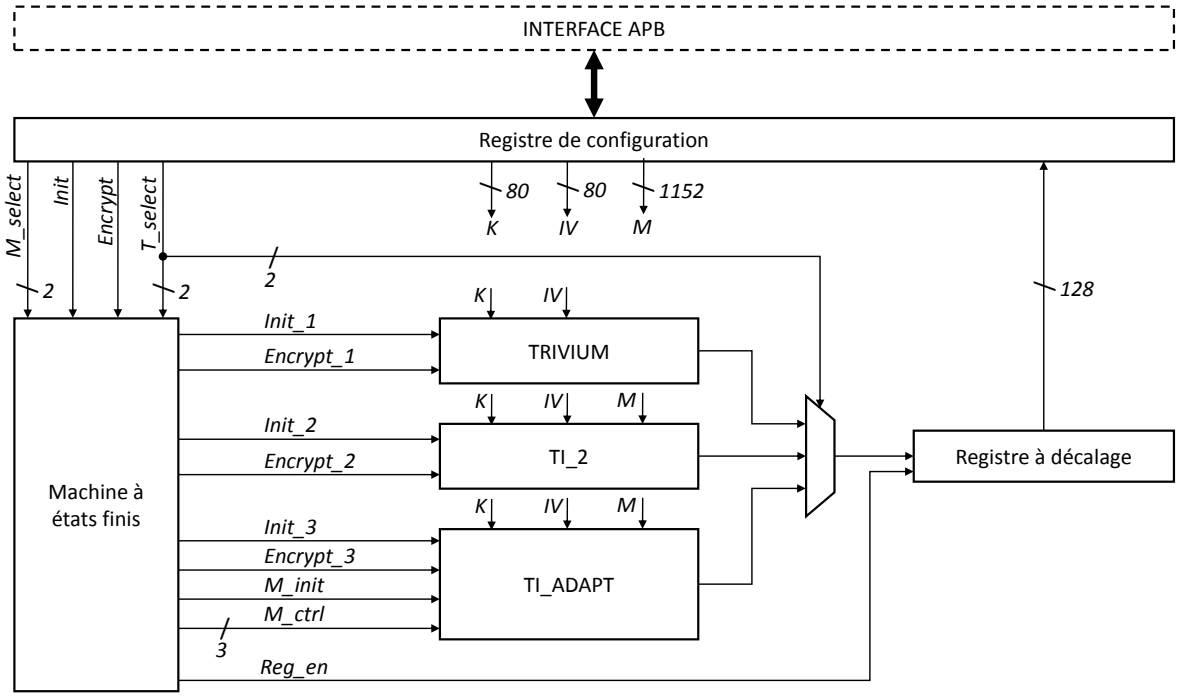


FIGURE 6.1 – Architecture du bloc fonctionnel contenant trois versions de Trivium, sur le circuit WARRIOR.

précises et reproductibles.

6.1.2 Implémentation de Trivium sur WARRIOR

Nous décrivons ici l'architecture interne de l'IP contenant Trivium sur le circuit WARRIOR, représentée sur la figure 6.1. Plusieurs données d'initialisation et signaux de contrôle sont écrits dans le registre de configuration via le bus APB : la clé K, le vecteur d'initialisation IV et les masques M, ainsi qu'un signal *T_select* permettant de sélectionner la version de Trivium parmi les trois implémentées, un signal *M_select* pour le choix du niveau de protection pour le masquage adaptatif, et deux signaux *Init* et *Encrypt* ayant respectivement pour but d'initier le chargement des données d'initialisation dans Trivium et de démarrer le chiffrement. Lors du chiffrement avec l'une des versions de Trivium, un flot de clé est écrit bit par bit dans un registre à décalage, et ces 128 bits sont écrits dans le registre de configuration à l'issue de la phase de chiffrement et peuvent être lus par le processeur via le bus APB.

Une machine à états finis permet de contrôler les trois versions de Trivium et le registre à décalage de 128 bits qui stocke au fur et à mesure le flot de clé généré par la version sélectionnée. Elle génère des signaux *Init_i* et *Encrypt_i*, avec $i \in \{1,2,3\}$, pour respectivement charger les données dans l'une des trois versions de Trivium et pour démarrer le chiffrement avec cette même version. Ces signaux *Init_i* et *Encrypt_i* dépendent directement des signaux *T_select*, *M_select* et *Encrypt* issus des registres de configuration. Durant le chiffrement avec l'un des trois Trivium, un signal *Reg_en* active le registre à décalage afin de stocker le flot de clé à mesure qu'il est généré, et un signal d'interruption est généré et transmis à l'extérieur de l'IP. Enfin, de la même manière que la machine à états finis décrite dans la section 5.3 pour le contrôle du masquage adaptatif, deux signaux *M_ctrl* et *M_init* permettent de diminuer ou d'augmenter le niveau de protection avec cette contre-mesure.

6.1.3 Implémentation de Trivium sur SamurAI

Une seule implémentation de Trivium, protégée par un encodage dynamique des registres à décalage, a été intégrée sur le circuit SamurAI. Par conséquent, l'IP contenant cette implémentation protégée est une version simplifiée de celle décrite précédemment et intégrée sur WARRIOR ;

en particulier, les signaux *T_select* et *M_select*, ainsi que les masques M, ne figurent pas dans l'IP implémentée sur SamurAI. Dans cette IP, un signal d'alarme est écrit dans le registre de configuration par le Trivium encodé lorsqu'une faute est détectée par l'encodage dynamique. On notera qu'aucune contrainte visant à réduire les fuites par canaux auxiliaire n'a été mise sur la synthèse et l'implémentation physique du Trivium protégé. On s'attend donc à ce que les chemins de données soient déséquilibrés, et à ce que plusieurs types de bascules différentes soient implémentées dans ceux-ci.

6.2 Implémentation et évaluation sur FPGA

Dans cette section, nous détaillons les mesures effectuées sur FPGA et leur exploitation afin d'analyser les fuites par canaux auxiliaires de différentes versions de Trivium : une première implémentation non protégée (notée TRIVIUM), et deux autres implémentations protégées respectivement avec le masquage adaptatif (TI_ADAPT) et le masquage énergétiquement efficace (TI_EFF) décrits dans le chapitre 5. Cette évaluation des fuites par canaux auxiliaires par des mesures physiques vient compléter l'analyse effectuée à partir de simulations décrite dans les chapitres précédents.

6.2.1 Plateforme d'évaluation

Les trois différentes versions de Trivium ont été implémentées et évaluées à tour de rôle sur un FPGA Intel Cyclone V [168]. Plusieurs mesures ont été effectuées avec le masquage adaptatif (circuit TI_ADAPT), afin d'évaluer la sécurité avec le masquage au premier et au second ordres, et lorsque le masquage est désactivé.

Un script en Python génère les clés, IV et masques, ainsi que les commandes envoyées au FPGA. Un processeur Nios-II sur le FPGA permet de faire l'interface entre l'ordinateur et le circuit testé, en recevant des commandes via l'interface JTAG et en les transmettant au Trivium grâce à un bus Avalon. Une sonde électromagnétique (EM) est utilisée afin de mesurer localement les fuites d'information par émissions EM. Comme expliqué dans la section 3.1, les émissions EM sont corrélées aux variations locales de courant dans un circuit intégré, et la mesure de ces émissions permet donc de cibler les fuites d'information provenant d'un endroit donné du circuit. La sonde est reliée à un oscilloscope Rohde & Schwarz RTO2014 [76], qui transmet ensuite la trace d'émissions EM à l'ordinateur. Un signal analogique est transmis par GPIO du circuit à l'oscilloscope à chaque nouvelle mesure, afin de synchroniser celui-ci avec l'exécution de Trivium. Une photographie de la plateforme d'évaluation est montrée sur la figure 6.2.

Dans le but d'évaluer au mieux les fuites d'information dans les émissions EM, on cherche à augmenter le rapport signal sur bruit en diminuant le bruit autant que possible. Une partie du bruit provient du circuit intégré lui-même, et notamment de tous les blocs fonctionnels qui ne contribuent pas directement à la fuite d'information, tels que ceux dédiés au contrôle et à la communication. On appellera cela par la suite le bruit de fonctionnement. Une autre composante du bruit est le bruit de mesure, inhérent à l'expérience elle-même et dépendant de divers facteurs liés à l'environnement de mesure.

Afin de réduire le bruit de fonctionnement lors des mesures, le FPGA a été partitionné de sorte à séparer physiquement le Trivium de la logique dédiée au test et à la communication. Le layout obtenu est représenté sur la figure 6.3. De cette manière, en plaçant la sonde EM au bon emplacement, il est possible de différencier les émissions EM dues à Trivium de celles provenant du reste du circuit. Une cartographie des émissions EM, au cours de laquelle la sonde EM est déplacée au-dessus du circuit, a été effectuée pour chaque acquisition afin de trouver l'emplacement présentant le meilleur rapport signal sur bruit. De plus, une horloge à 1 MHz sert à cadencer le Trivium, tandis que le reste du circuit utilise une horloge à 50 MHz. Ces fréquences sont choisies selon plusieurs critères : elles doivent être suffisamment distinctes, mais également suffisamment élevées afin de ne pas trop ralentir les temps de mesure, en restant suffisamment faibles afin de

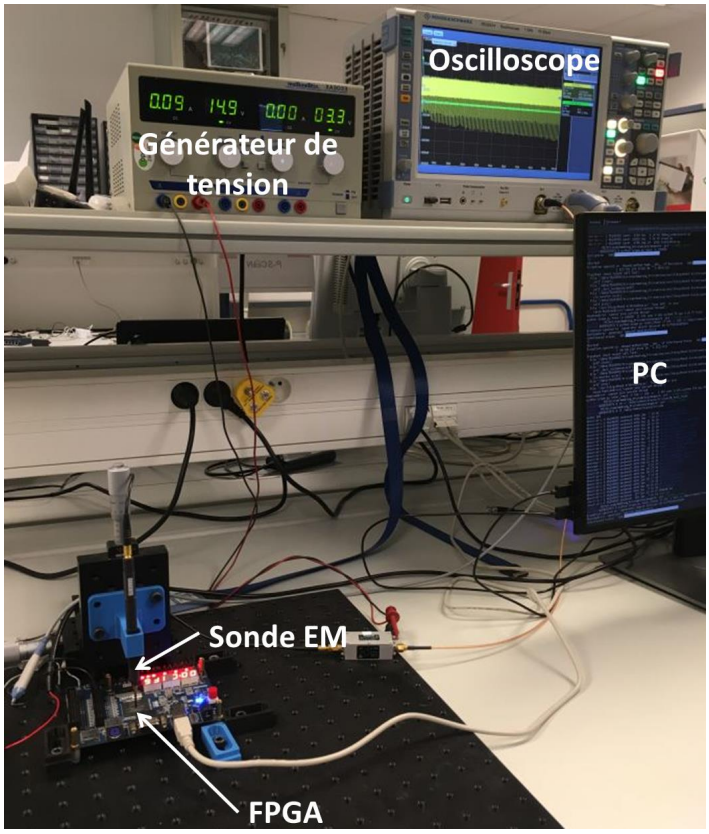


FIGURE 6.2 – Banc de mesure des émissions électromagnétiques sur FPGA (courtoisie du laboratoire LSOSP).

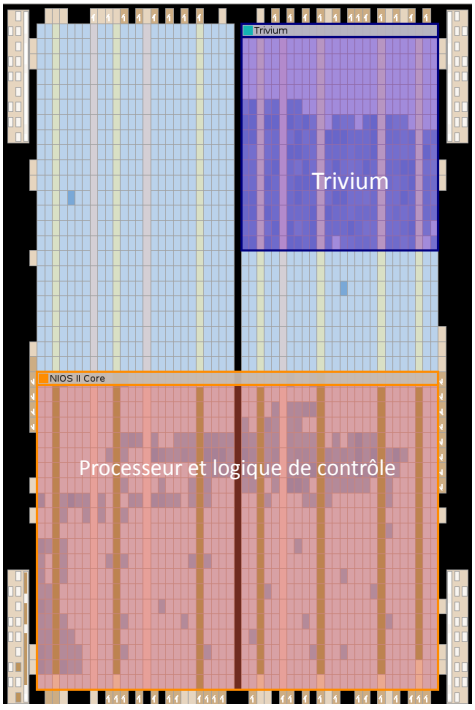


FIGURE 6.3 – Partitionnement sur le FPGA.

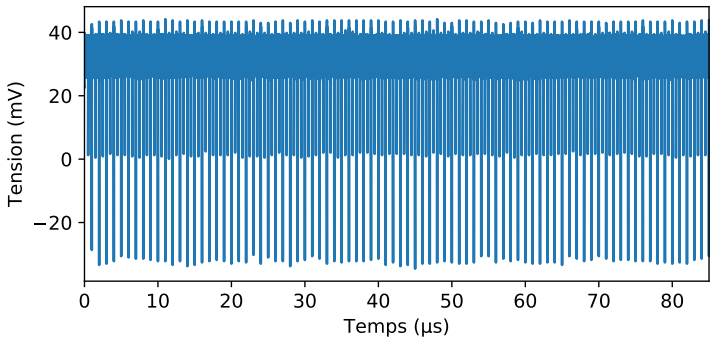


FIGURE 6.4 – Exemple de trace d'émissions EM pour le masquage adaptatif à l'ordre 2.

ne pas trop contraindre le placement dans le FPGA, ce qui pourrait avoir un impact sur les fuites observables par canaux auxiliaires. Les deux horloges sont décalées d'un demi-cycle de l'horloge la plus rapide, soit 10 ns. Cela permet de distinguer visuellement, sur les traces d'émissions EM, celles liées à Trivium de celles dues au reste du circuit.

Pour réduire le bruit de mesure, chaque trace d'émissions EM est acquise 100 fois de suite avec les mêmes paramètres (clé, IV et masques éventuels), et une moyenne est effectuée sur ces 100 traces par l'oscilloscope. Par la suite, nous appellerons donc « trace d'émissions EM » une trace moyennée 100 fois. Selon Standaert [129], ce procédé permet de réduire considérablement le nombre de traces nécessaires afin de détecter des fuites d'information d'ordres supérieurs. En effet, le bruit est amplifié lorsque les traces sont pré-traitées afin de mettre en évidence les fuites d'ordre supérieur, et à mesure que ce bruit augmente, le nombre de traces nécessaires pour détecter des fuites par canaux auxiliaires augmente exponentiellement avec l'ordre du masquage. Le fait de moyenner les traces est un outil d'évaluation puissant, qui n'est généralement pas à la portée d'un attaquant réel : cela suppose que l'attaquant soit en mesure de ré-utiliser plusieurs fois de suite non seulement les IV, mais aussi les masques. Or, un attaquant n'a généralement pas accès aux masques, qui sont générés aléatoirement au sein du circuit intégré.

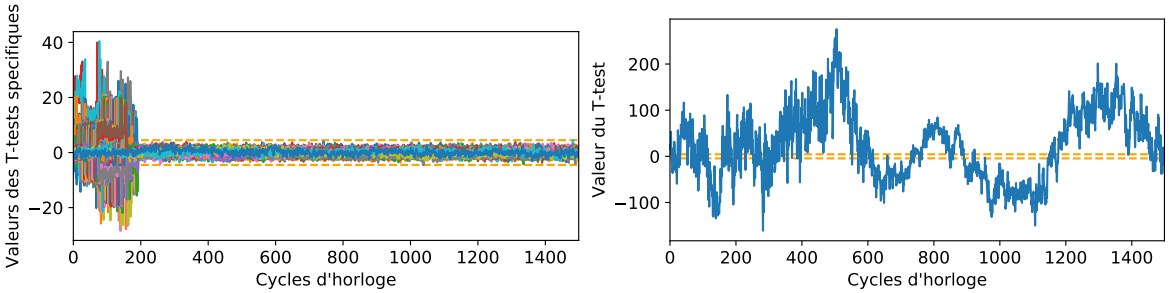
À titre d'exemple, une trace d'émissions EM couvrant les 85 premiers cycles de la phase d'initialisation du masquage adaptatif à l'ordre 2 est donnée sur la figure 6.4. Cette trace est acquise avec un échantillonnage à 5 GSa/s (5 milliards de points par seconde). Sur cette figure, les pics négatifs de tension correspondent aux pics d'émissions EM liées à l'activité de Trivium, les valeurs autour de ces pics étant assimilables à du bruit. Afin de gagner en mémoire et en temps de traitement, toutes les traces acquises par la suite sont compressées après leur acquisition en ne gardant qu'un seul point par cycle d'horloge, correspondant aux pics mentionnés précédemment.

6.2.2 Fuites d'information par canaux auxiliaires en l'absence de protections

Dans un premier temps, nous évaluons les fuites d'information par canaux auxiliaires durant les 1500 premiers cycles d'horloge de l'exécution de Trivium en l'absence de contre-mesures, selon la méthodologie détaillée dans la section 3.2.3. Pour cela, un T-test non-spécifique et 81 T-tests spécifiques utilisant les 81 valeurs intermédiaires σ_i sont effectués.

Chaque trace couvrant 1500 cycles d'horloge est mesurée avec une fréquence d'échantillonnage de 1 GSa/s. Cette fréquence d'échantillonnage de 1 GSa/s permet de récupérer 1000 poins par cycle d'horloge, ce qui est largement suffisant pour distinguer les pics d'émissions EM liés à l'activité de Trivium en l'absence de contre-mesures.

Pour le T-test non-spécifique, deux lots de 50.000 traces chacun ont été acquis avec une clé fixe, et respectivement un IV fixe et des IV aléatoires. Les T-tests spécifiques ont été effectués sur 50.000 traces générées avec un IV fixe et des clés aléatoires, il y a donc en moyenne 25.000 traces par lot. Les résultats de ces deux T-tests sont présentés sur la figure 6.5. Ces résultats permettent de valider expérimentalement, en présence de bruit, ceux obtenus précédemment à partir de simulations non bruitées : les attaques sur les valeurs intermédiaires σ_i ne sont possibles que pendant les



(a) T-tests spécifiques

(b) T-test non-spécifique

FIGURE 6.5 – T-tests spécifiques et non-spécifiques durant les 1500 premiers cycles d'exécution de Trivium, sur FPGA.

200 premiers cycles d'horloge environ, mais des fuites d'information par canaux auxiliaires sont observables durant les 1500 premiers cycles d'horloge de l'exécution de Trivium.

D'une part, ces résultats expérimentaux permettent de valider que le masquage énergétiquement efficace TI_EFF, activé seulement durant la phase d'initialisation, serait une contre-mesure suffisante pour protéger l'implémentation de Trivium contre les attaques par canaux auxiliaires publiées à l'heure actuelle. Les résultats de simulations post-layout de la consommation pour une implémentation sur ASIC et les résultats issus de mesures des émissions EM d'un FPGA indiquent tous deux que l'efficacité énergétique de cette protection pourrait encore être améliorée, car il n'est nécessaire d'activer le masquage que durant les 200 premiers cycles d'horloge environ. D'autre part, ces résultats justifient le besoin de protéger l'implémentation de Trivium contre de futures attaques exploitant les fuites observées au T-test non-spécifique. La sécurité matérielle adaptative permettra d'augmenter le niveau de protection lorsque de telles attaques seront publiées, tout en maintenant un niveau de consommation minimal lorsqu'un niveau de protection élevé n'est pas nécessaire.

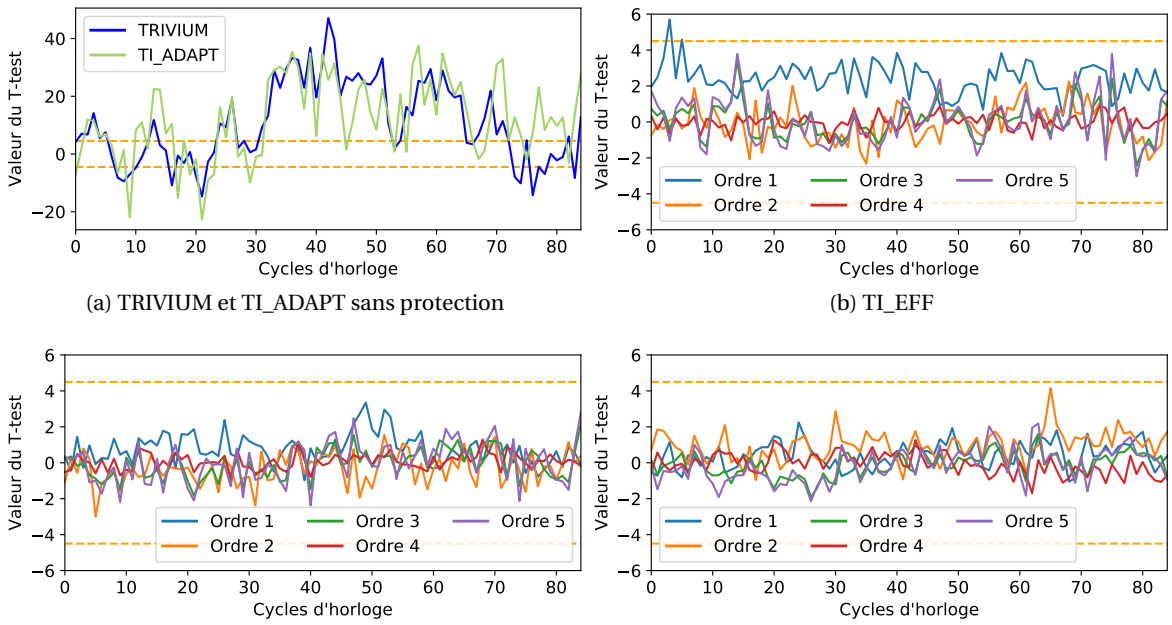
6.2.3 T-test non-spécifique sur les différentes versions de Trivium

Dans un second temps, nous évaluons la présence de fuites d'information par canaux auxiliaires pour toutes les versions de Trivium. Afin d'obtenir suffisamment de traces dans un temps d'acquisition raisonnable, les fuites sont étudiées uniquement durant les 85 premiers cycles d'horloge de l'exécution de Trivium, de même que pour l'analyse effectuée à partir de simulations post-layout. Toutes les traces couvrant ces 85 cycles d'horloge sont mesurées avec une fréquence d'échantillonnage de 5 GSa/s.

Des T-tests non-spécifiques permettent d'étudier la présence de fuites d'information. Un million de traces par lot sont acquises pour le masquage adaptatif aux premier et second ordres ainsi que pour le masquage énergétiquement efficace. Pour le Trivium non protégé, ainsi que pour le masquage adaptatif lorsque la protection est désactivée, seulement 10.000 traces par lot sont mesurées, afin d'écourter les temps d'acquisition. Pour chaque T-test, toutes les traces ont été acquises de manière entrelacée entre les deux lots, c'est-à-dire en alternant les mesures avec un IV fixe et un IV aléatoire. Cela permet d'atténuer l'impact sur les calculs des effets liés à l'environnement pouvant affecter les mesures sur le long terme, tels que les changements de température.

Environ 16 jours de mesures sont nécessaires pour acquérir deux lots contenant chacun un million de traces. À titre de comparaison, plusieurs jours sont nécessaires avec des simulations post-layout, jusqu'à plus d'une semaine en fonction de la taille du circuit simulé, pour obtenir un seul lot de 100.000 traces.

Les résultats des T-tests univariés pour chaque circuit sont représentés sur la figure 6.6. En particulier, des T-tests d'ordres supérieurs permettent d'évaluer les fuites d'information d'ordre supérieur des implémentations masquées, sur les figures 6.6b à 6.6d. Ces figures montrent la présence de fuites en l'absence de masquage, avec seulement 10.000 traces par lot. Aucune fuite d'in-



(a) TRIVIUM et TI_ADAPT sans protection

(b) TI_EFF

(c) TI_ADAPT - protection au premier ordre

(d) TI_ADAPT - protection au second ordre

FIGURE 6.6 – Valeurs du T-test pour les circuits évalués sur FPGA, aux différents niveaux de protection.

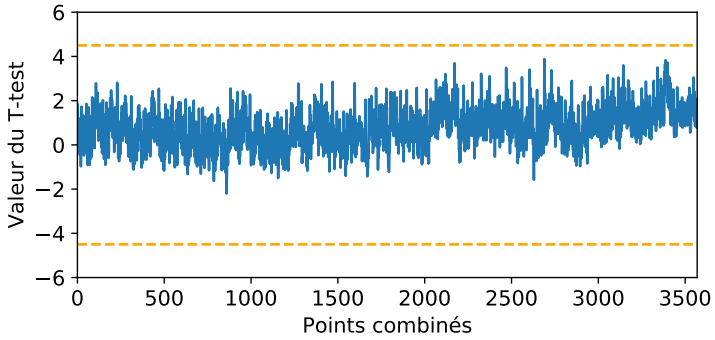
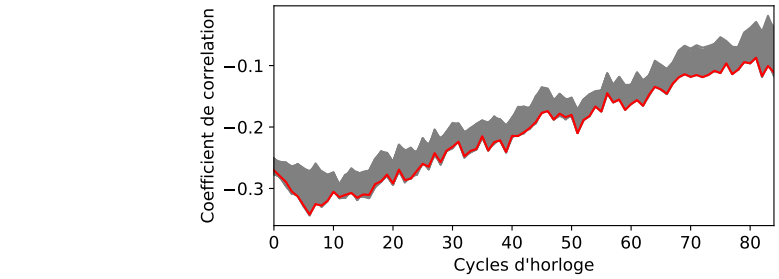


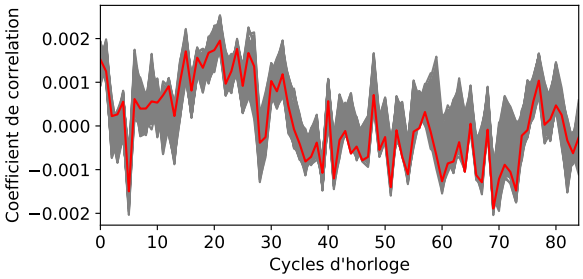
FIGURE 6.7 – T-test bivarié sur le masquage adaptatif au second ordre de Trivium, sur FPGA.

formation n'est visible pour le masquage adaptatif, lorsqu'il est activé à la fois au premier et au second ordre, avec un million de traces par lot. La valeur du T-test à l'ordre 1 dépasse le seuil pour le masquage énergétiquement efficace TI_EFF à un seul moment. L'absence de fuites à d'autres instants malgré le fonctionnement similaire de cette implémentation de Trivium à chaque cycle d'horloge, ainsi que l'absence de fuites aux ordres supérieurs, permettent d'avancer deux hypothèses probables pour expliquer cet unique dépassement de la valeur de seuil : celui-ci peut être dû à la présence d'un artefact de mesure, ou à des opérations effectuées dans d'autres parties du circuit, telles que le processeur ou les registres qui stockent sans protection les clés, IV et masques. En effet, chaque implémentation étudiée est placée différemment dans le FPGA, et la position de la sonde EM varie également d'une implémentation à l'autre afin d'obtenir le meilleur SNR possible ; il est donc possible que la sonde ait été placée plus proche des registres ou du processeur pour l'implémentation TI_EFF que pour les autres, ce qui pourrait expliquer une fuite à l'ordre 1 mais pas aux ordres supérieurs malgré la présence de masquage. Cela illustre la difficulté de réaliser des mesures physiques fiables dans des conditions identiques pour toutes les implémentations, à l'inverse des simulations qui s'affranchissent de ce type de problèmes en permettant de ne simuler qu'une sous-partie du circuit visé.

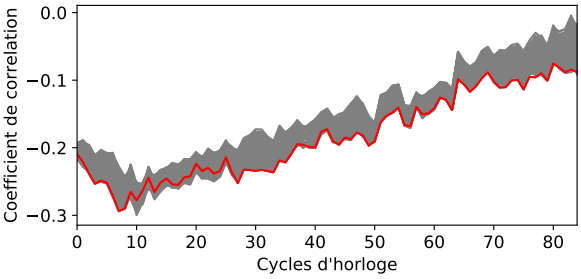
Un T-test bivarié est également effectué sur le masquage adaptatif au second ordre, suivant la méthodologie décrite dans la section 5.5.3. Celui-ci est représenté sur la figure 6.7, et démontre l'absence de fuites bivariées sur les 85 premiers cycles d'horloge avec un million de traces par



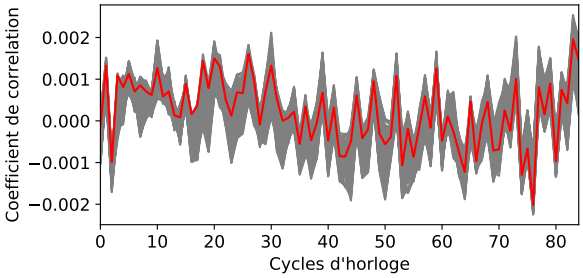
(a) TRIVIUM



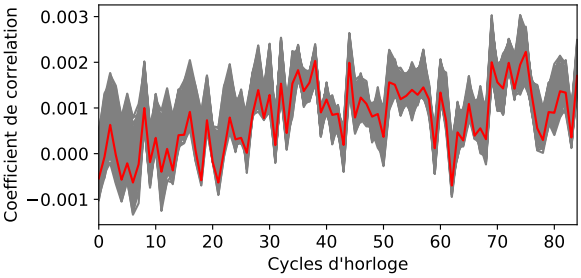
(b) TI_EFF



(c) TI_ADAPT sans protection



(d) TI_ADAPT - protection au premier ordre



(e) TI_ADAPT - protection au second ordre

FIGURE 6.8 – Évolution du coefficient de corrélation pour les circuits évalués sur FPGA, aux différents niveaux de protection.

lot. On notera que la démonstration expérimentale de fuites bivariées sur un NLFSR masqué à l'ordre 2 dans [78] a également été réalisée sur FPGA, avec le même nombre de traces que dans nos expériences. L'implémentation différente du masquage TI dans nos travaux, discutée dans la section 5.5.3, ou la complexité plus élevée de Trivium par rapport au NLFSR étudié par ces auteurs, pourraient expliquer cette absence de fuites.

6.2.4 Corrélation et CPA

Dans cette section, nous évaluons la possibilité d'exploiter les fuites d'information montrées précédemment en l'absence de contre-mesure afin d'obtenir la clé de chiffrement. Pour cela, nous analysons les variations du coefficient de corrélation pour les différentes implémentations étudiées, aux différents niveaux de protection. Bien qu'avec un million de traces par lot, aucune fuite d'information significative n'ait été observée avec le T-test non-spécifique pour les implémentations masquées, nous fournissons tout de même les résultats de l'étude du coefficient de corrélation pour ces implémentations, afin de souligner la difficulté d'effectuer une attaque CPA contre ces implémentations.

La méthodologie adoptée est similaire à celle décrite dans la section 3.2.2, avec quelques modifications décrites ci-après afin d'adapter celle-ci pour les traces d'émissions EM acquises sur FPGA. De même que précédemment, cette étude a été menée sur les 85 premiers cycles de l'initialisation de Trivium. Un million de traces sont acquises pour les implémentations masquées, et 10.000 traces pour les implémentations non masquées. Toutes ces traces ont été obtenues avec une clé fixe et des IV aléatoires.

TABLEAU 6.1 – Maximum du coefficient de corrélation, sur FPGA.

Circuit	Niveau de protection	Corrélation (valeur absolue)
TRIVIUM	-	0,3426
TI_EFF	-	0,0020
TI_ADAPT	$Aucun$	0,2939
	$1^{er} ordre$	0,0020
	$2^{nd} ordre$	0,0022

Le coefficient de corrélation est calculé de la manière décrite dans la section 3.2.2, pour 256 hypothèses effectuées sur les 8 premières valeurs intermédiaires σ_i, c'est-à-dire sur un octet de la clé de chiffrement. En particulier, un modèle en distance de Hamming est sélectionné afin de représenter au mieux les émissions EM liées à l'activité du circuit. La figure 6.8 montre, pour chaque implémentation, l'évolution du coefficient de corrélation pour les 256 hypothèses. Dans chaque cas, la bonne hypothèse est affichée en rouge. On constate que la bonne hypothèse de clé n'est pas distinguable des 255 mauvaises hypothèses pour les implémentations protégées ; la difficulté à distinguer la bonne hypothèse dans le cas des implémentations non protégées est discutée dans les paragraphes suivants.

Les coefficients de corrélation maximaux pour la bonne hypothèse sont donnés dans le tableau 6.1 pour toutes les versions étudiées, et à tous les niveaux de protection pour le masquage adaptatif. Globalement, ces résultats montrent l'absence de corrélation entre les émissions EM mesurées et le modèle d'émissions pour les implémentations masquées. On constate également que la corrélation maximale pour le Trivium non protégé vaut seulement 0,3426, quand elle valait 0,8950 avec les simulations post-layout ; cela est notamment dû à la présence de bruit lors de la mesure de ces traces. Ce coefficient de corrélation plus faible qu'en simulations pourrait également s'expliquer par l'utilisation d'un canal auxiliaire différent, et l'implémentation sur une cible différente : le modèle en distance de Hamming, qui représente bien la consommation d'une implémentation sur ASIC, n'est peut-être pas le plus adapté pour représenter les émissions EM d'une implémentation sur FPGA.

Lors de l'étude de la corrélation sur des traces obtenues en simulations post-layout, prendre le maximum du coefficient de corrélation à un instant donné suffit à déterminer la bonne hypothèse et à effectuer une CPA avec succès. Or, nous avons montré précédemment que même en simulations, les 256 hypothèses ont des corrélations très proches. En présence de bruit, et notamment lors de la mesure de traces d'émissions EM sur FPGA, il n'est plus possible de distinguer à coup sûr la bonne hypothèse grâce au maximum du coefficient de corrélation. Dans ce cas, l'intégration du coefficient de corrélation sur l'ensemble des points donne de meilleurs résultats pour effectuer une CPA, et la bonne hypothèse est alors celle qui présente la corrélation la plus élevée en moyenne sur l'ensemble des points. Cela s'explique par le fait que dans un registre à décalage, les valeurs intermédiaires ciblées par la CPA sont décalées d'un bit à chaque cycle d'horloge, et contribuent donc toujours à la consommation même après plusieurs dizaines de cycles d'horloge. Avec cette nouvelle méthode pour séparer les hypothèses, et en suivant la méthodologie décrite dans la section 3.1 par ailleurs, une CPA sur la clé complète a été effectuée avec succès avec 10.000 traces contre le Trivium non protégé. Cette CPA échoue avec un million de traces pour les circuits masqués. Cette CPA échoue également avec 10.000 traces pour l'implémentation du masquage adaptatif lorsque le masquage est désactivé. Plusieurs hypothèses peuvent être avancées afin d'expliquer l'échec de la CPA dans ce cas : le bruit de fonctionnement peut être trop élevé, la sonde EM peut être placée au mauvais endroit, ou le nombre de traces peut être trop faible. En effet, si 10.000 traces suffisent à montrer la présence de fuites par canaux auxiliaires avec le T-test pour cette implémentation, ce nombre de traces est peut-être trop faible pour exploiter ces fuites dans une CPA.

6.3 Conclusion

Dans ce chapitre, nous décrivons l'implémentation sur cibles physiques de différentes versions de Trivium, afin d'évaluer dans des conditions réelles la sécurité apportée par les contre-mesures matérielles présentées dans les chapitres 4 et 5, notamment en présence de bruit. Un autre objectif de cette implémentation physique est l'acquisition d'un nombre élevé de traces, les mesures physiques nécessitant un temps moins élevé que la simulation des signaux électriques dans tous les transistors d'un circuit.

Dans un premier temps, nous détaillons l'intégration de plusieurs variantes de Trivium sur deux ASIC différents, dans le nœud technologique 28 nm FDSOI. Ces implémentations sur ASIC sont similaires à celles étudiées en simulations dans les chapitres précédents, et les attaques sur ces ASIC permettront donc de valider expérimentalement les résultats présentés dans ces chapitres. De plus, des attaques en fautes, réalisées avec divers moyens d'injection de fautes, permettront d'estimer la difficulté d'injecter des fautes avec succès contre une implémentation protégée avec l'encodage dynamique.

Dans un second temps, nous décrivons l'intégration sur FPGA et l'analyse des émissions électromagnétiques de diverses variantes de Trivium, sans protection et avec plusieurs implémentations de masquage algorithmique. Cette évaluation à partir de mesures réelles s'affranchit notamment du biais dans le T-test qu'on peut observer pour les implémentations non protégées avec des simulations post-layout en l'absence de bruit. Ces mesures sur FPGA confirment la présence de fuites d'information par canaux auxiliaires durant les 1500 premiers cycles d'horloge de l'exécution de Trivium, et l'impossibilité d'exploiter ces fuites après 200 cycles d'horloge avec les attaques CPA publiées à l'heure actuelle contre Trivium. Elles confirment également l'absence de fuites bivariées et univariées (jusqu'à l'ordre 5) par canaux auxiliaires lorsque Trivium est protégé avec un masquage TI, grâce à des T-tests non-spécifiques effectués avec 1.000.000 de traces par lot. Enfin, une attaque CPA est réalisée avec succès avec 10.000 traces contre l'implémentation non protégée de Trivium. L'échec de cette CPA contre l'implémentation de masquage adaptatif lorsque le masquage est désactivé indique que cette attaque pourrait être améliorée, par exemple en utilisant plus de traces, en réduisant plus le bruit, ou en sélectionnant un autre modèle afin de représenter les variations des émissions électromagnétiques d'une implémentation sur FPGA.

Chapitre 7

Conclusion et perspectives

7.1 Conclusion

De nombreuses contraintes, parfois contradictoires, s'appliquent aux circuits intégrés pour l'Internet des Objets. D'une part, ces circuits ont généralement une durée de vie élevée, ce qui implique notamment une faible consommation énergétique s'ils fonctionnent sur batterie. D'autre part, ces circuits doivent être sécurisés, tout au long de leur durée de vie, contre diverses attaques, visant par exemple à mettre hors service ces circuits ou à récupérer des informations confidentielles contenues dans ceux-ci. Le sujet de cette thèse est l'étude de la relation entre sécurité et efficacité énergétique des circuits intégrés, et la proposition de solutions permettant de concilier ces deux contraintes. Deux aspects de cette relation entre sécurité des circuits intégrés et consommation d'énergie sont traités : nous nous intéressons d'une part à la sécurité d'un mécanisme de gestion de l'énergie, et d'autre part à l'efficacité énergétique des contre-mesures matérielles contre les attaques par injection de fautes et par analyse des canaux auxiliaires.

Le premier chapitre introduit le contexte de cette thèse, qui est celui des circuits intégrés conçus pour un usage dans l'IoT. Nous décrivons les caractéristiques des circuits intégrés pour l'IoT, et les différentes contraintes qui s'appliquent à ces circuits. Nous donnons un exemple de circuit intégré pour l'IoT avec la plateforme L-IoT développée au CEA LETI. Afin d'économiser de l'énergie, ce circuit est divisé en deux sous-parties : la partie *On-Demand* contient plusieurs composants ayant une consommation élevée, tels qu'une radio principale et un processeur, et est donc mise en veille la plupart du temps, tandis que la partie *Always-Responsive* est toujours réveillée, et contient une radio de réveil dont le but est de réveiller la partie *On-Demand* lors de la réception d'un code de réveil. Cette architecture de circuit intégré est prise comme référence dans nos travaux. Nous détaillons ensuite les enjeux de la sécurité dans l'IoT, et les mécanismes mis en œuvre afin d'assurer cette sécurité ; en particulier, nous présentons l'algorithme léger de chiffrement par flot Trivium, qui est représentatif des primitives cryptographiques légères conçues pour un usage dans l'IoT, et sur lequel sont évaluées toutes les contre-mesures matérielles présentées dans nos travaux. Nous expliquons également pourquoi la protection des circuits intégrés est primordiale afin d'empêcher de nombreuses attaques contre les objets connectés et les réseaux dans l'IoT ; cette protection des circuits intégrés a généralement un coût énergétique important, et nous motivons la nécessité d'effectuer un compromis entre les besoins en sécurité et en consommation énergétique, qui peuvent évoluer au cours du temps pour un même circuit intégré.

Dans un premier temps, nous abordons la relation entre sécurité et consommation d'énergie à travers la sécurisation d'un mécanisme de gestion de l'énergie utilisé sur certains circuits intégrés. Le chapitre 2 présente l'étude de la sécurité de la radio de réveil, qui a pour fonction de sortir de veille un circuit intégré lors de la réception d'un code de réveil spécifique. Si ce code est identique à chaque réveil, un attaquant peut aisément le renvoyer continuellement au circuit afin de le réveiller constamment et de vider ainsi sa batterie. C'est ce qu'on appelle une attaque par déni de sommeil. Bien que plusieurs méthodes existent afin de générer un code différent à chaque réveil et imprévisible pour un attaquant, ces méthodes présentent de nombreux inconvénients, dont

un coût énergétique potentiellement élevé du fait de communications supplémentaires sur la radio principale, de la mise en place d'un mécanisme de synchronisation temporelle, ou de calculs complexes. De plus, la plupart d'entre elles requièrent l'échange et la maintenance de clés secrètes dédiées au processus de génération de codes de réveil, ce qui a également un coût énergétique non négligeable et rajoute des contraintes sur le circuit pour la gestion sécurisée de ces clés. Nous proposons donc une nouvelle méthode de génération de codes de réveil, qui ne souffre pas des inconvénients des méthodes existantes. Cette méthode repose sur des hypothèses supplémentaires par rapport aux méthodes existantes : la communication sur la radio principale doit être sécurisée, et robuste face à la perte de messages. Ces hypothèses, réalistes pour le bon fonctionnement du réseau, sont compatibles avec de nombreux protocoles de communication communément utilisés dans l'IoT. Notre méthode est récursive, c'est-à-dire qu'à chaque réveil, elle permet de générer un nouveau code de réveil à partir de données calculées au réveil précédent, ces dernières étant combinées pour plus de sécurité à des données issues de la session de communication en cours. Une fonction de hachage cryptographique légère, SPONGENT-88, est sélectionnée afin de générer les codes de réveil avec un coût minimal en mémoire et en consommation. Une analyse de la sécurité de cette méthode montre que l'attaque la plus efficace pour obtenir un code de réveil est une attaque par force brute sur la radio de réveil ; cette attaque, qui doit être répétée dans son intégralité à chaque fois qu'un attaquant souhaite réveiller le circuit, requiert un temps très important pour être menée à bien. Ce temps est généralement supérieur à la durée de vie d'un code de réveil avant son renouvellement, et cette attaque est donc difficilement réalisable en pratique. Enfin, une simulation de l'implémentation de notre méthode sur un circuit intégré montre que, pour un cas d'usage typique pour l'IoT, celle-ci a un coût énergétique négligeable (0,08 %), tandis qu'elle permet d'augmenter par un facteur 120 la durée de vie d'un circuit visé par une attaque par déni de sommeil.

Dans un second temps, nous proposons des manières d'améliorer l'efficacité énergétique des contre-mesures qui visent à protéger l'implémentation de primitives cryptographiques contre les attaques matérielles. Nous présentons tout d'abord, dans le chapitre 3, les principes et la mise en œuvre des attaques matérielles par analyse des canaux auxiliaires et par injection de fautes. En particulier, nous simulons l'implémentation matérielle de l'algorithme de chiffrement par flot Trivium, et nous analysons la présence de fuites d'information par canaux auxiliaires et la possibilité d'exploiter ces fuites dans le but de retrouver la clé de chiffrement utilisée par cet algorithme. Cette évaluation montre la vulnérabilité de Trivium face à une analyse des canaux auxiliaires, et la nécessité de sécuriser cet algorithme contre celle-ci. Nous dressons ensuite un état de l'art des contre-mesures matérielles existantes, qui ont pour but de protéger l'implémentation d'une primitive cryptographique contre l'une ou l'autre de ces attaques matérielles, ou contre les deux attaques à la fois. Ces contre-mesures souffrent de nombreux désavantages, tels qu'une consommation énergétique, un coût silicium ou encore un temps d'exécution élevés. De plus, de nombreuses contre-mesures requièrent l'utilisation d'un nombre important de bits aléatoires, dont la génération est coûteuse en énergie. Par exemple, les contre-mesures mixtes, efficaces à la fois contre les attaques par analyse des canaux auxiliaires et par injection de fautes, ont une surface et une consommation d'énergie supérieures à six fois celles d'une implémentation non protégée, ce chiffre ne prenant pas en compte le coût de la génération des centaines ou des milliers de bits aléatoires nécessaires lors de chaque exécution de l'algorithme protégé. Ce chapitre permet donc de conclure sur la nécessité de concevoir de nouvelles contre-mesures ou d'implémenter différemment les contre-mesures existantes, afin de réduire le coût énergétique de la sécurité matérielle des circuits intégrés utilisés dans l'IoT.

Dans le chapitre 4, nous proposons une nouvelle contre-mesure matérielle mixte, visant à protéger l'implémentation matérielle d'une primitive cryptographique contre les attaques par analyse des canaux auxiliaires et par injection de fautes, et dont le coût énergétique est plus faible que celui des protections mixtes existantes. Cette contre-mesure, appelée encodage dynamique, est particulièrement adaptée pour protéger l'implémentation matérielle de primitives cryptographiques légères construites à partir de registres à décalage. Cette contre-mesure consiste en un lissage al-

gorithmique de la consommation, obtenu en encodant différemment les bits consécutifs de sorte à garantir à la fois une distance et un poids de Hamming totaux constants. L'encodage dynamique, qui diffère de la plupart des propositions existantes de lissage de la consommation dans la mesure où elle ne nécessite pas de phases de précharge et d'évaluation, permet également une détection intrinsèque des fautes. Selon le type d'encodage dynamique sélectionné, la détection de fautes peut être plus efficace que celle fournie par beaucoup de contre-mesures existantes, en considérant les moyens d'injection existants décrits dans le chapitre 3. Plusieurs variantes de l'encodage dynamique sont proposées, et sont appliquées à la protection de l'implémentation d'un registre à décalage et de Trivium. Une évaluation de la vulnérabilités aux attaques par canaux auxiliaires de ces différentes variantes est menée à partir de simulations post-layout de la consommation de puissance. Celle-ci montre l'efficacité du lissage de la consommation d'énergie par cycle d'horloge, et l'impossibilité de réaliser des attaques par canaux auxiliaires telles qu'elles existent actuellement contre Trivium. Une évaluation des fuites d'information par canaux auxiliaires montre l'efficacité de l'encodage dynamique aléatoire, qui consiste à générer un encodage différent à chaque exécution de l'algorithme protégé. Des fuites d'information semblent présentes pour les autres variantes, mais cela peut s'expliquer, du moins partiellement, par un biais de la méthodologie utilisée en l'absence d'aléa et de bruit. Cette évaluation à partir de simulations doit donc être complétée par des mesures physiques sur une intégration sur ASIC de cette contre-mesure. Enfin, nous montrons que l'encodage dynamique a un coût en surface et en consommation environ quatre fois plus élevé qu'une implémentation non protégée, et nécessite l'utilisation de seulement 0 à 8 bits aléatoires par exécution de l'algorithme protégé ; le coût de cette protection est donc bien plus faible que celui des autres contre-mesures mixtes existantes.

Dans le chapitre 5, nous cherchons à améliorer l'efficacité énergétique de contre-mesures matérielles en les implémentant de manière adaptative, c'est-à-dire de sorte à pouvoir les activer ou les désactiver à volonté, durant le fonctionnement de l'algorithme protégé. L'augmentation du niveau de sécurité lors de l'activation des contre-mesures permet de répondre à divers cas d'usage, tels que la modification de la politique de sécurité suite à la publication de nouvelles attaques, la manipulation de données sensibles, ou encore la détection d'attaques en temps réel grâce à divers capteurs intégrés au circuit. À l'inverse, la réduction de ce niveau de sécurité, via la désactivation de contre-mesures, permet d'économiser de l'énergie. Deux types de contre-mesures sont étudiées dans ce chapitre : le masquage algorithmique, qui est une contre-mesure mature et largement étudiée contre les attaques par canaux auxiliaires, et l'encodage dynamique que nous avons introduit précédemment, qui protège l'implémentation d'un algorithme à la fois contre les attaques par canaux auxiliaires et par injection de fautes. Après avoir décrit une implémentation existante de masquage de Trivium au premier ordre, nous expliquons comment masquer l'implémentation de cet algorithme au second ordre. Nous présentons ensuite une implémentation différente du masquage au premier ordre, plus efficace énergétiquement que les implémentations existantes, qui s'appuie sur les observations formulées dans le chapitre 3 sur la présence de fuites d'information d'information par canaux auxiliaires de l'implémentation matérielle Trivium. Cette protection, dite semi-adaptative, consiste à appliquer le masquage uniquement durant l'initialisation de Trivium, durant laquelle toutes les attaques par canaux auxiliaires publiées à ce jour contre Trivium sont effectuées, et à désactiver le masquage après cette phase afin d'économiser de l'énergie. Durant la phase de chiffrement, cette protection a une consommation de puissance supérieure de seulement 6 % à celle d'une implémentation non protégée, soit presque trois fois moins qu'une implémentation de référence du masquage au premier ordre. Nous proposons également une implémentation de masquage adaptatif, dans laquelle le masquage peut être soit effectué au premier ou au second ordre, soit désactivé, ainsi qu'une implémentation d'encodage dynamique adaptatif, qui consiste à activer ou désactiver l'encodage dynamique. Ces deux dernières contre-mesures permettent de modifier le niveau de sécurité durant le fonctionnement de Trivium. Enfin, nous évaluons la sécurité des contre-mesures adaptatives et semi-adaptatives proposées, ainsi que leur coût d'implémentation et le gain en énergie apporté par l'approche adaptative. Cette évaluation est effectuée à partir de simulations post-layout, selon la méthodologie

décrite dans les chapitres 3 et 4. Elle montre l'absence de fuites d'information par canaux auxiliaires avec 100.000 traces de consommation simulées pour les différentes implémentations du masquage, et la difficulté d'exploiter les éventuelles fuites par canaux auxiliaires observables pour l'implémentation de l'encodage dynamique adaptatif. Tandis que l'implémentation adaptative des contre-mesures présente un léger surcoût en consommation de puissance et en surface par rapport aux contre-mesures non adaptatives en considérant un niveau de sécurité fixé, elle permet de réaliser d'importants gains en énergie sur l'ensemble de la durée de vie du circuit intégré si le niveau de sécurité varie au cours de cette durée. Par exemple, le masquage adaptatif au second ordre consomme 4,6 % plus de puissance que le masquage de référence au second ordre, mais la désactivation de ce masquage adaptatif permet de réduire la consommation de 76 % par rapport à celle de ce masquage de référence. Si l'on considère que le masquage est désactivé la plupart du temps, par exemple selon un cas d'usage où il serait activé seulement lors de la détection d'une attaque, cette contre-mesure adaptative présente un réel intérêt du point de vue de la consommation énergétique.

Enfin, dans le chapitre 6, nous décrivons l'intégration sur cibles physiques des différentes variantes de Trivium étudiées dans les chapitres 3 à 5, dont une version de référence non protégée, et plusieurs versions protégées avec les contre-mesures proposées au cours de cette thèse. Une analyse des émissions électromagnétiques de plusieurs implémentations sur FPGA permet de confirmer expérimentalement la présence de fuites d'information par canaux auxiliaires lorsque Trivium n'est pas protégé, ainsi que l'absence de fuites d'information avec 1.000.000 traces d'émissions électromagnétiques pour différentes versions du masquage. Deux ASIC ont été réalisés, contenant plusieurs implémentations de Trivium : une version non protégée, deux versions masquées respectivement au premier et au second ordre, une version protégée par le masquage adaptatif, et une version protégée par l'encodage dynamique. L'évaluation de ces implémentations sur ASIC n'a pas encore été effectuée, du fait de temps de fabrication élevés. Les mesures sur ces ASIC permettront de valider expérimentalement la sécurité contre les attaques par injection de fautes et par analyse des canaux auxiliaires des contre-mesures proposées au cours de cette thèse.

7.2 Perspectives

Les travaux réalisés au cours de cette thèse ont ouvert plusieurs pistes de recherche, qui sont présentées dans cette section.

Évaluation sur ASIC de la génération de codes de réveil

Les travaux sur la sécurisation de la radio de réveil ont été réalisés en considérant un protocole générique sur la radio principale, à la fois sécurisé et robuste face à la perte de messages. Cette sécurité et cette robustesse face à la perte de messages sont mises en œuvre différemment par les différents protocoles de communication existants. De plus, selon le cas d'usage et le protocole utilisé, le nombre et le contenu des messages échangés sur la radio principale peut varier, ce qui impacte la manière dont les codes de réveil sont calculés. La mise en application de ces travaux sur le circuit intégré WARRIOR devrait permettre d'évaluer expérimentalement la compatibilité de notre méthode avec plusieurs protocoles de communication existants, tels que le Wi-Fi et le Bluetooth, et d'optimiser cette méthode au cas par cas en fonction des spécificités de chaque protocole. Cette implémentation sur ASIC de notre méthode permettra également d'affiner l'évaluation du coût du calcul des codes de réveil, selon différents cas d'usage, grâce à la mesure physique de plusieurs paramètres qui ont été approximés ou négligés dans le chapitre 2. Enfin, des attaques par déni de sommeil seront menées afin de quantifier expérimentalement la sécurité apportée par notre méthode.

Évaluation sur ASIC des contre-mesures matérielles

Les contre-mesures matérielles présentées dans nos travaux ont été implémentées sur ASIC, ce qui permettra d'évaluer la protection apportée par celles-ci face aux attaques matérielles. Dans un premier temps, des mesures physiques, effectuées en présence de bruit, permettront de compléter les résultats obtenus avec des simulations post-layout. Ce bruit inhérent aux mesures physiques permet d'une part d'éliminer le biais des T-tests calculés en l'absence d'aléa, et d'autre part de couvrir partiellement ou totalement les légères variations de la consommation subsistant avec l'encodage dynamique du fait des légers déséquilibres des différents chemins de données. Dans un second temps, des injections de fautes avec divers moyens physiques permettra d'évaluer en pratique l'efficacité de la détection de fautes fournie par l'encodage dynamique.

Encodage dynamique de la logique combinatoire

Dans nos travaux, nous avons évalué la sécurité apportée par l'application de l'encodage dynamique à l'ensemble de l'implémentation d'un algorithme de chiffrement. Cet algorithme contient de nombreuses portes logiques séquentielles et combinatoires, qui contribuent toutes aux variations de la consommation totale de puissance ; de plus, ces variations de la consommation dépendent en partie du placement et du routage du circuit. Il pourrait être intéressant, afin de mieux comprendre l'origine des éventuelles fuites d'information, d'étudier séparément la consommation de chaque porte logique élémentaire protégée, avec des simulations électriques précises. Cette évaluation de chaque porte logique permettrait également de mieux évaluer le coût, en consommation et en surface, lié à la protection de la logique combinatoire. De plus, nous avons étudié l'encodage de la logique combinatoire uniquement pour une table d'encodage en particulier, parmi les 168 tables d'encodage possible. D'autres équations seraient utilisées pour d'autres tables d'encodage, et les fuites d'information observables pourraient donc varier en fonction de la table d'encodage utilisée, ce qui représente une perspective d'étude intéressante.

Enfin, nous n'avons pas implémenté d'encodage dynamique aléatoire de la logique combinatoire. Cet encodage dynamique aléatoire permettrait de réduire, voire de supprimer, les fuites observables par canaux auxiliaires, mais également d'améliorer l'efficacité de la protection contre les injections de fautes. Pour cela, il faut tout d'abord développer une méthode pour générer les équations représentant les portes logiques élémentaires protégées, pour toutes les tables d'encodage possibles. Dans un second temps, plusieurs jeux de ces équations, pour plusieurs tables d'encodage différentes, doivent être implémentés en parallèle, ainsi qu'un mécanisme permettant de choisir le jeu d'équations à utiliser en fonction de la table d'encodage sélectionnée. Il est nécessaire d'évaluer non seulement la vulnérabilité aux attaques matérielles de l'implémentation obtenue, selon la méthodologie suivie dans notre travaux, mais également le coût en surface et en consommation de cette implémentation. L'étude du nombre optimal de tables d'encodages différentes pour réaliser cet encodage dynamique aléatoire pourrait permettre de réduire les coûts d'implémentation en conservant un niveau de sécurité suffisant.

Optimisation de l'implémentation physique de l'encodage dynamique

Dans nos travaux, aucune contrainte spécifique n'a été imposée aux outils de conception lors de l'implémentation physique de l'encodage dynamique de Trivium dans le but d'équilibrer la consommation et les temps de propagation dans les les différents chemins de données. Des chemins de données déséquilibrés ont pour conséquence une augmentation des fuites d'information par canaux auxiliaires. En choisissant soigneusement les cellules standard utilisées, le placement de celles-ci, et le routage entre ces cellules et avec les sources d'alimentation, ces déséquilibres peuvent être diminués, voire supprimés. Cependant, la suppression totale des fuites d'information dues à ces déséquilibres peut nécessiter des efforts importants, probablement incompatibles avec les contraintes de coût de conception et de temps de mise sur le marché de circuits intégrés complexes. Plusieurs optimisations de l'implémentation physique peuvent être étudiées, afin de déterminer quelles optimisation permettraient de réduire au maximum les fuites d'information

en conservant un flot de conception standard et en fournissant des efforts acceptables pour une implémentation à l'échelle industrielle.

Application des contre-mesures matérielles à la protection d'autres algorithmes

Les contre-mesures matérielles présentées dans les chapitres 4 et 5 ont été implémentées pour la protection de l'algorithme de chiffrement par flot Trivium. Ces travaux peuvent être appliqués à la protection de primitives cryptographiques ayant une structure interne similaire à celle de Trivium, telles que l'algorithme de chiffrement par flot Grain [31], l'algorithme de chiffrement par bloc KATAN [157], ou encore la fonction de hachage cryptographique QUARK [158]. Il serait également intéressant d'étendre ces contre-mesures à la protection d'autres primitives cryptographiques ayant une structure différente, ce qui nécessiterait d'adapter les mécanismes proposés dans nos travaux pour la protection d'autres fonctions combinatoires, potentiellement plus complexes, telles que les opérations de substitution effectuées par de nombreux algorithmes de chiffrement. Par exemple, l'AES [27] est un algorithme de chiffrement très largement répandu, et il est notamment utilisé par de nombreux protocoles de communication dans l'IoT pour assurer la confidentialité des échanges de messages. Des travaux ont démarré au sein du laboratoire afin d'appliquer le masquage adaptatif à l'AES ; ceux-ci n'ont pas encore abouti, et devraient être approfondis dans le futur.

Contrôle des contre-mesures adaptatives

Dans nos travaux, nous avons décrit la manière dont les contre-mesures adaptatives protègent les données manipulées, mais nous n'avons pas traité de l'implémentation du mécanisme qui effectue le choix du niveau de protection à appliquer avec ces contre-mesures. Ce mécanisme de contrôle des contre-mesures adaptatives a déjà été étudié dans plusieurs travaux, notamment par les auteurs de [89] et de [164], mais une implémentation plus adaptée aux contraintes de l'IoT et au circuit L-IoT serait une perspective intéressante pour mettre nos travaux en application. En particulier, les travaux existants n'abordent pas l'efficacité énergétique d'un tel mécanisme. De plus, l'implémentation de ce mécanisme de contrôle doit être protégée contre les attaques matérielles, et en particulier contre les injections de fautes qui auraient pour but de corrompre ce mécanisme afin de réduire artificiellement le niveau de sécurité fourni par les contre-mesures adaptatives. Cette protection doit être mise en œuvre localement, au niveau des signaux de contrôle de chaque contre-mesure adaptative, mais également de manière globale, au niveau du mécanisme de contrôle centralisé qui évalue la présence d'attaques grâce à divers capteurs et décide du niveau de sécurité à appliquer pour toutes les primitives cryptographiques implémentées sur le circuit intégré.

Bibliographie

[1] "Gartner Says 8.4 Billion Connected "Things" Will Be in Use in 2017, Up 31 Percent From 2016," 2017. https://www.gartner.com/en/newsroom/press-releases/2017-02-07-gartner-says-8-billion-connected-things-will-be-in-use-in-2017-up-31-percent-from-2016 (Accessed : 2019-07-08).

[2] A. M. Nia and N. K. Jha, "A Comprehensive Study of Security of Internet-of-Things," *IEEE Transactions on Emerging Topics in Computing*, vol. PP, no. 99, pp. 1–1, 2016.

[3] J. Suhonen, M. Kohvakka, V. Kaseva, T. D. Hämäläinen, and M. Hännikäinen, "Low-power Wireless Sensor Network Platforms," in *Handbook of Signal Processing Systems* (S. S. Bhattacharyya, E. F. Deprettere, R. Leupers, and J. Takala, eds.), pp. 123–160, Springer US, 2010.

[4] S. Sathyadevan, S. Prabhakaranl, and K. Bipin, "A Survey of Security Protocols in WSN and Overhead Evaluation," in *Proceedings of the 3rd International Conference on Frontiers of Intelligent Computing : Theory and Applications (FICTA) 2014* (S. C. Satapathy, B. N. Biswal, S. K. Udgata, and J. K. Mandal, eds.), no. 328 in Advances in Intelligent Systems and Computing, pp. 729–738, Springer International Publishing, 2015.

[5] P. Baronti, P. Pillai, V. W. C. Chook, S. Chessa, A. Gotta, and Y. F. Hu, "Wireless sensor networks : A survey on the state of the art and the 802.15.4 and ZigBee standards," *Computer Communications*, vol. 30, no. 7, pp. 1655–1695, 2007.

[6] J. M. Dutertre, J. J. A. Fournier, A. P. Mirbaha, D. Naccache, J. B. Rigaud, B. Robisson, and A. Tria, "Review of fault injection mechanisms and consequences on countermeasures design," in *2011 6th International Conference on Design Technology of Integrated Systems in Nanoscale Era (DTIS)*, pp. 1–6, Apr. 2011.

[7] I. Kuon and J. Rose, "Measuring the Gap Between FPGAs and ASICs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, pp. 203–215, Feb. 2007.

[8] S. S. Kumar, K. I. Lemke, and C. Paar, "Some Thoughts about Implementation Properties of Stream Ciphers," in *SASC - State of the Art of Stream Ciphers Workshop*, 2004. https://www.emsec.ruhr-uni-bochum.de/research/publications/some-thoughts-about-implementation-properties-stre (Accessed : 2019-09-24).

[9] S. Ray, T. Hoque, A. Basak, and S. Bhunia, "The power play : Security-energy trade-offs in the IoT regime," in *2016 IEEE 34th International Conference on Computer Design (ICCD)*, pp. 690–693, Oct. 2016.

[10] "ENCOS White Paper," tech. rep., Université catholique de Louvain, Apr. 2018. https://cdn.uclouvain.be/groups/cms-editors-icteam/iot/ENCOS_white%20paper.pdf (Accessed : 2019-07-29).

[11] I. Miro-Panades, "L-IOT : a Flexible Energy Efficient Platform Targeting Wide Range IoT Applications," in *DAC 2017 IP Track*, (Austin, TX), June 2017. www2.dac.com/54th/proceedings/slides/25_3.pptx (Accessed : 2019-09-24).

[12] A. Waterman, Y. Lee, D. A. Patterson, and K. Asanovíc, "The RISC-V Instruction Set Manual, Volume I : User-Level ISA, Version 2.0," *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2014-54*, May 2014. https://www2.eecs.berkeley.edu/Pubs/TechRpts/2014/EECS-2014-54.html (Accessed : 2017-11-21).

[13] B. Martineau, C. Jany, F. Todeschini, D. Morche, and E. Mercier, "Towards fully integrated 28nm UTBB FD-SOI IoT node : The sub-50µW RF receiver," in *2016 IEEE SOI-3D-Subthreshold Microelectronics Technology Unified Conference (S3S)*, pp. 1–2, Oct. 2016.

[14] J. Oller i Bosch, *Wake-up radio systems : design, development, performance evaluation and comparison to conventional medium access control protocols for wireless sensor networks*. PhD thesis, Universitat Politècnica de Catalunya, Mar. 2015. `https://upcommons.upc.edu/handle/2117/95668` (Accessed : 2019-08-06).

[15] V. Jelicic, M. Magno, D. Brunelli, V. Bilas, and L. Benini, "Analytic Comparison of Wake-up Receivers for WSNs and Benefits over the Wake-on Radio Scheme," in *Proceedings of the 7th ACM Workshop on Performance Monitoring and Measurement of Heterogeneous Wireless and Wired Networks*, PM2HW2N '12, (New York, NY, USA), pp. 99–106, ACM, 2012.

[16] F. A. Aoudia, M. Magno, M. Gautier, O. Berder, and L. Benini, "Analytical and Experimental Evaluation of Wake-Up Receivers Based Protocols," in *2016 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–7, 2016.

[17] E. Khorov, A. Lyakhov, A. Krotov, and A. Guschin, "A survey on IEEE 802.11ah : An enabling networking technology for smart cities," *Computer Communications*, vol. 58, pp. 53–69, Mar. 2015.

[18] R. Want, B. Schilit, and D. Laskowski, "Bluetooth LE Finds Its Niche," *IEEE Pervasive Computing*, vol. 12, pp. 12–16, Oct. 2013.

[19] "HP News - HP Study Reveals 70 Percent of Internet of Things Devices Vulnerable to Attack." `https://www8.hp.com/us/en/hp-news/press-release.html?id=1744676` (Accessed : 2019-07-29).

[20] A. Greenberg, "Hackers Remotely Kill a Jeep on the Highway—With Me in It," *Wired*, July 2015. `https://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway` (Accessed : 2019-08-02).

[21] C. Li, A. Raghunathan, and N. K. Jha, "Hijacking an insulin pump : Security attacks and defenses for a diabetes therapy system," in *2011 IEEE 13th International Conference on e-Health Networking, Applications and Services*, pp. 150–156, June 2011.

[22] D. Halperin, T. S. Heydt-Benjamin, B. Ransford, S. S. Clark, B. Defend, W. Morgan, K. Fu, T. Kohno, and W. H. Maisel, "Pacemakers and Implantable Cardiac Defibrillators : Software Radio Attacks and Zero-Power Defenses," in *2008 IEEE Symposium on Security and Privacy (sp 2008)*, pp. 129–142, May 2008.

[23] Y. Zou, J. Zhu, X. Wang, and L. Hanzo, "A Survey on Wireless Security : Technical Challenges, Recent Advances, and Future Trends," *Proceedings of the IEEE*, vol. 104, pp. 1727–1765, Sept. 2016.

[24] Agence nationale de la sécurité des systèmes d'information (ANSSI), "Référentiel Général de Sécurité - Annexe B1 : Règles et recommandations concernant le choix et le dimensionnement des mécanismes cryptographiques," Feb. 2014. `https://www.ssi.gouv.fr/uploads/2014/11/RGS_v-2-0_B1.pdf` (Accessed : 2019-07-25).

[25] R. L. Rivest, A. Shamir, and L. M. Adleman, "Patent - Cryptographic communications system and method," Sept. 1983. `https://patents.google.com/patent/US4405829/en` (Accessed : 2019-07-25).

[26] J. H. Kong, L.-M. Ang, and K. P. Seng, "A comprehensive survey of modern symmetric cryptographic solutions for resource constrained environments," *Journal of Network and Computer Applications*, vol. 49, pp. 15–50, Mar. 2015.

[27] M. J. Dworkin, E. B. Barker, J. R. Nechvatal, J. Foti, L. E. Bassham, E. Roback, and J. F. D. Jr, "Advanced Encryption Standard (AES)," *Federal Inf. Process. Stds. (NIST FIPS) - 197*, Nov. 2001. `https://www.nist.gov/publications/advanced-encryption-standard-aes` (Accessed : 2017-11-10).

[28] "The eSTREAM portfolio page." `http://www.ecrypt.eu.org/stream` (Accessed : 2018-06-05).

[29] C. D. Canniere and B. Preneel, "TRIVIUM Specifications," *eSTREAM, ECRYPT Stream Cipher Project*, vol. 2006, 2006. `https://www.ecrypt.eu.org/stream/e2-trivium.html` (Accessed : 2018-06-01).

[30] "ISO/IEC 29192-3 :2012 - Information technology — Security techniques — Lightweight cryptography — Part 3 : Stream ciphers." `https://www.iso.org/standard/56426.html` (Accessed : 2019-07-25).

[31] M. Agren, M. Hell, T. Johansson, and W. Meier, "Grain-128a : A New Version of Grain-128 with Optional Authentication," *Int. J. Wire. Mob. Comput.*, vol. 5, pp. 48–59, Dec. 2011.

[32] T. Good and M. Benaissa, "ASIC Hardware Performance," in *New Stream Cipher Designs* (M. Robshaw and O. Billet, eds.), no. 4986 in Lecture Notes in Computer Science, pp. 267–293, Springer Berlin Heidelberg, 2008.

[33] A. Chakraborti, A. Chattopadhyay, M. Hassan, and M. Nandi, "TriviA : A Fast and Secure Authenticated Encryption Scheme," in *Cryptographic Hardware and Embedded Systems – CHES 2015* (T. Güneysu and H. Handschuh, eds.), Lecture Notes in Computer Science, pp. 330–353, Springer Berlin Heidelberg, 2015.

[34] A. Canteaut, S. Carpov, C. Fontaine, T. Lepoint, M. Naya-Plasencia, P. Paillier, and R. Sirdey, "Stream Ciphers : A Practical Solution for Efficient Homomorphic-Ciphertext Compression," *Journal of Cryptology*, vol. 31, pp. 885–916, July 2018.

[35] J. M. Mora-Gutiérrez, C. J. Jiménez-Fernández, and M. Valencia-Barrero, "Multiradix Trivium Implementations for Low-Power IoT Hardware," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, pp. 3401–3405, Dec. 2017.

[36] R. Billure, V. M. Tayur, and M. V, "Internet of Things - a study on the security challenges," in *Advance Computing Conference (IACC), 2015 IEEE International*, pp. 247–252, June 2015.

[37] L. Markowsky and G. Markowsky, "Scanning for vulnerable devices in the Internet of Things," in *2015 IEEE 8th International Conference on Intelligent Data Acquisition and Advanced Computing Systems : Technology and Applications (IDAACS)*, vol. 1, pp. 463–467, Sept. 2015.

[38] I. Zeifman, "Breaking Down Mirai : An IoT DDoS Botnet Analysis," Oct. 2016. `https://www.incapsula.com/blog/malware-analysis-mirai-ddos-botnet.html` (Accessed : 2016-12-06).

[39] M. Dener, "Security Analysis in Wireless Sensor Networks," *International Journal of Distributed Sensor Networks*, vol. 10, p. 303501, Oct. 2014.

[40] A. Barenghi, L. Breveglieri, I. Koren, and D. Naccache, "Fault Injection Attacks on Cryptographic Devices : Theory, Practice, and Countermeasures," *Proceedings of the IEEE*, vol. 100, pp. 3056–3076, Nov. 2012.

[41] D. Karaklajić, J. M. Schmidt, and I. Verbauwhede, "Hardware Designer's Guide to Fault Attacks," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 21, pp. 2295–2306, Dec. 2013.

[42] N. F. Galathy, B. Yuce, and P. Schaumont, "A Systematic Approach to Fault Attack Resistant Design," in *Fundamentals of IP and SoC Security : Design, Verification, and Debug* (S. Bhunia, S. Ray, and S. Sur-Kolay, eds.), pp. 223–245, Cham : Springer International Publishing, 2017.

[43] S. Mangard, E. Oswald, and T. Popp, *Power Analysis Attacks : Revealing the Secrets of Smart Cards*. Springer US, 2007.

[44] K. Mai, "Side Channel Attacks and Countermeasures," in *Introduction to Hardware Security and Trust* (M. Tehranipoor and C. Wang, eds.), pp. 175–194, Springer New York, 2012.

[45] E. Ronen, A. Shamir, A. Weingarten, and C. O'Flynn, "IoT Goes Nuclear : Creating a ZigBee Chain Reaction," in *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 195–212, May 2017.

[46] N. Homma, Y.-i. Hayashi, N. Miura, D. Fujimoto, D. Tanaka, M. Nagata, and T. Aoki, "EM Attack Is Non-invasive? - Design Methodology and Validity Verification of EM Attack Sensor," in *Cryptographic Hardware and Embedded Systems – CHES 2014* (L. Batina and M. Robshaw, eds.), Lecture Notes in Computer Science, pp. 1–16, Springer Berlin Heidelberg, 2014.

[47] D. Utyamishev and I. Partin-Vaisband, "Real-Time Detection of Power Analysis Attacks by Machine Learning of Power Supply Variations On-Chip," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 1–1, 2018.

[48] F. Kenarangi and I. Partin-Vaisband, "Exploiting Machine Learning Against On-Chip Power Analysis Attacks : Tradeoffs and Design Considerations," *IEEE Transactions on Circuits and Systems I : Regular Papers*, vol. 66, pp. 769–781, Feb. 2019.

[49] G. Agosta, A. Barenghi, G. Pelosi, and M. Scandale, "Reactive side-channel countermeasures : Applicability and quantitative security evaluation," *Microprocessors and Microsystems*, vol. 62, pp. 50–60, Oct. 2018.

[50] R. Falk and H. J. Hof, "Fighting Insomnia : A Secure Wake-Up Scheme for Wireless Sensor Networks," in *2009 Third International Conference on Emerging Security Information, Systems and Technologies*, pp. 191–196, June 2009.

[51] A. Tang, S. Sethumadhavan, and S. Stolfo, "{CLKSCREW} : Exposing the Perils of Security-Oblivious Energy Management," in *26th {USENIX} Security Symposium ({USENIX} Security 17)*, pp. 1057–1074, 2017. `https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/tang` (Accessed : 2019-08-22).

[52] A. Kavoukis and S. Aljareh, "Efficient time synchronized one-time password scheme to provide secure wake-up authentication on wireless sensor networks," *International Journal of Advanced Smart Sensor Network Systems*, vol. 3, pp. 1–11, Jan. 2013. arXiv : 1302.1756.

[53] O. Stecklina, S. Kornemann, and M. Methfessel, "A secure wake-up scheme for low power wireless sensor nodes," in *2014 International Conference on Collaboration Technologies and Systems (CTS)*, pp. 279–286, 2014.

[54] D. M'Raihi, S. Machani, M. Pei, and J. Rydell, "IETF RFC6238 - TOTP : Time-based one-time password algorithm," tech. rep., 2011. `https://tools.ietf.org/html/rfc6238` (Accessed : 2019-09-24).

[55] J.-W. Liu, M. A. Ameen, and K.-S. Kwak, "Secure Wake-Up Scheme for WBANs," *IEICE Transactions on Communications*, vol. E93.B, no. 4, pp. 854–857, 2010.

[56] A. T. Capossele, V. Cervo, C. Petrioli, and D. Spenza, "Counteracting Denial-of-Sleep Attacks in Wake-Up-Radio-Based Sensing Systems," in *2016 13th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, pp. 1–9, June 2016.

[57] D. J. Wheeler and R. M. Needham, "TEA, a tiny encryption algorithm," in *Fast Software Encryption*, Lecture Notes in Computer Science, pp. 363–366, Springer, Berlin, Heidelberg, Dec. 1994.

[58] Q. H. Dang, "Secure Hash Standard," *Federal Inf. Process. Stds. (NIST FIPS) - 180-4*, Aug. 2015. `https://www.nist.gov/publications/secure-hash-standard` (Accessed : 2017-11-10).

[59] T. Instrument, "MSP430f15x, MSP430f16x, MSP430f161x mixed signal microcontroller - datasheet," 2011. `https://www.ti.com/lit/ds/symlink/msp430f1611.pdf` (Accessed : 2019-09-24).

[60] K.-F. Krentz and C. Meinel, "Denial-of-sleep defenses for IEEE 802.15.4 coordinated sampled listening (CSL)," *Computer Networks*, vol. 148, pp. 60–71, Jan. 2019.

[61] A. Bogdanov, M. Knežević, G. Leander, D. Toz, K. Varıcı, and I. Verbauwhede, "spongent : A Lightweight Hash Function," in *Cryptographic Hardware and Embedded Systems – CHES 2011*, pp. 312–325, Springer, Berlin, Heidelberg, Sept. 2011.

[62] "AS3932 - Programmable 3d Low Power LF Wake-up Receiver IC - ams." `https://ams.com/as3932` (Accessed : 2019-08-07).

[63] J. Balasch, B. Ege, T. Eisenbarth, B. Gérard, Z. Gong, T. Güneysu, S. Heyse, S. Kerckhof, F. Koeune, T. Plos, T. Pöppelmann, F. Regazzoni, F.-X. Standaert, G. V. Assche, R. V. Keer, L. v. O. t. Oldenzeel, and I. v. Maurich, "Compact Implementation and Performance Evaluation of Hash Functions in ATtiny Devices," in *Smart Card Research and Advanced Applications*, pp. 158–172, Springer, Berlin, Heidelberg, Nov. 2012.

[64] N. Mentens, "Hiding side-channel leakage through hardware randomization : A comprehensive overview," in *2017 International Conference on Embedded Computer Systems : Architectures, Modeling, and Simulation (SAMOS)*, pp. 269–272, July 2017.

[65] D. Agrawal, B. Archambeault, J. R. Rao, and P. Rohatgi, "The EM Side—Channel(s)," in *Cryptographic Hardware and Embedded Systems - CHES 2002* (B. S. Kaliski, e. K. Koç, and C. Paar, eds.), Lecture Notes in Computer Science, pp. 29–45, Springer Berlin Heidelberg, 2003.

[66] P. C. Kocher, "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems," in *Advances in Cryptology — CRYPTO '96* (N. Koblitz, ed.), Lecture Notes in Computer Science, pp. 104–113, Springer Berlin Heidelberg, 1996.

[67] D. Genkin, A. Shamir, and E. Tromer, "RSA Key Extraction via Low-Bandwidth Acoustic Cryptanalysis," *IACR Cryptology ePrint Archive*, no. 857, 2013. `https://eprint.iacr.org/2013/857` (Accessed : 2019-08-23).

[68] P. Kocher, J. Jaffe, and B. Jun, "Differential Power Analysis," in *Advances in Cryptology — CRYPTO' 99* (M. Wiener, ed.), Lecture Notes in Computer Science, pp. 388–397, Springer Berlin Heidelberg, 1999.

[69] STMicroelectronics, "STM32 32-bit Arm Cortex MCUs." `https://www.st.com/en/microcontrollers-microprocessors/stm32-32-bit-arm-cortex-mcus.html` (Accessed : 2019-08-23).

[70] W. Fischer, B. M. Gammel, O. Kniffler, and J. Velten, "Differential Power Analysis of Stream Ciphers," in *Topics in Cryptology – CT-RSA 2007*, Lecture Notes in Computer Science, pp. 257–270, Springer, Berlin, Heidelberg, Feb. 2007.

[71] D. Strobel, "Side Channel Analysis Attacks on Stream Ciphers," master thesis, Ruhr-Universität Bochum, 2009. `https://www.researchgate.net/publication/251447277_Side_Channel_Analysis_Attacks_on_Stream_Ciphers` (Accessed : 2017-09-11).

[72] Y. Jia, Y. Hu, F. Wang, and H. Wang, "Correlation power analysis of Trivium," *Security and Communication Networks*, vol. 5, no. 5, pp. 479–484, 2011.

[73] E. Tena-Sánchez and A. J. Acosta, "Optimized DPA attack on Trivium stream cipher using correlation shape distinguishers," in *2015 Conference on Design of Circuits and Integrated Systems (DCIS)*, pp. 1–6, Nov. 2015.

[74] E. Tena-Sánchez and A. J. Acosta, "DPA vulnerability analysis on Trivium stream cipher using an optimized power model," in *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1846–1849, May 2015.

[75] B. Gierlichs, L. Batina, C. Clavier, T. Eisenbarth, A. Gouget, H. Handschuh, T. Kasper, K. Lemke-Rust, S. Mangard, A. Moradi, and E. Oswald, "Susceptibility of eSTREAM candidates towards side channel analysis," in *Proceedings of SASC 2008*, 2008. `https://www.researchgate.net/publication/228371563_Susceptibility_of_eSTREAM_candidates_towards_side_channel_analysis` (Accessed : 2019-09-24).

[76] Rohde-Schwarz, "Rohde Schwarz RTO2000 oscilloscope." `https://www.rohde-schwarz.com/fr/produit/rto-page-de-demarrage-produits_63493-10790.html` (Accessed : 2019-07-25).

[77] G. Goodwill, B. Jun, J. Jaffe, and P. Rohatgi, "A testing methodology for side-channel resistance validation," in *NIST non-invasive attack testing workshop*, vol. 7, pp. 115–136, 2011. `https://www.semanticscholar.org/paper/A-testing-methodology-for-side-channel-resistance-Goodwill-Jun/97b6be2eaeebe1e13696e928e94f66b4c93719b8` (Accessed : 2018-10-29).

[78] T. Schneider and A. Moradi, "Leakage Assessment Methodology," in *Cryptographic Hardware and Embedded Systems – CHES 2015*, Lecture Notes in Computer Science, pp. 495–513, Springer, Berlin, Heidelberg, Sept. 2015.

[79] M. Agoyan, J.-M. Dutertre, D. Naccache, B. Robisson, and A. Tria, "When Clocks Fail : On Critical Paths and Clock Faults," in *Smart Card Research and Advanced Application* (D. Gollmann, J.-L. Lanet, and J. Iguchi-Cartigny, eds.), Lecture Notes in Computer Science, pp. 182–193, Springer Berlin Heidelberg, 2010.

[80] R. B. Carpi, S. Picek, L. Batina, F. Menarini, D. Jakobovic, and M. Golub, "Glitch It If You Can : Parameter Search Strategies for Successful Fault Injection," in *Smart Card Research and Advanced Applications* (A. Francillon and P. Rohatgi, eds.), Lecture Notes in Computer Science, pp. 236–252, Springer International Publishing, 2014.

[81] S. Anceau, P. Bleuet, J. Clédière, L. Maingault, J.-l. Rainard, and R. Tucoulou, "Nanofocused X-Ray Beam to Reprogram Secure Circuits," in *Cryptographic Hardware and Embedded Systems – CHES 2017* (W. Fischer and N. Homma, eds.), Lecture Notes in Computer Science, pp. 175–188, Springer International Publishing, 2017.

[82] A. Mirbaha, J. Dutertre, and A. Tria, "Differential analysis of Round-Reduced AES faulty ciphertexts," in *2013 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFTS)*, pp. 204–211, Oct. 2013.

[83] Sung-Ming Yen and M. Joye, "Checking before output may not be enough against fault-based cryptanalysis," *IEEE Transactions on Computers*, vol. 49, pp. 967–970, Sept. 2000.

[84] E. Biham and A. Shamir, "Differential fault analysis of secret key cryptosystems," in *Advances in Cryptology — CRYPTO '97* (B. S. Kaliski, ed.), Lecture Notes in Computer Science, pp. 513–525, Springer Berlin Heidelberg, 1997.

[85] M. Hojsík and B. Rudolf, "Floating Fault Analysis of Trivium," in *Progress in Cryptology - INDOCRYPT 2008*, Lecture Notes in Computer Science, pp. 239–250, Springer, Berlin, Heidelberg, Dec. 2008.

[86] Y. Hu, J. Gao, Q. Liu, and Y. Zhang, "Fault analysis of Trivium," *Designs, Codes and Cryptography*, vol. 62, pp. 289–311, Mar. 2012.

[87] F. E. Potestad-Ordóñez, C. J. Jiménez-Fernández, and M. Valencia-Barrero, "Vulnerability Analysis of Trivium FPGA Implementations," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, pp. 3380–3389, Dec. 2017.

[88] Y. Komano and S. Hirose, "Re-Keying Scheme Revisited : Security Model and Instantiations," *Applied Sciences*, vol. 9, p. 1002, Jan. 2019.

[89] B. Robisson, M. Agoyan, P. Soquet, S. Le-Henaff, F. Wajsbürt, P. Bazargan-Sabet, and G. Phan, "Smart security management in secure devices," *Journal of Cryptographic Engineering*, vol. 7, pp. 47–61, Apr. 2017.

[90] P. C. Liu, H. C. Chang, and C. Y. Lee, "A Low Overhead DPA Countermeasure Circuit Based on Ring Oscillators," *IEEE Transactions on Circuits and Systems II : Express Briefs*, vol. 57, pp. 546–550, July 2010.

[91] W. Shan, X. Fu, and Z. Xu, "A Secure Reconfigurable Crypto IC With Countermeasures Against SPA, DPA, and EMA," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, pp. 1201–1205, July 2015.

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

[93] N. Kamoun, L. Bossuet, and A. Ghazel, "A masked Correlated Power Noise Generator use as a second order DPA countermeasure to secure hardware AES cipher," in *ICM 2011 Proceeding*, pp. 1–5, 2011.

[94] L. Benini, E. Omerbegovic, A. Macii, M. Poncino, E. Macii, and F. Pro, "Energy-aware design techniques for differential power analysis protection," in *Design Automation Conference, 2003. Proceedings*, pp. 36–41, June 2003.

[95] F. Poucheret, L. Barthe, P. Benoit, L. Torres, P. Maurine, and M. Robert, "Spatial EM jamming : A countermeasure against EM Analysis ?," in *2010 18th IEEE/IFIP International Conference on VLSI and System-on-Chip*, pp. 105–110, Sept. 2010.

[96] Z. He, T. Ao, M. Wan, K. Dai, and X. Zou, "ERIST : An Efficient Randomized Instruction Insertion Technique to Counter Side-Channel Attacks.," *IAENG International Journal of Computer Science*, vol. 23, no. 1, 2016.

[97] S. Jerábek, J. Schmidt, M. Novotný, and V. Miškovský, "Dummy Rounds as a DPA Countermeasure in Hardware," in *2018 21st Euromicro Conference on Digital System Design (DSD)*, pp. 523–528, Aug. 2018.

[98] C. Clavier, J.-S. Coron, and N. Dabbous, "Differential Power Analysis in the Presence of Hardware Countermeasures," in *Cryptographic Hardware and Embedded Systems — CHES 2000* (e. K. Koç and C. Paar, eds.), no. 1965 in Lecture Notes in Computer Science, pp. 252–263, Springer Berlin Heidelberg, Aug. 2000.

[99] Shengqi Yang, W. Wolf, N. Vijaykrishnan, D. N. Serpanos, and Yuan Xie, "Power attack resistant cryptosystem design : a dynamic voltage and frequency switching approach," in *Design, Automation and Test in Europe*, pp. 64–69 Vol. 3, Mar. 2005.

[100] K. Baddam and M. Zwolinski, "Evaluation of Dynamic Voltage and Frequency Scaling as a Differential Power Analysis Countermeasure," in *20th International Conference on VLSI Design held jointly with 6th International Conference on Embedded Systems (VLSID'07)*, pp. 854–862, Jan. 2007.

[101] H. Geng, K. A. Kwiat, C. A. Kamhoua, and Y. Shi, "On Random Dynamic Voltage Scaling for Internet-of-Things : A Game-Theoretic Approach," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. PP, no. 99, pp. 1–1, 2017.

[102] A. Shamir, "Protecting Smart Cards from Passive Power Analysis with Detached Power Supplies," in *Cryptographic Hardware and Embedded Systems — CHES 2000* (e. K. Koç and C. Paar, eds.), no. 1965 in Lecture Notes in Computer Science, pp. 71–77, Springer Berlin Heidelberg, Aug. 2000.

[103] C. Tokunaga and D. Blaauw, "Securing Encryption Systems With a Switched Capacitor Current Equalizer," *IEEE Journal of Solid-State Circuits*, vol. 45, pp. 23–31, Jan. 2010.

[104] G. B. Ratanpal, R. D. Williams, and T. N. Blalock, "An on-chip signal suppression countermeasure to power analysis attacks," *IEEE Transactions on Dependable and Secure Computing*, vol. 1, pp. 179–189, July 2004.

[105] A. Attaran and M. Mirhassani, "An embedded low-overhead PLL-based countermeasure against DPA side channel attack," in *2015 International Symposium on Signals, Circuits and Systems (ISSCS)*, pp. 1–4, July 2015.

[106] A. Singh, M. Kar, S. Mathew, A. Rajan, V. De, and S. Mukhopadhyay, "Exploiting on-chip power management for side-channel security," in *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*, pp. 401–406, Mar. 2018.

[107] D. Das, M. Nath, B. Chatterjee, S. Ghosh, and S. Sen, "STELLAR : A Generic EM Side-Channel Attack Protection through Ground-Up Root-cause Analysis," in *2019 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pp. 11–20, May 2019.

[108] K. Tiri and I. Verbauwhede, "A Dynamic and Differential CMOS Logic Style to Resist Power and Timing Attacks on Security IC's.," *IACR Cryptology ePrint Archive*, vol. 2004, p. 66, 2004. `https://eprint.iacr.org/2004/066` (Accessed : 2019-09-24).

[109] K. Tiri and I. Verbauwhede, "A logic level design methodology for a secure DPA resistant ASIC or FPGA implementation," in *Proceedings of the Conference on Design, Automation and Test in Europe*, vol. 1, pp. 246–251, Feb. 2004.

[110] D. Suzuki, M. Saeki, and T. Ichikawa, "Random Switching Logic : A Countermeasure against DPA based on Transition Probability," *IACR Cryptology ePrint Archive*, no. 346, 2004. `https://eprint.iacr.org/2004/346` (Accessed : 2016-06-12).

[111] H. Marzouqi, M. Al-Qutayri, and K. Salah, "Review of gate-level differential power analysis and fault analysis countermeasures," *IET Information Security*, vol. 8, pp. 51–66, Jan. 2014.

[112] S. Moore, R. Anderson, P. Cunningham, R. Mullins, and G. Taylor, "Improving smart card security using self-timed circuits," in *Proceedings Eighth International Symposium on Asynchronous Circuits and Systems*, pp. 211–218, Apr. 2002.

[113] M. Mayhew and R. Muresan, "An overview of hardware-level statistical power analysis attack countermeasures," *Journal of Cryptographic Engineering*, vol. 7, pp. 213–244, Sept. 2017.

[114] D. Bellizia, G. Scotti, and A. Trifiletti, "TEL Logic Style as a Countermeasure Against Side-Channel Attacks : Secure Cells Library in 65nm CMOS and Experimental Results," *IEEE Transactions on Circuits and Systems I : Regular Papers*, vol. 65, pp. 3874–3884, Nov. 2018.

[115] R. E. Atani, S. Mirzakuchaki, S. E. Atani, and W. Meier, "On DPA-Resistive Implementation of FSR-based Stream Ciphers using SABL Logic Styles," *International Journal of Computers Communications & Control*, vol. 3, pp. 324–335, Dec. 2008.

[116] X. Li, C. Yang, J. Ma, Y. Liu, and S. Yin, "Energy-Efficient Side-Channel Attack Countermeasure With Awareness and Hybrid Configuration Based on It," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, pp. 3355–3368, Dec. 2017.

[117] D. Bellizia, S. Bongiovanni, P. Monsurrò, G. Scotti, A. Trifiletti, and F. B. Trotta, "Secure Double Rate Registers as an RTL Countermeasure Against Power Analysis Attacks," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, pp. 1368–1376, July 2018.

[118] D. Jayasinghe, A. Ignjatovic, J. A. Ambrose, R. Ragel, and S. Parameswaran, "Quadseal : Quadruple algorithmic symmetrizing countermeasure against power based side-channel attacks," in *2015 International Conference on Compilers, Architecture and Synthesis for Embedded Systems (CASES)*, pp. 21–30, IEEE, 2015.

[119] D. Jayasinghe, A. Ignjatovic, and S. Parameswaran, "NORA : Algorithmic Balancing without Pre-charge to Thwart Power Analysis Attacks," in *2017 30th International Conference on VLSI Design and 2017 16th International Conference on Embedded Systems (VLSID)*, pp. 167–172, Jan. 2017.

[120] S. Burman, D. Mukhopadhyay, and K. Veezhinathan, "LFSR Based Stream Ciphers Are Vulnerable to Power Attacks," in *Progress in Cryptology – INDOCRYPT 2007*, Lecture Notes in Computer Science, pp. 384–392, Springer, Berlin, Heidelberg, Dec. 2007.

[121] Y. Zhao, X. Yang, and R. Li, "Design of feedback shift register of against power analysis attack," *Computers, Materials and Continua*, vol. 58, no. 2, pp. 517–527, 2019.

[122] Z. Chen and Y. Zhou, "Dual-Rail Random Switching Logic : A Countermeasure to Reduce Side Channel Leakage," in *Cryptographic Hardware and Embedded Systems - CHES 2006* (L. Goubin and M. Matsui, eds.), Lecture Notes in Computer Science, pp. 242–254, Springer Berlin Heidelberg, 2006.

[123] T. Popp, M. Kirschbaum, T. Zefferer, and S. Mangard, "Evaluation of the Masked Logic Style MDPL on a Prototype Chip," in *Cryptographic Hardware and Embedded Systems - CHES 2007* (P. Paillier and I. Verbauwhede, eds.), Lecture Notes in Computer Science, pp. 81–94, Springer Berlin Heidelberg, 2007.

[124] A. Gornik, A. Moradi, J. Oehm, and C. Paar, "A Hardware-Based Countermeasure to Reduce Side-Channel Leakage : Design, Implementation, and Evaluation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, pp. 1308–1319, Aug. 2015.

[125] P. Schaumont and K. Tiri, "Masking and Dual-Rail Logic Don't Add Up," in *Cryptographic Hardware and Embedded Systems - CHES 2007* (P. Paillier and I. Verbauwhede, eds.), Lecture Notes in Computer Science, pp. 95–106, Springer Berlin Heidelberg, 2007.

[126] A. Moradi, T. Eisenbarth, A. Poschmann, C. Rolfes, C. Paar, M. T. M. Shalmani, and M. Salmasizadeh, "Information Leakage of Flip-Flops in DPA-Resistant Logic Styles.," *IACR Cryptology ePrint Archive*, vol. 2008, no. 188, 2008. `https://www.iacr.org/cryptodb/data/paper.php?pubkey=17865` (Accessed : 2019-09-24).

[127] A. Moradi, M. Kirschbaum, T. Eisenbarth, and C. Paar, "Masked Dual-Rail Precharge Logic Encounters State-of-the-Art Power Analysis Methods," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 20, pp. 1578–1589, Sept. 2012.

[128] J. Waddle and D. Wagner, "Towards Efficient Second-Order Power Analysis," in *Cryptographic Hardware and Embedded Systems - CHES 2004* (M. Joye and J.-J. Quisquater, eds.), Lecture Notes in Computer Science, pp. 1–15, Springer Berlin Heidelberg, 2004.

[129] F.-X. Standaert, "How (Not) to Use Welch's T-Test in Side-Channel Security Evaluations," in *Smart Card Research and Advanced Applications* (B. Bilgin and J.-B. Fischer, eds.), Lecture Notes in Computer Science, pp. 65–79, Springer International Publishing, 2019.

[130] O. Reparaz, B. Bilgin, S. Nikova, B. Gierlichs, and I. Verbauwhede, "Consolidating Masking Schemes," in *Advances in Cryptology – CRYPTO 2015* (R. Gennaro and M. Robshaw, eds.), Lecture Notes in Computer Science, pp. 764–783, Springer Berlin Heidelberg, 2015.

[131] Y. Ishai, A. Sahai, and D. Wagner, "Private Circuits : Securing Hardware against Probing Attacks," in *Advances in Cryptology - CRYPTO 2003* (D. Boneh, ed.), Lecture Notes in Computer Science, pp. 463–481, Springer Berlin Heidelberg, 2003.

[132] D. Canright and L. Batina, "A Very Compact "Perfectly Masked" S-Box for AES," in *Applied Cryptography and Network Security* (S. M. Bellovin, R. Gennaro, A. Keromytis, and M. Yung, eds.), no. 5037 in Lecture Notes in Computer Science, pp. 446–459, Springer Berlin Heidelberg, June 2008.

[133] S. Mangard, N. Pramstaller, and E. Oswald, "Successfully Attacking Masked AES Hardware Implementations," in *Cryptographic Hardware and Embedded Systems – CHES 2005* (J. R. Rao and B. Sunar, eds.), no. 3659 in Lecture Notes in Computer Science, pp. 157–171, Springer Berlin Heidelberg, Aug. 2005.

[134] S. Nikova, C. Rechberger, and V. Rijmen, "Threshold Implementations Against Side-Channel Attacks and Glitches," in *Information and Communications Security*, Lecture Notes in Computer Science, pp. 529–545, Springer, Berlin, Heidelberg, Dec. 2006.

[135] B. Bilgin, B. Gierlichs, S. Nikova, V. Nikov, and V. Rijmen, "Higher-Order Threshold Implementations," in *Advances in Cryptology – ASIACRYPT 2014*, Lecture Notes in Computer Science, pp. 326–343, Springer, Berlin, Heidelberg, Dec. 2014.

[136] A. Moradi, A. Poschmann, S. Ling, C. Paar, and H. Wang, "Pushing the Limits : A Very Compact and a Threshold Implementation of AES," in *Advances in Cryptology – EUROCRYPT 2011*, Lecture Notes in Computer Science, pp. 69–88, Springer, Berlin, Heidelberg, May 2011.

[137] T. D. Cnudde and S. Nikova, "Securing the PRESENT Block Cipher Against Combined Side-Channel Analysis and Fault Attacks," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, pp. 3291–3301, Dec. 2017.

[138] D. Shanmugam and S. Annadurai, "Secure Implementation of Stream Cipher : Trivium," in *Innovative Security Solutions for Information Technology and Communications*, Lecture Notes in Computer Science, pp. 253–266, Springer, Cham, June 2015.

[139] A. Moradi and A. Wild, "Assessment of Hiding the Higher-Order Leakages in Hardware," in *Cryptographic Hardware and Embedded Systems – CHES 2015* (T. Güneysu and H. Handschuh, eds.), Lecture Notes in Computer Science, pp. 453–474, Springer Berlin Heidelberg, 2015.

[140] T. Schneider, A. Moradi, and T. Güneysu, "ParTI – Towards Combined Hardware Countermeasures Against Side-Channel and Fault-Injection Attacks," in *Advances in Cryptology – CRYPTO 2016* (M. Robshaw and J. Katz, eds.), Lecture Notes in Computer Science, pp. 302–332, Springer Berlin Heidelberg, Aug. 2016.

[141] T. d. Cnudde and S. Nikova, "More Efficient Private Circuits II through Threshold Implementations," in *2016 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, pp. 114–124, Aug. 2016.

[142] O. Reparaz, L. De Meyer, B. Bilgin, V. Arribas, S. Nikova, V. Nikov, and N. Smart, "CAPA : The Spirit of Beaver Against Physical Attacks," in *Advances in Cryptology – CRYPTO 2018* (H. Shacham and A. Boldyreva, eds.), Lecture Notes in Computer Science, pp. 121–151, Springer International Publishing, 2018.

[143] L. D. Meyer, V. Arribas, S. Nikova, V. Nikov, and V. Rijmen, "M&M : Masks and Macs against Physical Attacks," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 25–50, 2019.

[144] T. D. Cnudde, O. Reparaz, B. Bilgin, S. Nikova, V. Nikov, and V. Rijmen, "Masking AES with d+1 Shares in Hardware," in *Cryptographic Hardware and Embedded Systems – CHES 2016*, Lecture Notes in Computer Science, pp. 194–212, Springer, Berlin, Heidelberg, Aug. 2016.

[145] C. Chen, M. Farmani, and T. Eisenbarth, "A Tale of Two Shares : Why Two-Share Threshold Implementation Seems Worthwhile—and Why It Is Not," in *Advances in Cryptology – ASIACRYPT 2016* (J. H. Cheon and T. Takagi, eds.), Lecture Notes in Computer Science, pp. 819–843, Springer Berlin Heidelberg, 2016.

[146] F. Bouesse, M. Renaudin, and G. Sicard, "Improving DPA Resistance of Quasi Delay Insensitive Circuits Using Randomly Time-shifted Acknowledgment Signals," in *Vlsi-Soc : From Systems To Silicon* (R. Reis, A. Osseiran, and H.-J. Pfleiderer, eds.), IFIP International Federation for Information Proc, pp. 11–24, Springer US, 2007.

[147] J. G. J. v. Woudenberg, M. F. Witteman, and F. Menarini, "Practical Optical Fault Injection on Secure Microcontrollers," in *2011 Workshop on Fault Diagnosis and Tolerance in Cryptography*, pp. 91–99, Sept. 2011.

[148] P. Maistri, "Countermeasures against fault attacks : The good, the bad, and the ugly," in *2011 IEEE 17th International On-Line Testing Symposium*, pp. 134–137, July 2011.

[149] X. Guo, D. Mukhopadhyay, C. Jin, and R. Karri, "Security analysis of concurrent error detection against differential fault analysis," *Journal of Cryptographic Engineering*, vol. 5, pp. 153–169, Sept. 2015.

[150] F. Amiel, K. Villegas, B. Feix, and L. Marcel, "Passive and Active Combined Attacks : Combining Fault Attacks and Side Channel Analysis," in *Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC 2007)*, pp. 92–102, Sept. 2007.

[151] Y. Yu, F. Marranghello, V. D. Teijeira, and E. Dubrova, "One-Sided Countermeasures for Side-Channel Attacks Can Backfire," in *Proceedings of the 11th ACM Conference on Security & Privacy in Wireless and Mobile Networks*, WiSec '18, (New York, NY, USA), pp. 299–301, ACM, 2018.

[152] M. Doulcier-Verdier, J.-M. Dutertre, J. Fournier, J.-B. Rigaud, B. Robisson, and A. Tria, "A side-channel and fault-attack resistant AES circuit working on duplicated complemented values," in *2011 IEEE International Solid-State Circuits Conference*, (San Francisco, CA, USA), pp. 274–276, IEEE, Feb. 2011.

[153] Y. Sung-Ming, S. Kim, S. Lim, and S. Moon, "A Countermeasure against One Physical Cryptanalysis May Benefit Another Attack," in *Information Security and Cryptology — ICISC 2001* (K. Kim, ed.), no. 2288 in Lecture Notes in Computer Science, pp. 414–427, Springer Berlin Heidelberg, Dec. 2001.

[154] K. J. Kulikowski, M. G. Karpovsky, and A. Taubin, "DPA on Faulty Cryptographic Hardware and Countermeasures," in *Fault Diagnosis and Tolerance in Cryptography* (L. Breveglieri, I. Koren, D. Naccache, and J.-P. Seifert, eds.), no. 4236 in Lecture Notes in Computer Science, pp. 211–222, Springer Berlin Heidelberg, 2006.

[155] Y. Ishai, M. Prabhakaran, A. Sahai, and D. Wagner, "Private Circuits II : Keeping Secrets in Tamperable Circuits," in *Advances in Cryptology - EUROCRYPT 2006* (S. Vaudenay, ed.), Lecture Notes in Computer Science, pp. 308–327, Springer Berlin Heidelberg, 2006.

[156] A. Poschmann, A. Moradi, K. Khoo, C.-W. Lim, H. Wang, and S. Ling, "Side-Channel Resistant Crypto for Less than 2,300 GE," *Journal of Cryptology*, vol. 24, pp. 322–345, Apr. 2011.

[157] C. De Cannière, O. Dunkelman, and M. Knežević, "KATAN and KTANTAN — A Family of Small and Efficient Hardware-Oriented Block Ciphers," in *Cryptographic Hardware and Embedded Systems - CHES 2009* (C. Clavier and K. Gaj, eds.), Lecture Notes in Computer Science, pp. 272–288, Springer Berlin Heidelberg, 2009.

[158] J.-P. Aumasson, L. Henzen, W. Meier, and M. Naya-Plasencia, "Quark : A Lightweight Hash," in *Cryptographic Hardware and Embedded Systems, CHES 2010* (S. Mangard and F.-X. Standaert, eds.), Lecture Notes in Computer Science, pp. 1–15, Springer Berlin Heidelberg, 2010.

[159] G. V. Bard, N. T. Courtois, J. Nakahara, P. Sepehrdad, and B. Zhang, "Algebraic, AIDA/Cube and Side Channel Analysis of KATAN Family of Block Ciphers," in *Progress in Cryptology - INDOCRYPT 2010* (G. Gong and K. C. Gupta, eds.), Lecture Notes in Computer Science, pp. 176–196, Springer Berlin Heidelberg, 2010.

[160] A. R. Kazmi, M. Afzal, M. F. Amjad, H. Abbas, and X. Yang, "Algebraic Side Channel Attack on Trivium and Grain Ciphers," *IEEE Access*, vol. 5, pp. 23958–23968, 2017.

[161] A. Berzati, C. Canovas, G. Castagnos, B. Debraize, L. Goubin, A. Gouget, P. Paillier, and S. Salgado, "Fault analysis of GRAIN-128," in *2009 IEEE International Workshop on Hardware-Oriented Security and Trust*, pp. 7–14, July 2009.

[162] L. Song and L. Hu, "Improved Algebraic and Differential Fault Attacks on the KATAN Block Cipher," in *Information Security Practice and Experience* (R. H. Deng and T. Feng, eds.), Lecture Notes in Computer Science, pp. 372–386, Springer Berlin Heidelberg, 2013.

[163] R. P. d. Canto, R. Korkikian, and D. Naccache, "Buying AES Design Resistance with Speed and Energy," in *The New Codebreakers*, Lecture Notes in Computer Science, pp. 134–147, Springer, Berlin, Heidelberg, 2016.

[164] G. Gogniat, T. Wolf, W. Burleson, J. P. Diguet, L. Bossuet, and R. Vaslin, "Reconfigurable Hardware for High-Security/ High-Performance Embedded Systems : The SAFES Perspective," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 16, no. 2, pp. 144–155, 2008.

[165] O. Reparaz, "A note on the security of Higher-Order Threshold Implementations," *IACR Cryptology ePrint Archive*, 2015. `https://eprint.iacr.org/2015/001` (Accessed : 2019-09-24).

[166] A. Wild, A. Moradi, and T. Güneysu, "Evaluating the Duplication of Dual-Rail Precharge Logics on FPGAs," in *Constructive Side-Channel Analysis and Secure Design* (S. Mangard and A. Y. Poschmann, eds.), Lecture Notes in Computer Science, pp. 81–94, Springer International Publishing, 2015.

[167] ARM, "AMBA Specifications – Arm." `https://www.arm.com/products/silicon-ip-system/embedded-system-design/amba-specifications` (Accessed : 2019-07-26).

[168] Intel, "Cyclone® V FPGA - Intel® FPGA." `https://www.intel.com/content/www/fr/fr/products/programmable/fpga/cyclone-v.html` (Accessed : 2019-07-25).

École Nationale Supérieure des Mines
de Saint-Étienne

NNT: 2019LYSEM032

Maxime MONTOYA

ADAPTIVE AND ENERGY-EFFICIENT SECURITY IN THE INTERNET OF THINGS

Speciality: Microelectronics

Keywords: Integrated circuits security, Energy efficiency, Wake-up radio, Denial-of-Service, Hardware attacks and countermeasures

Abstract:

Integrated circuits for the Internet of Things have to address various demands, such as a low energy consumption and a high security level against several attacks. However, this high security usually introduces a high energy consumption. The goal of this work is to propose new methods that provide both a high security and a high energy efficiency for integrated circuits.

On the one side, we study the security of a mechanism dedicated to energy management. Wake-up radios trigger the wake-up of integrated circuits upon receipt of specific wake-up tokens, but they are vulnerable to denial-of-sleep attacks, during which an attacker replays such a token indefinitely to wake-up a circuit and deplete its battery. We propose a new method to generate unpredictable wake-up tokens at each wake-up, which efficiently prevents these attacks at the cost of a negligible energy overhead.

On the other side, we improve on the energy efficiency of hardware countermeasures against fault and side-channel attacks, with two different approaches. First, we present a new combined countermeasure, which increases by four times the power consumption compared to an unprotected implementation, introduces no performance overhead, and requires less than 8 bits of randomness. Therefore, it has a lower energy overhead than existing combined protections. It consists in an algorithm-level power balancing that inherently detects faults. Then, we propose an *adaptive* implementation of hardware countermeasures, which consists in applying or removing these countermeasures on demand, during the execution of the protected algorithm, in order to tune the security level and the energy consumption. A security evaluation of all the proposed countermeasures indicates that they provide an efficient protection against existing hardware attacks.

École Nationale Supérieure des Mines
de Saint-Étienne

NNT : 2019LYSEM032

Maxime MONTOYA

SÉCURITÉ ADAPTATIVE ET ÉNERGÉTIQUEMENT EFFICACE DANS L'INTERNET DES OBJETS

Spécialité : Microélectronique

Mots clefs : Sécurité des circuits intégrés, Efficacité énergétique, Radio de réveil, Déni de service, Attaques et contre-mesures matérielles

Résumé :

De nombreuses contraintes sont imposées sur les circuits intégrés pour l'Internet des Objets. Ceux-ci doivent à la fois avoir une consommation énergétique la plus faible possible, et être protégés contre divers types d'attaques. Or, la mise en œuvre d'une sécurité appropriée est souvent coûteuse en énergie. Cette thèse a donc pour but de proposer de nouvelles manières de concilier sécurité et efficacité énergétique pour les circuits intégrés.

Dans un premier temps, la sécurisation d'un mécanisme de gestion de l'énergie est étudiée. Les radios de réveil permettent de gérer la sortie de veille d'objets connectés, en réveillant un tel objet lors de la réception d'un code de réveil spécifique, mais elles sont vulnérables aux attaques par déni de sommeil, qui consistent à réveiller constamment l'objet en répétant un même code de réveil de sorte à vider sa batterie. Une nouvelle manière de générer des codes de réveils est proposée, qui permet de contrer efficacement ces attaques avec un coût négligeable en énergie.

Dans un second temps, l'efficacité énergétique des contre-mesures contre les attaques matérielles par injection de fautes et par analyse des canaux auxiliaires est améliorée à travers deux approches différentes. Une nouvelle contre-mesure mixte, ayant une consommation énergétique plus faible que les protections mixtes existantes, est proposée ; elle consiste en un lissage algorithmique de la consommation offrant une détection intrinsèque des fautes. L'implémentation *adaptative* de contre-mesures matérielles est également proposée ; elle consiste à moduler le niveau de protection fourni par ces contre-mesures au cours du fonctionnement d'un algorithme protégé, afin d'optimiser la sécurité et la consommation énergétique. Une évaluation de la sécurité des contre-mesures montre qu'elles fournissent une protection efficace contre les attaques matérielles existantes.