UN FORMALISME UNIFIANT LES ATTAQUES PHYSIQUES SUR CIRCUITS CRYTOGRAPHIQUES ET SON EXPLOITATION AFIN DE COMPARER ET RECHERCHER DE NOUVELLES ATTAQUES

Hélène Le Bouder

NNT : 2014 EMSE 0759

THÈSE

Hélène LE BOUDER

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

UN FORMALISME UNIFIANT LES ATTAQUES PHYSIQUES SUR CIRCUITS CRYTOGRAPHIQUES ET SON EXPLOITATION AFIN DE COMPARER ET RECHERCHER DE NOUVELLES ATTAQUES.

soutenue à Gardanne, le 24 octobre 2014

Membres du jury

Président :	Christophe CLAVIER	Professeur, XLIM, Limoges
Rapporteurs :	Emmanuel PROUFF	Docteur HDR, ANSSI, Paris
	Louis GOUBIN	Professeur, UVSQ, Versailles
Examinateurs :	Jean-Jacques QUISQUATER	Professeur, UCL, Louvain-La-Neuve
	Sylvain GUILLEY	Professeur, TELECOM Paris-Tech, Paris
	Benoît GERARD	Docteur, DGA, Rennes
Directrice :	Assia TRIA	Docteur HDR, CEA, Gardanne
Encadrant :	Yanis LINGE	Docteur, STMicroelectronics, Rousset

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	IMAGE, VISION, SIGNAL	JC. Pinoli, Professeur
SCIENCES DE LA TERRE	B. Guy, Directeur de recherche	GENIE INDUSTRIEL	A. Dolgui, Professeur
SCIENCES ET GENIE DE L'ENVIRONNEMENT	D. Graillot, Directeur de recherche	MICROELECTRONIQUE	S. Dauzere Peres, Professeur

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

ABSI	Nabil	CR		CMP
AVRIL	Stéphane	PR2	Mécanique et ingénierie	CIS
BALBO	Flavien	PR2		FAYOL
BASSEREAU	Jean-François	PR		SMS
BATTON-HUBERT	Mireille	PR2	Sciences et génie de l'environnement	FAYOL
BERGER DOUCE	Sandrine	PR2		FAYOL
BERNACHE-ASSOLLANT	Didier	PR0	Génie des Procédés	CIS
BIGOT	Jean Pierre	MR(DR2)	Génie des Procédés	SPIN
BILAL	Essaid	DR	Sciences de la Terre	SPIN
BOISSIER	Olivier	PR1	Informatique	FAYOL
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
BURLAT	Patrick	PR2	Génie Industriel	FAYOL
COURNIL	Michel	PR0	Génie des Procédés	DIR
DARRIEULAT	Michel	IGM	Sciences et génie des matériaux	SMS
DAUZERE-PERES	Stéphane	PR1	Génie Industriel	CMP
DEBAYLE	Johan	CR	Image Vision Signal	CIS
DELAFOSSE	David	PR1	Sciences et génie des matériaux	SMS
DESRAYAUD	Christophe	PR2	Mécanique et ingénierie	SMS
DOLGUI	Alexandre	PR0	Génie Industriel	FAYOL
DRAPIER	Sylvain	PR1	Mécanique et ingénierie	SMS
FEILLET	Dominique	PR2	Génie Industriel	CMP
FEVOTTE	Gilles	PR1	Génie des Procédés	SPIN
FRACZKIEWICZ	Anna	DR	Sciences et génie des matériaux	SMS
GARCIA	Daniel	MR(DR2)	Génie des Procédés	SPIN
GERINGER	Jean	MA(MDC)	Sciences et génie des matériaux	CIS
GOEURIOT	Dominique	DR	Sciences et génie des matériaux	SMS
GRAILLOT	Didier	DR	Sciences et génie de l'environnement	SPIN
GROSSEAU	Philippe	DR	Génie des Procédés	SPIN
GRUY	Frédéric	PR1	Génie des Procédés	SPIN
GUY	Bernard	DR	Sciences de la Terre	SPIN
HAN	Woo-Suck	CR	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
LI	Jean-Michel		Microélectronique	CMP
MALLIARAS	Georges	PR1	Microélectronique	CMP
MOLIMARD	Jérôme	PR2	Mécanique et ingénierie	CIS
MONTHEILLET	Frank	DR	Sciences et génie des matériaux	SMS
MOUTTE	Jacques	CR	Génie des Procédés	SPIN
NEUBERT	Gilles			FAYOL
NIKOLOVSKI	Jean-Pierre			CMP
NORTIER	Patrice	PR1		SPIN
PIJOLAT	Christophe	PR0	Génie des Procédés	SPIN
PIJOLAT	Michèle	PR1	Génie des Procédés	SPIN
PINOLI	Jean Charles	PR0	Image Vision Signal	CIS
POURCHEZ	Jérémy	CR	Génie des Procédés	CIS
ROBISSON	Bruno			CMP
ROUSSY	Agnès	MA(MDC)		CMP
ROUSTANT	Olivier	MA(MDC)		FAYOL
ROUX	Christian	PR		CIS
STOLARZ	Jacques	CR	Sciences et génie des matériaux	SMS
TRIA	Assia	Ingénieur de recherche	Microélectronique	CMP
VALDIVIESO	François	MA(MDC)	Sciences et génie des matériaux	SMS
VIRICELLE	Jean Paul	MR(DR2)	Génie des Procédés	SPIN
WOLSKI	Krzystof	DR	Sciences et génie des matériaux	SMS
XIE	Xiaolan	PR1	Génie industriel	CIS
YUGMA	Gallian	CR	Génie industriel	CMP

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

BERGHEAU	Jean-Michel	PU	Mécanique et Ingénierie	ENISE
BERTRAND	Philippe	MCF	Génie des procédés	ENISE
DUBUJET	Philippe	PU	Mécanique et Ingénierie	ENISE
FEULVARCH	Eric	MCF	Mécanique et Ingénierie	ENISE
FORTUNIER	Roland	PR	Sciences et Génie des matériaux	ENISE
GUSSAROV	Andrey	Enseignant contractuel	Génie des procédés	ENISE
HAMDI	Hédi	MCF	Mécanique et Ingénierie	ENISE
LYONNET	Patrick	PU	Mécanique et Ingénierie	ENISE
RECH	Joël	PU	Mécanique et Ingénierie	ENISE
SMUROV	Igor	PU	Mécanique et Ingénierie	ENISE
TOSCANO	Rosario	PU	Mécanique et Ingénierie	ENISE
ZAHOUANI	Hassan	PU	Mécanique et Ingénierie	ENISE

Mise à jour : 28/03/2014

à Guillaume, Ronan et Olivier.

Il me semble plus aisé de poursuivre une thèse plutôt qu'un Minotaure.
Tout le monde ne peut pas être Thésée.
Simon Berjeaut.

Avant propos

Ma thèse s'est déroulée à l'École Nationale Supérieure des Mines de Saint-Étienne, dans le Centre Microélectronique de Provence sur le site Georges Charpak à Gardanne, au sein de l'équipe SAS (Systèmes et Architectures Sécurisées). Il s'agit d'une équipe de recherche commune entre le CEA-Tech et l'École des Mines de Saint-Étienne.

Les Thèmes de recherche y sont les suivants :

1. Sécuriser (contre des attaques physiques) les composants matériels de sécurité.
2. Assurer l'intégration optimale de ces circuits sur des systèmes électroniques.
3. Doter ces circuits de fonctions cryptographiques avancées.
4. Maîtriser les attaques actuelles.
5. Anticiper les attaques futures.

C'est dans les points 4 et 5 que s'inscrit le cadre ma thèse.

Cette thèse a été financée par le projet *Calisson-2*. La direction a été confiée à Assia Tria du CEA-Tech ; l'encadrement technique à Bruno Robisson du CEA-Tech puis, par Yanis Linge de ST-Microélectronics.

Diplômée du master CRYPTIS parcours mathématiques codages et application de l'Université de Limoges, cette thèse m'a permis de me confronter à un autre domaine scientifique qui touche à la sécurité des systèmes embarqués : la microélectronique. Ainsi, le contenu de ce manuscrit est à mi-chemin entre deux disciplines : les mathématiques et la microélectronique, les deux étant appliquées à la cryptographie des circuits électroniques.

Des manipulations expérimentales ont en partie été réalisées avec la plateforme *Micro-Packs*. Ces études ont été possibles grâce à Amine Dehbaoui, Driss Aboulkassimi et Loïc Zussa qui m'ont formée.

Remerciements

Pour commencer, je remercie le personnel de l'équipe SAS de m'avoir accueillie au sein de son laboratoire de recherche, à commencer par ma directrice de thèse Assia Tria et mon encadrant Yanis Linge. Dans cette thèse, ils ont su m'aider, m'orienter et me conseiller pour m'amener jusqu'à la soutenance.

Mes remerciements vont ensuite à Emmanuel Prouff et Louis Goubin mes rapporteurs pour m'avoir autorisé à soutenir et pour le temps qu'ils ont consacré à mon manuscrit. Je souhaite ensuite remercier tous les membres du jury, directrice, encadrant, rapporteurs et examinateurs à savoir, Jean Jacques Quisquater, Benoît Gérard, Christophe Clavier et Sylvain Guilley de m'avoir décerné le titre de docteur.

Un grand merci également à tous ceux qui ont travaillé avec moi sur différents sujets, en particulier, Christophe Clavier et Sylvain Guilley, membres du jury, mais également Antoine Wurcker, Gaël Thomas et Ronan Lashermes.

La thèse fut une épreuve difficile, ainsi je souhaite remercier chaleureusement Ronan Lashermes, Guillaume Reymond et Olivier Vallier de l'équipe SAS, pour leur soutien technique et psychologique qu'ils m'ont apporté au quotidien, durant ces trois ans. Toujours dans l'équipe SAS, je remercie Amine Dehbaoui, Driss Aboulkassimi et Loïc Zussa pour avoir pris le temps de me former aux différentes manipulations expérimentales ; ainsi que Jacques Fournier pour tous ses excellents conseils. Je souhaite également saluer les collègues doctorants et post-doctorants, en particulier Thomas Sarno, Marc Lacruche et Sébastien Tiran.

À titre plus personnel, je souhaite remercier tous mes proches, amis et famille, qui ont cru en moi. Je remercie les grimpeurs du GUMS d'Aix en Provence pour les sorties escalades qui m'ont permis d'aider à évacuer le stress accumulé, chaque semaine. Pour finir je remercie Padmée mon chat pour sa ronron thérapie quotidienne et Kim mon cochon d'inde, merci à elles deux pour le soutien psychologique qu'elles m'ont apporté à travers leur affection.

Table des matières

Partie I

Introduction

Sommaire

1 La cryptographie

Définition 1 : La **cryptologie** étymologiquement la science du secret, regroupe la cryptographie et la cryptanalyse.

1.1 Définition de la cryptographie

Définition 2 : Le sujet traditionnel de la cryptographie est le **chiffrement** (c'est également le sujet abordé dans cette thèse). Il s'agit de l'ensemble des méthodes permettant de rendre un message appelé **texte clair**, illisible à toute autre personne que le destinataire. Il ne s'agit pas de dissimuler le message ce qui est de la stéganographie ; mais de le chiffrer à l'aide d'une **clé**, en quelque chose d'incompréhensible : le **texte chiffré**. La méthode utilisée est appelée **algorithme de chiffrement**.

Il est important de différencier cryptographie et codage. Dans un codage il y a une notion d'apprentissage, écrire un texte en français est un codage, l'écrire en anglais en est un autre. Il faut donc connaître la langue et ses règles de grammaire, conjugaison et orthographe. En cryptographie il n'y a pas besoin d'apprentissage, le concept important est la notion de clé.

Un principe fondamental de la cryptographie est le principe de Kerckoffs [1]. Il énonce que la sécurité d'un algorithme cryptographique ne doit pas reposer sur la connaissance de celui-ci. Il doit pouvoir « *tomber sans inconvénient aux mains de l'ennemi* ». Autrement dit la sécurité doit reposer uniquement sur la protection de la clé, ou plus généralement d'un secret.

La cryptographie âgée d'environ 4000 ans, a très longtemps été considérée comme une arme de guerre. C'est la prolifération des systèmes de communication actuels qui a permis à la cryptographie de sortir du domaine militaire.

La **cryptographie moderne** a de nombreux usages :
- **confidentialité** : assurer que l'information n'est accessible qu'aux personnes autorisées,
- **intégrité** : assurer que l'information ne peut pas être altérée,
- **authentification** : valider l'origine d'une entité,
- **non répudiation** : permettre à une personne de prendre part à un contrat avec impossibilité de le renier ultérieurement.

1.2 Chiffrement symétrique et asymétrique

La cryptographie se scinde en deux parties : la cryptographie symétrique et asymétrique.

Définition 3 : Les **chiffrements symétriques** ou à clé secrète, reposent sur une même clé qui sert au chiffrement et au déchiffrement.

Le principal problème de la cryptographie symétrique est l'échange de la clé. Comment deux personnes souhaitant communiquer de manière sécurisée peuvent le faire si elles n'ont pas de moyen fiable de s'échanger la clé ? C'est pourquoi en 1976 dans [2], pour répondre à ce grand problème, les premiers concepts de cryptographie asymétrique sont inventés.

Définition 4 : Les **chiffrements asymétriques** ou à clé publique, utilisent deux clés distinctes liées entre-elles, l'une pour chiffrer et l'autre pour déchiffrer.

Ainsi dans la cryptographie à clé publique, tout le monde peut communiquer avec tout le monde de manière sécurisée. La clé publique servant à chiffrer est dévoilée aux yeux de tous, par exemple dans un annuaire. La clé privée reste aux mains du destinataire qui ne doit pas la dévoiler à qui que ce soit. Il n'y a pas de partage de clé.

Définition 5 : Le **chiffrement par blocs** est une des deux grandes catégories de chiffrement en cryptographie symétrique, l'autre étant le chiffrement à flot. Le principe est de découper le texte clair en **blocs** de taille fixe. Les blocs sont ensuite chiffrés les uns après les autres. Pour cela une **fonction de tour** est appliquée itérativement.

Définition 6 : Un **système embarqué** est un système électronique et informatique autonome, spécialisé dans une tâche bien précise. Il est généralement composé de circuits dédiés et optimisés pour une application.

Les ressources d'un système embarqué sont généralement limitées. Les chiffrements par blocs y sont très utilisés. L'exemple type de système embarqué est la carte à puce. Elle a été inventée par Roland Moreno en 1974. Aujourd'hui, l'utilisation de la carte à puce est très répandue, notamment dans le domaine bancaire. Ainsi la présence de la cryptographie dans les systèmes embarqués est nécessaire voire indispensable.

Dans le cadre de cette thèse, nous nous sommes focalisés sur des algorithmes de chiffrement à clé privée, plus précisément aux algorithmes de chiffrement par blocs pouvant être embarqués.

1.3 Propriétés cryptographiques

La confusion et la diffusion sont deux propriétés utilisées dans un algorithme de chiffrement. Elles ont été introduites par Shannon dans [3].

Définition 7 : La **confusion** correspond à une volonté de rendre la relation entre le texte clair et le texte chiffré la plus complexe possible.

Définition 8 : La **diffusion** est l'idée qu'un biais sur le texte clair ou sur la clé ne doit pas se retrouver sur le texte chiffré. En d'autres termes, les études statistiques sur les textes chiffrés doivent donner le moins possible d'informations sur le texte clair et la clé.

Pour un algorithme de chiffrement avec une bonne diffusion, l'inversion d'un seul bit en entrée doit changer chaque bit en sortie avec une probabilité de $\frac{1}{2}$.

Dans les algorithmes la confusion est introduite par des boites S (substitution), appelées aussi s-boxes. Celles-ci introduisent également de la diffusion. Quant aux permutations et transpositions, elles augmentent la diffusion.

1.4 Algorithmes standards

Dans tout le manuscrit, les notations suivantes sont utilisées : T désigne un texte clair appelé aussi simplement clair, C un texte chiffré appelé aussi simplement chiffré et K une clé de chiffrement. Le numéro de tour r est indexé en bas d'une variable (par exemple K_r est la clé de tour r). Les tours d'un algorithme de chiffrement par blocs sont numérotés de 0 à $\mathbf{r}$. La fonction « ou exclusif », souvent appelée « xor » est notée $\oplus$. L'ensemble des notations utilisées dans cette thèse est reporté en annexe F.

1.4.1 Data Encryption Standard, DES

Le Data Encryption Standard (DES [4]) a été approuvé par le NIST (National Institute of Standards and Technology) en 1977. Il s'agit d'un algorithme de chiffrement par blocs, suivant un schéma de Feistel de 16 tours. Il est représenté en Figure 1.

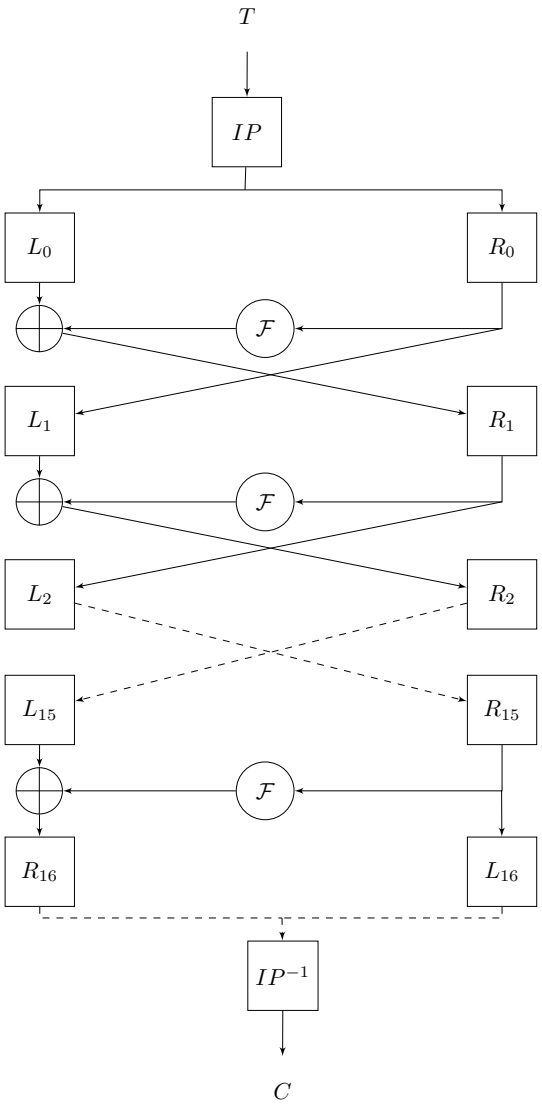


Figure 1 : Schéma général du Data Encryption Standard.

Le DES utilise des clés de 64 bits ; 56 bits sont réellement utilisés pour le chiffrement, les bits restant sont des bits de parité. La notion de schéma de Feistel est définie en détails dans la partie IV. Le DES commence par une permutation IP de 64 bits (Tableau 24 en annexe A i), et se termine par l'inverse de cette même permutation IP^{-1}.

a) Description de la fonction de tour

La fonction de tour illustrée en Figure 2, est composée d'une fonction dite de Feistel $\mathcal{F}$ et d'un « xor »avec le bloc de gauche. La fonction de Feistel est sur 32 bits et se décompose en 4 étapes (dans l'ordre, expansion, xor de clé, s-box et permutation) :

- L'**expansion** E (voir Tableau 25 en annexe A.i) transforme 32 bits en 48 bits. Pour cela, certains bits sont doublés.

- Le « **xor** » avec les 48 bits de la clé de tour K_r, $r \in [\![0, 15]\!]$.

- Les **S-boxes** S_i, $i \in [\![1, 8]\!]$ sont des fonctions Booléennes de 6 bits vers 4. Elles sont représentées sous forme de tableaux de 4 lignes et 16 colonnes comme dans le Tableau 1 qui représente la 1ère s-box du DES, les autres s-boxes sont en annexe A.i dans les Tableaux 27 à 33.

14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	14	3	10	0	6	13

Tableau 1 : S-box 1 du DES

Soit x une entrée, le bit de poids fort et le bit de poids faible déterminent la ligne et les quatre autres bits (ceux du milieu) la colonne de la case de la s-box, dans laquelle se trouve la valeur de sortie qui lui est associée. Prenons un exemple pour la première s-box, $S_1(x) = y$. Si $x = (25)_{10} = (011001)_2$, le numéro de la ligne est $(01)_2 = (1)_{10}$ et le numéro de la colonne est $(1100)_2 = (12)_{10}$, $\Rightarrow y = (9)_{10}$.

- La **permutation** P, de 32 bits (voir Tableau 26 en annexe A.i).

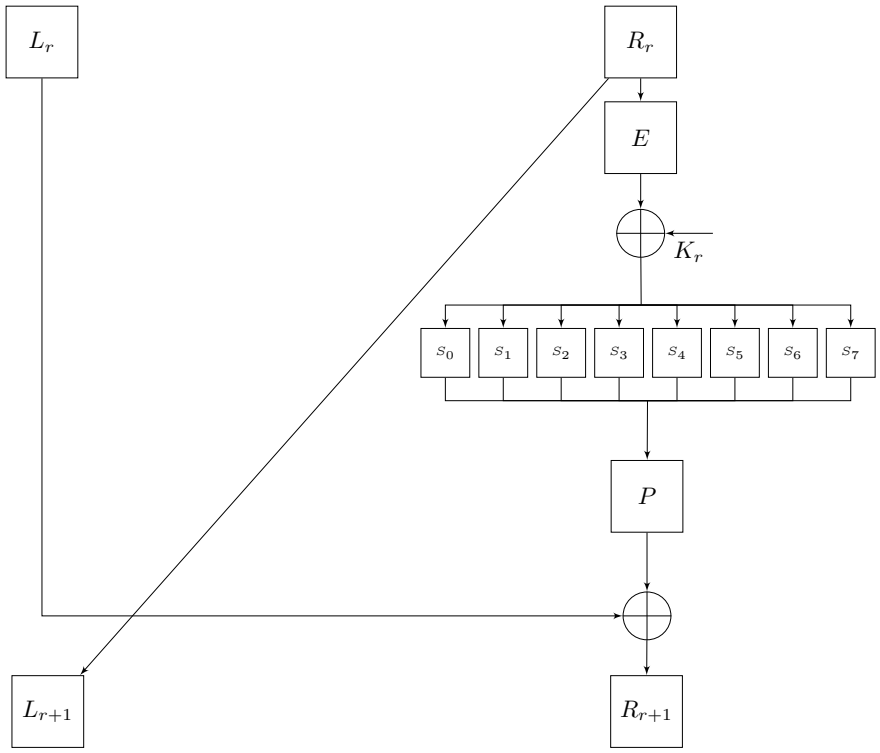


Figure 2 : Fonction de tour du DES.

b) Les propriétés des s-boxes

Les s-boxes du DES respectent un certain nombre de propriétés afin de résister à la cryptanalyse classique (comme par exemple les attaques différentielles) et satisfaire au maximum leur tâche de confusion.

Brickell dans [5] a justifié quelles sont les propriétés respectées par les s-boxes du DES.

Propriété P1 : Chaque ligne d'une s-box est une permutation des entiers de 0 à 15.

Propriété P2 : Les s-boxes ne sont ni des fonctions linéaires ni des fonctions affines.

Propriété P3 : Changer un bit à l'entrée d'une s-box implique en changer au moins deux en sortie.

Propriété P4 : $S_i(x)$ et $S_i(x \oplus (001100)_2)$ doivent différer d'au moins deux bits.

Propriété P5 : $S_i(x) \neq S_i(x \oplus (00ab00)_2)$ pour tout a et b.

Propriété P6 : Les s-boxes doivent être choisies de façon à minimiser la différence entre le nombre de bits à 1 et le nombre de bits à 0, pour toutes les sorties, quand un bit est fixé en entrée.

Ces propriétés sont vérifiées pour de nombreux algorithmes de chiffrement autres que le DES. Des travaux plus récents sur les propriétés des s-boxes d'un algorithme de chiffrement ont été présentés par Carlet dans [6].

1.4.2 Advanced Encryption Standard, AES

L'AES, Advanced Encryption Standard [7] est un algorithme de chiffrement symétrique, approuvé par le NIST en 2001. Il existe sous trois formes 10, 12 ou 14 tours avec des clés de respectivement 128, 192 et 256 bits. Il est basé sur un réseau de substitution-permutation et non sur un schéma de Feistel comme le DES. Il traite des blocs de 16 octets vus comme une matrice 4×4 appelée **state**. La Figure 3 illustre l'algorithme.

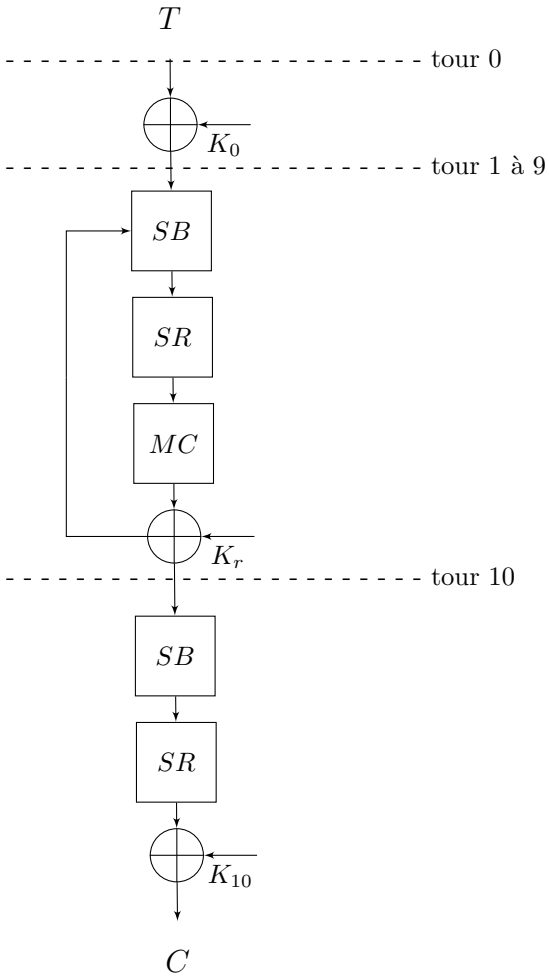


Figure 3 : Schéma de l'AES de 10 tours

La fonction de tour de l'AES se décompose en 4 sous-fonctions.

- Le **SubBytes** est composé de 16 s-boxes identiques. Cette fonction s-box notée SB, est une fonction booléenne de 8 bits vers 8 qui s'applique à tous les octets du state. Il s'agit de la fonction qui transforme les éléments de $\mathbb{F}_{256}$ en leurs inverses (0 reste 0), suivie d'une fonction affine. Elle est décrite dans le Tableau 34 en annexe A.ii.

- Le **ShiftRow** noté SR, déplace cycliquement vers la gauche les trois dernières lignes du state de respectivement un, deux et trois éléments.

- Le **MixColumn** noté MC, applique une transformation linéaire dans $GF(2^8)$ aux colonnes du state. Plus précisément c'est le produit d'un vecteur colonne du state et d'une matrice définie dans la Figure 42 en annexe A.ii.

- Le **AddRoundKey** effectue le "xor" avec la clé de tour.

Les fonctions SR et MC servent à la diffusion ; un bit changé sur un octet affecte l'ensemble du state, après deux tours. Il est important de préciser que le tour 0 de l'AES ne fait qu'un xor avec la première clé, et le dernier tour ne contient pas de MC.

Une clé de tour d'AES est représentée comme le state sous la forme d'une matrice 4×4. Ainsi l'emplacement de chacun des octets est noté par un couple (l, c), représentant respectivement la ligne et la colonne dans la matrice.

Pour un AES_{128}, la clé maître K est la clé du tour 0, K_0. La clé de tour K_{r+1} est obtenue à partir de la clé du tour précédent K_r, en utilisant la fonction KeyExpansion décrite dans l'équation (1) suivante :

$$\begin{cases} K_{r+1}^{l,0} = SB\left(K_r^{(l+1) \bmod 4,3}\right) \oplus rcon(l,r) \oplus K_r^{l,0} & \forall l \in [\![0,3]\!] \\ K_{r+1}^{l,c} = K_r^{l,c} \oplus K_{r+1}^{l,c-1} & \forall l \in [\![0,3]\!] \text{ and } c \in [\![1,3]\!] \end{cases} \tag{1}$$

avec SB la fonction s-box et $rcon$ désigne une matrice constante de taille 4×10.

Pour un AES_{192} la clé maître est de taille 6 colonnes (clé du tour 0 et les deux premières colonnes du tour 1). La transformation est la même que pour AES_{128} appliquée toutes les 6 colonnes. Pour un AES_{256} la clé maître est de taille 8 colonnes (clés des tours 0 et 1). Tous les states des clés de tours suivants sont calculés alternativement par la même transformation et une autre transformation simplifiée, juste une SB.

2 La cryptanalyse

La cryptanalyse dans au sens générale du terme a pour but d'invalider des propriétés cryptographiques. C'est pourquoi, nous précisons que cette partie est orientée pour la cryptanalyse des algorithmes de chiffrement.

2.1 Définition de la cryptanalyse

Définition 9 : La **cryptanalyse** consiste à tenter de retrouver un message ayant été chiffré sans posséder la clé de chiffrement, on parle aussi de décryptage. Il peut aussi s'agir d'être capable de retrouver tous les messages en retrouvant la clé de chiffrement.

Définition 10 : Un **attaquant** est une personne qui souhaite obtenir des informations protégées par de la cryptographie qui ne lui sont pas destinées. Pour cela, il va devoir faire de la cryptanalyse.

De manière générale, ce que recherche l'attaquant est appelée la **cible**, il peut s'agir de la clé de chiffrement, des textes clairs.

La Figure 4 présente une vision schématique du positionnement de la cryptanalyse par rapport à la cryptographie.

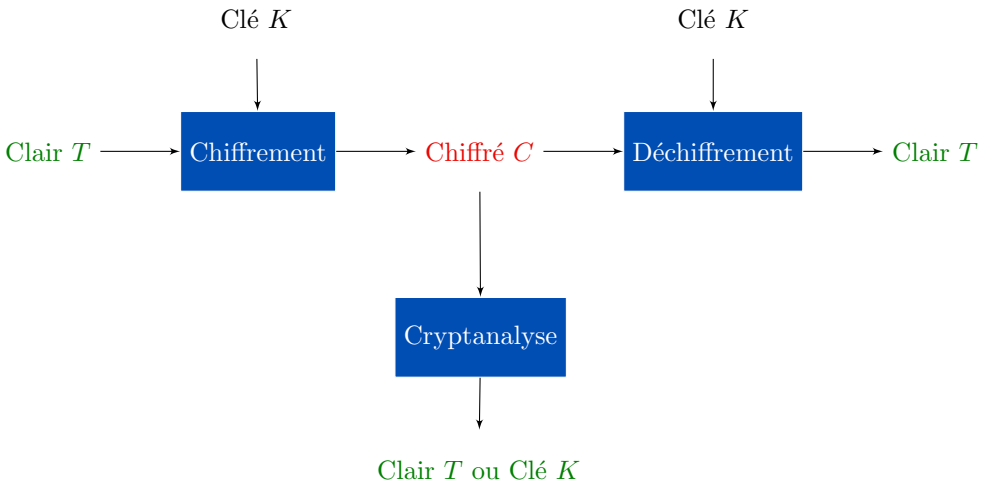


Figure 4 : Schéma de la cryptographie et de la cryptanalyse

Le terme d'**attaque** est également employé pour évoquer la cryptanalyse. Un attaquant peut avoir accès à différents niveaux de connaissance.

- Les **attaques à texte(s) chiffré(s) seulement :**
  l'attaquant n'a connaissance que d'un ou plusieurs textes chiffrés.

- Les **attaques à clairs connus :**
  l'attaquant connaît plusieurs couples : (textes clairs, textes chiffrés).

- Les **attaques à clairs choisis :**
  l'attaquant a accès aux couples : (textes clairs qu'il choisit, textes chiffrés).

– Les **attaques à chiffrés choisis :**
l'attaquant peut déchiffrer des chiffrés de son choix, mais il ne connaît pas la clé.

2.2 La cryptanalyse classique ou théorique

La cryptanalyse classique se scinde en deux grandes familles, la cryptanalyse linéaire et la cryptanalyse différentielle.

Définition 11 : La **cryptanalyse linéaire** consiste à trouver une relation linéaire entre certains bits du texte clair et certains bits du texte chiffré.

Les algorithmes modernes de chiffrement sont résistants à la cryptanalyse linéaire.

Définition 12 : La **cryptanalyse différentielle** est une attaque à clair choisi, introduite dans [8] par Biham et Shamir. Elle consiste à considérer des couples de clairs présentant une différence fixée et à étudier comment peut se propager cette différence au fur et à mesure du chiffrement.

La cryptanalyse différentielle, plus récente que la cryptanalyse linéaire, a été très employée afin d'attaquer les algorithmes de chiffrement par blocs. Ainsi, les algorithmes récents sont conçus pour résister (au mieux) à ce type d'attaques.
En 2007, un nouveau genre de cryptanalyse appelé attaques algébriques, a été introduit. Une approche des attaques algébriques sur AES est présentée en [9].

Définition 13 : La **cryptanalyse algébrique** réécrit le chiffrement sous forme d'un système d'équations polynomiales. Les variables représentant les bits de clé, l'attaque consiste à résoudre ce système.

2.3 Les attaques physiques

2.3.1 Attaques physiques matérielles

Un algorithme cryptographique est conçu pour être robuste mathématiquement, résistant à la cryptanalyse classique. Cependant, une fois implémenté dans un circuit, un attaquant peut utiliser les failles de ce dernier. Il s'agit alors d'attaques physiques.

Définition 14 : Les **attaques physiques** sont l'ensemble des techniques de cryptanalyse exploitant le fonctionnement du circuit. Elles s'opposent à la cryptanalyse au sens classique car elles ne permettent pas d'attaquer l'algorithme en soit, mais son implémentation logicielle ou matérielle.

Dans le domaine des attaques physiques, il est important de différencier les données (exemple : texte clair T) des mesures (exemple : mesures de la consommation de courant).

Définition 15 : Une **donnée** est une variable discrète, traitée par un algorithme. Une **mesure** est une approximation d'une grandeur physique (continue) par un outil.

Dans cette thèse, nous nous sommes focalisés sur les attaques physiques matérielles et non logicielles. Dans cette catégorie des attaques physiques matérielles, trois classes d'attaques se distinguent entre elles.

– Les **attaques invasives** sont généralement les attaques les plus coûteuses, tant en termes de temps que d'équipements spécifiques. Elles sont généralement menées par des experts et se déroulent en deux étapes, la préparation du circuit, puis l'attaque proprement dite. Ces attaques pénètrent physiquement le circuit afin d'opérer à des observations directes ou des modifications. Elles sont extrêmement performantes, mais présentent cependant deux inconvénients majeurs. Elles altèrent parfois la fonctionnalité du circuit et elles ont un fort taux d'échec.

– Les **attaques semi-invasives** nécessitent généralement d'avoir accès à la surface du circuit, donc d'effectuer une décapsulation totale ou partielle. Cependant aucune connexion directe n'est nécessaire. Elles sont moins coûteuses que les attaques invasives, nécessitent moins de matériel, et permettent d'obtenir des résultats rapidement.

– Les **attaques non-invasives** consistent en l'analyse du comportement du circuit sans affecter son intégrité.

Il existe deux grandes familles d'attaques physiques matérielles :
– attaques par observation ou par canaux auxiliaires [1],
– attaques par injections de fautes.

2.3.2 Attaques par observation

Définition 16 : Les attaques **par observation** ou **par canaux auxiliaires** notées **SCA** (Side Channel Analysis) se basent sur l'observation du circuit pendant l'exécution des calculs liés au chiffrement.

Si un algorithme est théoriquement « sûr », le circuit dans lequel il est implémenté ne l'est pas forcément. C'est la frontière entre la théorie et la pratique. L'algorithme de chiffrement est théorique, le circuit lui est bien concret. Ainsi ce dernier a une consommation de courant, un temps de calcul etc. Ce sont ces paramètres physiques qui peuvent nuire à la sécurité de notre algorithme de chiffrement. Ces paramètres ne sont pas faciles à anticiper d'un point de vue théorique, on parle de **fuite d'information** par un canal auxiliaire. De plus, seules des mesures de ces paramètres physiques peuvent être obtenues, ces mesures dépendent donc à la fois de la fuite mais aussi de l'appareil de mesure.

Ce type d'attaque est théoriquement non-invasive ; en pratique, l'attaquant peut vouloir acquérir des mesures à un emplacement précis du circuit et finalement la réalisation implique une attaque invasive ou semi-invasive.

2.3.3 Attaques par injection de fautes

Définition 17 : Les attaques **par injection de fautes** notées **FIA** (Fault Injection Analysis) exploitent l'effet d'une perturbation intentionnelle sur le fonctionnement du circuit.

Les attaques par injection de fautes se basent sur l'idée de venir perturber le circuit. Pour une attaque, une ou plusieurs injection(s) peu(ven)t avoir lieu. Chaque injection conduit à une perturbation appelée faute ou erreur notée e. Deux cas se distinguent alors, le comportement normal du

1. Dans la littérature le terme attaques par canaux auxiliaires est parfois utilisé pour l'ensemble des attaques physiques. Le terme d'attaque par canaux cachés est également utilisé pour les attaques par observation, cependant il est utilisé pour un canal non répertorié (voir [10]).

circuit et le comportement perturbé du circuit.

Les attaques FIA sont généralement puissantes. Cependant, ce sont généralement des attaques invasives ou semi-invasives, elles comportent un risque non négligeable qui est **l'endommagement** voir **la destruction** du circuit. En effet, les outils d'injection tels que le laser (Figure 5) peuvent détruire une partie du circuit. C'est pourquoi, lors d'une attaque par injection de fautes, l'attaquant cherche généralement à minimiser le nombre de fautes à injecter, afin de limiter les risques de destruction même partielle du circuit.

Les moyens pour injecter une faute dans un circuit sont divers et variés : laser, modification de la tension, modification de l'horloge interne du circuit, etc.

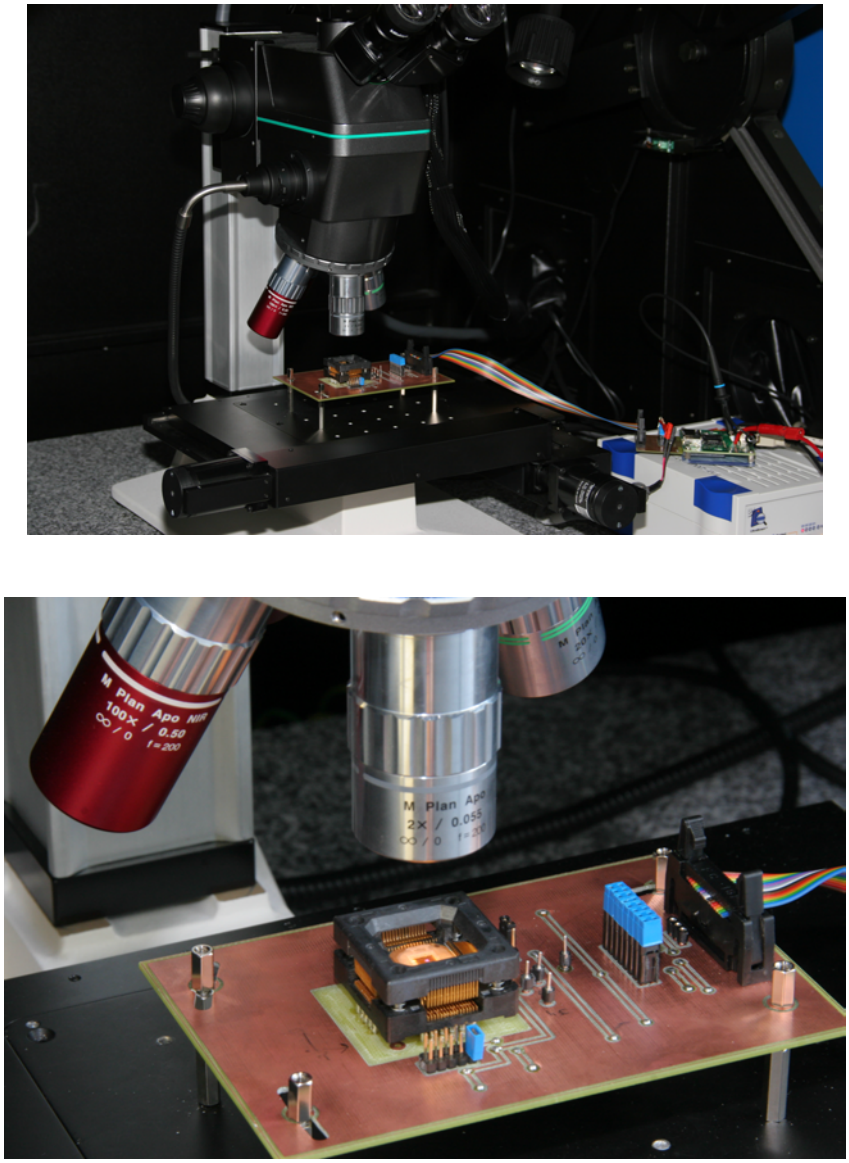


Figure 5 : Le laser, un outil semi-invasif pour injecter des fautes et réaliser des attaques. Le laser présenté ici, provient de la plateforme *Micro-Packs.*

Définition 18 : Les attaques de type DFA (Differential Fault Analysis) sont une technique de cryptanalyse qui exploitent des résultats erronés obtenus par injection de fautes. L'analyse de ces résultats se fait au niveau des données.

Des exemples de DFA sont en [11, 12, 13, 14, 15, 16, 17, 18]. Lorsqu'une donnée manipulée par l'algorithme devient erronée elle est dite **fautée** et est notée d'une étoile en exposant *. La différentielle entre une valeur a et une valeur a^* est notés Δ_a.

2.3.4 Attaques hybrides

Les principes des attaques par observation et par injection de fautes, semblent très éloignés. Cependant, des attaques récentes combinent de plus en plus canaux auxiliaires et injection de fautes, elles sont dites **hybrides** (exemples : la FSA (Fault Sensitive Analysis [19]) et la DBA (Differential Behaviour Analysis [20])). Il y a bien des injections de fautes, cependant les résultats fautés ne sont pas utilisés comme dans une DFA. Ce qui intéresse l'attaquant c'est sous quelles conditions la faute apparaît. Il observe le comportement du circuit en réponse au processus d'injection de fautes. La réaction à la tentative d'injection de fautes est perçue comme une fuite via un canal auxiliaire.

2.3.5 Les cibles des attaques physiques matérielles

En cryptanalyse classique la cible est le texte clair ou la clé de chiffrement. Cependant les attaques physiques ont d'autres utilisations, par exemple : faire de la rétro-conception, détecter des chevaux de Troie etc. Dans cette thèse, nous nous intéressons à deux objectifs des attaques physiques : la recherche de clé et la rétro-conception.

Définition 19 : On appelle **rétro-conception** d'un algorithme cryptographique, la recherche d'une partie ou de la totalité de celui-ci.

Bien que le principe de Kerckhoff [1] insiste sur le fait que la sécurité d'un algorithme ne doit pas reposer sur la connaissance de celui-ci , il existe un certain nombre d'algorithmes privés. Par exemple l'algorithme de chiffrement par blocs Cryptomeria (C2) [21] est un algorithme propriétaire pour la protection de DVDs. Un autre exemple est la suite d'algorithmes A3/A5/A8 utilisés dans les applications GSM. Dans [22], Clavier *et al.* les restituent en utilisant des attaques physiques.

Cependant créer un nouvel algorithme de chiffrement n'est pas chose aisée. C'est pourquoi, de nombreux algorithmes privés s'inspirent d'algorithmes déjà existants. Parfois il s'agit juste de modifier les tables de permutation et de substitution. L'attaque présentée par Wurcker et *al.* [23] est une parfaite illustration des attaques pour la rétro-conception ; dans ce papier, leur but est de retrouver un AES où toutes les tables (SB, MC et SR) sont inconnues. Un autre exemple est l'attaque de Rivain et *al.* [24].

Les attaques physiques pour la rétro-conception utilisant les canaux auxiliaires sont notées SCARE (Side Channel Attack Reverse Engineering) et FIRE (Fault Injection Reverse Engineering) celles utilisant des injections de fautes.

3 Les Circuits électroniques

3.1 Les différents niveaux d'abstraction en informatique.

Définition 20 : Un **système informatique** est un ensemble d'éléments de type logiciel (software) et matériel (hardware), mis ensemble pour collaborer dans l'exécution d'une application.

Entre l'algorithme conçu de manière théorique et sa mise en pratique par un circuit (appelée aussi implémentation logicielle ou matérielle) il y a différents niveaux d'abstraction. À chaque niveau d'abstraction, de nouvelles failles sont possibles, c'est pour cela que les algorithmes de chiffrement théoriques ne peuvent pas anticiper toutes les attaques (présentes et futures) à la fois. Nous donnons la liste exhaustive des différents niveaux.

1. **Algorithme :** la théorie.

2. **Langage de programmation** (C, java, matlab *etc.*.)
   Une fois conçu, l'algorithme est décrit par un informaticien en langage de programmation. Le programme est alors compilé afin d'être traduit en langage machine en un ensemble d'instructions différentes.

3. **Architecture de la machine et micro-architecture**
   Le **processeur** ou CPU (Central Processing Unit) est le composant de l'ordinateur qui exécute les instructions machines. Un processeur construit en un seul circuit intégré est appelé **microprocesseur**. Un programmeur peut choisir de se placer directement à ce niveau et coder directement en assembleur (langage machine) ; ce qui est souvent le cas dans la conception de circuits sécurisés.

4. **Circuits logiques**
   Une instruction de type ALU (Arithmetic Logic Unit) se décompose en fonctions booléennes. Elles sont mises en œuvre en électronique sous forme de portes logiques.

5. **Transistors** : dispositifs semi-conducteurs qui permet de contrôler un courant (ou une tension). Les portes logiques sont construites à partir de plusieurs transistors connectés de manière adéquate. Dans un transistor, un 0 logique correspond à une tension en dessous d'une certaine valeur sinon cela correspond à un 1 logique.

Les attaques physiques matérielles se situent au niveau « circuits logiques » ou « transistors » ; dans cette thèse nous nous situons au niveau : « circuits logiques ».

3.2 Les circuits utilisés dans la thèse.

Bien que cette thèse ait un cadre relativement théorique, des attaques physiques ont néanmoins été réalisées en pratique sur des microcontrôleurs : un ATmega, un STM32 et sur un FPGA.

Définition 21 : Un **microcontrôleur** est un circuit intégré synchrone qui rassemble les éléments essentiels d'un ordinateur : processeur, mémoires, unités périphériques et interfaces d'entrées-sorties. Ils sont fréquemment utilisés dans les systèmes embarqués.

Définition 22 : Un FPGA (**Field-Programmable Gate Array**) est un circuit intégré logique qui peut être reconfiguré. On se substitue au processeur en choisissant les connexions des portes logiques entre elles. Il permet de simuler un circuit dédié. C'est un compromis très rentable qui permet d'économiser le temps et le coût de fabrication d'un vrai circuit.

3.2.1 Contraintes temporelles des circuits.

Définition 23 : Un circuit **synchrone** est un circuit électronique numérique qui fonctionne à un rythme cadencé par une **horloge interne**.

Définition 24 : Un circuit **asynchrone** par opposition au circuit synchrone est un circuit électronique numérique qui n'utilise pas de signal d'horloge global pour synchroniser ses différents éléments.

Les circuits utilisés dans cette thèse sont tous des circuits synchrones. Un circuit synchrone est constitué d'un ou plusieurs blocs logiques encadrés par des registres cadencés par une horloge interne commune, notée clk (clock). Les blocs logiques correspondent aux fonctions utilisées par l'algorithme théorique. Les **registres** sont des « éléments mémoire » du circuit, correspondant aux blocs de l'algorithme de chiffrement par blocs.

L'horloge interne donne à intervalles réguliers, une impulsion électrique, simultanément à tous les composants du processeur. À chaque front montant de l'horloge, les données sont mises à jour à la sortie des registres. La Figure 6 représente la logique synchrone des circuits, les droites verticales rouges représentent les fronts montants de l'horloge.

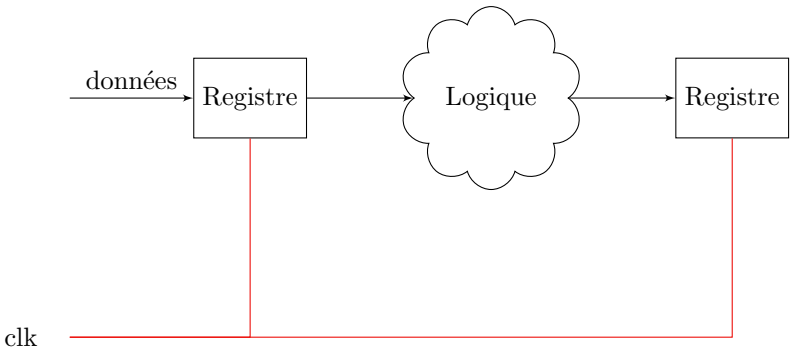


Figure 6 : Représentation de la logique synchrone

Il apparaît évident que la période d'horloge doit être suffisamment longue pour assurer un bon déroulement des calculs. Plus précisément, nous avons les notations suivantes :

- t_{clk} : la période d'horloge.
- t_{Max} : le temps auquel a lieu le dernier changement de valeur des données en entrée des registres avant d'être stable, C'est le temps du bit qui a le chemin le plus long pour aller d'un registre à un autre.
- t_s : le temps pendant lequel la donnée doit rester stable en entrée d'un registre pour être échantillonnée correctement.

L'équation suivante doit toujours être vérifiée :

$$t_{\text{clk}} > t_{\text{Max}} + t_s \quad . \tag{2}$$

3.2.2 Consommation de courant et émission d'ondes électromagnétiques

Un **courant électrique** est un déplacement de charges électriques, généralement des électrons, au sein d'un matériau conducteur. Chaque instruction réalisée par un circuit met en œuvre un certain nombre de transistors. À chaque instant, la mesure de l'intensité du courant électrique consommée (appelée **consommation de courant**) reflète l'activité du circuit. Certaines opérations, plus coûteuses, augmentent la consommation électrique du circuit, notamment à cause de l'utilisation de plus de composants.

En cryptanalyse matérielle, l'analyse de consommation du courant a pour but de découvrir des informations secrètes.

Les **émissions d'ondes électromagnétiques** des circuits sont causées par la circulation de courant au sein du circuit. Cette circulation de charges dans le conducteur produit un champ électromagnétique composé d'un champ magnétique et d'un champ électrique, mathématiquement reliés par les équations de Maxwell. Ces ondes électromagnétiques se propagent dans l'espace et ainsi peuvent être captées par une sonde. La sonde doit généralement être positionnée aussi près que possible de la surface active du circuit.

En cryptanalyse matérielle, l'analyse d'émission d'ondes électromagnétiques notée (EM) consiste à étudier le rayonnement électromagnétique provenant d'un circuit pour obtenir des informations secrètes.

L'avantage majeur des analyses EM par rapport aux analyses par consommation de courant est le fait que les mesures sont prises localement. Cependant, les mesures EM sont souvent plus sensibles au bruit et sont très dépendantes du positionnement de la sonde.

La consommation de courant et l'EM sont étroitement liées, ainsi elles sont souvent modélisées et utilisées de manière très similaire.

4 Description des bancs

Dans ce paragraphe les bancs EM et « clock glitch », qui ont permis de réaliser des attaques en pratique, sont présentés.

4.1 Banc EM

Les mesures EM ont été réalisées, en suivant les protocoles expérimentaux de Dehbaoui [25] et d'Aboulkassimi *et al.* [26].

Le banc d'acquisition des courbes EM est constitué d'une sonde EM RFU-2 de la marque Langer, d'un oscilloscope Tektronix DPO7104 avec amplificateur (Figure 7) et d'un ordinateur de contrôle.

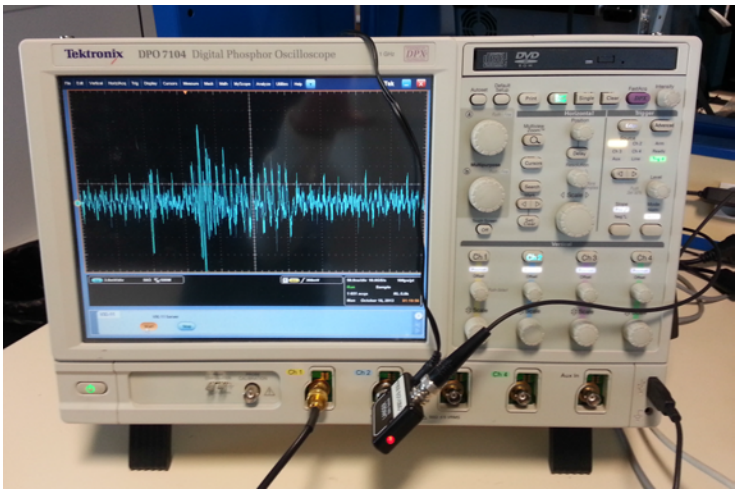


Figure 7 : Oscilloscope, sur l'écran nous pouvons voir l'exécution d'un AES.

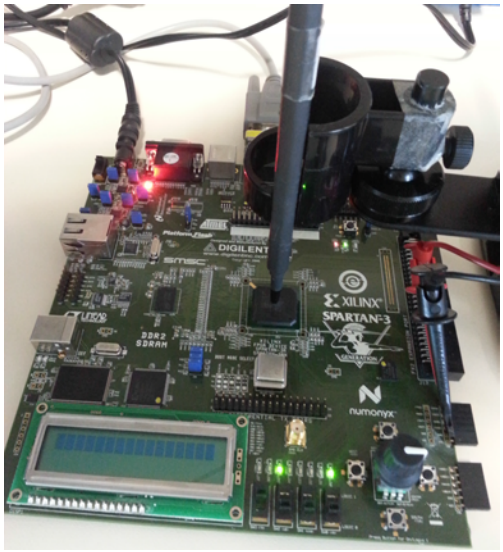


Figure 8 : Sonde EM au contact d'un FPGA

La sonde est mise au contact du circuit et est branchée à un oscilloscope qui enregistre les

mesures, comme illustré dans la Figure 8. Le circuit (FPGA, STM32 ou ATmega) est lui aussi branché à l'oscilloscope pour envoyer le signal de trigger. L'ordinateur contrôle le circuit, c'est à dire que c'est lui qui envoie la commande de chiffrement. Le signal du trigger informe l'oscilloscope du moment où il doit enregistrer les données fournies par la sonde. Pour finir l'oscilloscope envoie les courbes à l'ordinateur de contrôle qui peut alors les analyser (dans notre cas, avec Matlab). La Figure 9 illustre le protocole expérimental.

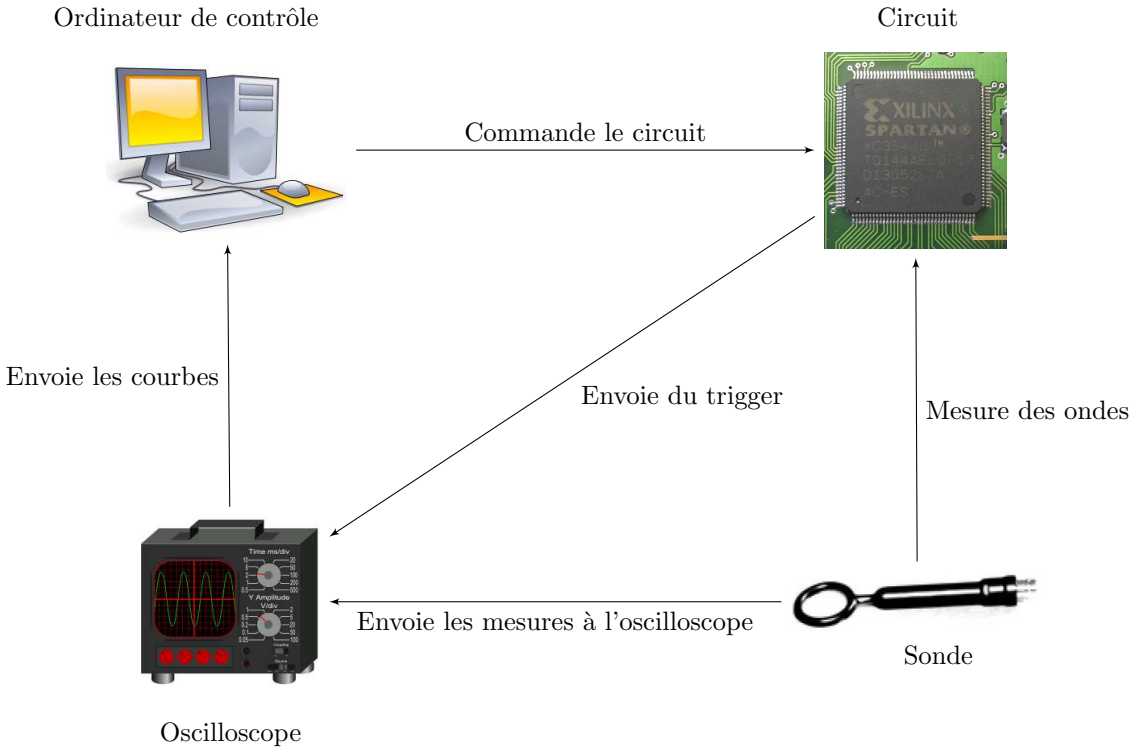


Figure 9 : Représentation schématique du banc EM.

4.2 Banc d'injection de fautes

Les attaques qui perturbent la synchronisation de l'horloge sont appelées attaques par *clock glitch*. Dans cette thèse, des attaques de ce type ont été réalisées sur FPGA Spartan Virtex 5, en suivant le protocole expérimental décrit dans les publications de Zussa *et al.* [27, 28].

Dans une attaque par *clock glith*, l'horloge est perturbée soit par modification soit par substitution. L'inéquation (2) indique qu'en réduisant la période d'horloge t_{clk} à une période suffisamment courte t^*_{clk}, les calculs ne sont pas évalués correctement, ce phénomène est appelé **violation des contraintes temporelles**. Nous avons l'inééquation suivante :

$$t^*_{\text{clk}} < t_{\text{Max}} + t_s < t_{\text{clk}} \quad .$$

La modification de l'horloge est illustrée en Figure 10.

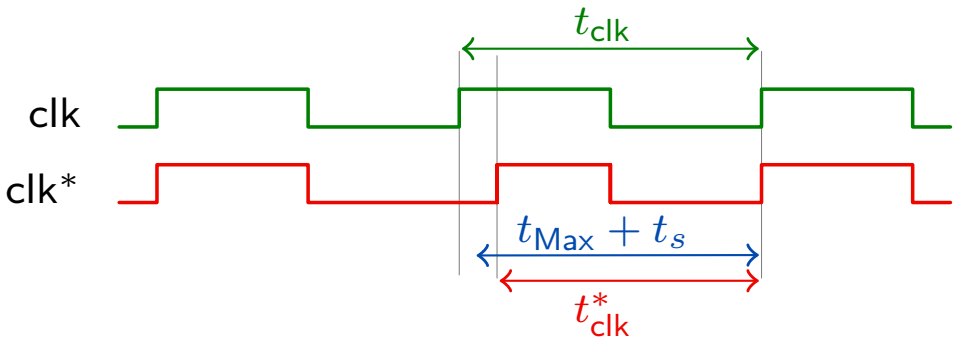


Figure 10 : Schéma du glitch d'horloge théoriquement injecté

En pratique, la Figure 11 est une photo du banc d'injection de glitch d'horloge. Le FPGA de gauche (Virtex 5) génère l'horloge « glitchée » qui cadence le FPGA de droite (Spartan 3) sur lequel l'AES est implémenté. La synchronisation se fait avec un signal du trigger.

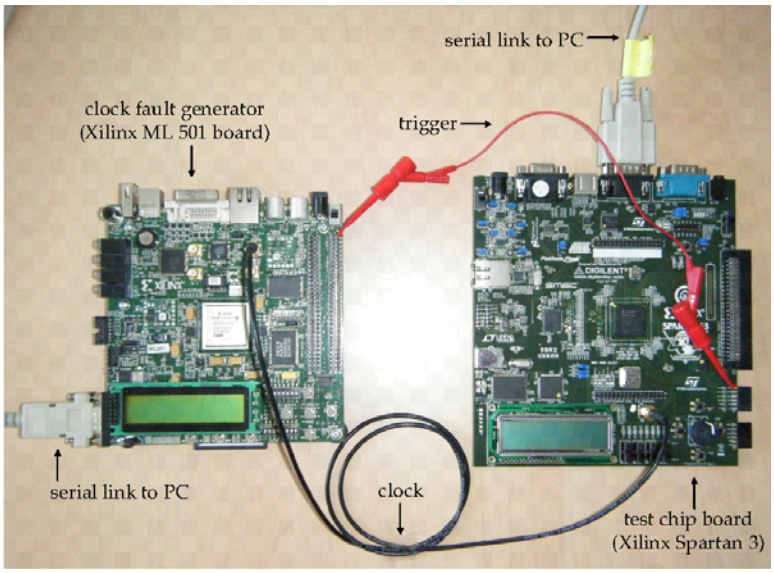


Figure 11 : Banc d'injection de fautes par glitchs d'horloge

La Figure 12 présente le « glitch » d'horloge injecté, observé sur un oscilloscope. La première courbe bleue représente l'horloge interne normale, la seconde l'horloge modifiée. La courbe en rose représente le temps de calcul d'un AES.

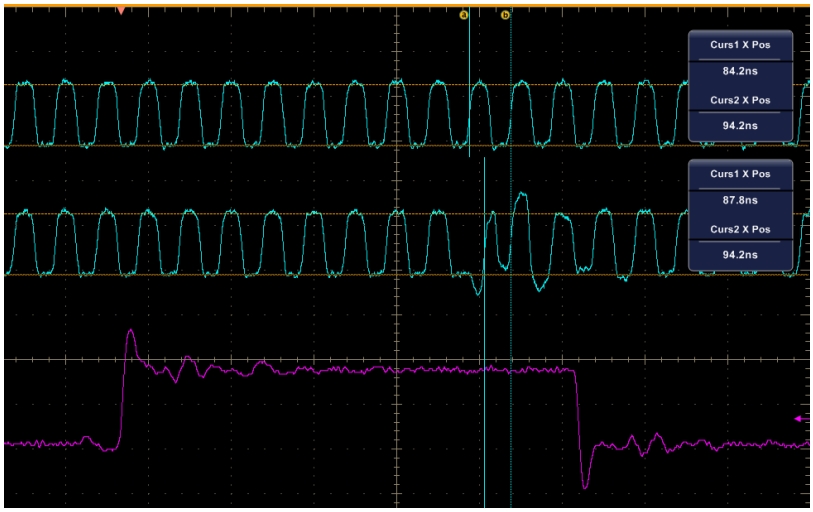


Figure 12 : Capture d'oscilloscope d'un glitch d'horloge

5 Plan de la thèse

Dans un premier temps (chapitre II), le formalisme des attaques physiques est abordé. La description des attaques physiques par le formalisme, puis l'outil de comparaison y sont présentés. Des exemples d'attaques sont également décrits afin d'illustrer la mise en pratique du formalisme. Ces travaux ont été exposés à Chip-to Cloud 2013 [29] et publiés sur Cryptology ePrint [30].

Au chapitre III une nouvelle attaque physique issue du formalisme est présentée de manière théorique, elle n'a pas encore été réalisée en pratique. Ce chapitre est un travail collaboratif avec Ronan Lashermes doctorant au CEA-Tech, Jean-Yves Zie Diali stagiaire au CEA-tech, Antoine Wurcker doctorant à Limoges au laboratoire Xlim et son directeur Christophe Clavier. Ce chapitre est donc en collaboration avec la thèse d'Antoine Wurcker portant sur : ≪ *l'étude de la sécurité d'algorithmes de cryptographie embarquée vis-à-vis des attaques par analyse de la consommation de courant* ≫. Cette thèse concerne la conception de nouvelles attaques et de nouvelles contre-mesures dans le domaine de l'analyse de courant sur des fonctions cryptographiques.

Le chapitre IV présente une étude générique d'attaques par injection de fautes, sur un type de chiffrement bien précis qui sont les schémas de Feistel généralisés.

L'objectif est d'anticiper de manière théorique, les registres à cibler pour réaliser une attaque de type DFA. Cette étude est un travail collaboratif avec Gaël Thomas doctorant à Limoges au laboratoire Xlim, encadré et dirigé par Thierry Berger et Marine Minier. Les travaux de recherche de Gaël Thomas portent sur ≪ *l'utilisation en cryptographie symétrique de certaines classes d'automates dont la fonction de transition dispose d'une structure algébrique forte* ≫. Il a notamment étudié les propriétés dites structurelles des schémas de Feistel généralisés qui sont, avec les réseaux substitution-permutation, l'une des grandes familles de chiffrement par blocs.

Les travaux de ce chapitre seront présentés à FDTC 2014.

Le chapitre V s'intéresse à la rétro-conception d'un pseudo DES dont les s-boxes sont inconnues. Une nouvelle attaque de type FIRE y est décrite. Elle a été présentée à Chip-to Cloud 2013 [31], et publiée à FPS 2013 [32].

Ce chapitre est un travail collaboratif avec Sylvain Guilley de Telecom Paris-Tech.

Pour finir la conclusion est en chapitre VI.

Partie II

Formalisme des attaques physiques

Sommaire

1 Objectifs et état de l'Art

Bien que les attaques physiques aient des principes et des cibles distincts, cette thèse présente un formalisme qui permet d'unifier toutes ces attaques. En écrivant les attaques physiques sous une même forme, nous constatons qu'il est plus facile de les comparer entre elles. Le formalisme peut être vu comme une sorte de guide. Il a un rôle descriptif et comparatif.

A formalism to compare them all,
A formalism to combine them,
A formalism to find them all,
And in the cryptanalysis bind them.

De nombreux formalismes ont déjà été introduits. Cependant, ils sont axés soit sur les attaques SCA [33, 34, 35, 36, 37, 38, 39, 40, 41] soit les attaques FIA [42, 43, 44], mais jamais ils ne regroupent les deux familles. Pourtant, les attaques hybrides attestent l'existence d'un lien entre les deux.

Dans [37], Standaert *et al.* s'intéressent aux liens entre pratique et théorie pour les attaques par observation. Notre formalisme élargit l'étude aux attaques par injection de fautes. Le débat pour savoir si le terme « attaques par canaux auxiliaires » désigne les attaques par observations ou attaques physiques en générale ; met en évidence le fait que la distinction entre attaques par observation et attaques par injection de fautes est parfois difficile. Ces types d'attaques sont en effet très proches, d'un point de vue théorique. Dans cette thèse, grâce au formalisme, toutes les attaques physiques matérielles se décrivent de la même manière sans distinction.

Notre formalisme décompose les attaques physiques en différentes étapes. Notre approche est assez proche de [37] au niveau de la description des variables et de [36] dans la décomposition en étapes des attaques. Par ailleurs, cette décomposition nous permet d'évaluer l'attaque en détails et non uniquement dans sa globalité. Dans le formalisme, une attaque est vue comme un ensemble de paramètres choisis à chaque étape, pouvant être modifiés quasi séparément les uns des autres, pour former de nouvelles attaques.

2 Description du formalisme en trois étapes

Dans cette partie, nous proposons une méthode pour décrire les attaques physiques via notre formalisme. Ce dernier décompose les attaques physiques en trois étapes. La cible est notée $\mathcal{K}$ quelle que soit sa nature (clé ou partie d'algorithme).

2.1 Étape 1 : campagne.

Définition 25 : Une **expérience** $\mathcal{E}$ est l'exécution du processus de chiffrement par un circuit, avec ou sans perturbation. Un ensemble de n expériences est appelé **campagne**.

Définition 26 : Une **observable** est une donnée ou une mesure que l'attaquant acquiert durant une expérience. Le résultat d'une expérience est un couple d'observables (O_S, O_R). O_S est appelé observable **stimulus** et O_R observable **réaction**.

Les observables les plus souvent rencontrées sont :
- le texte clair T,
- le texte chiffré C,
- un texte chiffré fauté C^*,
- une mesure de la consommation de courant (noté $Conso$),
- une mesure du temps de calcul,
- une sensibilité à une faute,
- une mesure EM.

L'exemple typique de couple d'observables, dans une attaque par observation est :

$$(O_S = T, O_R = Conso) \quad .$$

L'exemple typique de couples d'observable dans une attaque par injection de fautes est :

$$(O_S = C, O_R = C^*) \quad .$$

Définition 27 : La **relation exploitable** ou **chemin d'attaque** $\mathcal{R}$ permet de faire le lien entre les observables (O_S, O_R) et la cible $\mathcal{K}$. Elle est composée d'une ou plusieurs fonction(s) physique(s) notées f_j et de fonctions g_j basées sur l'algorithme [1].

$$\begin{array}{rcl} O_R & = & \mathcal{R}(O_S, \mathcal{K}) \\ \mathcal{R} & = & f_1 \circ g_1 \circ \cdots \circ f_n \circ g_n \end{array} \tag{3}$$

Les fonctions physiques sont liées au comportement du circuit. Il est difficile de déterminer tous les paramètres qu'elles utilisent. Ainsi, elles n'ont pas d'écriture algébrique connue. Dans [45, 46], les fonctions physiques pour les attaques par observations, sont décrites comme des fonctions auxquelles est ajoutée une valeur aléatoire représentant un bruit. Dans le cadre de cette thèse, nous choisissons la définition suivante.

Définition 28 : Une **fonction physique** est une transformation liée au circuit, que l'attaquant ne sait pas décrire mathématiquement. En effet, c'est une transformation stochastique, à une même entrée elle n'associe pas toujours la même sortie.

1. Il est important de préciser que la composition $\circ$ n'est pas la composition au sens mathématique classique du terme car les fonctions physiques ne sont pas déterministes.

Une fonction physique est appelée **fonction de fuite** dans les attaques par observation et **fonction d'erreur** dans les attaques par injection de fautes.

Le plus souvent, il n'y a qu'une seule fonction physique dans le chemin d'attaque. Un exemple classique de chemin d'attaque par observation de courant, sur l'AES, pour retrouver la clé du premier tour est :

$$\mathcal{R} = f \circ g = \underbrace{conso}_{f} \circ \underbrace{SB \circ \oplus}_{g} \quad . \tag{4}$$

Ce qui donne :

$$\mathcal{R}(T, K_0) = conso(SB(K_0 \oplus T)) = Conso \quad .$$

Les exemples les plus connus de fonctions physiques sont les suivants :
- la consommation de courant *conso*,
- l'émission d'ondes électromagnétiques *em*,
- le processus d'injection de fautes $\lightning$,
- le temps de calcul.

Les fonctions g_j sont des fonctions de l'algorithme attaqué ou leurs inverses ou une fonction composée de ces mêmes fonctions. Ainsi, dans le cas de l'AES les g_j se composent dans $\{SB, SR, MC, \oplus\}$ et leurs inverses.

Pour être tout à fait exact et cohérent avec les notations, les fonctions g_j et f_j prennent jusqu'à 3 valeurs en entrée ($\mathcal{K}$, O_S, une image retournée par une fonction précédente du chemin d'attaque). Ceci est illustré et détaillé dans les exemples section 3.

2.2 Étape 2 : prédictions

Les attaques physiques sont de type « diviser pour régner ». La cible n'est jamais attaquée entièrement en une seule fois, elle est partitionnée en petits fragments. Les hypothèses sur la cible $\mathcal{K}$ sont notées k. La bonne hypothèse sur la cible est notée $\hat{k}$. Le domaine de définition des hypothèses noté $\mathbb{K}$, doit avoir un cardinal suffisamment petit. La puissance doit être suffisante pour pouvoir prendre en compte toutes les hypothèses k. Par exemple, si le but est de retrouver la clé de chiffrement d'un AES, généralement seul un octet d'une clé de tour est attaqué, à la fois. Ainsi, $\mathbb{K} = [\![0, 255]\!]$ et son cardinal est 256.

Le but de l'attaquant est de discerner la bonne hypothèse des autres. Pour cela il va reprendre le chemin d'attaque $\mathcal{R}$ pour construire une **prédiction** pour chacune des hypothèses k. Le problème vient alors des fonctions physiques f qui n'ont pas d'écriture mathématique connue.

Définition 29 : Un **modèle** est une fonction mathématique ayant pour but d'approcher le comportement d'une fonction physique.

En reprenant l'équation (3), chaque fonction physique est remplacée par un modèle. La relation $\mathcal{R}_m$ est alors appelée **chemin d'attaque théorique**. Pour chaque observable O_S et chaque hypothèse k, une prédiction $P_{m,k}$ associée à un ensemble m de modèles m_j choisis (un par fonction physique), est calculée. L'ensemble des prédictions est noté $\mathbb{P}$.

$$\begin{aligned} P_{m,k} &= \mathcal{R}_m(O_S, k) \\ \mathcal{R}_m &= m_1 \circ g_1 \circ \cdots \circ m_n \circ g_n \end{aligned} \tag{5}$$

En pratique, aucun modèle ne peut pas correspondre parfaitement à la réalité. De plus, différents

modèles peuvent approcher la même fonction physique. Dans ce cas, un ensemble d'ensembles de modèles est noté $m = \{m_i\}$, tel que $m_i = \{m_{i,j}\}$, j faisant référence à la fonction physique et i aux différents modèles possibles. Par abus de notation, nous notons m_i sans indice j lorsqu'il n'y a qu'une fonction physique f dans $\mathcal{R}$.

Deux exemples de modèles :

1. Les modèles les plus courants pour les fonctions de fuite sont le poids de Hamming HW et la distance de Hamming HD.

   Définition 30 : Le **poids de Hamming** noté HW d'un mot binaire est le nombre de bits à 1. La **distance de Hamming** notée HD entre deux mots binaires est le nombre de bits ayant changés de valeurs.

   En reprenant l'exemple du paragraphe précédent, en équation (4), cela donne, par exemple :

   $$\mathcal{R}_{\mathcal{HW}} = m \circ g = \underbrace{HW}_{m} \underbrace{\circ SB \circ \oplus}_{g} \quad .$$

2. Pour une injection de faute mono-bit sur un octet, la fonction « xor » peut permettre de construire 8 modèles différents :

   $$m = \{\oplus 2^0, \oplus 2^1, \oplus 2^2, \oplus 2^3, \oplus 2^4, \oplus 2^5, \oplus 2^6, \oplus 2^7\}.$$

Certains modèles sont meilleurs que d'autres. Le choix est alors le suivant : savoir si les modèles sont utilisés séparément ou non. Généralement, dans l'exemple 1 ci dessus, le choix est un « ou exclusif » : HW ou HD ; alors que dans l'exemple 2, les 8 modèles sont pris en considération simultanément.

2.3 Étape 3 : confrontation

2.3.1 Retrouver la cible grâce à un distingueur

Cette ultime étape permet de retrouver la cible $\mathcal{K}$, si l'attaque a réussi.

Définition 31 : Un **distingueur** est un outil statistique qui permet de faire ressortir une hypothèse k_d en utilisant O_R et les $P_{m,k}$ comme des variables aléatoires. L'attaque a réussi lorsque $k_d = \hat{k}$.

Généralement le choix du distingueur dépend de la quantité des échantillons de la campagne et de la qualité du ou des modèles. Plus le modèle est proche de la fonction physique, moins le choix du distingueur n'a d'impact sur la réussite de l'attaque.

De nombreuses recherches portent sur la comparaison des distingueurs et la recherche du distingueur idéal. Un travail de comparaison des distingueurs a été réalisé dans [47]. Dans [34], et [40] Standaert *et al.* ramènent les attaques SCA classiques à la forme de DPA (Differential Power Analysis). Dans [48] Doget *et al.* montrent l'équivalence de nombreux distingueurs. Ces travaux font remarquer que l'étape du distingueur ne définit pas l'attaque en elle-même puisqu'ils considèrent comme équivalentes plusieurs attaques où seul le distingueur diffère. Pour cela ils se ramènent à un distingueur commun.

Cependant le débat sur les différents distingueurs et la recherche de nouveaux toujours plus performants n'est pas un axe de recherche de cette thèse, ainsi seule une brève description des plus connus est présentée.

2.3.2 Les distingueurs les plus utilisés

$P_{m,k}$ et O_R sont deux variables aléatoires pouvant être discrètes ou continues. Afin d'améliorer la clarté de nos propos dans la suite de cette partie 2.3, les variables aléatoires X et Y sont parfois utilisées ; avec $x \in \mathcal{X}$ l'Univers de X, remplaçant ainsi $P_{m,k}$ et O_R. De plus, certaines formules mathématiques sont rappelées en annexe C.

a) Distingueurs basés sur l'égalité

Ces distingueurs ne sont utilisés que si les deux variables $P_{m,k}$ et O_R discrètes.

Définition 32 : Un **crible** élimine toutes les hypothèses k telles qu'il existe au moins un stimulus O_S tel que $P_{m,k} \neq O_R$. Un **compteur** retourne l'hypothèse k_d pour laquelle $P_{m,k_d} = O_R$ pour le plus grand nombre de stimuli O_S .

Ils supposent que les modèles choisis m_j sont "égaux" aux fonctions physiques f_j ; du moins que l'ensemble d'arrivée de $\mathcal{R}$ est le même que $\mathcal{R}_m$. Ils sont essentiellement utilisés dans les attaques par injection de fautes.

b) Distingueurs basés sur la dépendance linéaire

Historiquement, le premier distingueur proposé dans les attaques SCA [49] est la **différence de moyennes**. Celle-ci a été rapidement remplacée par la **corrélation** de Pearson [50]. Il s'agit d'un distingueur supposant l'existence d'une dépendance affine entre le modèle et la fonction physique.

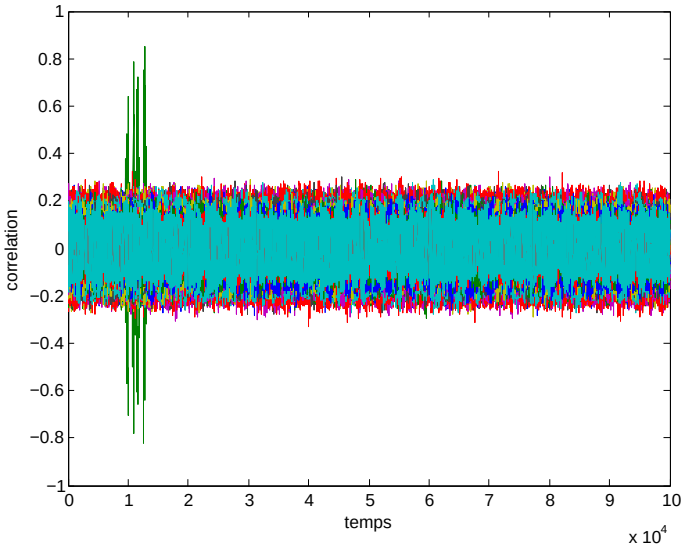


Figure 13 : Résultat d'une attaque CPA (Correlation Power Analysis) sur un AES implémenté dans un ATmega, du premier octet de clé du premier tour $K_1^{0,0}$. La bonne hypothèse de clé est associée à la courbe verte.

Définition 33 : La corrélation de Pearson entre deux variables aléatoires X et Y est définie comme suit :

$$\mathrm{Cor}(X,Y) = \frac{\mathrm{Cov}(X,Y)}{\sqrt{\mathrm{Var}(X)\cdot\mathrm{Var}(Y)}} \quad .$$

Ce distingueur retourne l'hypothèse k_d pour laquelle la valeur absolue de la corrélation est la plus proche de 1, comme le montre la Figure 13. Si réellement il y avait une dépendance linéaire (c'est à dire $f(x) = \alpha \cdot m(x) + \beta$), la valeur absolue de la corrélation serait égale 1 pour la bonne hypothèse $\hat{k}$.

c) Distingueurs basés sur l'entropie des lois de probabilité

Définition 34 : L'**entropie** de Shanon est « une mesure de l'incertitude ». C'est une fonction qui, intuitivement, correspond à la quantité d'information contenue ou délivrée par une source d'information.

$$H(X) = -\sum_{x \in \mathcal{X}} Pr(X = x) \cdot \log_2(Pr(X = x))$$

L'**entropie conditionnelle** de deux variables aléatoires X et Y est :

$$H(X|Y) = -\sum_{x \in \mathcal{X}, y \in \mathcal{Y}} Pr(X = x, Y = y) \cdot \log_2(Pr(X = x|Y = y)) \quad .$$

L'**entropie jointe** de deux variables aléatoires X et Y est :

$$H(X, Y) = -\sum_{x \in \mathcal{X}, y \in \mathcal{Y}} Pr(X = x, Y = y) \cdot \log_2(Pr(X = x, Y = y)) \quad .$$

Définition 35 : L'**information mutuelle** entre deux variables représente le degré de dépendance d'un point de vue probabiliste.

$$I(X;Y) = H(X) - H(X|Y) = H(X) + H(Y) - H(X,Y) = I(Y;X) \tag{6}$$

L'information mutuelle est un distingueur très utilisé, comme le montrent, les travaux suivants [51, 52, 53, 54, 55].
Pour calculer une information mutuelle, une loi de probabilité est nécessaire. Celle-ci est alors soit estimée par histogrammes, soit supposée suivre une loi normale.

d) Distingueur basé sur le test de Kolmogorov Smirnov.

Définition 36 : En statistiques, **le test de Kolmogorov-Smirnov** est un test d'hypothèses, utilisé pour déterminer si un échantillon suit bien une loi donnée connue par sa fonction de répartition continue, ou bien si deux échantillons suivent la même loi.

Le test de Kolmogorov-Smirnov peut être utilisé comme distingueur, comme dans [56, 57], dans le sens où il compare des distributions.

e) Distingueurs basés sur la variance inter et intra classes.

Définition 37 :
L'espérance de la variance conditionnelle : $\mathbb{E}(Var(X|Y))$ est appelée **variance intra-classes**.
La variance de l'espérance conditionnelle : $Var(\mathbb{E}(X|Y))$ est appelée **variance inter-classes**.

A l'origine, la composante principale est un outil statistique utilisé pour la compression et la classification de données. L'objectif est de trouver un sous-ensemble ayant perdu un minimum

d'information. Plus précisément, nous cherchons une transformation linéaire qui projette les données dans un sous-espace qui maximise la variance. Les nouvelles variables sont ordonnées en fonction de l'information qu'elles contiennent, plus exactement en fonction de leurs variances inter-classes. La première variable de ce classement est appelée **composante principale**.

L'analyse par composante principale a été utilisée pour la première fois comme distingueur dans une attaque par canaux auxiliaires, de Guilley *et al.*, dans [58]. Le principe repose sur le fait que :

$$\mathrm{Var}(X) = \mathbb{E}(\mathrm{Var}(X|Y)) + \mathrm{Var}(\mathbb{E}(X|Y)) \quad .$$

La composante principale maximise la variance intra-classe. Lors d'une attaque, une mauvaise hypothèse sur la cible a une variance inter-classes nulle, c'est à dire :

$$\mathrm{Var}(\mathbb{E}(X|Y)) = 0 \quad \Rightarrow \quad \mathrm{Var}(X) = \mathbb{E}(\mathrm{Var}(X|Y)) \quad .$$

Le discriminant linéaire quant à lui, maximise le rapport entre la variance inter-classes et la variance intra-classes : La bonne hypothèse est celle associée au plus grand discriminant. Il est utilisé comme distingueur dans les attaques [59] et [60].

2.4 Cas particulier des attaques à plusieurs niveaux d'étapes

Dans certaines attaques la cible n'est pas retrouvée directement, il faut passer par des données intermédiaires. Mais chacune de ces données intermédiaires est retrouvée en suivant le même schéma des trois étapes décrites précédemment. Ainsi pour chaque niveau de recherche il y a une-sous cible $\mathcal{K}^i$, des observables (O_S^i, O_R^i) et un chemin d'attaque $\mathcal{R}^i$. Le niveau suivant est défini par l'équation suivante :

$$\mathcal{R}^{i+1}(O_S^{i+1}, \mathcal{K}^{i+1}) = O_R^{i+1} \text{ avec } \mathcal{K}^i = O_S^{i+1} \text{ ou } O_R^{i+1} \quad .$$

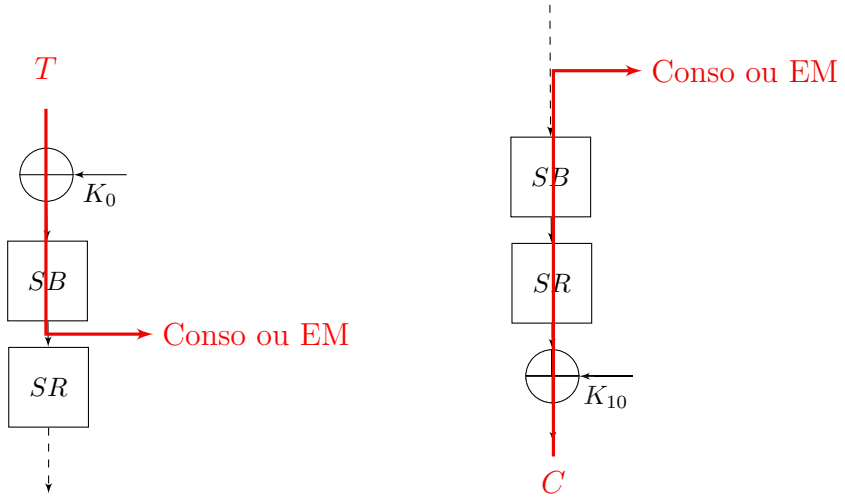
Un exemple d'attaque pour la rétro-conception est décrit en annexe B.ii.c.

3 Formalisme appliqué à des exemples d'attaques physiques

Dans cette partie un échantillon représentatif des attaques physiques est parcouru. L'idée est de décrire ces attaques grâce au formalisme comme dans la partie précédente (section 2). L'algorithme choisi pour illustrer le formalisme est l'AES. Des attaques similaires sur le DES, ainsi que d'autres attaques sur l'AES sont décrites en annexe B.

3.1 Attaques par observation sur AES

Des d'attaques classiques de type SCA sur AES sont présentées en [61, 50]. Il s'agit d'attaques à clairs ou chiffrés connus. L'idée des attaques par consommation de courant ou d'émission d'ondes électromagnétique est que ces canaux auxiliaires fuient souvent en poids et/ou distance de Hamming. Généralement l'attaquant se place au niveau des s-boxes.



(a) premier tour (b) dernier tour

Figure 14 : Chemins d'attaques SCA sur AES

- $\mathcal{K} = K_0^{l,c}$ ou $\mathcal{K} = K_{10}^{l,c}$, la cible est un octet de la clé du premier ou dernier tour. $k \in \mathbb{K} = [\![0, 255]\!]$. Il y a 256 hypothèses k. Il existe une variante où la cible est juste un bit de l'octet.

- $O_S = T$ ou $O_S = C$, l'observable stimulus est le texte clair si l'attaque est au premier tour, le texte chiffré si l'attaque est au dernier.

- $O_R = EM$ ou $Conso$, l'observable réaction est une mesure de consommation de courant ou une mesure d'émission d'ondes électromagnétiques.

– Le chemin d'attaque est le suivant : $\mathcal{R} = f \circ g$, avec :
$f = em$ ou $f = conso$,
$g = SB \circ \oplus$ si l'attaque est au premier tour,
$g = SB^{-1} \circ SR^{-1} \circ \oplus$ si l'attaque est au dernier tour.
Ces deux chemins d'attaque sont illustrés en Figure 14.

– $\mathcal{R}_m = m_i \circ g$, avec : $i = \{1, 2\}$
$m_1 = HW$ poids de Hamming ou
$m_2 = HD$ distance de Hamming (Définition 30).
Généralement le poids de Hamming est choisi pour les circuits qui exécutent les calculs sur un octet à la fois. Dans le cas d'une distance de Hamming, g retourne deux éléments par exemple :
$g(K_0^{l,c}, T^{l,c}) = (SB(K_0^{l,c} \oplus T^{l,c}), T^{l,c})$ ainsi le modèle $m_2 = HD$ a bien deux valeurs en entrée.
Dans les deux cas $\mathbb{P} = [\![0, 8]\!]$, il y a 9 valeurs de prédictions possibles.

– Le distingueur peut être une différence de moyennes, une corrélation équation (33), une information mutuelle équation (6), une composante principale, ou un discriminant linéaire (paragraphe e section 2.3.2).

3.2 Attaque par injection de fautes sur AES

L'attaque présentée ici est une DFA classique sur l'AES, il en existe plusieurs dont les plus connues sont [12, 11, 14, 15]. Il s'agit d'attaques à chiffrés connus. Nous rappelons que la DFA confronte les textes chiffrés aux textes chiffrés fautés afin de réduire l'espace d'hypothèses possibles.

Dans cet exemple, l'emplacement choisi pour injecter une faute, est l'entrée de tour 10 ; cependant d'autres emplacements sont possibles. Par exemple, dans la publication [12] de Giraud *et al.*, plusieurs autres attaques par injection de fautes sont présentées, injection de fautes sur K_{10} avant le KeyExpansion, sur K_9 avant le KeyExpansion et sur la sortie du 9ème tour. L'emplacement de la faute varie uniquement pour avoir plus de bits fautés à l'entrée du dernier tour, la cible reste la même.

Étudions les cas où la faute est injectée dans l'algorithme avant le dernier tour comme illustré dans la figure 15, les variantes sont présentées en annexe B.i.a.

– $\mathcal{K} = K_0^{l,c}$ ou $K_{10}^{l,c}$, la cible est un octet de la clé du premier ou dernier tour.
$k \in \mathbb{K} = [\![0, 255]\!]$. Il y a 256 hypothèses k. Il existe une variante où la cible est juste un bit de l'octet.

– $O_S = C$, l'observable stimulus est le texte chiffré.

– $O_R = C^*$, l'observable réaction est le chiffré fauté.

– Le chemin d'attaque est illustré en Figure 15.
$\mathcal{R} = g_2 \circ f \circ g_1$, avec :
$f = \lightning$ processus d'injection de fautes,
$g_1 = SB^{-1} \circ SR^{-1} \circ \oplus$ et
$g_2 = \oplus \circ SR \circ SB$.

– $\mathcal{R}_m = g_2 \circ m_i \circ g_1$, avec :
$m_i = \oplus 2^i$, $i \in [\![0, 7]\!]$.
Cet ensemble de 8 modèles est utilisé lorsque l'attaquant pense avoir fait une injection de faute mono-bit en entrée du dernier tour. $\mathbb{P} = [\![0, 255]\!]$, il y a 256 prédictions possibles.
Si la faute n'est pas mono-bit ou a été injectée plus tôt dans l'algorithme ou les deux,
$\mathcal{R}_{m'} = g_2 \circ m'_i \circ g_1$, avec :
$m'_i = \oplus i$, $i \in [\![0, 255]\!]$.

– Le distingueur peut être un crible ou un compteur (Définition 32) ou une entropie (Définition 34) pour les modèles m. Pour les modèles m', si un seul modèle est considéré à la fois, cela suppose qu'on répète toujours la même faute à chaque expérience, ce sont les mêmes distingueurs (crible compteur ou entropie) qui peuvent être utilisés. Sinon, dans le cas où l'attaquant suppose que la faute varie, seule l'entropie est un bon distingueur (comme réalisé dans l'attaque de Lashermes *et al.* [14]). En effet, pour le crible ou le compteur il y a toujours une faute permettant d'expliquer n'importe quelle hypothèse ; à chaque modèle $m'_i = \oplus i$, il existe une hypothèse de clé k valide.

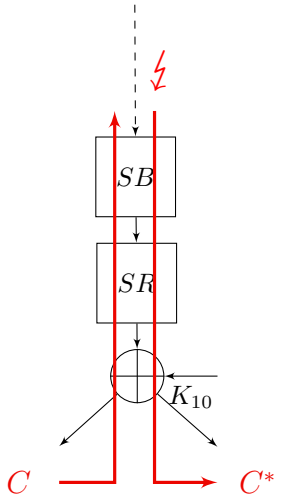


Figure 15 : Chemin d'attaque par injection de fautes sur AES au dernier tour

3.3 Exemple d'attaque hybride sur AES

La Fault Sensitive Ananlysis (FSA [19]) est une attaque hybride ; elle combine habilement les principes des attaques par injection de fautes et ceux des attaques par observation. En effet, il y a bien des injections de fautes, mais ce ne sont pas les résultats fautés qui sont analysés. C'est la « sensibilité » à cette faute, qui est utilisée comme un canal auxiliaire.

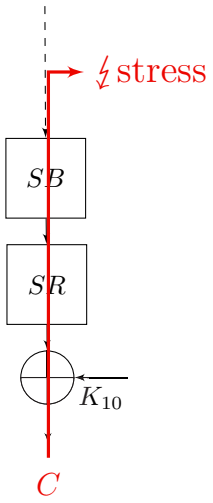


Figure 16 : Chemin d'attaque de la FSA sur AES.

- $\mathcal{K} = K_{10}^{l,c}$, la cible est un octet de la clé du tour 10. Il existe une variante où un seul bit de clé est attaqué à la fois. $k \in \mathbb{K} = [\![0, 255]\!]$. Il y a 256 hypothèses k.

- $O_S = C^*$, l'observable stimulus est le texte chiffré fauté.

- $O_R = I$ l'observable réaction est la sensibilité à la faute.

- Le chemin d'attaque est illustré en Figure 16.
  $\mathcal{R} = f \circ g$, avec :
  $f = \lightning$ processus d'injection de fautes et
  $g = SB^{-1} \circ SR^{-1} \circ \oplus$

- $\mathcal{R}_m = m \circ g$, avec $m = HW$.

- Le distingueur est une corrélation.

Une autre attaque hybride similaire : la Differential Behaviour Analysis (DBA), est présentée en annexe B i.b.

4 Outil d'évaluation et de comparaison

4.1 Pourquoi comparer ?

Il est naturel de se poser la question : « *Comment comparer des attaques physiques ?* ». Généralement, les attaques physiques sont comparées en termes d'équipements, de coût (de l'équipement, du temps de réalisation), de la précision exigée (de mesure pour les SCA et d'injection pour les FIA), la facilité à adapter le dispositif à un autre circuit etc. Par exemple dans [62], les auteurs comparent les différentes fautes pouvant être injectées pour un équipement donné. Pour les SCA, c'est l'efficacité des différents distingueurs qui est comparée (voir [47]). Un outil très utilisé est le taux de succès d'une attaque. Ces outils de comparaison ont un défaut, en cas d'échec, il n'est pas possible d'identifier quelle étape de l'attaque est inappropriée.

De nombreux formalismes, comme dans [47, 63, 41], comparent les attaques physiques à travers le distingueur. Or celui-ci n'est utilisé qu'à la dernière étape d'une attaque. Notre idée est de comparer les attaques physiques avant l'étape finale de confrontation. Bien sur, le choix du distingueur est crucial, car un mauvais choix peut faire échouer l'attaque. Cependant, de nombreux travaux sur les distingueurs ont déjà été menés. Notre idée est de regarder les deux premières étapes, sans se soucier de la dernière. L'outil de comparaison présenté dans cette section est donc complémentaire aux études menées sur le distingueur.

4.2 Évaluation

Des approche similaires à nos travaux ont été présentées dans [64, 65].

4.2.1 Étude théorique du chemin d'attaque

Dans la partie 2.2 des prédictions sont calculées. Une prédiction pour la bonne hypothèse sur la cible $P_{m,\hat{k}}$ (7) n'est pas équivalente à connaître la cible $\mathcal{K} = \hat{k}$ elle-même.

$$P_{m,\hat{k}} = \mathcal{R}_m(O_S, \hat{k}) \tag{7}$$

Rappelons que $\mathbb{P}$ est l'ensemble des prédictions possibles et $\mathbb{K}$ est l'ensemble des hypothèses possibles, ces ensemble peuvent avoir des cardinaux différents [2]. Le cardinal $\#\mathbb{P}$ est noté N.

Nous introduisons alors, un oracle Θ_m (Figure 17) associé au(x) modèle(s) m. À une observable stimulus O_S, Θ_m retourne la prédiction pour la bonne hypothèse sur la cible $P_{m,\hat{k}}$.

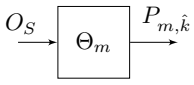


Figure 17 : Oracle Θ_m

Le nombre moyen d'appels qu'il faut faire à Θ_m pour retrouver $\hat{k}$ est noté θ ; autrement dit le nombre de $P_{m,\hat{k}}$ qu'il faut en moyenne pour retrouver la cible $\hat{k}$.

2. # signifie cardinal

Dans le cas où il y a m, un ensemble de modèles m_i, Θ_m est défini comme l'oracle qui pour un O_S fait appel aléatoirement à un des oracles Θ_{m_i} . En reprenant l'exemple de la fonction physique qui injecte des fautes mono-bit, où l'attaquant choisit la fonction « xor » comme modèle, avec les 9 modèles possibles : $m = \{\oplus 2^i\}, i \in [\![0, 8]\!]$; Θ_m fait appel aléatoirement à $\Theta_{\oplus 2^i}$.

Cette première étape, permet d'évaluer le chemin d'attaque théorique.
Plus θ est petit, meilleur est le chemin d'attaque théorique $\mathcal{R}_m$.

4.2.2 Étude des modèles par rapport aux réactions

Dans ce paragraphe, nous nous intéressons au lien entre les bonnes prédictions $P_{m,\hat{k}}$ et les observables réactions O_R, pour un ensemble de modèles m donné. C'est une manière de comparer les fonctions physiques aux modèles utilisés.

Construisons un tableau de répartition (Tableau 2) des $P_{m,\hat{k}}$ et des O_R. Pour cela la cible $\mathcal{K} = \hat{k}$ est supposée connue. À chaque expérience $\mathcal{E}$, O_R est observée et $P_{m,\hat{k}}$ calculée ; alors la valeur dans la case correspondant à ce couple $(O_R, P_{m,\hat{k}})$ est incrémentée de 1. Il est bien précisé, à chaque expérience $\mathcal{E}$ et non pas à chaque O_S, en effet pour deux expériences il peut y avoir le même stimulus O_S et pourtant pas la même réaction O_R. Pour réaliser un tel tableau, la campagne (n expériences) doit être suffisamment grande. Le nombre de valeurs différentes O_R observées est noté M. Attention M n'est pas le cardinal des valeurs possibles de O_R, car O_R n'est pas obligatoirement discret. De plus, comme l'ont fait remarquer les travaux [33, 36, 37, 40], il n'y a pas forcément bijection entre l'espace des $P_{m,\hat{k}}$ et des O_R, d'où $N \neq M$. Les valeurs des $P_{m,\hat{k}}$ sont notées ρ_i.

	$O_R = o_1$	$O_R = o_2$	$\cdots$	$O_R = o_M$	Total
$P_{m,\hat{k}} = \rho_1$	$a_{1,1}$	$a_{1,2}$	$\cdots$	$a_{1,M}$	$\sum_{j=1}^{M} a_{1,j}$
$P_{m,\hat{k}} = \rho_2$	$a_{2,1}$	$a_{2,2}$	$\cdots$	$a_{2,M}$	$\sum_{j=1}^{M} a_{2,j}$
$\vdots$	$\vdots$	$\vdots$	$\cdots$	$\vdots$	$\vdots$
$P_{m,\hat{k}} = \rho_N$	$a_{N,1}$	$a_{N,2}$	$\cdots$	$a_{N,M}$	$\sum_{j=1}^{M} a_{N,j}$
Total	$\sum_{i=1}^{N} a_{i,1}$	$\sum_{i=1}^{N} a_{i,2}$	$\cdots$	$\sum_{i=1}^{N} a_{i,M}$	n

Tableau 2 : Tableau de répartition des O_R et $P_{m,\hat{k}}$

Une fois le Tableau 2 construit, ce qui nous intéresse c'est de mesurer la répétabilité d'une expérience. Une expérience $\mathcal{E}$ est associée à un couple $(O_R = \hat{o} = o_{j_1}, P_{m,\hat{k}} = \hat{\rho} = \rho_{i_1})$. Quelle est la probabilité qu'une deuxième expérience $\mathcal{E}_2$ ayant la même valeur de mesure ait aussi la même valeur de prédiction ? Cette probabilité que nous allons estimer est notée **p**.

Pour estimer **p**, l'expérience suivante est décrite à l'aide du Tableau 2. Une première urne (l'urne 0) contient $a_{i,j}$ boules (ρ_i, o_j) soient n boules au total est considérée [3]. D'autre part M urnes différentes numérotées de 1 à M, chaque urne j contenant $a_{i,j}$ boules (ρ_i) sont également considérées [4]. L'expérience est la suivante :

1. Premier tirage : tirage aléatoire d'une boule dans l'urne 0.
   Le résultat est noté : $(\hat{o} = o_{j_1}, \hat{\rho} = \rho_{i_1})$.
2. Second tirage : tirage aléatoire d'une boule dans l'urne j_1 (j_1 définit par le tirage précédent).
   Le résultat est noté ρ_{i_2}.

3. Elle représente le tableau en entier.
4. Elles représentent chacune une colonne du tableau.

Intéressons nous à la probabilité :

$$\mathbf{p} = Pr(\rho_{i_2} = \rho_{i_1}) \quad .$$

Sachant les résultats du premier tirage, nous avons :

$$Pr(\rho_{i_2} = \rho_{i_1} | o_{j_1}) = \sum_{i=1}^{N} \frac{a_{i,J}^2}{n^2} \quad .$$

Ainsi, en prenant en compte tous les résultats possibles, du premier tirage nous obtenons :

$$\mathbf{p} = \sum_{j=1}^{M} Pr(\rho_{i_2} = \rho_{i_1} | o_j) \cdot Pr(o_j) = \sum_{j=1}^{M} \left(\frac{o_j}{n} \cdot \sum_{i=1}^{N} \frac{a_{i,j}^2}{n^2} \right) \quad . \tag{8}$$

La probabilité **p** permet d'évaluer la pertinence du modèle par rapport à la fonction physique sans utiliser de distingueur. Plus la campagne est grande, meilleure est l'estimation de **p**.

Plus p est grand, meilleure est l'approximation de(s) fonction(s) physique(s) par le(s) modèle(s).

4.2.3 Outil d'évaluation et de comparaison

Définition 38 : Un **oracle probabiliste** est un oracle qui a une probabilité non nulle de se tromper.

Soit $\Theta_{\mathbf{p}}$ un oracle probabiliste qui, à une observable O_S retourne $P_{m,\hat{k}}$ avec une **probabilité de succès p**. Cet oracle est par construction l'oracle de la partie 4.2.1 avec une probabilité de succès égale à celle définie dans le paragraphe 4.2.2. Il est illustré dans la Figure 18

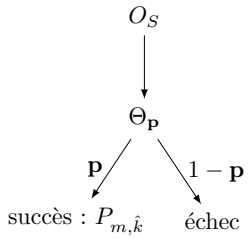


Figure 18 : Oracle avec une probabilité d'erreur **p**

L'oracle $\Theta_{\mathbf{p}}$ combine θ et **p** pour évaluer globalement les deux premières étapes d'une attaque physique, *c.à.d* évaluer une attaque physique avant le distingueur. Il peut aussi être vu comme un schéma de Bernoulli. Pour une campagne de n expériences, la probabilité d'avoir au moins θ succès peut être calculée avec l'équation (9).

$$\mathcal{P} = Pr(\text{obtenir au moins } \theta \text{ succès}) = \sum_{i=\theta}^{n} \binom{n}{i} \mathbf{p}^i \cdot (1 - \mathbf{p})^{n-i} \tag{9}$$

Une autre possibilité pour comparer les attaques est de déterminer combien faut-il d'expériences pour avoir une telle probabilité $\mathcal{P}$ supérieure à $0,99$?

Cette probabilité a deux défauts :

1. Elle ne prend pas en compte le fait qu'une réponse erronée de l'oracle nécessite éventuellement plusieurs réponses correctes pour être compensée.
2. Cet outil est pour une attaque finie, la campagne est fixée. La probabilité varie d'une campagne à une autre surtout, notamment en fonction de sa taille.

La meilleure attaque a la plus grande probabilité $\mathcal{P}$ pour une campagne de taille n.
Une autre manière de comparer est la suivante :
La meilleure attaque est celle qui nécessite le moins d'expériences pour obtenir $\mathcal{P} > 0,99$.

En conclusion, cet outil permet de comparer des attaques foncièrement différentes sur un même circuit. Il permet aussi de mettre en évidence les points faibles d'une attaque. Les différents paramètres de l'attaque sont étudiés séparément.

Étudier une attaque avec notre formalisme se fait alors en trois étapes :

1. Analyser si le chemin d'attaque théorique $\mathcal{R}_m$ est pertinent, en calculant θ.
2. Considérer la pertinence du (des) modèle(s) avec la (les) fonction(s) physique(s) sans se soucier du distingueur utilisé, juste en évaluant $\mathbf{p}$.
3. Pour finir si les deux premières étapes paraissent cohérentes, ajuster l'attaque en jouant sur le distingueur.

4.3 Mises en pratique

Certaines attaques décrites dans ce même chapitre en section 3 ont été réalisées sur un FPGA - Xilinx Spartan3 700A. Pour chacune de ces attaques les mêmes 5000 textes ont été joués avec la même clé : `A3B0D09804584269DC7FB0CDABAD57F8`. La cible est toujours un octet de clé du dernier tour $K_{10}^{l,c} = \hat{k}$.

4.3.1 Réalisation en pratique d'une SCA sur AES

La première attaque est celle de l'exemple de SCA sur le dernier tour de l'AES de la partie 3.1. Le modèle choisi est une distance de Hamming entre le chiffré et la valeur en entrée des s-boxes du dernier tour. Pour réaliser cette attaque le banc EM décrits en chapitre I paragraphe 4.1 est utilisé.

1. Évaluer θ, *c.à.d* le nombre d'appels à Θ_m pour retrouver $\hat{k}$ revient ici à évaluer le nombre de couples $(C, P_{HD,\hat{k}})$ nécessaires pour retrouver $\hat{k}$. Dans ce cas précis, $\theta = 4$; ce résultat provient d'une moyenne effectuée sur un grand nombre de couples.

2. La deuxième étape consiste à estimer $\mathbf{p}$ la probabilité de succès, construite avec les mesures O_R comme expliqué dans le paragraphe 4.2.2.

3. Pour finir le nombre d'expériences n pour avoir une probabilité $\mathcal{P}$ supérieure à $0,99$ est calculé. Ici il s'agit du nombre de mesures d'émission EM, pour avoir une probabilité $\mathcal{P}$ supérieure à $0,99$ d'avoir $\theta = 4$ prédictions $P_{HD,\hat{k}}$ correctes.

Les résultats pour chacun des octets sont reportés dans le Tableau 3.
L'attaque n'a pas fonctionné pour les octets de clé 3 et 13, *c.à.d* $K_{10}^{3,1}$ et $K_{10}^{1,4}$.

octet	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
p	0,22	0,22	0,21	×	0,22	0,22	0,21	0,21	0, 20	0,20	0,21	0,21	0,23	×	0,21	0,22
n	43	43	45	×	43	43	45	45	47	47	45	45	45	×	45	43

Tableau 3 : Résultats d'évaluation de l'attaque de type SCA

4.3.2 Réalisation en pratique d'une DFA sur AES

Cette attaque est une réalisation possible de l'attaque décrite dans l'exemple 3.2. La méthode d'injection de fautes est la méthode de « *clock glitch* » présentée en I paragraphe 4.2. Les modèles choisis sont les $\oplus 2^i$ sur l'entrée du dernier tour.

1. Évaluer θ, *c.à.d* le nombre d'appels à Θ_m pour retrouver $\hat{k}$ revient ici à évaluer le nombre de couples $(C, P_{\oplus 2^i, \hat{k}})$ avec i tiré aléatoirement, pour retrouver $\hat{k}$. Dans ce cas précis, $\theta = 2,24$ que nous arrondissons à 3 car θ doit être entier pour être utilisé dans l'équation (9). Ce résultat a été montré dans [14].

2. La deuxième étape consiste à estimer **p** la probabilité de succès, construite avec les données O_R comme expliqué dans le paragraphe 4.2.2.

3. Pour finir le nombre d'expériences n pour avoir une probabilité $\mathcal{P}$ supérieure à $0,99$ est calculé. Ici il s'agit du nombre d'injections de fautes nécessaires, pour avoir une probabilité supérieure à $0,99$ d'avoir $\theta = 3$ prédictions $P_{\oplus 2^i, \hat{k}}$ correctes.

Les résultats pour chacun des octets sont reportés dans le Tableau 4.

octet	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
p	0,68	0,27	0,35	0,35	0,50	1,00	0,65	0,38	0,69	0,32	0,43	0,68	0,15	0,60	0,74	0,65
n	6	28	21	21	14	3	10	19	10	23	18	6	54	12	9	10

Tableau 4 : Résultats d'évaluation de l'attaque de type FIA

4.3.3 Réalisation en pratique d'une FSA sur AES

Ce paragraphe présente l'attaque FSA introduite en 3.2. Les mesures expérimentales proviennent du même banc d'injection de fautes par « *clock glitch* » présenté en I 4.2 que l'attaque de type DFA.

1. Évaluer θ, *c.à.d* le nombre d'appels à Θ_{HW} pour retrouver $\hat{k}$ revient ici à évaluer le nombre de couples $(C, P_{HW, \hat{k}})$ nécessaires, pour retrouver $\hat{k}$. Dans ce cas précis, $\theta = 4,0$ arrondi à 4.

2. La deuxième étape consiste à estimer **p** la probabilité de succès, construite avec les O_R comme expliqué dans le paragraphe 4.2.2.

3. Pour finir le nombre d'expériences n pour avoir une probabilité $\mathcal{P}$ supérieure à $0,99$ est calculé. Ici il s'agit du nombre d'injections de fautes, pour avoir une probabilité supérieure à $0,99$ d'avoir $\theta = 4$ prédictions $P_{HW, \hat{k}}$ correctes.

Les résultats pour chacun des octets sont reportés dans le Tableau 5.

octet	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
p	0,20	0,20	0,20	0,20	0.21	0,20	0,20	0,20	0,20	0,20	0,20	0,20	0,22	0,21	0,20	0,20
n	48	48	48	48	46	48	48	48	48	48	48	48	43	46	48	48

Tableau 5 : Résultats d'évaluation de l'attaque hybride

4.3.4 Comparaison des 3 attaques

Les attaques peuvent être comparées puisqu'elles ont été menées sur le même circuit. Nous remarquons que si les 5000 textes sont utilisés pour retrouver tous les octets de clé dans l'attaque SCA, ce n'est pas le cas des deux autres attaques. Ainsi, le calcul de **p** est plus précis pour cette première attaque, c'est pourquoi **p** n'est pas choisi comme outil de comparaison.

La comparaison avec n, le nombre d'expériences pour avoir $\mathcal{P} > 0,99$ montre que l'attaque de type injection de fautes (paragraphe 4.3.3) est la meilleure. En effet, en moyenne elle a besoin de moins d'expériences. Nous remarquons également que les octets qui posent problèmes (3 et 13) pour la SCA (paragraphe 4.3.1) ne sont pas les octets les plus difficile à retrouver avec les deux autres attaques. Par ailleurs l'octet 12 a un plus grand n pour la FIA, mais il a le plus petit n dans les deux autres attaques.

Les résultats de la FIA et de la FSA montrent qu'avec un même dispositif d'attaque, les mêmes manipulations, le chemin d'attaque et le modèle changent fondamentalement les résultats. La même campagne peut être traitée de manières très différentes.

5 Bilan et perspectives du formalisme

Dans cette thèse, le formalisme a atteint une partie des objectifs envisagés.

Une description des attaques physiques de manière structurée dans un cadre mathématique a été exposée. La description repose sur la décomposition des attaques en étapes et paramètres.

De plus, en définissant les attaques comme un ensemble de paramètres pouvant être modifiés quasi séparément les uns des autres, il est plus aisé de construire une nouvelle attaque. Dans les chapitres III et V, des attaques issues du formalisme sont présentées.

Par ailleurs, cette décomposition permet d'étudier une attaque en détails et non uniquement dans sa globalité. Elle a permis d'élaborer un outil d'évaluation et de comparaison. Celui-ci permet de comparer des attaques sur un même circuit. En pratique, trois types d'attaque ont été réalisées sur un FGA (section 4.3). L'outil de comparaison peut également permettre de comparer la même attaque sur différents circuits.

En conclusion notre formalisme permet deux types de comparaison :

- Pour une attaque donnée, comparer des circuits et conclure si l'un est plus vulnérable que les autres.
- Pour un circuit donné, tester différentes attaques et conclure à laquelle, il est le plus vulnérable.

Bien évidemment, pour effectuer de telles comparaisons, les tailles de campagne doivent être identiques, car la précision de l'outil de comparaison dépend du nombre d'expériences.

Les perspectives pour le formalisme sont vastes et nombreuses.

La recherche de manière systématique des nouvelles attaques, afin de les anticiper, en combinant les différents paramètres reste à élaborer. Dans cette thèse, la question de la création d'un générateur de toutes les attaques possibles pour un algorithme s'est posée. Un tel générateur permettrait de trouver la meilleure attaque. Malheureusement, nous n'avons pas réussi à programmer un tel générateur. Néanmoins, dans le chapitre IV l'idée est déjà amorcée dans le cas précis des schémas de Feistel et des attaques par injection de fautes.

De plus, le formalisme devrait permettre de revoir la pertinence des contre-mesures existantes. Éventuellement, les contres mesures elles-mêmes pourraient s'écrire sous une même forme.

Pour finir, il serait pertinent d'étendre le formalisme à la cryptographie asymétrique.

Partie III

Attaque par consommation de courant uniquement

Sommaire

1 Introduction

Dans ce chapitre, nous nous sommes intéressés aux attaques par observation de courant uniquement, sur AES, dans le but de retrouver la clé de chiffrement. En décrivant les attaques via le formalisme, il est apparu que dans le couple d'observables (O_S, O_R), O_S est souvent une donnée. L'idée est alors venue d'étudier des attaques où les deux observables sont des mesures. Plus spécifiquement, nous avons choisi des mesures de consommation de courant ou d'émission d'onde EM, à l'entrée du AddRoundKey et à la sortie des s-boxes. Les textes clairs et chiffrés ne sont jamais utilisés. Il s'agit d'attaques à clairs et chiffrés inconnus, seules des mesures sont exploitées. Ainsi l'attaque peut se faire à n'importe quel tour.

Dans ce chapitre, les notations suivantes sont utilisées :
- X désigne la variable aléatoire discrète définie sur un octet en entrée du AddRoundKey, un événement est noté $X = x$ avec $x \in [\![0, 255]\!]$.
- Y désigne la variable aléatoire discrète définie sur un octet en sortie de s-box, un événement est noté $Y = y$ avec $y \in [\![0, 255]\!]$.
- $K_r^{l,c} = \hat{k}$, K désigne la variable aléatoire discrète des hypothèses sur un octet de clé, un événement est noté $K = k$ avec $k \in [\![0, 255]\!]$. K étant fixe lors des différentes expériences, on note $\hat{k}$ la valeur correcte.
- $H_X = HW(X = x)$ désigne la variable aléatoire discrète représentant le poids de Hamming de X, un événement est noté $H_X = h_x$ avec $h_x \in [\![0, 8]\!]$.
- $H_Y = HW(Y = y)$ désigne la variable aléatoire discrète représentant le poids de Hamming de Y, un événement est noté $H_Y = h_y$ avec $h_y \in [\![0, 8]\!]$.
- H'_X désigne la variable aléatoire représentant le poids de Hamming de X estimé (mesuré), un événement est noté $H'_X = h'_x$ avec $h'_x \in [\![0, 8]\!]$ si H'_X est discret, $h'_x \in \mathbb{R}$ si H'_X est continue.
- H'_Y désigne la variable aléatoire représentant le poids de Hamming de Y estimé, un événement est noté $H'_Y = h'_y$ avec $h'_y \in [\![0, 8]\!]$ si H'_Y est discret, $h'_y \in \mathbb{R}$ si H'_Y est continue.

Ce chapitre présente des travaux en cours, qui explorent un domaine peu étudié : des attaques par observation de mesures sans connaissance d'aucune donnée. Il décrit l'enchainement des pistes exploitées durant cette thèse. Certaines ont abouti d'autres non. Ainsi, la méthode la plus pertinente est décrite en paragraphe 5.2.

2 Étude théorique de l'attaque

2.1 Chemin d'attaque

Nous souhaitons réaliser une attaque où les deux observables sont des mesures. Aucune donnée n'est utilisée. Deux points de mesures pertinents pour la consommation de courant (un pour l'observable O_S et un pour l'observable O_R) sont nécessaires à la réalisation de cette attaque. Notre choix est :

- l'instant avant le xor de clé de tour (représenté par la variable X)
- l'instant après la sortie des s-boxes (représenté par la variable$Y$).

Ainsi, l'attaque peut se faire à n'importe quel tour de l'AES à condition de détecter les points d'intérêt.

Définition 39 : Les **points d'intérêt** notés PoI sont les instants où les variables ciblées sont manipulées.

Décrivons l'attaque à l'aide du formalisme comme présenté dans le chapitre précédent :

- $\mathcal{K} = K_r^{l,c}$, la cible est un octet de la clé de n'importe quel tour.
  $k \in [\![0, 255]\!]$. Il y a 256 hypothèses k.
- $O_1 = Conso$, la première observable est une mesure de consommation de courant.
- $O_2 = Conso$, la seconde observable est une mesure de consommation de courant [1].
- Le chemin d'attaque est illustré en Figure 19.
  $\mathcal{R} = f_2 \circ g \circ f_1$, avec :
  $f_1 = conso^{-1}$ est la fonction inverse d'une consommation de courant et $f_2 = conso$ est une consommation de courant et $g = SB \circ \oplus$.
- $\mathcal{R}_m = m_2 \circ g \circ m_1$, avec :
  $m_2 = HW$, poids de Hamming et avec $m_1 = HW^{-1}$ l'inverse du poids de Hamming.
- Le choix du distingueur n'est pas trivial, il est détaillé dans ce chapitre.

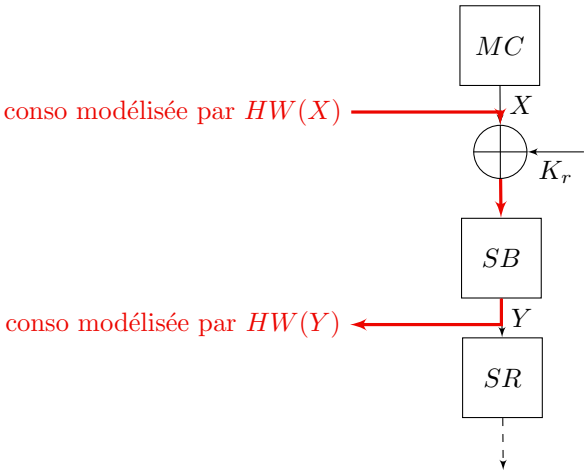


Figure 19 : Chemin d'attaque

1. Il y a bien deux observables, il faut cependant reconnaître que les termes stimulus et réaction ne sont pas adaptés à ce chapitre car ils n'ont pas vraiment de sens, c'est pourquoi nous parlons de permière et seconde observables.

2.2 Analyse du chemin d'attaque théorique.

Avant de réaliser l'attaque, commençons par étudier le pertinence du chemin d'attaque théorique, comme le suggère le formalisme.

Pour cela nous avons besoin de définir la notion de poids de Hamming inverse HW^{-1}. La fonction HW n'étant pas inversible car non injective, $HW^{-1}(h_x)$ représente l'ensemble des antécédents de h_x par HW.

Soit $\mathcal{X}_{h_x}$ l'ensemble des antécédents de h_x pour la fonction HW.

$$\mathcal{X}_{h_x} = \{x / HW(x) = h_x\}$$

Le cardinal de cet ensemble dépend de la valeur de x comme montré dans le Tableau 6.

h_x	0	1	2	3	4	5	6	7	8
$\#(\mathcal{X}_{h_x})$	1	8	28	56	70	56	28	8	1

Tableau 6 : Le cardinal de $\mathcal{X}_{h_x}$ selon la valeur du poids de Hamming observé h_x.

Ceci étant précisé, revenons à l'attaque théorique en soit.

Le chemin d'attaque théorique est le suivant :

$$\mathcal{R}_m(k, h_x) = HW\left(SB\left(k \oplus HW^{-1}(h_x)\right)\right) \quad . \tag{10}$$

Il est particulier car il ne définit pas une fonction mais un ensemble.

Pour la bonne hypothèse de clé $\hat{k}$, l'équation suivante est vérifiée :

$$SB(\hat{k} \oplus x) = y \quad .$$

De plus l'attaquant ne connaît que des réalisations $H_X = h_x$ et $H_Y = h_y$.

L'ensemble des hypothèses k est $\mathbb{K} = [\![0, 255]\!]$. Pour chaque couple (h_x, h_y) seulement un sous-ensemble d'hypothèses k est valide.

Soit $\mathbb{K}_{h_x,h_y}$ le sous-ensemble d'hypothèses de clé défini comme suit :

$$\mathbb{K}_{h_x,h_y} = \{k / \exists x \in \mathcal{X}_{h_x} / HW\left(SB\left(k \oplus x\right)\right) = h_y\} \quad .$$

Soit $\mathbb{K}_{\{(h_x,h_y)_i\}_{i\in[\![1,n]\!]}}$ l'intersection de n ensembles $\mathbb{K}_{h_x,h_y}$ construits avec n couples $(h_x, h_y)_i$:

$$\mathbb{K}_{\{(h_x,h_y)_i\}_{i\in[\![1,n]\!]}} = \bigcap_{i=1}^{n} \mathbb{K}_{(h_x,h_y)_i} \quad .$$

La bonne hypothèse $\hat{k}$ est incluse dans $\mathbb{K}_{\{(h_x,h_y)_i\}_{i\in[\![1,n]\!]}}$. La première idée est alors,d'utiliser un crible pour discriminer les mauvaises hypothèses de clés. Cependant, cet outil est limité car il n'élimine pas toutes les mauvaises hypothèses.

Une mauvaise hypothèse k' peut ne pas être discriminée si $\forall (h_x, h_y)$ générés par k'

$$\exists x, x' \in \mathcal{X}_{h_x} \ c.à.d \ HW(x) = HW(x') = h_x \quad .$$

tel que :

$$HW(SB(\hat{k} \oplus x)) = HW(SB(k' \oplus x')) = h_y \quad .$$

Si $\mathbb{K}_{\{(h_x,h_y)_i\}_{i\in[\![1,n]\!]}}$ est construit pour tous les couples possibles (n est au moins égal à 256) alors, il existe des cas où :

$$\mathbb{K}_{\{(h_x,h_y)_i\}_{i\in[\![1,n]\!]}} = \{\hat{k}, k'\} \text{ avec } k' \neq \hat{k} \quad .$$

Il reste une mauvaise hypothèse de clé k' qui n'a pas été discriminée.

Concrètement, les hypothèses de clé erronées non discriminées sont $\{25, 55, 88, 167, 200, 230\}$. Par exemple, il y a le cas où :

$$\mathbb{K}_{\{(h_x,h_y)_i\}_{i\in[\![1,n]\!]}} = \{25, 62\} \quad .$$

Or si $\hat{k} = 25$, et que toutes les valeurs de x possibles ont servi à construire $\mathbb{K}_{\{(h_x,h_y)_i\}_{i\in[\![1,n]\!]}}$, alors :

$$\mathbb{K}_{\{(h_x,h_y)_i\}_{i\in[\![1,n]\!]}} = \{25\} \quad .$$

Nous pouvons en déduire que si nous sommes convaincus que $\mathbb{K}_{\{(h_x,h_y)_i\}_{i\in[\![1,n]\!]}}$ a été construit avec toutes les valeurs de x et que le crible retourne :

$$\mathbb{K}_{\{(h_x,h_y)_i\}_{i\in[\![1,n]\!]}} = \{25, 62\}$$

alors :

$$\hat{k} = 62.$$

Finalement, le défaut du crible est corrigé. Autrement dit l'ensemble des couples des poids de Hamming possibles $\{(h_x, h_y)_i\}$ est différent pour chacune des 256 hypothèses k.

Le chemin d'attaque théorique permet bien de retrouver la cible. L'attaque est donc théoriquement réalisable.

3 Les difficultés liées à la pratique

Dans le paragraphe précédent 2.2 la pertinence du chemin d'attaque théorique a été démontrée. Dans cette partie, nous introduisons toute la difficulté de passer de cette attaque théorique à l'attaque en pratique.

3.1 Estimation du poids de Hamming : les approches qui n'ont pas fonctionné

Pour commencer les approches qui n'ont pas abouti sont présentées. Elles permettent de bien se familiariser avec les problèmes. L'acquisition de courbes EM sur STM32 et ATmega a été menée sur le banc décrit en I 4.1.

3.1.1 Génération de poids de Hamming de manière exhaustive

La première idée a été de générer tous les vecteurs de poids de Hamming possibles de manière exhaustive, puis de faire la corrélation avec les consommations de courant. Nous rappelons qu'il s'agit d'une attaque diviser pour régner, ainsi nous n'attaquons qu'un seul octet à la fois. Le poids de Hamming sur cet octet varie entre 0 et 8, soit 9 valeurs. Ainsi pour une campagnes de n textes il y a 9^n vecteurs de prédiction de poids de Hamming à générer.

- Pour 1 texte il y a 9 valeurs à tester : $0, 1, 2, 3, 4, 5, 6, 7, 8$.
- Pour 2 textes il y a 81 vecteurs de taille 2 à tester : $(0,0), (0,1), (0,2), \ldots, (8,7), (8,8)$.
- Pour 5 textes il y a $9^5 = 59049$ vecteurs, c'est encore possible à générer.
- Pour 10 textes il y a $2^{31} < 9^{10} < 2^{32}$ vecteurs à générer, ce qui est déjà très conséquent.

Pour 10 textes, la corrélation entre un vecteur de poids de Hamming aléatoire et le bon vecteur de poids de Hamming n'est pas efficace comme le montre la Figure 20.

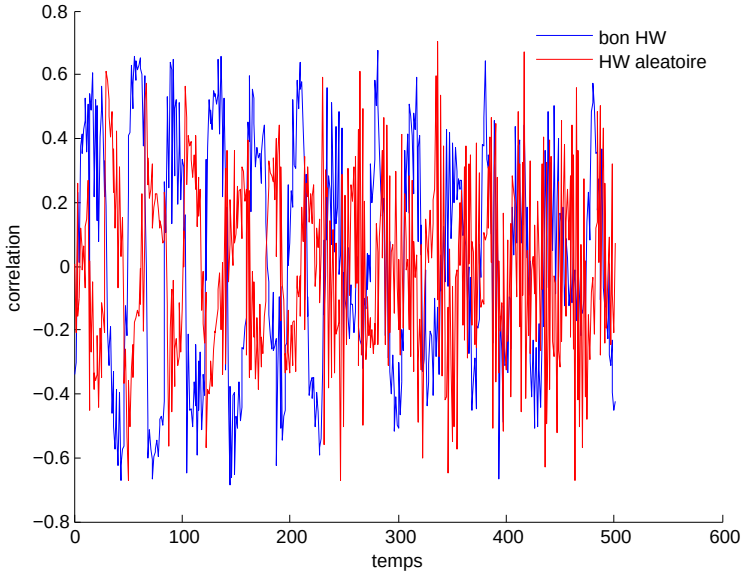


Figure 20 : Comparaison de la corrélation entre : les mesures et un vecteur de poids de Hamming aléatoire et les mesures et le bon vecteur de poids de Hamming pour une campagne de 10 textes. Les mesures sont issues d'un STM32.

En effet, 100 textes est un minimum raisonnable pour distinguer une différence entre un vecteur aléatoire et le vecteur correct (voir Figure 21). Faire une recherche exhaustive sur la clé de l'AES a une complexité de 2^{128} qui est inférieure à celle de la génération des $9^{100} \simeq 2^{317}$ vecteurs.

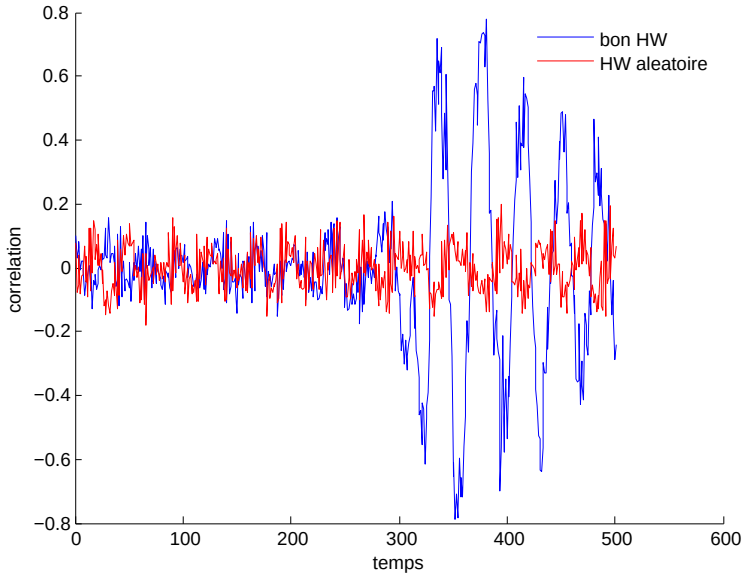


Figure 21 : Comparaison de la corrélation entre un vecteur de poids de Hamming aléatoire et le vecteur correct de poids de Hamming pour une campagne de 100 textes. Les mesures sont issues d'un STM32

3.1.2 Utilisation de l'information mutuelle

Le paragraphe précédent montre qu'il est impossible de générer tous les vecteurs de poids de Hamming de manière exhaustive dés qu'une campagne dépasse 10 textes. Ce qui est un problème puisque la corrélation n'est pertinente que pour une campagne 10 fois plus grande.

Une propriété de l'information mutuelle voir équation (6), est qu'elle joue sur la fréquence d'apparition des valeurs et non les valeurs elles-mêmes. Autrement dit, pour z et z' deux variables aléatoires ; l'information reste la même si z est remplacé par une bijection F de z :

$$I(z, z') = I(F(z), z')$$

Pour $n = 3$: les vecteurs de poids de Hamming $(0,1,0), (0,2,0), \cdots (0,8,0), \cdots (8,0,8), \cdots, (8,7,8)$ ont la même information mutuelle et donc le vecteur $(0,1,0)$ suffit à les représenter. L'idée est de générer tous les vecteurs de motifs différents et non de poids de Hamming différents. Cela réduit notre espace de génération. Cependant, pour 100 textes, ce n'est toujours pas suffisant. Nous ne retirons qu'un facteur $9! = 362880$ au 9^{100} vecteurs.

De plus, il peut tester jusqu'à 9! vecteurs à corréler une fois le vecteur de motifs correct obtenu, en supposant qu'il n'y a pas d'erreur dans ce dernier.

3.1.3 Génération par algorithme évolutionnaire

La dernière approche qui a été tentée est de rechercher le vecteur de poids de Hamming de manière aléatoire, en s'inspirant des algorithmes génétiques. Pour une campagne de taille n, un vecteur de poids de Hamming de taille n est généré aléatoirement. La corrélation avec les mesures est alors calculée. Puis, une des valeurs du vecteur est changée aléatoirement, la nouvelle corrélation est calculée. Si celle-ci est meilleure, le nouveau vecteur est gardé, sinon il faut reprendre le vecteur précédent. Un tel algorithme nécessite une condition d'arrêt, le dépassement d'une certaine valeur de corrélation est choisie. Cette approche n'a pas abouti. En effet, l'algorithme retourne systématiquement un vecteur de poids Hamming qui n'est pas le bon et qui corrèle bien mieux que celui qui est correct.

3.2 Les limites du crible et du compteur

L'estimation du poids de Hamming n'est pas chose aisée, comme le montre le paragraphe précédent (3.1). C'est pourquoi, nous supposons qu'il peut y avoir des erreurs d'estimation. Dans le cas où il y a des erreurs dans l'estimation des couples de poids de Hamming (H_X, H_Y), l'utilisation d'un crible ne peut plus fonctionner.

Pour une campagne de taille n, si n est suffisamment grand toutes les valeurs d'hypothèses de clé sont éliminées, en reprenant les notations du paragraphe 2.2 cela s'exprime ainsi :

$$\mathbb{K}_{\{(h_x,h_y)_i\}_{i\in[\![1,n]\!]}} = \varnothing \quad .$$

La première idée est d'utiliser un compteur à la place du crible. Cela consiste à associer un compteur c_k, initialisé à 0, pour chaque hypothèse $k \in [\![0, 255]\!]$. Puis à chaque exécution, si l'hypothèse k est cohérente avec l'observation $(h_x = h_y)$, *c.à.d* $k \in \mathbb{K}_{h_x,h_y}$, alors c_k est incrémenté.

À la fin c_k est le nombre de fois où k est suggérée comme une hypothèse de clé valide. Le compteur est normalisé par le nombre n d'exécutions. Le vecteur des $\frac{c_k}{n}$ est noté U. Malheureusement, intrinsèquement quelques valeurs de clés sont moins souvent de bonnes candidates que d'autres (voir paragraphe 2.2). Par conséquent, le compteur ne peut pas être utilisé tel quel, il faut l'adapter ou utiliser un autre distingueur.

3.3 Problématiques

Tout le questionnement de notre étude réside dans le passage de la consommation de courant mesurée au poids de Hamming. L'estimation du poids de Hamming présente deux difficultés dans ce contexte.

1. La détection des points d'intérêt est nécessaire. En effet, le paragraphe 3.1 montre qu'à chaque instant il existe une corrélation possible entre la consommation de courant et le poids de Hamming. Il est indispensable pour l'attaquant d'être capable de déterminer les *PoI*, *c.à.d* à quel instant est la fuite de sécurité du circuit.

2. Une fois les *PoI* ciblés, il reste encore la question de l'inférence des poids de Hamming à partir des mesures. Le paragraphe 3.1 a montré qu'il n'est pas possible de tous les générer. De plus, le vecteur de poids qui corrèle le mieux est rarement le bon.

La difficulté de retrouver le poids de Hamming à partir de mesures de consommation de courant ou d'émission EM, sans connaissance sur les données (textes) est évidente. Il faut admettre que la méthode à employer ne peut être parfaite. Il n'est pas possible de retrouver les poids de Hamming de manière complètement fiable. Le crible n'est alors pas un bon candidat au distingueur, il reste à en sélectionner un efficace.

4 Des courbes de mesures au poids de Hamming

4.1 Détection des *PoI*

Une méthode performante pour détecter les *PoI* est celle présentée dans [66]. Pour chaque point $conso_t$ de la courbe, associé à un instant t, les auteurs calculent le NICV (Nomalized Inter-Class Variance) en fonction du texte T.

$$\text{NICV} = \frac{Var(\mathbb{E}(conso_t|T))}{Var(conso_t)} \quad .$$

Le point ayant le plus grand NICV est le *PoI*. Cette méthode nécessite la connaissance des textes.

Le problème de trouver les *PoI* sans aucun texte a déjà été abordé dans la littérature dans [67]. L'idée est de calculer la variance en chaque point de mesure, les points ayant une variance suffisamment élevée sont des *PoI*.

L'inconvénient majeur de cette méthode est qu'elle est « aveugle » à la signification de la variable manipulée en ce point. Elle mesure une variance élevée mais pouvant être en rapport avec n'importe quoi comme données, y compris le bruit et non celles que nous souhaitons manipulées. Ainsi, cette méthode n'a pas fonctionné sur les courbes que nous avons obtenues.

4.2 Estimation des poids de Hamming

4.2.1 Approche par templates

Définition 40 : Une attaque par **templates** est une attaque de type SCA, où l'attaquant dispose d'un double du circuit attaqué, sur lequel il a une maitrise totale. Dans un premier temps il génère un ensemble de mesures sur ce circuit en faisant varier les différentes valeurs possibles de la cible. Ces mesures sont traitées de manières statistiques, afin d'obtenir une « mesure type », témoin, appelée **template** du comportement du circuit selon les hypothèses de cible.

D'un point de vue formalisme, l'attaquant crée en quelque sorte, lui-même ses prédictions $P_{m,k}$ et sa fonction modèle m. Puis, il confronte les mesures sur le second circuit (le circuit cible) avec celles de son modèle témoin, afin d'extraire la bonne hypothèse de cible, via bien évidemment le distingueur de son choix.

Les méthodes par templates [68] sont très efficaces et permettent à la fois de détecter les *PoI* et en même temps de passer des mesures aux poids de Hamming. Ainsi, dans [69] les auteurs retrouvent le poids de Hamming des données manipulées avec une probabilité de $0,993$. Le défaut est que celles-ci nécessitent de posséder un second circuit sur lequel les textes sont connus, pour construire les templates.

4.2.2 Méthodes par classement

Supposons que les *PoI* ont été détectés correctement. L'attaquant dispose alors de deux vecteurs de mesures de courant :

$$(\mathcal{C}'_X(x_1), \cdots, \mathcal{C}'_X(x_n)) \text{ et } (\mathcal{C}'_Y(y_1), \cdots, \mathcal{C}'_Y(y_n)) \,.$$

L'objectif est de retrouver les vecteurs des poids de Hamming correspondant à ces vecteurs de mesures :

$$(h_{x_1}, \cdots, h_{x_n}) \text{ et } (h_{y_1}, \cdots, h_{y_n}) \quad .$$

Si la campagne est assez grande, et sous l'hypothèse de manipuler des données construites avec des textes aléatoires et uniformément distribués, nous pouvons classer les mesures pour en déduire les poids de Hamming. Deux méthodes de classement sont proposées.

1. Premier classement.
   En supposant que la plus petite valeur de courant corresponde au poids de Hamming 0. Pour conserver la distribution des poids de Hamming (voire Tableau 6, en paragraphe 2.2), les $\frac{n}{256}$ plus petites valeurs sont associées au poids de Hamming 0, les $\frac{8 \cdot n}{256}$ valeurs suivantes correspondent au poids de Hamming 1 etc.

2. Second classement.
   Supposons que l'ensemble des mesures (pour tous les poids de Hamming confondus) suit une loi normale $\mathcal{N}(\mu, \sigma")$. Les valeurs de la moyenne μ et de l'écart type $\sigma"$ sont estimées à partir des réalisations mesurées. Des intervalles sont alors construits pour retrouver les poids de Hamming. Les poids de Hamming sont alors assignés suivant les règles :

$$
\begin{array}{llcll}
- & a & \leqslant \frac{\mathcal{C}-\mu}{\sigma" \cdot \sqrt{2}} < & - \operatorname{erf}^{-1}\left(\frac{254}{256}\right) & \Rightarrow h = 0 \\
- & \operatorname{erf}^{-1}\left(\frac{254}{256}\right) & \leqslant \frac{\mathcal{C}-\mu}{\sigma" \cdot \sqrt{2}} < & - \operatorname{erf}^{-1}\left(\frac{238}{256}\right) & \Rightarrow h = 1 \\
- & \operatorname{erf}^{-1}\left(\frac{238}{256}\right) & \leqslant \frac{\mathcal{C}-\mu}{\sigma" \cdot \sqrt{2}} < & - \operatorname{erf}^{-1}\left(\frac{180}{256}\right) & \Rightarrow h = 2 \\
- & \operatorname{erf}^{-1}\left(\frac{180}{256}\right) & \leqslant \frac{\mathcal{C}-\mu}{\sigma" \cdot \sqrt{2}} < & - \operatorname{erf}^{-1}\left(\frac{70}{256}\right) & \Rightarrow h = 3 \\
- & \operatorname{erf}^{-1}\left(\frac{70}{256}\right) & \leqslant \frac{\mathcal{C}-\mu}{\sigma" \cdot \sqrt{2}} \leqslant & \operatorname{erf}^{-1}\left(\frac{70}{256}\right) & \Rightarrow h = 4 \\
 & \operatorname{erf}^{-1}\left(\frac{70}{256}\right) & < \frac{\mathcal{C}-\mu}{\sigma" \cdot \sqrt{2}} \leqslant & \operatorname{erf}^{-1}\left(\frac{180}{256}\right) & \Rightarrow h = 5 \\
 & \operatorname{erf}^{-1}\left(\frac{180}{256}\right) & < \frac{\mathcal{C}-\mu}{\sigma" \cdot \sqrt{2}} \leqslant & \operatorname{erf}^{-1}\left(\frac{238}{256}\right) & \Rightarrow h = 6 \\
 & \operatorname{erf}^{-1}\left(\frac{238}{256}\right) & < \frac{\mathcal{C}-\mu}{\sigma" \cdot \sqrt{2}} \leqslant & \operatorname{erf}^{-1}\left(\frac{254}{256}\right) & \Rightarrow h = 7 \\
 & \operatorname{erf}^{-1}\left(\frac{254}{256}\right) & < \frac{\mathcal{C}-\mu}{\sigma" \cdot \sqrt{2}} \leqslant & a & \Rightarrow h = 8
\end{array}
$$

   Avec erf la fonction d'erreur de Gauss définie par :

$$\operatorname{erf}(z) = \frac{2}{\sqrt{\pi}} \int_0^z e^{-t^2} \, \mathrm{d}t$$

   Nous pouvons également exclure les données extrêmes et donc improbables ; celles qui dépassent un seuil défini par a positif. Dans un cas parfait où il n'y a pas d'erreur de mesure $a = \infty$. En pratique, a est choisi empiriquement en fonction de nos mesures.

Des résultats expérimentaux sur des courbes de simulation sont présentés en 5.3.1.

4.2.3 Approche utilisant un modèle de la consommation de courant

Pour commencer, nous définissons un modèle de la consommation de courant classique. La consommation dite parfaite et notée $\mathcal{C}$ est issue du modèle suivant :

$$\mathcal{C}(X) = \alpha \cdot H_X + \beta \quad .$$

La consommation « mesurée », notée $\mathcal{C}'$, est égale à la consommation parfaite avec un bruit de mesure $W_X \in \mathbb{R}$ qui suit une loi normale $\mathcal{N}(0, \sigma')$.

$$\begin{aligned} \mathcal{C}'(X) &= \mathcal{C}(X) + W_X \\ \mathcal{C}'(X) &= \alpha \cdot H_X + \beta + W_X \\ \mathcal{C}'(X) &= \alpha \cdot H'_X + \beta \end{aligned}$$

Une méthode classique pour déterminer α et β est la régression linéaire. Cette méthode suppose dans notre cas, la connaissance de H_X et $\mathcal{C}'(X)$. Comme pour les attaques en templates, cette méthode impose à l'attaquant de posséder un circuit témoin.

Nous proposons une autre approche qui consiste à transformer, à l'aide d'une application affine l'intervalle des mesure de $\mathcal{C}'$ en l'intervalle $]0 - \sigma"; 8 + \sigma"[$; où $\sigma"$ est l'écart type de $\mathcal{C}'$. Cette méthode ne requiert pas de circuit témoin, mais l'estimation obtenue de H'_X est moins précise.

Avec la connaissance de α et β alors nous pouvons obtenir H'_X. En effet, sachant que :

$$\alpha \cdot H_X + \beta + W_X = \alpha \cdot H'_X + \beta \quad .$$

nous avons :

$$H'_X = H_X + W_{H_X}$$

W_{H_X} suit la loi normale $\mathcal{N}(0, \frac{\sigma'}{|\alpha|})$ d'après la propriété d'invariance par dilatation de l'écart type (voir P8 en annexe C.ii.c). Par la suite, elle est notée $\mathcal{N}(0, \sigma)$.

5 Les propositions d'attaques

5.1 Approche par acceptation de l'erreur

Cette première approche suppose que l'attaquant a pu évaluer des poids de Hamming (par la méthode des templates 4.2.1 ou des classements 4.2.2), ces derniers pouvant être erronés.

L'approche présentée maintenant, étudie le comportement des différentes hypothèses k avec des variables aléatoires.

Deux nouvelles variables aléatoires uniformément distribuées et **indépendantes** sont introduites : X" et Y" dans $[\![0, 255]\!]$. Nous définissons également :

$$H_X" = HW(X") \text{ et } H_Y" = HW(Y") \quad .$$

Les variables H_X" et H_Y" sont construites pour représenter une erreur, elles n'ont pas de lien avec les données manipulées par l'AES. Dans ce cas, le vecteur des c_k est calculé avec un simulateur pour un grand nombre n de couples $(X", Y")$, ici $n = 100000$. Toutes les hypothèses k n'ont pas la même probabilité d'être dans $\mathbb{K}_{\{(h_x", h_y")_i\}_{i\in[\![1,n]\!]}}$.

$$\mathbb{K}_{\{(h_x", h_y")_i\}_{i\in[\![1,n]\!]}} = \bigcap_{i=1}^{n} \mathbb{K}_{(h_x", h_y")_i}$$

Les variables X" et Y" sont uniformément aléatoires, donc H_X"et H_Y" suivent la distribution habituelle du poids de Hamming. Dans ce contexte, le vecteur résultat qui stocke les 256 valeurs de compteur normalisées $\frac{c_k}{n}$ est noté U".

D'autre part, pour toutes les hypothèses de clé k, pour chaque valeur x de X, $x \in [\![0, 255]\!]$, Y est fixé à un y, par la relation suivante :

$$y = SB(x \oplus k)$$

Dans ce contexte pour chacune des hypothèses de clé k, nous construisons un vecteur U_k, qui stocke les 256 valeurs de compteur $\frac{c_k}{n}$.

Nous remarquons que $U_k \neq U"$.

Un couple de poids de Hamming a une probabilité d'être correcte de p_{hw}.

$$P((H_X, H_Y) = (H'_X, H'_Y)) = p_{hw}$$

En pratique nous avons :

$$p_{hw} = p_{hx} \cdot p_{hy} \quad .$$

avec p_{hx} la probabilité que le poids de Hamming H_X soit correctement évalué et p_{hy} celle de H_Y. Il y a deux cas particuliers :

- Si les poids de Hamming sont connus ($p_{hw} = 1$), $U = U_k$.
- Si les poids de Hamming sont aléatoires ($p_{hw} = 0$), $U = U"$.

Finalement pour s'approcher d'un cas plus réaliste, un vecteur de prédiction Up_{hw} peut être

calculé avec l'équation suivante :

$$Up_{hw} = p_{hw} \cdot U_k + (1 - p_{hw})U" \quad .$$

Pour chacune des hypothèses k la corrélation entre U et Up_{hw} est calculée. U étant le vecteur de compteur défini en paragraphe 3.2. L'hypothèse retournée est celle ayant la plus grande corrélation.

5.2 Approche par inférence bayésienne

L'idée de d'étudier des lois jointes comme fonction de la clé, est abordée dans les attaques algébriques [9, 69, 70, 71] ou encore dans la thèse de Linge [72].

Définition 41 : L'inférence bayésienne est une méthode d'inférence permettant de déduire la probabilité d'un événement à partir de celles d'autres événements déjà évalués. Elle s'appuie principalement sur le théorème de Bayes.

5.2.1 Principe et objectif

Dans cette partie, une approche probabiliste est envisagée. Une approche similaire pour les templates est présenté en section 6.6 de [73]. L'idée n'est plus de corriger des poids de Hamming potentiellement faux ; mais d'associer une probabilité à chaque hypothèse de clé selon des couples de mesures (H'_X, H'_Y) obtenus, comme décrit en paragraphe 4.2.3. Ce qui nous intéresse est la probabilité de chaque hypothèse de clé sachant un ensemble de mesures :

$$A = Pr\left(K = k | \{(H'_X, H'_Y) = (h'_x, h'_y)_i\}_{i \in [\![1,n]\!]}\right) \quad . \tag{11}$$

La Figure 22 représente le contexte sous forme de réseau de croyances (comme décrit dans [74]). Il s'agit d'un graphe où chaque nœud est une variable. L'influence d'une variable aléatoire sur une autre est symbolisée par une flèche. Les variables incluses dans le rectangle changent à chaque exécution, celle située à l'extérieur reste fixe.

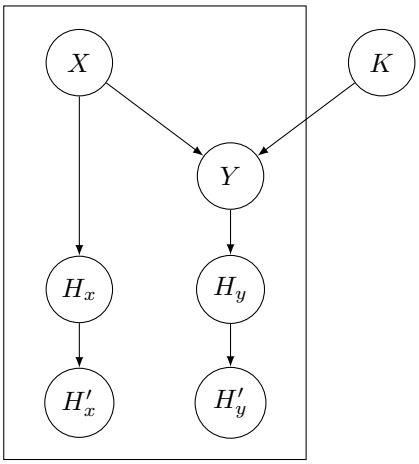


Figure 22 : Modélisation du problème par un graphe

Des rappels sur les outils de probabilité utilisés dans cette section sont en annexe C.ii.

5.2.2 Calcul des probabilités

Dans toute cette section F_Z est la fonction de densité de probabilité de la variable Z.

a) Calculs préliminaires

A priori toutes les hypothèses de clé doivent être équiprobables.

$$Pr(K = k) = \frac{1}{256} \tag{12}$$

Les probabilités : $Pr\left((H_X, H_Y) = (h_x, h_y)|K = k\right)$ et $Pr\left(K = k|(H_X, H_Y) = (h_x, h_y)\right)$ se calculent très facilement par dénombrement sur les valeurs de X.

On considère que le bruit en X suit la loi Normale $\mathcal{N}(0, \sigma_X)$; et ainsi nous avons la fonction de densité de probabilité suivante :

$$F_{B_X}(z) = \frac{e^{-\frac{1}{2}\cdot\left(\frac{z}{\sigma_X}\right)^2}}{\sigma_X \cdot \sqrt{2\pi}} \quad .$$

De même, on considère que le bruit en Y suit la loi Normale $\mathcal{N}(0, \sigma_Y)$. Ainsi, nous avons la fonction de densité de probabilité suivante :

$$F_{B_Y}(z) = \frac{e^{-\frac{1}{2}\cdot\left(\frac{z}{\sigma_Y}\right)^2}}{\sigma_Y \cdot \sqrt{2\pi}} \quad .$$

b) Calcul des probabilités des différentes hypothèses de clé.

Nous cherchons à estimer la probabilité A, voir équation (11), que $K = k$ sachant un ensemble de mesures. La formule de Bayes, rappelée en annexe C.ii équation (39), donne la relation suivante :

$$A = \frac{\overbrace{F_{(H'_X,H'_Y)}\left(\{(h'_x, h'_y)_i\}|K = k\right)}^{A_1} \cdot Pr(K = k)}{\underbrace{F_{(H'_X,H'_Y)}\left(\{(h'_x, h'_y)_i\}\right)}_{A_0}} \quad .$$

Le dénominateur A_0 est obtenu par normalisation, donc n'a pas besoin d'être calculé.

Intéressons nous au calcul de A_1. Les mesures de couples $(H'_X, H'_Y)_i$ sont des variables indépendantes et identiquement distribuées. C'est à dire que tous les couples ont la même loi de probabilité et tous les couples sont mutuellement indépendants (voir Définition 50 en annexe C.ii). En d'autres termes un couple $(H'_X, H'_Y)_1$ ne permet pas de prédire le couple suivant $(H'_X, H'_Y)_2$. C'est pourquoi, nous avons :

$$A_1 = \prod_{i=1}^{n} F_{(H'_X,H'_Y)}\left((h'_x, h'_y)_i|K = k\right) \quad .$$

Maintenant, il faut calculer la même probabilité pour un unique couple d'observables.

$$A_2 = F_{(H'_X,H'_Y)}\left((h'_x, h'_y)|K = k\right) \quad .$$

L'application de la formule des probabilités totales (34) donne :

$$A_2 = \sum_{i \in [\![0,8]\!]^2} \underbrace{F_{(H'_X,H'_Y)}\left((h'_x,h'_y)_i | (H_X,H_Y) = (h_x,h_y)_i\right)}_{A_3} \cdot Pr((H_X,H_Y) = (h_x,h_y)_i | K = k) \quad .$$

Intéressons nous maintenant au calcul de A_3.

$$A_3 = F_{(H'_X,H'_Y)}\left((h'_x,h'_y) | (H_X,H_Y) = (h_x,h_y)\right)$$

Sachant (H_X, H_Y), H'_X et H'_Y sont indépendants. Pour h_x fixé H'_X est indépendant de H_Y donc :

$$F_{H'_X}\left(h'_x | (H_X,H_Y) = (h_x,h_y)\right) = F_{H'_X}\left(h'_x | H_X = h_x\right) \quad .$$

De même, pour h_y fixé H'_Y est indépendant de H_X donc :

$$F_{H'_Y}\left(h'_y | (H_X,H_Y) = (h_x,h_y)\right) = F_{H'_Y}\left(h'_y | H_Y = h_y\right) \quad .$$

Ceci implique :

$$A_3 = F_{H'_X}\left(h'_x | H_X = h_x\right) \cdot F_{H'_Y}\left(h'_y | H_Y = h_y\right) \quad .$$

Or $Pr(h'_x | H_X = h_x)$ correspond à un bruit centré en h_x d'où :

$$F_{H'_X}\left(h'_x | H_X = h_x\right) = F_{B_X}(|h'_x - h_x|) = \frac{e^{-\frac{1}{2}\cdot\left(\frac{h'_x - h_x}{\sigma_X}\right)^2}}{\sigma_X \cdot \sqrt{2\pi}} \quad .$$

De même :

$$F_{H'_Y}\left(h'_y | H_Y = h_y\right) = F_{B_Y}(h'_y - h_y) = \frac{e^{-\frac{1}{2}\cdot\left(\frac{h'_y - h_y}{\sigma_Y}\right)^2}}{\sigma_Y \cdot \sqrt{2\pi}} \quad .$$

Finalement, nous avons A proportionnel au produit de la somme suivant :

$$A \propto \prod_{(h'_x,h'_y)_i}^{n} \sum_{(h_x,h_y) \in [\![0,8]\!]^2} \frac{e^{-\frac{1}{2}\cdot\left(\frac{h'_x - h_x}{\sigma_X}\right)^2}}{\sigma_X \cdot \sqrt{2\pi}} \cdot \frac{e^{-\frac{1}{2}\cdot\left(\frac{h'_y - h_y}{\sigma_Y}\right)^2}}{\sigma_Y \cdot \sqrt{2\pi}} \cdot Pr((H_X,H_Y) = (h_x,h_y) | K = k) \quad (13)$$

Cette approche par inférence Bayésienne peut être utilisée dans la détection de *PoI* voir annexe D.

5.3 Résultats et comparaison des deux approches

Des simulations ont été faites dans le langage de programmation Julia. Des textes ont été générés aléatoirement. Les poids de Hamming des points d'intérêt ont été calculés et un bruit suivant la loi normale $\mathcal{N}(0, \sigma)$ a été appliqué à chaque poids. Nos résultats sont en fonction de l'écart type σ et n la taille de la campagne (nombre de textes). Nous précisons que σ correspond ici à σ_X et σ_Y. La Figure 23 illustre les différentes densités de probabilité des poids de Hamming bruités simulés, pour différents écart types.

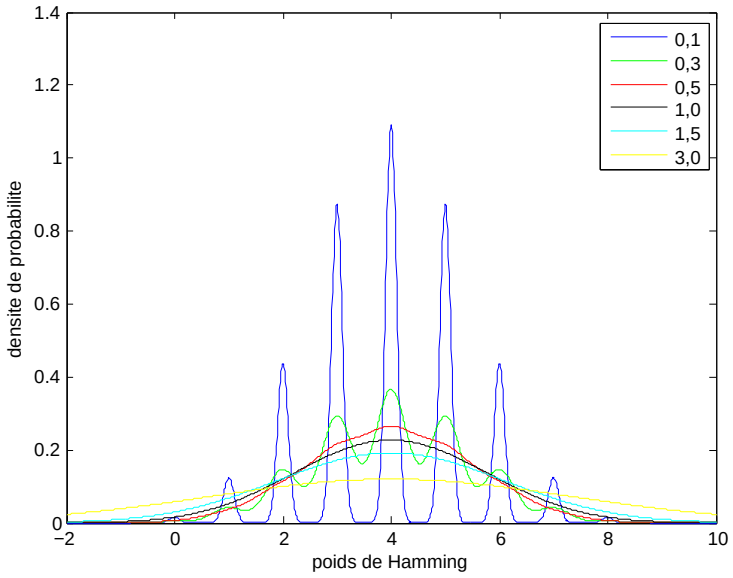


Figure 23 : Densités de probabilité des poids de Hamming bruités en fonction des différentes valeurs d'écart type σ. ($\sigma \in \{0,1; 0,2; 0,3; 0,5; 1; 1,5; 2; 3\}$)

5.3.1 Résultats approche par acceptation

La méthode par acceptation de l'erreur suppose l'obtention de poids de Hamming plus ou moins erronés. Nous commençons par comparer les deux méthodes de classement décrites en section 4.2.2. Les tableaux 7 et 8 présentent des résultats moyens pour 10 attaques. Plus le nombre de textes augmente, plus nous nous approchons du cas optimal (cas où la campagne est constituée d'une infinité de textes).

$n \setminus \sigma$	0,1	0,2	0,3	0,5	1,0	1,5	2,0	3,0
100	0,74	0,71	0,76	0,59	0,4	0,33	0,27	0,25
500	0,9	0,91	0,85	0,68	0,43	0,34	0,29	0,25
1000	0,94	0,94	0,88	0,69	0,44	0,35	0,3	0,25
5000	0,98	0,97	0,91	0,71	0,45	0,35	0,3	0,26
10000	0,98	0,98	0,91	0,71	0,44	0,35	0,3	0,26
50000	0,99	0,99	0,91	0,71	0,45	0,35	0,3	0,26
100000	0,99	0,99	0,91	0,71	0,45	0,35	0,3	0,26

Tableau 7 : Probabilités en moyenne d'avoir trouvé le bon poids de Hamming par la méthode du premier classement

$n \setminus \sigma$	0,1	0,2	0,3	0,5	1,0	1,5	2,0	3,0
100	1	0,94	0,85	0,66	0,43	0,35	0,3	0,27
500	1	0,98	0,89	0,7	0,44	0,35	0,29	0,26
1000	1	0,98	0,9	0,7	0,44	0,35	0,3	0,25
5000	1	0,99	0,91	0,71	0,45	0,35	0,3	0,26
10000	1	0,99	0,91	0,7	0,44	0,35	0,3	0,26
50000	1	0,99	0,91	0,71	0,45	0,35	0,3	0,26
100000	1	0,99	0,91	0,71	0,44	0,35	0,3	0,26

Tableau 8 : Probabilités en moyenne d'avoir trouvé le bon poids de Hamming par la méthode du second classement

Nous constatons que la méthode du second classement est la plus performante. C'est pourquoi, c'est celle que nous avons retenue pour la suite de l'analyse.

Le Tableau 9 présente des résultats pour un octet de clé fixé arbitrairement à la valeur 42, en faisant varier le paramètre σ et n la taille de la campagne (nombre de textes). Il indique le rang moyen dans le classement des hypothèses de clé, pour 10 attaques en utilisant la méthode par acceptation de l'erreur.

$n \setminus \sigma$	0,1	0,2	0,3	0,5	1,0	1,5	2,0	3,0
100	12,2	23,5	28,6	100,2	111,5	91,4	157,4	120,1
500	1,1	1	1,6	18,9	84,9	112,5	128,2	157,4
1000	1	1	1	16,1	80	121,3	128,6	141
5000	1	1	1	1	34,8	76,1	111,8	143,1
10000	1	1	1	1	50	86,3	92,4	164,1
50000	1	1	1	1	23,9	84,8	107,6	165,8
100000	1	1	1	1	17,2	96,7	117	182,7

Tableau 9 : Résultats du rang de la clé 42 en fonction du nombre de textes et de l'écart type de la loi normale associée au bruit.

5.3.2 Résultats approche par inférence Bayésienne

Le Tableau 10 présente des résultats pour un octet de clé fixé arbitrairement à la valeur à 42, en faisant varier le paramètre σ et n la taille de la campagne (nombre de textes). Le Tableau 10 indique le rang moyen dans le classement des hypothèses de clé, pour 10 attaques.

Nous observons que plus l'écart type augmente, plus il faut de textes pour que la bonne hypothèse reste bien classée. Nous constatons également que cette méthode est plus efficace que celle par acceptation de l'erreur. Il est tout à fait probable que le fait que les méthodes par classement sont très rapidement peu efficaces lorsque l'écart type σ du bruit augmente, justifie cette différence dans les résultats.

Par ailleurs, nous nous sommes intéressés aux clés 25 et 62, qui représentaient une difficulté dans l'attaque théorique avec un crible. Ainsi les tableaux suivants 13, 12 et 11 présentent des résultats de 10 attaques de 5000 textes, faisant varier les écarts types. Le rang moyen est donné pour les trois hypothèses 25, 42 et 62. Pour le tableau 13, nous précisons que les rangs 1 et 2 correspondent bien à deux probabilités différentes et non à des scores égaux.

$n \setminus \sigma$	0,1	0,2	0,3	0,5	1,0	1,5	2,0	3,0
100	1	1,3	1,9	27,5	68	115	133,2	139,6
500	1	1	1	1	34,4	79,8	86,6	91,5
1000	1	1	1	1	10,8	39,9	67,8	92,1
5000	1	1	1	1	1	16,5	39,7	70,4
10000	1	1	1	1	1	2	36,4	56
50000	1	1	1	1	1	1,1	2,9	18,9
100000	1	1	1	1	1	1	1	8,7

Tableau 10 : Résultats du rang de la clé 42 en fonction du nombre de textes et de l'écart type de la loi normale associée au bruit.

rang de l'hypothèse $\setminus \sigma$	0.1	0.2	0.3	0.5	1.0	1.5	2.0	3.0
25	67,1	70,7	85,6	133,8	145,1	181	152,4	149,5
42	142,4	101,1	60,8	31,9	23,9	53,6	59,5	76,3
62	1	1	1	1	1,4	17,9	43,6	90,6

Tableau 11 : Résultats sur 10 attaques de 5000 textes où l'octet de clé $\hat{k} = 62$

rang de l'hypothèse $\setminus \sigma$	0,1	0,2	0,3	0,5	1,0	1,5	2,0	3,0
25	136,5	160,5	183	144,1	96,2	84,3	108,8	82,5
42	1	1	1	1	1	4,3	44	74,2
62	7,2	7,3	10,5	7,3	18	47,6	92,5	112,2

Tableau 12 : Résultats sur 10 attaques de 5000 textes où l'octet de clé $\hat{k} = 42$

rang de l'hypothèse $\setminus \sigma$	0,1	0,2	0,3	0,5	1,0	1,5	2,0	3,0
25	1	1	1	1	1	4,6	36,2	38,3
42	110,2	109,5	87,7	42,2	26,2	18,6	40	79,1
62	2	2	4,8	12,7	84,2	94,5	120,2	155,3

Tableau 13 : Résultats sur 10 attaques de 5000 textes où l'octet de clé $\hat{k} = 25$

Nous remarquons que l'attaque réussit même dans le cas litigieux des clés 25 et 62.

5.4 Amélioration de l'attaque

Dans [75], les auteurs présentent une méthode pour énumérer les clés obtenues par SCA octet par octet. Cet article répond à la question suivante : « Si à la fin d'une attaque SCA la clé retournée n'est pas la bonne (sachant qu'on attaque octet par octet) dans quelle ordre tester les autres clés de manière efficace ? »

Dans cette partie, nous souhaitons améliorer le résultat final en combinant les propriétés du KeyExpansion à la méthode de Veyrat-Charvillon *et al.* [75]. L'attaque présentée retourne les valeurs des octets de clé des tours 1 à 9, de la clé la plus probable. Il est alors évident que le résultat retourné est partiellement faux. Cependant, le fait que l'attaque ait lieu pour tous les octets de plusieurs tours est un avantage majeur. Certes, nous ne disposons pas de texte clair ni chiffré pour tester les clés, mais les octets de clé sont reliés entre eux par le KeyExpansion. L'objectif est le suivant : retourner la clé dont tous les octets de chacun des tours soient les mieux classés possibles dans notre attaque vérifiant les équations du KeyExpansion.

5.4.1 Lien entre les clés

Le système d'équations (1) (chapitre I, section 1.4.2) implique qu'un octet de clé est relié directement à deux autres octets. En effet, chaque octet est utilisé pour le calcul de deux autres octets et est lui-même issu de deux octets. Les Tableaux 24 et 25 illustrent les composantes du calcul d'un octet de clé (un bleu et un jaune pour faire un vert).

$K_r^{0,0}$	$K_r^{0,1}$	$K_r^{0,2}$	$K_r^{0,3}$
$K_r^{1,0}$	$K_r^{1,1}$	$K_r^{1,2}$	$K_r^{1,3}$
$K_r^{2,0}$	$K_r^{2,1}$	$K_r^{2,2}$	$K_r^{2,3}$
$K_r^{3,0}$	$K_r^{3,1}$	$K_r^{3,2}$	$K_r^{3,3}$

$K_{r+1}^{0,0}$	$K_{r+1}^{0,1}$	$K_{r+1}^{0,2}$	$K_{r+1}^{0,3}$
$K_{r+1}^{1,0}$	$K_{r+1}^{1,1}$	$K_{r+1}^{1,2}$	$K_{r+1}^{1,3}$
$K_{r+1}^{2,0}$	$K_{r+1}^{2,1}$	$K_{r+1}^{2,2}$	$K_{r+1}^{2,3}$
$K_{r+1}^{3,0}$	$K_{r+1}^{3,1}$	$K_{r+1}^{3,2}$	$K_{r+1}^{3,3}$

Figure 24 : Lien entre les clés

$K_r^{1,1}$	$K_r^{1,2}$	$K_r^{1,3}$	$K_r^{1,4}$
$K_r^{2,1}$	$K_r^{2,2}$	$K_r^{2,3}$	$K_r^{2,4}$
$K_r^{3,1}$	$K_r^{3,2}$	$K_r^{3,3}$	$K_r^{3,4}$
$K_r^{4,1}$	$K_r^{4,2}$	$K_r^{4,3}$	$K_r^{4,4}$

$K_{r+1}^{1,1}$	$K_{r+1}^{1,2}$	$K_{r+1}^{1,3}$	$K_{r+1}^{1,4}$
$K_{r+1}^{2,1}$	$K_{r+1}^{2,2}$	$K_{r+1}^{2,3}$	$K_{r+1}^{2,4}$
$K_{r+1}^{3,1}$	$K_{r+1}^{3,2}$	$K_{r+1}^{3,3}$	$K_{r+1}^{3,4}$
$K_{r+1}^{4,1}$	$K_{r+1}^{4,2}$	$K_{r+1}^{4,3}$	$K_{r+1}^{4,4}$

Figure 25 : Lien entre les clés, cas particulier de la première colonne

Au final, chaque octet de clé doit vérifier les trois relations suivantes (les formules sont à adapter pour les octets de la première colonne qui passent par la fonction du KeyExpension, ligne 2 de l'équation (1)) :

1. $K_r^{l,c} = K_{r+1}^{l,c-1} \oplus K_{r+1}^{l,c}$
2. $K_r^{l,c} = K_{r-1}^{l,c+1} \oplus K_r^{l,c+1}$
3. $K_r^{l,c} = K_r^{l,c-1} \oplus K_{r-1}^{l,c}$.

5.4.2 Algorithme

Le problème : « retourner la clé dont tous les octets de chacun des tours soient les mieux classés possibles dans notre attaque et qui vérifient les équations du KeyExpansion » est un problème compliqué.

C'est pourquoi, l'Algorithme 1 proposé n'est pas optimal. Celui-ci retourne une clé dont tous les octets de chacun des tours sont correctement classés et qui vérifient les équations du KeyExpansion, à condition qu'une telle clé existe. De plus, sa complexité explose si les résultats expérimentaux ne sont pas satisfaisants. Ainsi, en fonction de la puissance de calcul et de la confiance que l'attaquant donne à ses mesures, il doit choisir un seuil a.

Algorithme 1 Retourne une clé dont tous les octets de chacun des tours sont correctement classés et qui vérifient les équations du KeyExpansion

Entrée : Le classement des hypothèses sur tous les octets de clé de chaque tour de 1 à 9
Sortie : Une clé ou échec

1: **Pour** $l \in [\![0,3]\!]$ **faire :**
2: Chercher le meilleur triplet : $(K_2^{l,0}, K_1^{l,0}, K_1^{(l+1) \bmod 4,3})$.
3: Chercher le meilleur couple : $(K_2^{l,1}, K_1^{l,1})$.
4: Chercher le meilleur couple : $(K_2^{l,2}, K_1^{l,2})$.
5: **Fin de boucle pour**
6: Regarder le classement général de la clé entière.
7: **Si** la clé entière est bien classée **alors :**
8: Retourner la clé.
9: **Sinon**
10: Essayer toutes les valeurs de $(K_2^{l,2}, K_1^{l,2})$.
11: **Si** toujours pas de clé bien classée **alors :**
12: Essayer la meilleure valeur suivante de couple $(K_2^{l,1}, K_1^{l,1})$.
13: Retourner en étape 4.
14: **Si** toujours pas de clé bien classée **alors :**
15: Essayer la meilleure valeur de triplet suivante $(K_2^{l,0}, K_1^{l,0}, K_1^{(l+1) \bmod 4,3})$.
16: Retourner en étape 3.
17: **Fin de si**
18: **Fin de si**
19: **Fin de si**

À la fin de l'attaque, un score de corrélation ou une probabilité (suivant si l'attaquant choisit la méthode par acceptation de l'erreur, paragraphe 5.1 ou par inférence bayesienne, paragraphe 5.2) est associé(e) à chaque octet de clé des tours 1 à 9. Pour chacun des octets de clé, un classement des hypothèses des valeurs qu'il peut prendre, lui est attribué. Ce dernier est pris en entrée de l'algorithme 1.

L'idée est de se focaliser sur les octets des premiers tours. Nous considérons qu'un ensemble d'octets de clé est bien classé si il fait parti des a premiers. En utilisant la méthode de Veyrat-Charvillon *et al.* l'algorithme 1 sélectionne le triplet le mieux classé pour les 4 lignes $\forall l \in [\![0,3]\!]$ (ligne 2 de l'agorithme) :

$$(K_2^{l,0}, K_1^{l,0}, K_1^{(l+1) \bmod 4,3}) \quad .$$

Puis, avec $K_2^{l,0}$ le meilleur couple (ligne 3) :

$$(K_2^{l,1}, K_1^{l,1}) \quad .$$

Puis, le meilleur couple avec $K_2^{l,2}$ (ligne 2) :

$$(K_2^{l,2}, K_1^{l,2}) \quad .$$

Ainsi les $K_2^{l,3}$ sont entièrement déterminés. Les octets clés des tours 3 à 9 également. Si toutes les clés ainsi choisies sont bien classées alors cette hypothèse est gardée sinon l'algorithme 1 modifie son choix, en supposant toujours en premier que c'est à la dernière étape qui contient une erreur de sélection.

Si l'algorithme a échoué, il faut soit revoir le seuil, soit les mesures expérimentales qui ont permis d'obtenir le classement.

5.4.3 Perspectives

Définition 42 : Un **algorithme par propagation de croyances** s'appuie sur un graphe. Chaque nœud du graphe connaît certaines informations. À chaque itération, chaque nœud échange ses informations avec ses voisins et met à jour ses propres informations en fonction de ce qu'il a reçu, et ainsi de suite, jusqu'à « convergence ».

Un algorithme par propagation de croyances peut être vu comme une « rumeur ». L'algorithme de Gallager [76] est un algorithme itératif et probabiliste à propagation de croyances utilisé pour les codes LDPC (codes à matrice de parité creuse). Le principe est de calculer la probabilité à posteriori du mot de code, sachant les mots reçus et utilisant des contraintes qui régissent un mot de code. Cet algorithme se base sur un graphe de Tanner. Un graphe de Tanner est constitué de nœuds de données représentant les bits et de nœuds fonctionnels représentant les équations liant les bits entre eux. Un nœud de donnée est relié à un nœud fonctionnel si le bit qu'il représente intervient dans l'équation du nœud fonctionnel.

Une perspective pour notre étude est d'adapter les algorithmes par propagation de croyances à notre problème [2]. La difficulté est de transformer les relations du KeyExpansion qui sont sur des octets en relation sur des bits. Les nœuds de données du graphe de Tanner adapté à notre étude représenteraient alors, les bits des octets de clé.

2. Le problème étant toujours : retourner la clé dont tous les octets de chacun des tours soient les mieux classés possibles dans notre attaque et qui vérifient les équations du KeyExpansion.

6 Conclusion et perspectives

Dans ce chapitre, nous nous sommes intéressés aux attaques par observation de courant uniquement sur AES, dans le but de retrouver la clé de chiffrement. Les textes clairs et chiffrés ne sont jamais utilisés. L'idée n'est pas non plu d'utiliser des templates, l'attaquant ne dispose pas d'un circuit clone d'essai. Seules des mesures de consommation de courant, en deux points stratégiques (avant le AddRoundKey et en sortie des s-boxes) sont exploitées.

Les difficultés de ce type d'attaque reposent essentiellement sur deux points :

1. la détection des *PoI*,
2. le passage des mesures aux poids de Hamming.

Dans ce chapitre nous décrivons un court état de l'art sur la détection des *PoI*.

Les mesures ne permettent pas de donner les poids de Hamming recherchés de manière infaillible. Les travaux de ce chapitre portent essentiellement sur la prise en compte de cette erreur liée aux mesures. Deux approches ont été présentées :

1. une approche par acceptation des erreurs liées aux mesures,
2. une approche par inférence Bayésienne.

La seconde approche est plus pertinente que la première. Malheureusement ces approches n'ont été testées qu'en simulation. Une perspective essentielle de ce chapitre est la validation de nos approches sur des courbes issues d'un vrai circuit.

L'attaque pouvant se faire à n'importe quel tour, il est possible d'affiner le résultat obtenu en utilisant le KeyExpansion. Une perspective possible est d'améliorer la méthode en introduisant un algorithme par propagation de croyances.

Cette méthode permet aussi d'attaquer un AES masqué, à condition que le masque soit fixe ou qu'il ne soit pas appliqué sur l'ensemble des tours ; mais également d'autres algorithmes tels que le DES.

Partie IV

Étude théorique du meilleur registre pour l'injection de fautes mono-bit dans un schéma de Feistel généralisé

Sommaire

1 Objectifs

Dans le cadre du formalisme (section II.1), l'un des objectifs est de trouver de nouvelles attaques physiques. Une nouvelle attaque de type SCA a été présentée dans le chapitre III. Dans ce chapitre, c'est de manière systématique que nous souhaitons mettre en évidence des attaques de type FIA. Mieux que cela, nous souhaitons être capable de justifier le choix de l'emplacement de l'injection des fautes plutôt qu'un autre.

L'objectif de cette étude est le suivant :
Soit un algorithme de chiffrement basé sur un schéma de Feistel, quel est le registre idéal pour minimiser le nombre de fautes mono-bit injectées dans celui-ci, pour retrouver la clé du dernier tour ?
Nous considérons que les fautes injectées sont de type mono-bit sur un registre (bloc). Cette hypothèse d'injection de fautes n'est pas une hypothèse forte. Il est possible d'injecter des fautes mono-bit comme le montrent les travaux de Dutertre *et al.* dans [77, 78, 79]. Nous rappelons que le fait d'injecter des fautes dans un circuit l'expose au risque d'être détérioré. Ainsi, le registre idéal pour injecter les fautes est celui qui implique le moins de d'injection pour retrouver la clé.

2 Schémas de Feistel Généralisés

2.1 Définition

Les schémas de Feistel doivent leur nom au cryptologue allemand Horst Feistel.

Définition 43 : Un **schéma de Feistel** est une construction itérative permettant de créer une bijection à partir de fonctions qui ne le sont pas forcément. Pour cela, le schéma sépare son entrée en deux blocs (L et R) de taille identique. Un de ces deux blocs passe alors dans une fonction dépendante de la clé et dite **fonction de Feistel**, puis le résultat est « xoré » au second bloc. Cette construction est alors itérée plusieurs fois en inversant les rôles des deux blocs à chaque tour.

La Figure 26 illustre un schéma de Feistel classique.

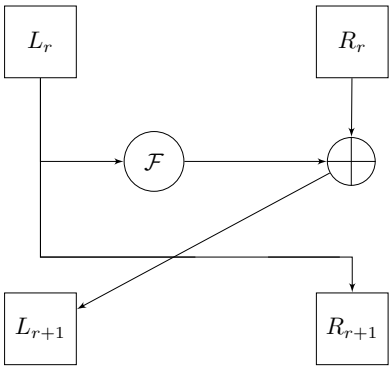


Figure 26 : Structure classique d'un schéma de Feistel

Les caractéristiques principales de ce schéma sont les suivantes :
- à l'aide de la fonction de Feistel, seule une partie de l'état interne est modifiée à chaque tour ;
- le schéma est inversible même si la fonction de Feistel utilisée ne l'est pas ;
- la bijection réciproque suit le même schéma mais en inversant l'ordre des clés de tour.

Définition 44 : Les **schémas de Feistel généralisés** introduits à CRYPTO'89 par Zheng *et al.* [80], notés GFN, sont une classe plus générale de schémas reprenant les idées principales du schéma de Feistel, mais en subdivisant l'état interne en plus de deux blocs.

D'après Berger *et al.* dans [81], un tour de GFN (Genralized Feistel Network) est divisé en deux transformations successives : la couche non-linéaire et la couche de permutation.
- La couche non-linéaire est constituée d'une ou plusieurs fonctions de Feistel $\mathcal{F}$ dépendant d'un bloc de clé de tour.
- La couche de permutation consiste à réarranger les différents blocs de l'état interne.

Attention, les tours d'indice r vont de 0 à $\mathbf{r}$ et le découpage en blocs de 0 à $\mathbf{b}-1$. La Figure 27 est un exemple de schéma de Feistel à 8 blocs.

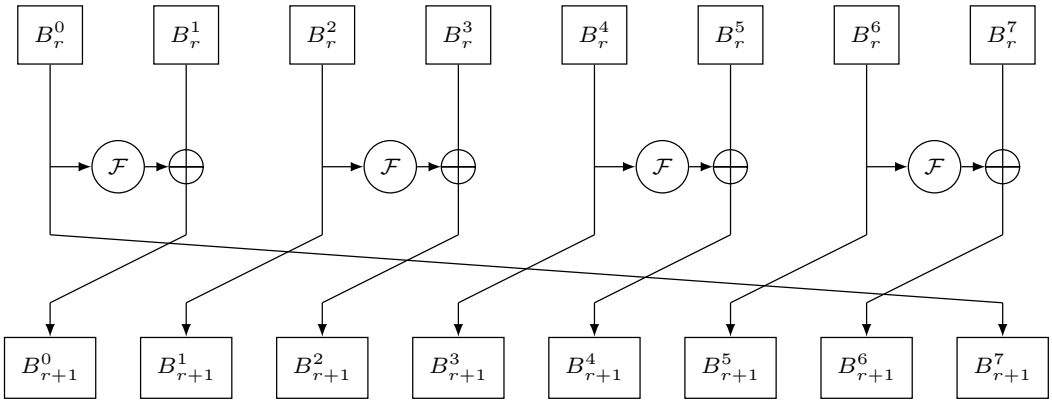


Figure 27 : Exemple de schéma de Feistel généralisé avec 8 blocs.

2.2 Fonction de Feistel

La fonction de tour est composée d'une fonction de Feistel $\mathcal{F}$ et du « xor » avec un autre bloc. Une fonction de Feistel est construite à partir de trois types de fonctions.

1. Le « xor » avec la clé de tour K_r^λ. Nous remarquons que dans un même tour, il peut y avoir plusieurs blocs qui passent dans la fonction $\mathcal{F}$, ainsi un exposant λ est ajouté à la clé pour différencier les différents blocs de la clé de tour ($\lambda \in [\![0, \Lambda - 1]\!]$). Par exemple dans la Figure 27, il y a 4 blocs de clé par tour.
2. Les s-boxes, qui ne sont pas linéaires, assurent la confusion (voir chapitre I Définition 7).
3. Les fonctions de "mélange", c'est à dire les fonctions linéaires, assurent la diffusion au sein d'un bloc (voir chapitre I, Définition 8).

2.3 Exemples de schémas de Feistel et de GFN

Le schéma de Feistel le plus connu est le DES [4], il est décrit en chapitre I.1.4.1. Cependant, sa particularité est que les blocs traités par la fonction de tour sont ceux de droites contrairement aux autres schémas de Feistel classiques.

2.3.1 MIBS

MIBS est un algorithme de chiffrement basé sur un schéma de Feistel classique, il suit le schéma de la Figure 26. Il doit son nom aux initiales des auteurs Maryam Izadi et Babak Sadeghiyan qui l'ont présenté à CANS 2009 [82]. Il se compose de 32 itérations d'un schéma de Feistel sur 64 bits, avec des clés de 64 ou 80 bits.

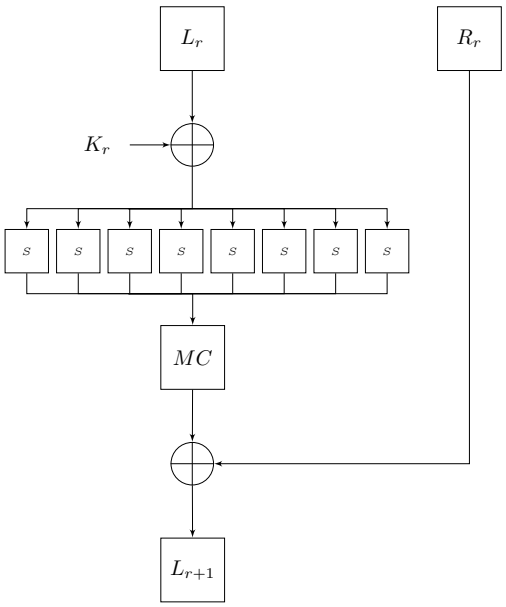


Figure 28 : Schéma de la fonction de Feistel du MIBS

La fonction de Feistel est illustrée en Figure 28. Elle opère sur 32 bits répartis en 8 nibbles [1] et se compose de 3 étapes :

- le « xor » avec la clé de tour ;
- une couche de s-boxes bijectives en parallèle, prenant en entrée un nibble (La s-box est donnée dans le Tableau 14.) ;

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$S(x)$	4	15	3	8	13	10	12	0	1	5	7	14	2	6	1	9

Tableau 14 : S-box de MIBS en notation décimale.

- MC une application linéaire de diffusion.
  Soit y_i les sorties des s-boxes et y'_i les paquets de nibbles à la sortie de la fonction de Feistel après MC. Les y'_i sont définis comme suit :

$$\begin{aligned}
y'_0 &= y_0 \oplus y_1 \oplus y_3 \oplus y_4 \oplus y_6 \oplus y_7 \\
y'_1 &= y_1 \oplus y_2 \oplus y_3 \oplus y_4 \oplus y_5 \oplus y_6 \\
y'_2 &= y_0 \oplus y_1 \oplus y_2 \oplus y_3 \oplus y_5 \oplus y_7 \\
y'_3 &= y_0 \oplus y_2 \oplus y_3 \oplus y_4 \oplus y_7 \\
y'_4 &= y_1 \oplus y_2 \oplus y_3 \oplus y_6 \oplus y_7 \\
y'_5 &= y_0 \oplus y_1 \oplus y_3 \oplus y_4 \oplus y_5 \\
y'_6 &= y_0 \oplus y_1 \oplus y_2 \oplus y_5 \oplus y_6 \\
y'_7 &= y_0 \oplus y_2 \oplus y_3 \oplus y_5 \oplus y_6 \oplus y_7
\end{aligned}$$

1. agrégat de 4 bits

2.3.2 TWINE

TWINE, dont le nom signifie natter, entortiller, est un algorithme de chiffrement de 64 bits. Il a été présenté à SAC 2012 [83]. Le but recherché est une implémentation matérielle réduite, tout en gardant de bonnes performances logicielles. Il s'agit d'un schéma de Feistel généralisé agissant sur 16 branches de 4 bits chacune, avec des clés de 80 ou 128 bits.

La permutation des branches du schéma vise à améliorer la diffusion [84]. Elle est présentée en Figure 29. Par rapport à un décalage circulaire, celle-ci requiert moitié moins de tours pour qu'une différence dans une des branches se diffuse dans toutes les branches. Le chiffrement complet consiste à itérer ce schéma 36 fois pour les deux longueurs de clés.

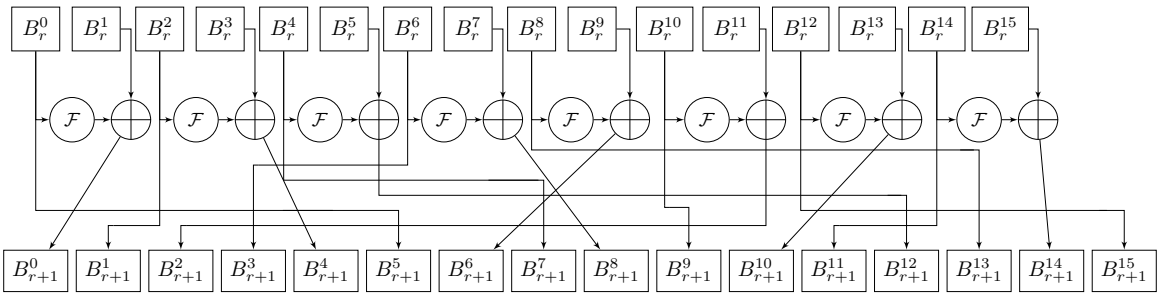


Figure 29 : Schéma de TWINE

La fonction de Feistel est appelée 8 fois par tour et se compose simplement d'un « xor » de 4 bits de clé, suivi d'un passage dans une s-box bijective, décrite dans le Tableau 15.

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$S(x)$	12	0	15	10	2	11	9	5	8	3	13	7	1	14	6	4

Tableau 15 : S-box de TWINE en notation décimale.

2.3.3 CLEFIA

CLEFIA dont le nom est dérivé du mot « CLEF », est un algorithme de chiffrement de 128 bits pour des tailles de clés de 128, 192 ou 256 bits. Il a été présenté à FSE 2007 [85]. Il s'agit d'un schéma de Feistel généralisé agissant sur 4 branches de 32 bits chacune, illustré en Figure 30.

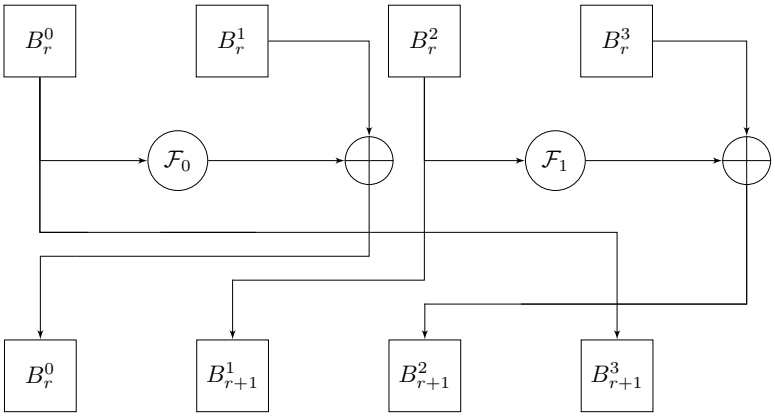


Figure 30 : Schéma de CLEFIA

À chaque tour, il utilise deux fonctions de Feistel différentes, quoique construites de la même manière, $\mathcal{F}_0$ et $\mathcal{F}_1$ illustrées en Figure 31. Chacune se compose de trois étapes :

- un « xor » avec une clé de tour ;
- 4 s-boxes de 8 bits vers 8, en parallèle, dont l'ordre est différent d'une fonction à l'autre, plus précisément (S_1, S_0, S_1, S_0) pour $\mathcal{F}_0$ et (S_0, S_1, S_0, S_1) pour $\mathcal{F}_1$ (CLEFIA utilise deux s-boxes différentes S_0 et S_1 données en annexe A.iii) ;
- deux applications linéaires de diffusion, MC_0 pour $\mathcal{F}_0$ et MC_1 pour $\mathcal{F}_1$, de type MixColumns (application linéaire sur les corps finis).

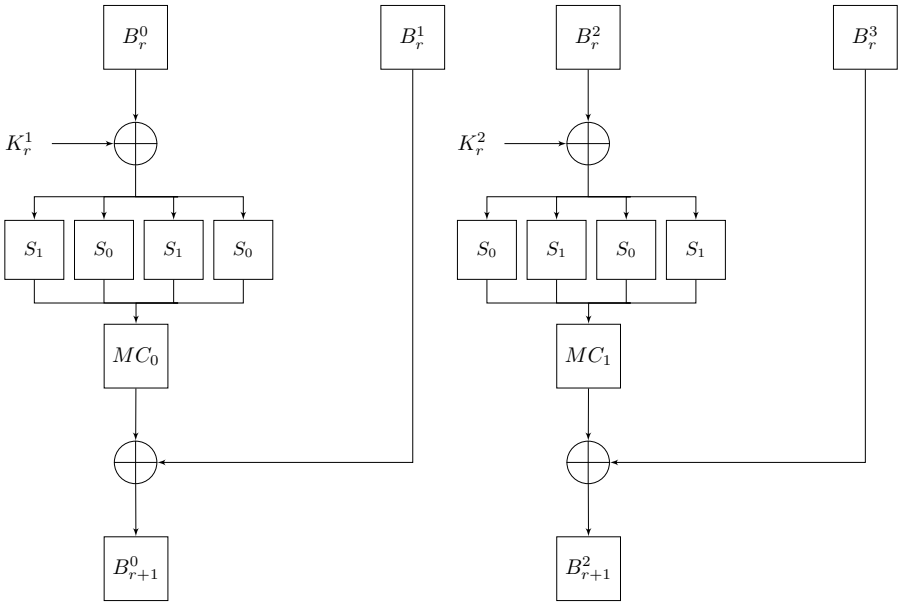


Figure 31 : Les fonctions de tours $\mathcal{F}_0$ et $\mathcal{F}_1$ de CLEFIA.

3 DFA sur un schéma de Feistel classique un généralisé

Le chemin d'attaque usuel pour une attaque de type DFA sur un schéma Feistel se fait sur le dernier tour, comme dans les DFA classiques [11] sur le DES, présentées en annexe B ii.a.

Le chemin d'attaque est illustré en Figure 32, il est répété pour chaque bloc de clé de tour $K^{\lambda}_{\mathbf{r}}$. D'un point de vue formalisme des attaques physiques, nous avons : $O_S = \left(B^{i}_{\mathbf{r}}, B^{i*}_{\mathbf{r}}\right)$ et $O_R = \Delta_{B^{o}_{\mathbf{r}}}$.

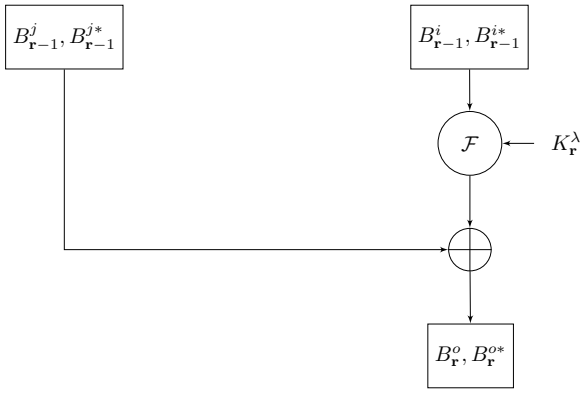


Figure 32 : Schéma d'attaque de type DFA dans un schéma de Feistel classique ou généralisé

Ainsi, nous avons la relation suivante :

$$\begin{aligned} \Delta_{B^{o}_{\mathbf{r}}} &= B^{o}_{\mathbf{r}} \oplus B^{o*}_{\mathbf{r}} \\ &= \mathcal{F}\left(B^{i}_{\mathbf{r}-1}, K^{\lambda}_{\mathbf{r}}\right) \oplus \mathcal{F}\left(B^{i*}_{\mathbf{r}-1}, K^{\lambda}_{\mathbf{r}}\right) \oplus \Delta_{B^{j}_{\mathbf{r}-1}} \quad . \end{aligned} \tag{14}$$

Néanmoins, il existe le cas où la clé de tour est xorée à la fin de la fonction de Feistel. Dans ce cas précis nous avons :

$$\begin{aligned} \Delta_{B^{o}_{\mathbf{r}}} &= \mathcal{F}\left(B^{i}_{\mathbf{r}-1}\right) \oplus K^{\lambda}_{\mathbf{r}} \oplus \mathcal{F}\left(B^{i*}_{\mathbf{r}-1}\right) \oplus K^{\lambda}_{\mathbf{r}} \oplus \Delta_{B^{j}_{\mathbf{r}-1}} \\ &= \mathcal{F}\left(B^{i}_{\mathbf{r}-1}\right) \oplus \mathcal{F}\left(B^{i*}_{\mathbf{r}-1}\right) \oplus \Delta_{B^{j}_{\mathbf{r}-1}} \quad . \end{aligned}$$

Il n'y a plus de dépendance avec la clé de tour, il faut donc trouver un autre chemin d'attaque ; c'est par exemple le cas de SIMON [86] ou de Piccolo [87].

Un autre cas à gérer est lorsque $B^{i}_{\mathbf{r}-1}$ et $B^{i*}_{\mathbf{r}-1}$ sont masqués. Il n'est alors pas possible de les obtenir directement. C'est le cas pour l'algorithme Piccolo. Il existe des solutions à ce problème, seulement il ne faut pas oublier de le prendre en compte. Ces cas particuliers ne sont pas pris en compte dans notre étude.

Les attaques physiques étant de type diviser pour régner, le bloc de clé cible $K^{\lambda}_{\mathbf{r}}$ est partitionné en Γ fragments (un par s-box). Au final, la clé de tour est donc décomposée en Λ blocs $K^{\lambda}_{\mathbf{r}}$, eux-mêmes partitionnés en Γ fragments $K^{\lambda,\gamma}_{\mathbf{r}}$.

Une hypothèse sur $K^{\lambda,\gamma}_{\mathbf{r}}$ est notée k et sa valeur exacte est notée $\hat{k}$.

4 Notre approche

Cette étude porte sur la recherche du bloc B_r^e idéal pour injecter une faute mono-bit. L'emplacement est idéal pour injecter les fautes lorsque le nombre de fautes nécessaires pour retrouver la clé est minimal. Autrement dit, ce que nous souhaitons c'est attaquer un maximum de fragments sur un maximum de blocs de clé à chaque faute injectée. Nous souhaitons avoir une bonne diffusion de la faute, de manière à influencer un maximum de fragments de blocs de clé. D'où, a priori, la recherche du registre idéal dépend de deux choses :

1. la structure générale du schéma de Feistel classique ou généralisé,
2. la fonction de Feistel.

4.1 Raisonnement au niveau du schéma

4.1.1 Délai de diffusion

Le bloc B_0^i **influence** le bloc B_r^j si B_0^i apparaît effectivement dans l'expression de B_r^j vu comme fonction de $B_0^0, \cdots, B_0^{\mathbf{b}-1}$.

B_0^i **a diffusé** au tour r si B_0^i influence tous les B_r^j pour $j \in [\![0, \mathbf{b}-1]\!]$.

Si tous les blocs B_0^i ont diffusé au tour r, on dit que le schéma de Feistel a atteint **l'état de pleine diffusion**, c'est-à-dire que tous les blocs en sortie B_r^j dépendent de tous les blocs en entrée B_0^i.

L'état de pleine diffusion du GFN présenté en exemple dans la Figure 27 est illustré en Figure 33.

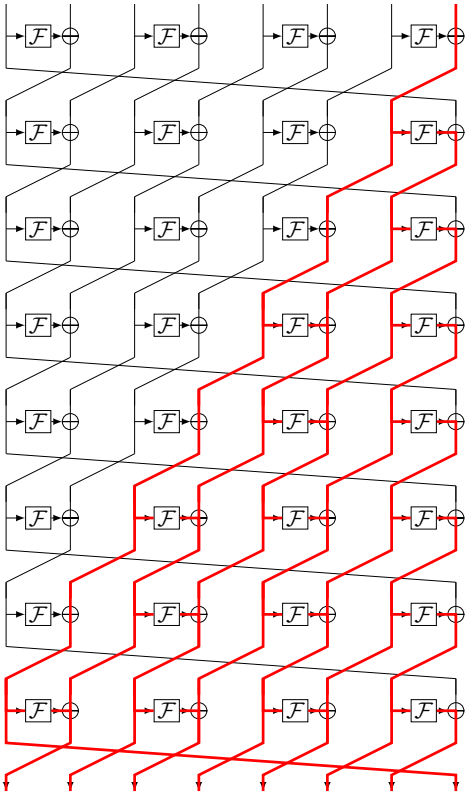


Figure 33 : Figure illustrant l'état de plein diffusion d'un GFN.

Définition 45 : Le nombre minimal de tours requis pour atteindre la pleine diffusion est appelé **délai de (pleine) diffusion**, il est noté d.

La diffusion dans les GFNs a été étudié notamment par Suzaki et Minematsu [84].
En partant du dernier tour, il n'est pas pertinent d'injecter des fautes au delà du délai de diffusion. Au delà, la faute est en effet propagée à tous les blocs comme au délais de diffusion, mais son étude est plus complexe. Notre étude s'intéresse donc aux blocs B_r^e, avec $e \in [\![0, \mathbf{b}-1]\!]$, $r \in [\![\mathbf{r}-(d+1), \mathbf{r}-1]\!]$, où e indique l'indice du bloc où la faute mono-bit est injectée.

4.1.2 Représentation matricielle du schéma

Des représentations matricielles des schémas de Feistel ont été proposées par Kim *et al.* dans [88] puis par Berger *et al.*, dans [81]. Notre représentation est une adaptation de ces travaux à nos besoins.

Soit $\mathbb{D} = \mathbb{N} \cup \{-\infty\}$, $(\mathbb{D}, \max, +)$ un dioïde commutatif idempotent [2]. C'est à dire $(\mathbb{D}, \max, +)$ est un demi-anneau car max n'est pas inversible. C'est un dioïde car la relation induite par la première loi max est une relation d'ordre ($a \leq b$ s'il existe c tel que $\max(a, c) = b$). Il est commutatif car les deux lois le sont et idempotent car $\max(x, x) = x$. L'élément neutre pour la loi max est $-\infty$ et 0 celui de la loi $+$. Nous remarquons que $-\infty$ est absorbant pour la loi $+$.

Cet objet mathématique peu usuel permet de représenter des degrés de polynômes, en attribuant la valeur $-\infty$ au degré du polynôme nul. Plus précisément dans notre cas il s'agit de degré de polynôme en $\mathcal{F}$.

Soient M_1 et M_2 deux matrices à coefficients dans $\mathbb{D}$, il est utile de redonner les opérations de base sur ce type de matrice [3].

- Le produit par un scalaire a, $M_3 = a \cdot M_1$ est donc la somme au sens mathématique classique avec ce scalaire :
  $M_3(i, j) = M_1(i, j) + a$
- La somme, $M_3 = M_1 + M_2$ est le maximum sur les coefficients :
  $M_3(i, j) = \max(M_1(i, j), M_2(i, j))$
- Le produit $M_3 = M_1 \cdot M_2$ est donc le max sur les sommes des coefficients :
  $M_3(i, j) = \max_{0 \leqslant l \leqslant \mathbf{l}-1}(M_1(i, l) + M_2(l, j))$ ou $\mathbf{l}$ est le nombre de lignes de M_1 et donc de colonnes de M_2.

Ainsi, un schéma de Feistel peut être représenté à l'aide d'une matrice $\mathcal{M}$ à coefficients dans $\mathbb{D}$.

- $\mathcal{M}(i, j) = 1$ si $B_{r+1}^i = \mathcal{F}(B_r^j) \oplus \cdots$ c'est à dire B_r^j influence B_{r+1}^i par une fonction de Feistel $\mathcal{F}$.
- $\mathcal{M}(i, j) = 0$ si $B_{r+1}^i = B_r^j$ ou $B_{r+1}^i = \cdots \oplus B_r^j$ c'est à dire B_r^j influence B_{r+1}^i directement.
- $\mathcal{M}(i, j) = -\infty$ sinon.

Après n tours $\mathcal{M}^n(i, j)$ est le plus grand nombre de fonctions $\mathcal{F}$ traversées pour aller de B^j à B^i. En d'autres termes, il existe d'autres chemins reliant B^j à B^i, mais le nombre de fonctions traversées est inférieur ou égal à $\mathcal{M}^n(i, j)$ Les polynômes n'ont pas de coefficient négatif, c'est pourquoi, après n tours $\mathcal{M}^n(i, j)$ est le nombre de fonctions $\mathcal{F}$ traversées.

2. Des rappels d'algèbre sont présentés en annexe C i.
3. Les matrices représentant des schémas de Feistel généralisés sont des matrices carrées. Par conséquent, Il n'y a pas de condition spéciale sur les dimensions

Il est important de faire remarquer l'inégalité suivante :

$$\mathcal{M}^n(i,j) \neq (\mathcal{M}(i,j))^n \quad .$$

Par exemple, la matrice $\mathcal{M}$ ci-dessous est la matrice associée au schéma défini en Figure 27.

$$\mathcal{M} = \begin{pmatrix} 1 & 0 & -\infty & -\infty & -\infty & -\infty & -\infty & -\infty \\ -\infty & -\infty & 0 & -\infty & -\infty & -\infty & -\infty & -\infty \\ -\infty & -\infty & 1 & 0 & -\infty & -\infty & -\infty & -\infty \\ -\infty & -\infty & -\infty & -\infty & 0 & -\infty & -\infty & -\infty \\ -\infty & -\infty & -\infty & -\infty & 1 & 0 & -\infty & -\infty \\ -\infty & -\infty & -\infty & -\infty & -\infty & -\infty & 0 & -\infty \\ -\infty & -\infty & -\infty & -\infty & -\infty & -\infty & 1 & 0 \\ 0 & -\infty & -\infty & -\infty & -\infty & -\infty & -\infty & -\infty \end{pmatrix}$$

4.1.3 Utilisation de la forme matricielle

Pour qu'un bloc $K^{\lambda}_{\mathbf{r}}$ puisse être attaqué à l'aide d'injections de fautes, il faut que le bloc $B^i_{\mathbf{r}-1}$ en entrée de la fonction de Feistel $\mathcal{F}_{\mathbf{r}}$ soit fauté. Il faut donc que le bloc sur lequel la faute a été injectée influence le bloc d'entrée de la fonction de Feistel.

Soit B^e_r le bloc dans lequel est injectée une faute. Soit V un vecteur de taille $\mathbf{b}$ valant $-\infty$ sauf à l'indice de bloc e, à cet indice sa valeur est 0. Le produit matrice vecteur $V_{\mathcal{F}}$ défini dans l'équation (15) permet de savoir pour chaque bloc en sortie du dernier tour, s'il est fauté et par combien de fonctions de tour est passée l'erreur qu'il reçoit. La puissance $\mathbf{r}-(r+1)$ correspond au nombre de tours à parcourir entre le tour r et la sortie.

$$V_{\mathcal{F}} = \mathcal{M}^{\mathbf{r}-(r+1)} \cdot V \quad . \tag{15}$$

Si l'indice i correspondant au bloc $B^i_{\mathbf{r}-1}$ nous observons :
- $-\infty$: alors le bloc n'est pas fauté,
- 0 : le bloc contient la faute mono-bit,
- $x \in \mathbb{N}$: le bloc est fauté et la faute est passé x fois dans une fonction de Feistel.

Ce produit permet entre autres de définir pour chaque faute le nombre n_λ de blocs de clé attaqué.

4.2 Raisonnement au niveau de la fonction de Feistel

Il est important de rappeler que dans une fonction de Feistel, une couche de non-linéarité est requise ; ce sont les Γ s-boxes. Ainsi, chaque bloc de clé est découpé en Γ fragments (un par s-box). Cette division a un fort impact sur la propagation d'une faute. L'étude de la fonction de Feistel doit permettre d'identifier combien de fragments de clé sont attaqués à chaque faute injectée. Ce nombre est noté n_γ.

L'étude de la fonction de Feistel doit également permettre de retrouver la différentielle :

$$\Delta_{B^j_{\mathbf{r-1}}} = B^j_{\mathbf{r-1}} \oplus B^{j*}_{\mathbf{r-1}} \quad .$$

En effet, $B^j_{\mathbf{r-1}}$ pouvant être fauté, il est important de connaître les différentes valeurs possibles de la différentielle afin d'utiliser l'équation (14).

L'étude qui suit est en fait une étude de la diffusion des bits dans la fonction de Feistel.

4.2.1 Nombre de fragments de bloc de clé attaqués

La première question est : « combien de fragments de clé peuvent être attaqués ? » C'est-à-dire combien d'entrées de s-boxes sont fautées. En regardant la fonction de Feistel, il est possible de savoir quels bits de sortie peuvent être fautés avec l'injection d'une faute mono-bit.

Les bits en entrées des s-boxes sont linéairement dépendants des bits d'entrée de la fonction de Feistel, il est possible de déduire quelles s-boxes sont fautées à la fonction de Feistel suivante.

Le nombre de fragments de clé attaqués pour une faute injectée, noté n_γ, peut être évalué, à partir de la diffusion des bits dans la fonction de Feistel. La i-ème coordonnée $V_\mathcal{F}(i)$ du vecteur $V_\mathcal{F}$ indiquant le nombre de passages dans la fonction n_γ, peut être calculé.

Attention, il faut aussi regarder le passage dans la fonction de Feistel ciblée ; c'est-à-dire le début de la fonction de Feistel. Par exemple dans le cas du DES, un unique bit fauté en entrée de fonction de Feistel peut impacter jusqu'à 2 s-boxes à cause de l'expansion E.

4.2.2 Recherche de la différentielle $\Delta_{B^j_{\mathbf{r-1}}}$

Si la faute n'est pas propagée dans $B^j_{\mathbf{r-1}}$, la différentielle $\Delta_{B^j_{\mathbf{r-1}}}$ est nulle. Sinon elle est inconnue, ce qui pose soucis dans l'équation (14). La j-ème coordonnée $V_\mathcal{F}(j)$ du vecteur $V_\mathcal{F}$ indique si $B^j_{\mathbf{r-1}}$ a été fauté ou non, mais aussi combien de fois la faute est passée par une fonction de Feistel.

L'étude de la diffusion des bits dans une fonction de Feistel et la connaissance du nombre de passage dans celle-ci, permettent d'évaluer le nombre de bits potentiellement fautés dans $B^j_{\mathbf{r-1}}$.

Par exemple, si la faute n'est passée dans aucune fonction de Feistel, la différentielle $\Delta_{B^j_{\mathbf{r-1}}}$ est mono-bit, il y a autant de différentielles possibles que de bits dans un bloc. Dans le cas du DES, 32 différentielles $\Delta_{B^j_{\mathbf{r-1}}}$ sont possibles. Cependant, cette approche peut être améliorée en recherchant le bit fauté de B^{e*}_r. La seule différentielle connue est celle observée sur le dernier tour. Pour cela un nouveau vecteur est introduit :

$$W_\mathcal{F} = \mathcal{M}^{\mathbf{r}-r} \cdot V = \mathcal{M} \cdot V_\mathcal{F} \quad .$$

La valeur minimale positive de $W_\mathcal{F}$, $\min(W_\mathcal{F})$ correspond au bloc qui contient la différentielle non nulle la plus simple (la moins diffusée par les fonctions de tour). Différents cas sont alors possibles Si $\min(W_\mathcal{F}) = 0$, une faute mono-bit est observée sur la différentielle $\Delta_{B^{\min(W_\mathcal{F})}_{\mathbf{r}}}$, la faute a juste été recopiée, ainsi l'erreur de B^{e*}_r est connue. Pour une faute mono-bit, l'énumération de toutes les différentielles possibles à la sortie de la fonction de Feistel est pertinente. En effet, toutes les différentielles ne sont pas possibles. Ainsi, si $\min(W_\mathcal{F}) = 1$, connaître $\Delta_{B^{\min(W_\mathcal{F})}_{\mathbf{r}}}$ permet de restreindre l'ensemble des bits fautés possibles de B^{e*}_r. Si $\min(W_\mathcal{F}) = 2$, la même approche est encore envisageable. Au delà de 2 passages par la faute dans la fonction de Feistel, tous les bits en sortie sont susceptibles d'être fautés, sinon c'est que la fonction de Feistel a de mauvaises propriétés de diffusion. De plus, l'algorithme de recherche des différences a une complexité exponentielle en nombre de fonctions traversées.

4.3 Raisonnement au niveau s-box

Dans ce paragraphe, nous réduisons le chemin d'attaque au niveau des s-boxes, en considérant que les sous-fonctions de la fonction de tour $\mathcal{F}$ situées avant le xor avec la clé sont connues et que seules les s-boxes ne sont pas linéaires. Nous nous ramenons à l'équation (16) ou x, x^* et Δ_y sont connus.

$$S(x \oplus \hat{k}) \oplus S(x^* \oplus \hat{k}) = \Delta_y \quad . \tag{16}$$

L'idée est de s'inspirer des méthodes de la cryptanalyse différentielle classique. La démonstration qui suit est composée de résultats bien connus que l'on retrouve nouement dans [89, 90, 6, 91, 92, 93]

En reprenant l'équation (16) nous posons $e = x \oplus x^*$, nous avons alors l'équation suivante :

$$S(x \oplus \hat{k}) \oplus S(x \oplus e \oplus \hat{k}) = \Delta_y \quad . \tag{17}$$

D'une expérience à l'autre, les valeurs de x, e et Δ_y varient tandis que $\hat{k}$ reste constant. Nous recherchons le nombre minimal n d'expériences (injections de fautes), tel qu'étant donnés : $x_0, \cdots, x_{n-1}, e_0, \cdots, e_{n-1}$ et $\Delta_{y_0}, \cdots, \Delta_{y_{n-1}}$, il n'y ait qu'une seule clé qui vérifie les n équations (17).

La s-box S est une fonction booléenne de $\mathbb{F}_2^q$ dans $\mathbb{F}_2^{q'}$. Soit $u \in \mathbb{F}_2^q$ et $v \in \mathbb{F}_2^{q'}$, l'ensemble des entrées de la s-box telles que si la différence a est appliquée en entrée, la différence b est observée en sortie, est noté $\mathcal{S}_{u,v}$. En d'autres termes :

$$\mathcal{S}_{u,v} = \{z \in \mathbb{F}_2^q / S(z) \oplus S(z \oplus u) = v\} \quad . \tag{18}$$

Par symétrie de l'équation (18), le cardinal $\#\mathcal{S}_{u,v}$ est pair. Cet ensemble $\mathcal{S}_{u,v}$ est défini par :

$$\forall u \in \mathbb{F}_2^q \backslash \{0\} \text{ et } \forall v \in \mathbb{F}_2^{q'} \quad .$$

Le logarithme en base 2 de la taille maximum des ensembles $\mathcal{S}_{u,v}$ est noté δ.

$$\begin{aligned} \delta &= \log_2 \max_{u \neq 0,\ v} \#\mathcal{S}_{u,v} \\ 2^\delta &= \max_{u \neq 0,\ v} \#\mathcal{S}_{u,v} \end{aligned}$$

Puisque les $\mathcal{S}_{u,v}$ ne sont pas tous vides et sont tous de cardinal pairs, cela implique que $\delta \geq 1$. Plus généralement : $\delta \geq q - q'$. En effet, soit une différence en entrée $a \neq 0$ quelconque fixée, alors :

$$\sum_{v \in \mathbb{F}_2^{q'}} \#\mathcal{S}_{u,v} = 2^q \quad .$$

Donc, il existe un v tel que :

$$\#\mathcal{S}_{u,v} \geq \frac{2^q}{2^{q'}} \text{ et donc } \delta \geq q - q' \quad .$$

D'autre part, si z est tiré uniformément dans $\mathbb{F}_2^q$ alors :

$$Pr\left(z \in \mathcal{S}_{u,v}\right) = \frac{\#\mathcal{S}_{u,v}}{2^q} \leq 2^{\delta - q} \quad .$$

Ainsi, k est solution de l'équation (17) si et seulement si $x \oplus k \in \mathcal{S}_{e,\Delta_y}$. Ce qui implique que le

nombre de solutions de l'équation (17) est $\#\mathcal{S}_{e,\Delta_y}$.

L'ensemble des z tel que $z \oplus x \in \mathcal{S}_{e_i,\Delta y_i}$est noté $x \oplus \Delta_{e,\Delta_y}$. Alors k est solution de (17) si et seulement si $k \in x \oplus \mathcal{S}_{e,\Delta_y}$.

Pour n expériences, nous nous intéressons à la probabilité suivante :

$$Pr\left(k \in \bigcap_{i=0}^{n-1} (x_i \oplus \mathcal{S}_{e_i,\Delta_{y_i}})\right) \quad .$$

Les fautes e_i sont indépendantes les unes aux autres. Les Δ_{y_i} et les x_i indépendantes et identiquement distribuées. C'est-à-dire que ce sont des variables aléatoires qui suivent toutes la même loi de probabilité et sont mutuellement indépendantes, nous avons :

$$\begin{aligned} Pr\left(k \in \bigcap_{i=0}^{n-1} \left(x_i \oplus \mathcal{S}_{e_i,\Delta_{y_i}}\right)\right) &= \prod_{i=0}^{n-1} Pr\left(k \in (x_i \oplus \mathcal{S}_{e_i,\Delta_{y_i}})\right) \\ &\leq 2^{n(\delta-q)} \quad . \end{aligned}$$

Par ailleurs, la probabilité cherchée est au moins égale à 2^{-q} car il y a une solution au problème. Le nombre n de fautes recherché, est tel que $2^{n(\delta-q)} \leq 2^{-q}$ d'où [4] :

$$n \geq \left\lceil \frac{q}{q-\delta} \right\rceil \quad . \tag{19}$$

En pratique, il existe principalement deux cas possibles :

- La s-box est bijective (q bits vers q bits), optimale pour la différentielle, plus précisément avec δ aussi petit que possible typiquement : $q = 4$ ou $q = 8$. $\delta \geq 1$. Cependant, à l'heure actuelle les seuls exemples connus atteignant cette borne sont pour q impair ou pour $q = 6$. En revanche, beaucoup d'exemples de s-boxes avec $\delta = 2$ sont connus. Le cas $q = 4$ a été bien étudié, comme le montrent les classifications de Leander et Poschmann [94] et de Saarinen [95]. Il est aujourd'hui facile de trouver une s-box 4×4 avec un δ minimal. L'autre cas usuel de s-box bijective est le cas $q = 8$. Si leur classification reste un problème ouvert à l'heure actuelle, de nombreux exemples de s-box avec $\delta = 2$ existent, la plus connue étant la s-box de l'AES (décrit en chapitre I paragraphe 1.4.2) [5].

- La s-box n'est pas bijective (q bits vers q' bits). C'est le cas du DES avec des s-boxes (6 bits vers 4 bits), alors $\delta \geq 6 - 4 = 2$; $n = 2$. Cependant, à cause de leur structure, les s-boxes du DES n'atteignent pas cette borne, mais vérifient en fait $\delta = 4$, ce qui donne $n = 3$.

De manière générale, avec les paramètres q et q' utilisés, en pratique, il est toujours possible de calculer δ en un temps raisonnable.

4. $\lceil x \rceil$ représente la partie entière supérieure.
5. Nous rappelons que l'AES ne suit pas un schéma de Feistel.

Maintenant que se passe-t-il quand il y a une incertitude sur la valeur Δ_y ? Il s'agit du cas de l'équation (20), avec n_Δ le nombre de Δ_y possibles :

$$S(x \oplus \hat{k}) \oplus S(x \oplus e \oplus \hat{k}) = \Delta_{y_1} \text{ ou } \cdots \text{ ou } \Delta_{y_{n_\Delta}} \quad . \tag{20}$$

Puisque à une différence d'entrée u fixée, les ensembles $\mathcal{S}_{u,v}$ sont disjoints deux à deux nous avons :

$$Pr\left(z \in \bigcup_{i=1}^{n_\Delta} \mathcal{S}_{u,v_i}\right) = \sum_{i=1}^{n_\Delta} P\left[z \in \mathcal{S}_{u,v_i}\right] \quad .$$

Comme dans le cas où il y a seulement un Δ_y, k est solution de l'équation (20) si et seulement si :

$$k \ \in \ x \ \oplus \ \bigcup_{i=1}^{n_\Delta} \mathcal{S}_{e,\Delta_{y_i}} \quad .$$

L'union des hypothèses de fragments de clé vérifiant des équations de type (20) n'est pertinente que si elle est plus restreinte que l'ensemble de toutes les hypothèses $\mathbb{K}$.

$$\bigcup_i (x \oplus \mathcal{S}_{e,\Delta y_i}) \subsetneq \mathbb{K} \quad .$$

Pour n expériences nous avons :

$$Pr\left(k \in \bigcap_{j=0}^{n-1} \left(x_j \oplus \bigcup_{i=1}^{n_\Delta} \mathcal{S}_{e_j,\Delta_{y_{j,i}}}\right)\right) = \prod_{j=1}^{n-1} Pr\left(k \in x_j \oplus \bigcup_{i=1}^{n_\Delta} \mathcal{S}_{e_j,\Delta_{y_{j,i}}}\right) \quad . \tag{21}$$

Par conséquent, n_f le nombre minimum de fautes pour récupérer la clé est le plus petit n dans l'équation précédente (21) tel que le terme soit inférieure à 2^{-q}. Plus précisément (avec n_Δ le nombre moyen de Δ_y possibles), nous avons :

$$n_f = \left\lceil \frac{q}{q - \delta - \log_2 n_\Delta} \right\rceil \quad . \tag{22}$$

5 Algorithme et résultats

5.1 Algorithme

Cette partie décrit la méthode pour sélectionner le meilleur bloc dans lequel injecter des fautes mono-bit pour un algorithme de chiffrement basé sur schéma de Feistel classique ou généralisé.

L'algorithme 2 décrit notre procédure ; il prend en entrée la fonction de Feistel $\mathcal{F}$ et la matrice $\mathcal{M}$ représentant le schéma.

Pour tous les blocs concernés, c'est-à-dire les B_r^e avec $e \in [\![0, \mathbf{b}-1]\!]$ et $r \in [\![\mathbf{r}-(d+1), \mathbf{r}-1]\!]$, il retourne le nombre de fragments de clé et le nombre de différentielles à prendre en compte.

Algorithme 2 Retourne des informations pertinentes à la sélection du meilleur bloc dans lequel injecter des fautes mono-bit pour que l'attaque soit optimisée, c.à.d injecter le moins de fautes possibles.

Entrée : $\mathcal{F}$, $\mathcal{M}$
Sortie : B_r^e avec leur n_λ, les n_γ, les n_Δ et n_f, associés

Calculer d le délai de diffusion à l'aide de $\mathcal{M}$.
Déduire Γ de $\mathcal{F}$.
Pour 0 à d (nombre de passages dans un $\mathcal{F}$) **faire :**
  Calculer le n_γ associé.
  Calculer le n_Δ associé.
Fin de boucle pour
Pour B_r^e $e \in [\![0, \mathbf{b}]\!], r \in [\![\mathbf{r}-(d+1), \mathbf{r}-1]\!]$ **faire :**
  Créer un vecteur $-\infty$ V de taille $\mathbf{b}$
  $V(i) \leftarrow 0$
  $V_{\mathcal{F}} \leftarrow \mathcal{M}^{\mathbf{r}-(r+1)} \cdot V$
  Déduire n_λ
  Pour tous les blocs $K_{\mathbf{r}}^\lambda$ concernés **faire :**
    Identifier le bloc d'entrée de $B_{\mathbf{r}-1}^i$ et le bloc xoré de $B_{\mathbf{r}-1}^j$
    Déduire n_γ de $V_{\mathcal{F}}(i)$
    Si $V_{\mathcal{F}}(j) \neq -\infty$ **alors :**
      Si $V_{\mathcal{F}}(j) \geq 2$ **alors :**
        Retirer le fragment de la liste $K_{\mathbf{r}}^\lambda$.
      Sinon
        Déduire le nombre des $\Delta_{B_{\mathbf{r}-1}^j}$ possibles.
        Si $\min(W) \leq 1$ **alors :**
          Réduire le nombre des $\Delta_{B_{\mathbf{r}-1}^j}$ possibles.
        Fin de si
        Déduire les différentielles n_Δ.
        Si $n_\Delta \geq \#\{k\}$ **alors :**
          Retirer le fragment de la liste $K_{\mathbf{r}}^\lambda$.
        Fin de si
      Fin de si
    Fin de si
    Calculer n_f avec la formule (22).
  Fin de boucle pour
Fin de boucle pour

Pour chaque bloc étudié comme candidat à la faute injectée, l'algorithme commence par définir le nombre n_λ de blocs de clé attaqués. Puis pour chacun des blocs $K_{\mathbf{r}}^\lambda$, l'algorithme calcule le

nombre de fragments attaqués (optimal et minimal). Ensuite si $V_{\mathcal{F}}(j) \neq -\infty$, c'est à dire si le bloc xoré en sortie de fonction de Feistel $B^j_{\mathbf{r}-1}$ est fauté, une première estimation du nombre de différentielles possibles est donnée. Puis, l'algorithme réutilise le schéma de Feistel pour identifier le blocs le moins influencé (influencé quand même) par les fautes, pour réduire encore le nombre de différentielles possibles. Si le nombre de différentielles ne permet plus d'éliminer des clé, $K^{\lambda,\gamma}_{\mathbf{r}}$ n'est plus considéré comme un fragment de clé attaqué. Pour finir, le nombre de fautes nécessaires n_f pour retrouver les fragments de clé, en fonction du nombre n_Δ (nombre moyen) de différentielles Δ_y, est calculé.

Cet algorithme donne une première idée sur où injecter les fautes, il ne dit pas quel bloc est le meilleur. Rien n'empêche de trouver une attaque encore plus performante qui tire parti de certaines spécificités de l'algorithme, venant améliorer nos résultats. Par ailleurs, une étude similaire avec des fautes multi-bits est à envisager en perspective.

5.2 Résultats sur les exemples

L'algorithme 2 a été codé en magma et a permis de tester notre approche sur différents exemples. Dans cette partie les matrices sont à coefficients dans le dioïde, avec les conventions faites en section 4.1.2. Les $-\infty$ sont remplacés par des $\ll . \gg$ (point) pour plus de lisibilité.

5.2.1 DES

Le DES a été choisi car c'est le plus connu des schémas de Feistel classique.
Dans le cadre de notre étude :
- le nombre de tour est $\mathbf{r} = 16$;
- le nombre de blocs est $\mathbf{b} = 2$;
- le délai de diffusion est $d = 2$;
- la matrice de diffusion $\mathcal{M}$ est la suivante :

$$\mathcal{M} = \begin{pmatrix} . & 0 \\ 0 & 1 \end{pmatrix}$$

- le nombre de blocs de clé à attaquer est : $\Lambda = 1$ ($B^i_{\mathbf{r}-1} = R_{15}$ et $\Delta_{B^j_{\mathbf{r}-1}} = \Delta_{L_{15}}$) ;
- le nombre de fragments de blocs de clé (le nombre de s-box) est : $\Gamma = 8$;
- le nombre moyen de fautes pour retrouver un fragments de clé avec des équations de type (16) est : $n = 3$.

L'étude de la diffusion dans la fonction de Feistel est la suivante :
- du fait de l'expansion E, un bit modifié en entrée de fonction de Feistel peut impacter jusqu'à 2 s-boxes ;
- une des propriétés des s-boxes du DES (P3) exprime le fait que modifier un bit en entrée implique d'en modifier au moins deux en sortie ;
- la permutation P est telle que les bits en sortie des s-boxes d'un tour sont répartis sur 4 s-boxes différentes à l'entrée du tour suivant.

En résumé, si la faute n'est passée dans aucune fonction de Feistel, 2 s-boxes peuvent quand même être impacter. Le passage de la faute dans une seule fonction de Fesitel implique qu'il y a 2 à 8 bits fautés en sortie, ce qui induit $32 * (256 - 1 - 8)$ différentielles possibles.

Inversement, le passage dans une fonction de Feistel est assez caractéristique pour que l'observation des bits fautés en sortie permette de diminuer le nombre des 32 différentielles en entrée à

seulement 2.

Dans l'exemple du DES, on élimine l'étude d'injection de fautes dans L_{15} qui ne permet pas d'attaquer la clé. Les résultats de notre analyse sont dans le Tableau 16 où n_γ est le nombre de fragments de blocs de sous-clé attaqués et n_Δ est le nombre d'hypothèses sur $\Delta_{L_{15}}$.

blocs	$V_\mathcal{F}$	$W_\mathcal{F}$	n_γ	n_Δ
R_{15} or L_{14}	$(.,0)$	$(1,0)$	$1 \leq n_\gamma \leq 2$	1
R_{14} or L_{13}	$(0,1)$	$(2,1)$	$2 \leq n_\gamma \leq 8$	2
R_{13}	$(1,2)$	$(3,2)$	$2 \leq n_\gamma \leq 8$	$32 * 247 = 1504$

Tableau 16 : Résultats sur le DES

Notre étude montre qu'il est plus judicieux d'injecter des fautes dans R_{14} pour le DES. Pourtant, Biham et Shamir ont montré dans [11] que pour le DES, l'attaque est plus efficace en injectant la faute dans R_{13}. Pour justifier ce résultat, ils ont étudié la fonction de Feistel en profondeur, tandis que notre approche est générique. Notre approche confirme que, si nous sommes en mesure de réduire le nombre d'hypothèses sur $\Delta_{L_{15}}$, davantage de fragments de clé sont attaqués en injectant les fautes sur R_{13} plutôt que sur R_{14}.

De plus, l'étude de Rivain dans [18], montre qu'en injectant les fautes au tour 12,11,10 et 9 (au delà du délai de diffusion), l'attaque nécessite davantage de fautes.

5.2.2 MIBS

Cet algorithme est choisi en exemple car comme le DES il n'est découpé qu'en deux blocs, ainsi c'est surtout sur la fonction de Feistel que se joue notre étude. De plus, contrairement au DES, ses s-boxes sont bijectives. Ce qui est plus classique.

Dans le cadre de notre étude :

- le nombre de tour est : $\mathbf{r} = 32$;
- le nombre de blocs est : $\mathbf{b} = 2$;
- le délai de diffusion est : $d = 2$;
- la matrice de diffusion $\mathcal{M}$ est la suivante :

$$\mathcal{M} = \begin{pmatrix} 1 & 0 \\ 0 & . \end{pmatrix}$$

- le nombre de blocs de clé à attaquer : $\Lambda = 1$ ($B^i_{\mathbf{r}-1} = L_{31}$ et $\Delta_{B^j_{\mathbf{r}-1}} = \Delta_{R_{31}}$) ;
- le nombre de fragments de blocs de clé (le nombre de s-boxes) est : $\Gamma = 8$;
- le nombre moyen de fautes pour retrouver un fragment de clé avec des équations de type (16) est : $n = 2$.

La diffusion par la fonction de Feistel préserve le découpage en nibbles défini par les s-boxes de MIBS. L'étude de la diffusion dans la fonction de Feistel est la suivante :

- fauter un bit en entrée implique fauter une s-box et donc jusqu'à 4 bits ;
- si la faute est passée dans une fonction de Feistel, il y a alors 5 ou 6 fragments de clé attaqués ;
- si nous observons un résultat fauté (faute passée une fois dans la fonction de Feistel) nous pouvons en déduire quelle s-box était fautée et par conséquent, il y a 4 bits susceptibles de contenir la faute en entrée ;

– si la faute est passée dans plus d'une fonction de Feistel, tous les fragments sont attaqués.

Les résultats de notre analyse sont dans le Tableau 17 où n_γ est le nombre de fragments de blocs de sous-clé attaqués et n_Δ est le nombre d'hypothèses sur $\Delta_{R_{31}}$. Comme pour le DES, il n'y a qu'un bloc de clé, car il s'agit d'un schéma de Feistel classique.

Blocs	$V_\mathcal{F}$	$W_\mathcal{F}$	n_γ	n_Δ
L_{31} or R_{30}	$(0,.)$	$(0,1)$	1	1
L_{30} or R_{29}	$(1,0)$	$(1,2)$	$5 \leq n_\gamma \leq 6$	4
L_{29}	$(2,1)$	$(2,3)$	8	112

Tableau 17 : Résultats sur le MIBS

Les résultats indiquent qu'il est préférable d'attaquer en L_{30}, mais l'attaquant ne cible pas tous les fragments à la fois.

En L_{29}, au contraire, tous les fragments sont attaqués. Cependant, le nombre de différentielles possibles sur $\Delta_{R_{31}}$ est grand. Une étude plus approfondie en L_{29} permettrait peut-être de diminuer le nombre de différentielles, mais elle n'est pas triviale.

5.2.3 TWINE

Cet exemple est choisi car il est l'exact inverse du DES et de MIBS dans notre étude, la fonction de Feistel ne change rien au découpage toute l'étude se fait sur le schéma de Feistel lui-même.

Dans le cadre de notre étude :

– le nombre de tour est : $\mathbf{r} = 36$;

– le nombre de blocs est : $\mathbf{b} = 16$;

– le délai de diffusion est : $d = 8$;

– la matrice de diffusion $\mathcal{M}$ est la suivante :

$$\begin{pmatrix}
1 & 0 & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & 0 & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & 1 & 0 & . & . & . & . \\
. & . & . & . & . & . & 0 & . & . & . & . & . & . & . & . & . \\
. & . & 1 & 0 & . & . & . & . & . & . & . & . & . & . & . & . \\
0 & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & 1 & 0 & . & . & . & . & . & . \\
. & . & . & . & 0 & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & 1 & 0 & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & 0 & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & 1 & 0 & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & 0 & . \\
. & . & . & . & 1 & 0 & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & 0 & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & 1 & 0 \\
. & . & . & . & . & . & . & . & . & . & . & . & 0 & . & . & .
\end{pmatrix}$$

– le nombre de blocs de clé à attaquer est : $\Lambda = 8$;

– le nombre de fragments de blocs de clé (le nombre de s-box) $\Gamma = 1$;

– le nombre moyen de fautes pour retrouver un fragment de clé avec des équations de type (16) est : $n = 2$.

La fonction de Feistel de TWINE n'est composée que d'un xor de clé et d'une seule s-box ($n_\gamma = 1$). De ce fait, l'étude de celle-ci est très simple : soit le bloc est fauté soit il ne l'est pas. De plus, si nous observons la sortie d'un bloc fauté par une faute passée dans une seule fonction de Feistel, cela laisse les 4 bits en entrée possibles. Donc, si la faute de $B^{j*}_{\mathbf{r}-1}$ n'est passée que dans une seule fonction de Feistel, il y a $n_\Delta = 14$ différentielles possibles, voire 7 si la faute est connue.

Par conséquent, il existe trois cas favorables :

1. $B^j_{\mathbf{r}-1}$ n'est pas fauté : $n_\Delta = 1$;
2. $B^j_{\mathbf{r}-1}$ contient exactement la valeur de la faute : $n_\Delta = 1$ si la valeur de la faute est connue, sinon $n_\Delta = 4$;
3. $B^j_{\mathbf{r}-1}$ est passé dans exactement une fonction de Feistel : $n_\Delta = 7$ si la valeur de la faute est connue, sinon $n_\Delta = 14$.

Les résultats détaillés sont donnés en annexes E.i, car il y a beaucoup de cas à énumérer (8 tours et 8 blocs soient 64 cas).

Le meilleur registre pour l'injection de fautes sur TWINE, est alors celui où les fautes atteignent la plupart des fonctions de Feistel ; à condition que les différentielles respectives soient dans l'un des trois cas ci-dessus.

Notre algorithme propose les B^i_{31} en l'entrée des fonctions de tours comme meilleur candidat. De cette façon, il est possible d'attaquer simultanément $n_\lambda = 5$ blocs de clé sur le dernier tour. Dans cette configuration, quatre différentielles $\Delta_{B^j_{\mathbf{r}-1}}$ sont connues ($n_\Delta = 1$) et pour le dernier cas : $n_\Delta = 7$.

Injecter les fautes plus tôt dans l'algorithme (tour 30 et avant) ne permet d'attaquer au plus que 4 blocs de clé, ce qui n'est pas aussi intéressant que le cas précédent. En effet, plus de blocs sont fautés, cependant le nombre de différentielles possibles n'est plus réduit significativement.

Injecter les fautes plus tard dans l'algorithme (tour 32 et après) ne permet d'attaquer que 3 blocs de clé à la fois.

5.2.4 CLEFIA

CLEFIA est choisi comme exemple de « *juste milieu* » entre les schémas de Feistel classiques à 2 blocs et fonction de Feistel conséquente et les GFN à nombreux blocs et petite fonction de Feistel. L'étude se fait avec de la diffusion dans la fonction de Feistel et étude de la diffusion au niveau du schéma de Feistel. Il est l'exemple type où notre étude doit habilement mélanger la diffusion dans la fonction de Feistel et étude de la diffusion au niveau du GFN.

Dans le cadre de notre étude :

– le nombre de tour est : $\mathbf{r} = 18$ ou 22 ou 26, cela dépend de la taille de la clé, pour notre étude nous avons choisi $\mathbf{r} = 18$;
– le nombre de blocs est : $\mathbf{b} = 4$;
– le délai de diffusion est : $d = 4$:

– la matrice de diffusion $\mathcal{M}$ est la suivante :

$$\mathcal{M} = \begin{pmatrix} 1 & 0 & . & . \\ . & . & 0 & . \\ . & . & 1 & 0 \\ 0 & . & . & . \end{pmatrix}$$

– le nombre de blocs de clé à attaquer est : $\Lambda = 2$;
– le nombre de fragments de blocs de clé (le nombre de s-box) est : $\Gamma = 4$;
– e nombre moyen de fautes pour retrouver un fragment de clé avec des équations de type (16) est : $n = 2$.

L'étude de la diffusion dans la fonction de Feistel est la suivante :
– chaque bit d'entrée de la fonction impacte exactement une seule s-box.
– modifier un bit en entrée signifie modifier 1 à 8 bits en sortie (car les s-boxes sont bijectives) ;
– si la faute a traversé une fonction de Feistel, alors toutes les s-boxes sont impactées (par les propriétés de diffusion des matrices MC_0 et MC_1 de CLEFIA) ;
– en observant une sortie de fonction, et supposant qu'un seul bit d'entrée est fauté, il est possible de déduire quelle s-box est fautée et ainsi la faute d'entrée est réduite à 8 possibilités.

Les résultats détaillés sont donnés en annexes E.ii, un résumé est en Tableau 18.

Blocs B	$V_{\mathcal{F}}$	$W_{\mathcal{F}}$	n_λ	n_γ	n_Δ
B_{17}^0	$(0,.,.,.)$	$(0,1,.,.)$	1	$(1,0)$	$(1,-)$
B_{16}^0	$(1,.,.,0)$	$(1,2,.,0)$	1	$(4,0)$	$(1,-)$
B_{15}^0	$(2,.,0,1)$	$(2,3,0,1)$	2	$(4,1)$	$(1,\ \leq 127)$
B_{14}^0	$(3,0,1,2)$	$(3,4,1,2)$	2	$(4,4)$	$(4,\text{grand})$
B_{13}^0	$(4,1,2,3)$	$(4,5,2,3)$	2	$(4,4)$	$(946,\text{grand})$

Tableau 18 : Résultats de l'analyse sur CLEFIA

Les résultats indiquent que l'injection dans B_{17}^0 n'est pas optimale. En effet, si l'injection est faite un tour plus tôt B_{16}^0, cela permet d'attaquer plus de blocs de clés simultanément. Ce constat a déjà été exposé dans l'attaque menée par Chen *et al.* [96] où 18 fautes sont nécessaires pour réaliser l'attaque.

Si un attaquant souhaite attaquer deux blocs de clés en même temps, l'un des deux a une différentielle non triviale. Le cas B_{15}^0 est encore gérable, mais une seule s-box de la deuxième fonction est attaquée, c'est le choix de Takahashi et Fukunaga dans [97]. L'étude approfondie des fonctions de CLEFIA leur permet de réduire significativement le nombre de fautes nécessaires pour réaliser leur attaque. Ce résultat est en adéquation avec notre analyse et permet de conclure qu'il est préférable d'injecter la faute sur B_{15}.

D'autre part, une attaque pourrait être menée en ciblant B_{14}^0, cela permettrait d'attaquer les 8 blocs à la fois, pour cela, une meilleure étude de la fonction de Feistel reste à mener.

6 Conclusion

Les travaux de ce chapitre ont été publiés à FDTC 2014.

Il a été mis en évidence que certains blocs des schémas de Feistel généralisés, sont plus pertinents que d'autres dans le choix d'injection de fautes mono-bit, en vue de retrouver la clé de chiffrement. Une méthode générique pour identifier ces blocs a été présentée. Elle peut être utilisée dans le cas où l'attaquant est face à un algorithme de type GFN peu connu. Cette étude peut être également utilisée par les concepteurs afin d'anticiper les attaques et de protéger leur circuit en conséquence.

Le caractère générique de la méthode proposée implique que l'évaluation de la vulnérabilité n'est pas optimale, mais elle peut être affinée par des connaissances précises sur l'algorithme ciblé.

Cette méthode a été implémentée en Magma et ainsi testée sur différents algorithmes (DES, TWINE, MIBS et CLEFIA). Les résultats obtenus montrent que notre étude est donc en adéquation avec les travaux déjà menés sur ces différents exemples.

Les perspectives pour cette étude sont les suivantes :
- étendre l'étude aux fautes multi-bits ;
- étendre l'étude à la rétro-conception (les résultats du chapitre suivant montrent que l'étude fonctionne dans le cas du DES).
- affiner l'analyse de la diffusion dans la fonction de Feistel ;
- étendre l'étude aux schémas de Feistel tel que Piccolo où la clé est xorée en fin de fonction de Feistel.

Partie V

Nouvelle attaque de type FIRE sur pseudo-DES

Sommaire

1 État de l'Art sur la rétro-conception sur de pseudo-DES

Notre étude porte sur la recherche de s-boxes d'un pseudo-DES. Retrouver une s-box S signifie pour toute entrée x, connaître la sortie y qui lui est associée, $S(x) = y$. La clé de chiffrement est connue.

Toujours dans l'idée de limiter le nombre de fautes injectées, comme dans le chapitre IV, l'objectif est de trouver un registre optimisé dans lequel injecter des fautes mono-bit. Comme expliqué précédemment dans cette thèse, le but est de limiter le nombre d'injections de fautes pour réaliser l'attaque. Ce qui joue un rôle capital dans les attaques FIA, comme l'expliquent Sakiyama *et al.* dans [44]. Pour la rétro-conception, cette optimisation est d'autant plus cruciale ; en effet l'attaquant peut n'avoir qu'un seul circuit à disposition. Si le circuit se retrouve endommagé par l'injection de fautes, l'attaque est de facto considérée comme échouée.

1.1 Attaque SCARE

Avant de parler de l'attaque FIRE sur pseudo DES, l'attaque SCARE sur pseudo-DES est présentée à l'aide du formalisme. Cette attaque peut permettre de finaliser une attaque FIRE. Dans ce cas, il s'agit d'une attaque à deux niveaux ; cet exemple d'attaque est illustré en annexe B.ii.c. L'attaque de type SCARE sur le DES a été présentée dans [98].

La différence avec une attaque SCA classique réside dans la nature et la taille de la cible. Les s-boxes du DES sont des fonctions booléennes $b_{6\to4}$, ce qui donne $2^{4\cdot2^6} = 2^{256}$ hypothèses. Il n'est alors pas possible de prendre en considération toutes les hypothèses. L'astuce présentée dans l'attaque SCARE est de décomposer les fonctions s-boxes en fonctions booléennes $b_{2\to1}$; pour cela un masque doit être appliqué en entrée et un autre en sortie. Il existe exactement 16 fonctions booléennes $b_{2\to1}$, ce qui rend le « diviser pour régner »possibles. Ces fonctions présentées par Knuth dans [99], sont décrites dans le Tableau 19.

indices de b	$b(1,1)$	$b(1,0)$	$b(0,1)$	$b(0,0)$	Nom	Équation booléenne
0	0	0	0	0	Zero	0
1	0	0	0	1	Nor2	$\overline{A+B}$
2	0	0	1	0	And2B	$A\cdot\overline{B}$
3	0	0	1	1	NotB	$\overline{B}$
4	0	1	0	0	And2A	$B\cdot\overline{A}$
5	0	1	0	1	NotA	$\overline{A}$
6	0	1	1	0	Xor2	$A\oplus B$
7	0	1	1	1	Nand2	$\overline{A\cdot B}$
8	1	0	0	0	And2	$A\cdot B$
9	1	0	0	1	XNor2	$\overline{A\oplus B}$
10	1	0	1	0	A	A
11	1	0	1	1	Or2B	$A+\overline{B}$
12	1	1	0	0	B	B
13	1	1	0	1	Or2A	$B+\overline{A}$
14	1	1	1	0	Or2	$A+B$
15	1	1	1	1	One	1

Tableau 19 : Les fonctions booléennes $b_{2\to1}$

- Les s-boxes décomposées en plusieurs fonctions booléennes $b_{2\to1}$.
  La cible est une fonction booléenne $b_{2\to1}$. $\mathcal{K} = b_{2\to1}$.
  L'ensemble des hypothèses possibles $\mathbb{K}$ est l'ensemble des 16 fonctions booléennes décrites dans le Tableau 19.
- $O_S = C$, l'observable stimulus est le texte chiffré.
- $O_R = Conso$, l'observable réaction est une mesure de consommation de courant.
- $\mathcal{R} = f \circ g$ avec :
  $f = conso$ consommation de courant.
  $g = (\oplus K_{16}) \circ E \circ IP$
  Le chemin d'attaque est illustré en Figure 34.
- $\mathcal{R}_m = m \circ g$, avec :
  $m = HW$.
- Le distingueur est une corrélation.

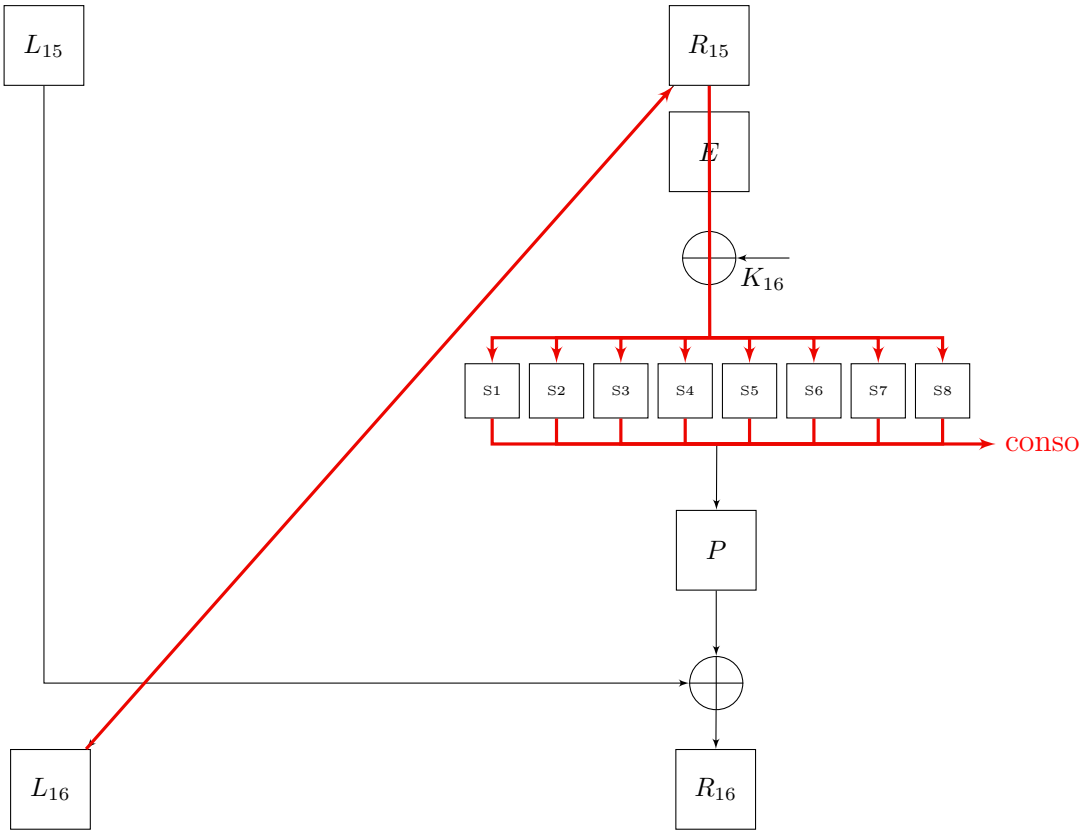


Figure 34 : Chemin d'attaque de SCARE

1.2 Première attaque FIRE

L'attaque présentée dans ce chapitre, est une amélioration de l'attaque FIRE de San Pedro *et al.* [100]. Une attaque similaire a été présentée dans [101] sur des schémas de Feistel, cependant leur étude n'inclut pas le DES. Leur résultats sont évalués sur un Twofish-like crypto-système. L'algorithme d'origine, le Twofish est décrit dans [102]. En moyenne, ils ont besoin de 234 fautes mono-octet.

Dans l'attaque de San-Pedro, une faute mono-bit est injectée dans R_{15}. Par conséquent il y a au plus 2 bits de différence, entre x l'entrée des s-boxes et x^* l'entrée fautée des s-boxes. Comme pour les attaques de DFA classique, C et C^* sont connus donc L_{16}, R_{16}, L^*_{16} et R^*_{16} aussi. La clé n'étant pas une inconnue, les entrées des s-boxes x et x^* sont connues.

$$\left\{ \begin{array}{lcccl} x & = & E(R_{15}) \oplus K_{16} & = & E(L_{16}) \oplus K_{16} \\ x^* & = & E(R^*_{15}) \oplus K_{16} & = & E(L^*_{16}) \oplus K_{16} \end{array} \right. \tag{23}$$

Les sortie fautées et non fautées (y and y^*) des s-boxes ne peuvent être calculées car L_{15} est inconnu.

$$y = P^{-1}(R_{16} \oplus L_{15})$$
$$y^* = P^{-1}(R^*_{16} \oplus L_{15})$$

Cependant, L_{15} n'étant pas fauté, la différentielle $\Delta_y = y \oplus y^*$ en sortie des s-boxes peut être calculée :

$$\Delta_y = P^{-1}(R_{16} \oplus R^*_{16} \oplus L_{15} \oplus L_{15}) \quad .$$

Donc :

$$\Delta_y = P^{-1}(R_{16} \oplus R^*_{16}) \quad .$$

Ainsi, dans une attaque de type FIRE seules les entrées x, x^* et la différentielle en sortie Δ_y des s-boxes sont connues. Pour chacune des 8 s-boxes, nous avons $S_i(x_i) \oplus S_i(x^*_i) = \Delta_{y_i}$, pour $i \in [\![1, 8]\!]$.

En considérant la relation entre deux entrées de s-boxes, nous avons :

$$S_i(x_{i_0}) \oplus S_i(x_{i_1}) = \Delta_{y_i} \quad . \tag{24}$$

Certaines relations mathématiques permettent d'améliorer l'attaque. Soient x_{i_0}, x_{i_1} et x_{i_2} trois entrées de s-box S_i, alors :

$$\left. \begin{array}{l} S_i(x_{i_0}) \oplus S_i(x_{i_1}) = \Delta_{y_i} \\ S_i(x_{i_0}) \oplus S_i(x_{i_2}) = \Delta'_{y_i} \end{array} \right\} \Rightarrow S_i(x_{i_1}) \oplus S_i(x_{i_2}) = \Delta_{y_i} \oplus \Delta'_{y_i} \quad . \tag{25}$$

Les relations (24) ne permettent pas de retrouver y_i and y^*_i. L'attaque ne permet pas de définir les fonctions s-boxes. Les s-boxes sont dites **définies à une translation près**. Pour une s-box donnée, la connaissance d'un couple entrée sortie (x_i, y_i) suffit alors à définir toute la s-box. L'attaque doit encore tester 16 valeurs pour une sortie de chaque s-box, soit $16^8 = 2^{32}$ hypothèses pour retrouver toutes les s-boxes. Pour cette raison, une attaque FIRE sur les s-boxes d'un DES doit finir en exhaustive ou en SCARE (voir annexe B ii.c).

Dans l'attaque [100] pour retrouver une s-box à une translation près, 130 fautes sont nécessaires, soit au total $130 \cdot 8 = 1040$ fautes pour définir à une translation près les 8 s-boxes.

En reprenant les notations du Formalisme, la description de l'attaque FIRE de San Pedro *et al.* [100] est la suivante.

- $\mathcal{K} = \Delta_y$, où $\Delta_y = S(x \oplus x^*)$
  Il y a 15 hypothèses k.
- $O_S = (L_{16}, L_{16}^*)$, l'observable stimuli est la partie de gauche du couple chiffré, chiffré fauté, après IP.
- $O_R = R_{16}^* \oplus R_{16}$, l'observable réaction est la différentielle de la partie de droite après IP.
- $\mathcal{R} = g \circ f$
  avec $f = \lightning$ processus d'injection de fautes et
  $g = P \circ (\oplus K_{16}) \circ E$
  Le chemin d'attaque est illustré en Figure 35.
- $\mathcal{R}_m = g \circ m_i$, avec :
  $m_i = \oplus 2^i$, $i \in [\![0, 5]\!]$.
- Le distingueur est un crible.

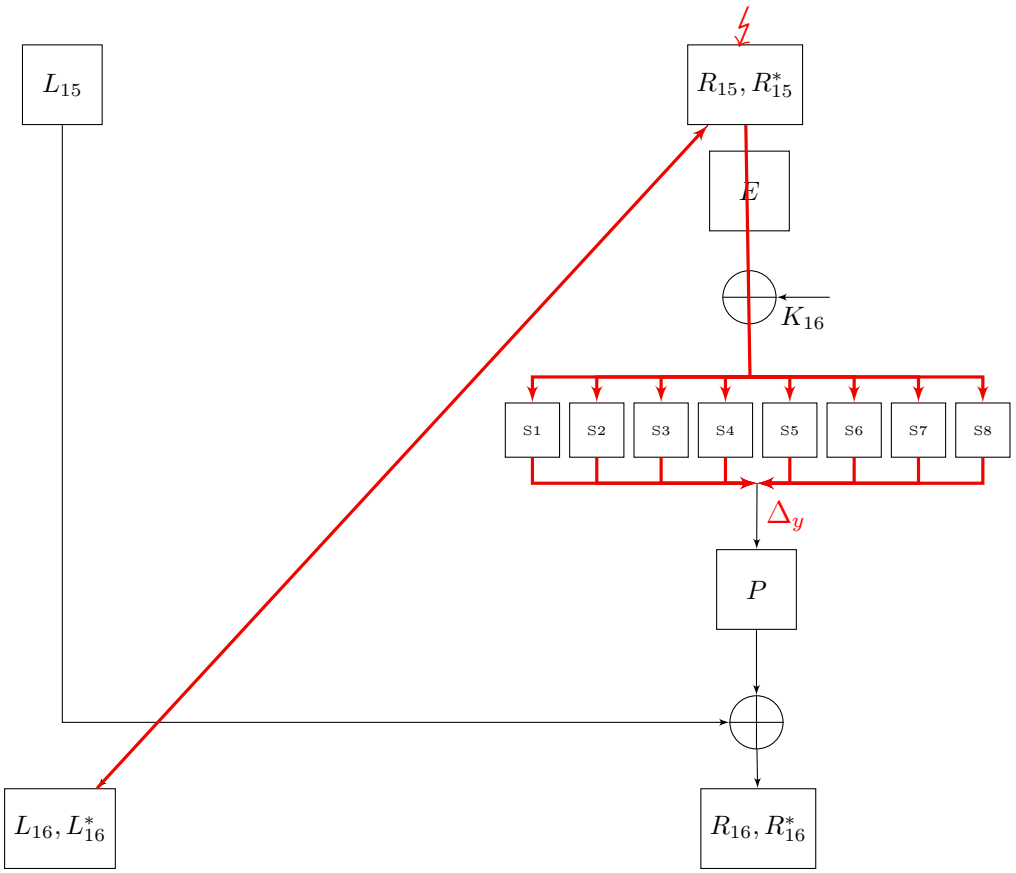


Figure 35 : Chemin d'attaque de FIRE

2 Notre attaque FIRE

Dans cette section, nous présentons en détails notre attaque améliorée, qui a pour but de limiter le nombre de fautes mono-bits injectées.

2.1 Injection de fautes sur R_{14} pour une meilleure diffusion

Dans le chapitre précédent (IV), il a été montré qu'il est plus intéressant d'attaquer la clé du DES sur R_{14} que sur R_{15}. Cette étude étend le résultat à la rétro-conception des s-boxes.

La valeur de la faute injectée sur R_{14}^* est mono-bit et notée e.

$$e = R_{14} \oplus R_{14}^* \quad .$$

La Figure 36 présente les deux derniers tours du DES et l'emplacement de la faute.

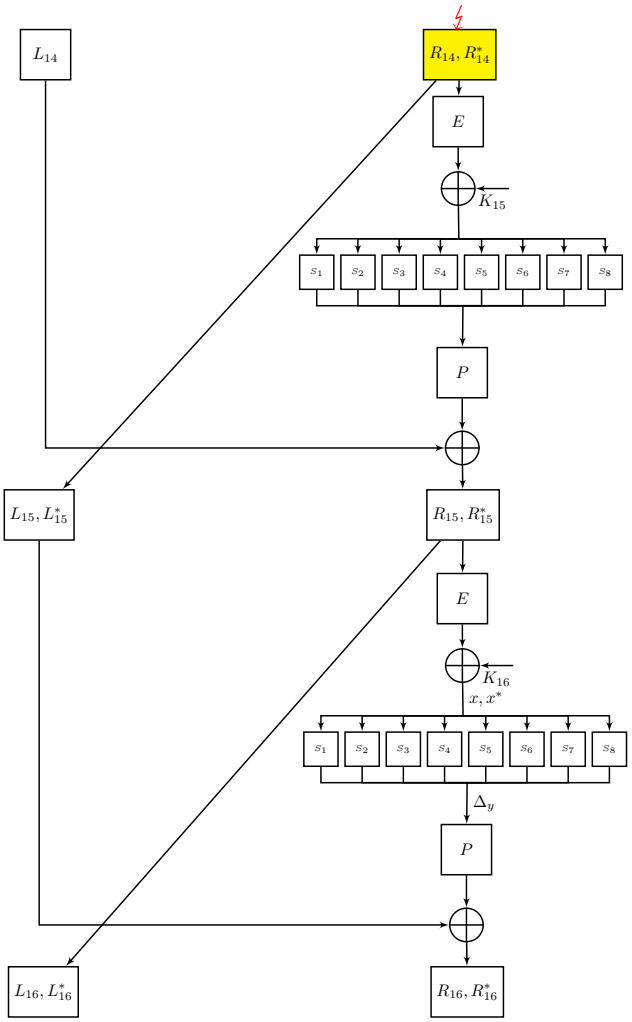


Figure 36 : Injection de la faute dans R_{14}

Les différentes étapes de la diffusion des bits sont détaillées et illustrées en Figure 37.

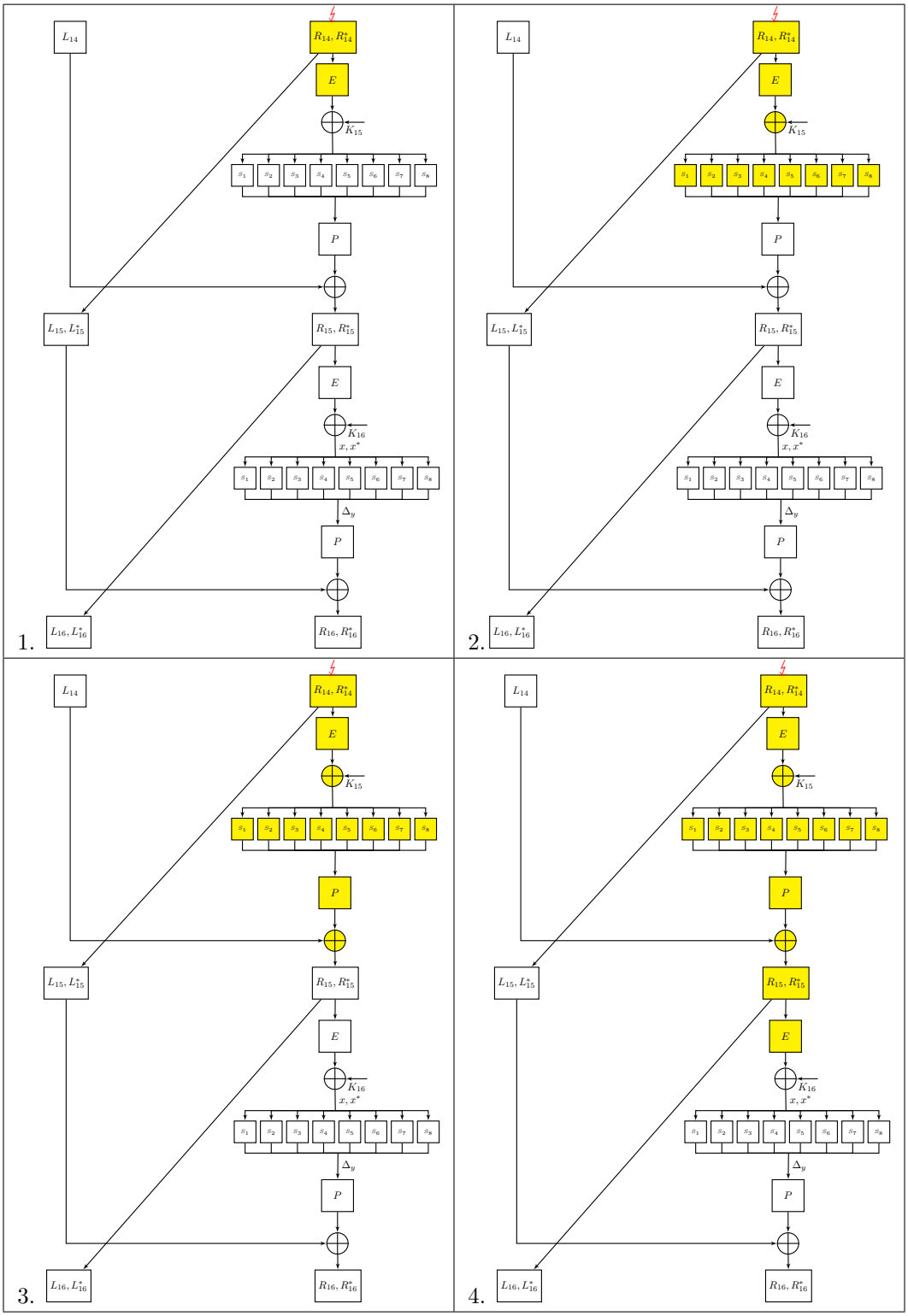


Figure 37 : Les différentes étapes importantes de la diffusion de la faute injectée

1. L'expansion E duplique la moitié des bits de R_{14} (Case 1 de la Figure 37). Deux s-boxes peuvent avoir une entrée fautée au tour 15.
2. Un bit en sortie des s-boxes peut fauter jusqu'à 4 bits en sortie. Combiné à l'expansion, il peut y avoir jusqu'à 8 bits fautés en sortie des s-boxes du tour 15 (Case 2 de la Figure 37).
3. La permutation P du tour 15 vient mélanger les 8 bits qui peuvent être fautés et les répartir sur les différentes s-boxes du tour suivant (Case 3 de la Figure 37).
4. L'expansion du tour 16 va dupliquer à nouveau les bits fautés, soient jusqu'à 16 bits fautés (Case 4 de la Figure 37).

Injecter la faute sur R_{14} permet bien de diffuser à toutes les s-boxes du tour 16.

De plus une faute mono-bit est rarement complètement uniforme. Si certains bits sont moins souvent impactés par les injections de fautes, il est alors possible que certaines s-boxes soient plus compliquées à retrouver que d'autres, dans l'attaque FIRE initiale [100]. En injectant la faute mono-bit sur R_{14} ce problème non négligeable disparaît grâce à la diffusion.

2.2 Les inconvénients de la diffusion de la faute : une différentielle de sortie inconnue.

Comme expliqué dans le chapitre IV, les attaques par injection de fautes de type DFA, dans les schémas de Feistel, se font sur le dernier tour. Les entrées des s-boxes x et x^*, R_{16}, L_{16}, R^*_{16} et L^*_{16} sont connues et calculées comme dans l'équation (23). Cependant, la faute est diffusée en L^*_{15}. Ce problème est illustré en Figure 38a.

$$R^*_{14} = L^*_{15} \neq L_{15}$$

donc :

$$\begin{aligned} \Delta_y &= P^{-1}(R_{16} \oplus R^*_{16} \oplus L_{15} \oplus L^*_{15}) \\ &= P^{-1}(R_{16} \oplus R^*_{16} \oplus e) \quad . \end{aligned}$$

Δ_y est inconnue. Nous calculons alors $\Delta_{R_{16}}$.

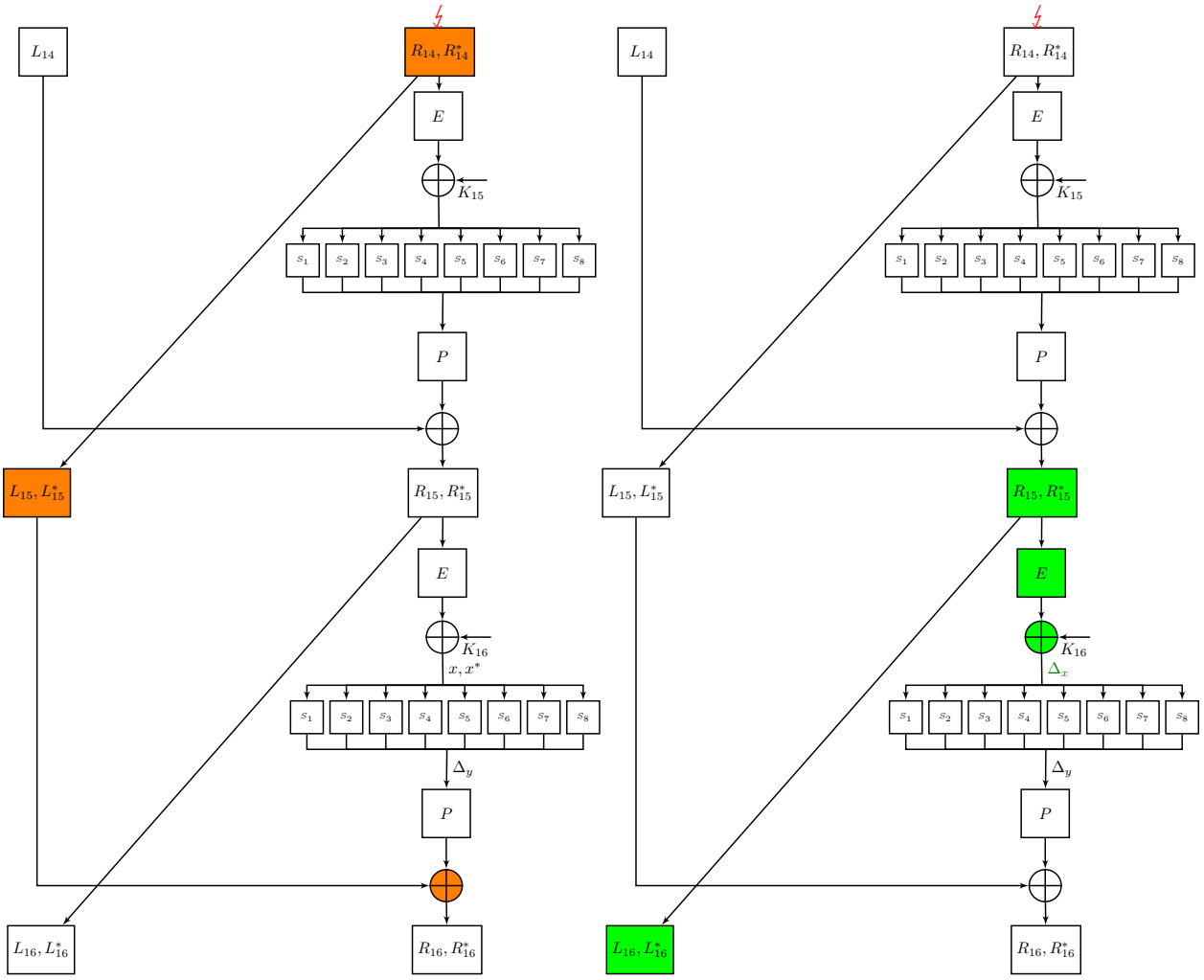
$$\Delta_z = P^{-1}(\Delta_{R_{16}}) = P^{-1}(R_{16} \oplus R^*_{16}) \tag{26}$$

Du fait que $L^*_{15} = L_{15} \oplus e$ et e non nulle, $\Delta_z \neq \Delta_y$. Pour reprendre les notations du chapitre sur les schémas de Feistel : $V_{\mathcal{F}} = (0, 1)$. La faute a juste été recopiée sur L_{15}.

$$\Delta_y = \Delta_z \oplus P^{-1}(e) \tag{27}$$

Retrouver e n'est pas impossible. En effet, en reprenant les notations du chapitre précédent (IV), nous observons $W_{\mathcal{F}} = (1, 2)$. L'étude est donc possible.

Nous nous intéressons aux $\Delta_{x_i} = x_i \oplus x^*_i$, les différentielles en entrée des s-boxes. Elles sont connues comme le montre la Figure 38b. Plusieurs configurations sont possibles, celle où la faute est retrouvée et celle où la faute n'est pas retrouvée.



(a) Le problème de la faute diffusée sur L_{15} (b) Δ_x est connue

Figure 38 : Connaissance au dernier tour

2.2.1 Cas où e est découverte

Pour une s-box S_i donnée, si les entrées x_i et x_i^* sont égales, alors $y_i = y_i^*$, autrement dit :

$$\Delta_{x_i} = 0 \Rightarrow \Delta_{y_i} = 0 \quad . \tag{28}$$

Soit Δ_{z_i} les 4 bits de Δ_z au niveau de la s-box S_i. Si $\Delta_{x_i} = 0$ et $\Delta_{z_i} \neq 0$, alors $e = P(\Delta_{z_i})$. Le bit fauté est le bit de Δ_{z_i} qui est égal à 1 à la permutation P^{-1} près. Alors les différentielles de sortie Δ_{y_i} de toutes les s-boxes sont connues. En pratique, nous avons observé que pour 1000 fautes mono-bit et aléatoires sur R_{14}, ce cas arrive dans 56% des cas.

2.2.2 Cas où e reste inconnue

Dans ce paragraphe les fautes possibles sont listées. L'hypothèse d'erreur e tel que $HW(e_j) = 1$ est notée e_j, plus précisément seul le bit j est à 1 :

$$e = (\underbrace{0 \cdots 0}_{0 \cdots j-1} \underbrace{1}_{j} \underbrace{0 \cdots 0}_{j+1 \cdots 31})_2$$

Il s'agit du cas où il n'y a pas de couple : (différentielle en entrée nulle, sortie non nulle) *c.à.d* $\Delta_{x_i} = 0$ avec $\Delta_{z_i} \neq 0$. Il est possible de remonter la différentielle jusqu'à la sortie des s-boxes de l'avant dernier tour, comme illustré dans la Figure 39. Ceci est possible car L_{14} n'est pas fauté.

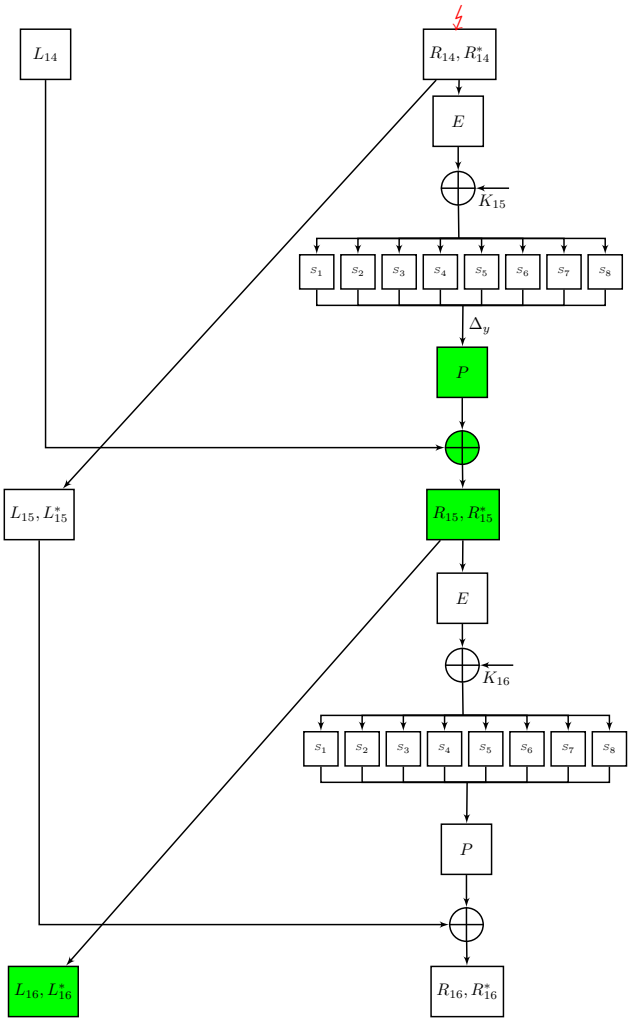


Figure 39 : Remontée au tour 15

La différentielle $\Delta_{x_i} \neq 0$ indique quelles s-boxes ont une sortie fautée à l'avant dernier tour. Le Tableau 20 présente une sorte de diffusion inversée, pour chacun des bits en entrée des s-boxes du tour 16. Cela permet de savoir quelles s-boxes sont fautées au tour 15.

round 16 \ bits	1	2	3	4	5	6
Δ_{x_1}	S_7	S_4	S_2	S_5	S_6	S_8
Δ_{x_2}	S_6	S_8	S_3	S_7	S_5	S_1
Δ_{x_3}	S_5	S_1	S_4	S_6	S_7	S_2
Δ_{x_4}	S_7	S_2	S_5	S_8	S_3	S_1
Δ_{x_5}	S_3	S_1	S_2	S_6	S_4	S_8
Δ_{x_6}	S_4	S_8	S_7	S_1	S_3	S_5
Δ_{x_7}	S_3	S_5	S_4	S_8	S_2	S_6
Δ_{x_8}	S_2	S_6	S_3	S_1	S_7	S_4

Tableau 20 : Provenance des 6 bits de Δ_{x_i} en fonction des s-boxes du tour 15.

Finalement il y a deux cas : une ou deux s-boxes ont une sortie fautée au tour 15.

a) Deux s-boxes ont une sortie fautée

Une s-box ayant une sortie fautée implique obligatoirement une entrée fautée. Si l'attaquant observe deux s-boxes avec des sorties fautées à l'avant dernier tour, uniquement 2 bits de R_{14}^* peuvent être à l'origine de ces sorties fautées, elles sont notées j_1 et j_2. Le Tableau 21 indique la répartition des bits d'un registre R en fonction des entrées des s-boxes du même tour. Il y a deux erreurs possibles sur R_{14}^* qui sont : e_{j_1} and e_{j_2}. Soient deux hypothèses pour Δ_y au dernier tour :

$$\Delta_y = \Delta_z \oplus P^{-1}(e_{j_1}) \text{ ou } \Delta_z \oplus P^{-1}(e_{j_2}) \quad .$$

Nous remarquons que sur les 8 s-boxes, 6 différentielles Δ_{y_i} sont connues.

S_1	32	1	2	3	4	5
S_2	4	5	6	7	8	9
S_3	8	9	10	11	12	13
S_4	12	13	14	15	16	17
S_5	16	17	18	19	20	21
S_6	20	21	22	23	24	25
S_7	24	25	26	27	28	29
S_8	28	29	30	31	32	1

Tableau 21 : Expansion E : les bits en fonction des entrées des s-boxes

Prenons un exemple pour illustrer nos propos, supposons que les bits suivants ne sont pas nuls : 6 de Δ_{x_1}, 4 de Δ_{x_2}, 1 et 4 de Δ_{x_4}. Le Tableau 20 indique qu'au tour 15 les s-boxes S_7 et S_8 ont une sortie fautée. Le Tableau 21 indique que seuls les bits 28 et 29 de $R14$ sont à la fois en entrée de S_7 et de S_8. Il n'y a que deux fautes possibles e : e_{28} ou e_{29}. La permutation inverse nous donne la position de ces bits dans Δ_z.

$$P^{-1}(\{28, 29\}) = \{6, 22\}$$

Ainsi il y a bien 2 différentielles possibles :

$$\Delta_y = \Delta_z \oplus P^{-1}(e_{28}) \text{ ou } \Delta_z \oplus P^{-1}(e_{29}) \quad .$$

De plus pour S_i, $i \in \{1,3,4,5,7,8\}$, $\Delta_{y_i} = \Delta_{z_i}$.

b) Une seule s-box a une sortie fautée

A l'avant dernier tour, observer une seule s-box avec une sortie fautée, ne veut pas dire qu'une seule s-box a une entrée fautée. En effet, les s-boxes du DES sont de 6 bits vers 4, ainsi elles ne sont pas bijectives puisque l'ensemble d'arrivée est plus petit que l'ensemble de départ. Une s-box peut être fautée en entrée et pas en sortie. Le Tableau 21 indique qu'une s-box partage deux bits d'entrée avec ses voisines. Ainsi, l'observation d'une sortie de s-box fautée S_i peut en réalité correspondre à deux s-boxes fautées en entrée : soit S_{i-1} et S_i ou S_i et S_{i+1} [1]. Par conséquent, il y a 6 bits fautés possibles sur R_{14} : e_{j_1}, e_{j_2}, e_{j_3}, e_{j_4}, e_{j_5} et e_{j_6}. Il s'agit des 6 bits d'entrée de S_i. Soient deux hypothèses pour Δ_y au dernier tour :

$$\Delta_y = \Delta_z \oplus P^{-1}(e_{j_1}) \text{ ou } \Delta_z \oplus P^{-1}(e_{j_2}) \text{ ou } \cdots \text{ ou } \Delta_z \oplus P^{-1}(e_{j_6})$$

Nous remarquons que sur les 8 s-boxes, 2 différentielles Δ_{y_i} sont connues.

Par exemple, si nous observons que les bits suivants sont non nuls : 6 de Δ_{x_4}, 2 de Δ_{x_5} et 3 de Δ_{x_7}. Le Tableau 20 indique qu'au tour 15, seule S_1 a une sortie fautée. Le Tableau 21 indique que les bits $\{32,1,2,3,4,5\}$ de R_{14} définissent l'entrée de S_1. Il y a 6 fautes possibles e : e_{32}, e_1, e_2, e_3, e_4 et e_5. La permutation inverse nous donne la position de ces bits dans Δ_z.

$$P^{-1}(\{32,1,2,3,4,5\}) = \{25,16,7,20,21,29\}$$

Ainsi, il y a bien 6 différentielles possibles :

$$\Delta_y = \begin{cases} \Delta_z \oplus P^{-1}(e_1) \text{ ou} \\ \Delta_z \oplus P^{-1}(e_2) \text{ ou} \\ \Delta_z \oplus P^{-1}(e_3) \text{ ou} \\ \Delta_z \oplus P^{-1}(e_4) \text{ ou} \\ \Delta_z \oplus P^{-1}(e_5) \text{ ou} \\ \Delta_z \oplus P^{-1}(e_{32}) \end{cases} .$$

De plus pour S_i, $i \in \{1,3\}$, $\Delta_{y_i} = \Delta_{z_i}$.

1. Évidemment la numérotation des s-boxes est circulaire, *c.à.d* $S_{1-1} = S_8$ et $S_{8+1} = S_1$. En d'autres termes après S_8 nous reprenons à S_1

2.3 Utilisation des propriétés des s-boxes

Les s-boxes d'un pseudo DES ne sont pas a priori, n'importe quelle fonction booléenne $b_{\{0,1\}^6 \to \{0,1\}^4}$. C'est pourquoi, nous pouvons prendre en compte quelques propriétés des s-boxes afin d'améliorer notre attaque. Les propriétés utilisées sont P1 et P3 (introduite au chapite I, section 1.4.1.b).

P3 implique qu'une faute mono-bit en entrée de s-box change au minimum 2 bits en sortie. Dans le paragraphe 2.2.2.b, cela réduit le nombre de bits potentiellement fautés de 6 à 2. En effet, le cas où une s-box est fautée en entrée et non en sortie n'existe plus.

En reprenant l'exemple du paragraphe 2.2.2. a. Il n'y a plus que 2 fautes possibles e : e_2 e_3 qui sont les deux bits se trouvant exclusivement en entrée de S_1. La permutation inverse nous donne la position de ces bits dans Δ_z.

$$P^{-1}(\{2,3\}) = \{7,20\}$$

Ainsi, il y a 2 différentielles possibles :

$$\Delta_y = \Delta_z \oplus P^{-1}(e_2) \text{ ou } \Delta_z \oplus P^{-1}(e_3) \quad .$$

De plus pour S_i, $i \in \{1,3,\}$, $\Delta_{y_i} = \Delta_{z_i}$. Cette propriété P3 simplifie l'étude, il y a au maximum deux différentielles possibles Δ_y.

La propriété P1 implique que dans une s-box, sur une même ligne, toutes les valeurs sont différentes. Si 15 cases d'une ligne sont connues, la valeur de la 16ème case manquante est la valeur manquante de l'intervalle entier $[\![0,15]\!]$. La propriété P1 réduit le nombre de s-boxes possibles à $(16!)^{4\cdot 8} \approx 2^{1376}$. Il est naturel de penser que $4 \cdot 16 - 1 = 63$ relations sont nécessaires pour définir une s-box à une translation près. Cependant ,P1 implique que 14 relations adéquates suffisent à définir une ligne. Finalement, $4 \cdot 14 = 56$ relations sont suffisantes.

P1 permet également d'éliminer de fausses hypothèses de différentielles Δ_{y_i}. En effet, soient x_{i_0} et x_{i_1} deux entrées associées à la même ligne telles que :

$$S_i(x_{i_0}) = y_{i_0} \neq y_{i_1} = S(x_{i_1}) \quad .$$

$\forall x_{i_2}$, tel que $S_i(x_{i_2}) = y_{i_2}$, nous avons :

$$S_i(x_{i_0}) \oplus S_i(x_{i_2}) = y_{i_0} \oplus y_{i_2} \neq y_{i_1} \oplus y_{i_2} = S_i(x_{i_1}) \oplus S_i(x_{i_2}) \quad .$$

En pratique pour une relation, telle que $x_{i_0} \neq x_{i_1}$ les propriétés suivantes sont vérifiées :

– si x_{i_0}, x_{i_1} sont sur une même ligne et x_{i_2} une entrée différente de x_{i_0} et x_{i_1} alors :

$$S_i(x_{i_0}) \oplus S_i(x_{i_1}) = \Delta_{y_i} \Rightarrow \begin{cases} S_i(x_{i_0}) \oplus S_i(x_{i_2}) \neq \Delta_{y_i} \\ S_i(x_{i_1}) \oplus S_i(x_{i_2}) \neq \Delta_{y_i} \end{cases} \quad . \tag{29}$$

– si x_{i_0} et x_{i_1} sont respectivement sur les lignes l et l' et soient $x_{i_2} \neq x_{i_0}$ sur l et $x_{i_3} \neq x_{i_1}$ sur l' alors :

$$S_i(x_{i_0}) \oplus S_i(x_{i_1}) = \Delta_{y_i} \Rightarrow \begin{cases} S_i(x_{i_0}) \oplus S_i(x_{i_3}) \neq \Delta_{y_i} \\ S_i(x_{i_1}) \oplus S_i(x_{i_2}) \neq \Delta_{y_i} \end{cases} \quad . \tag{30}$$

Dans la première attaque FIRE [100], P1 n'est pas exploitée complètement et les propriétés (29) et (30) ne le sont pas du tout.

2.4 Description de notre attaque en utilisant le Formalisme

Dans ce paragraphe nous décrivons notre attaque via le formalisme présenté en chapitre II.

- $\mathcal{K} = \Delta_y$ la cible est une différentielle de sortie des s-boxes.
- $O_S = (L_{16}, L_{16^*})$, l'observable stimulus est le couple.
- $O_R = \Delta_{R_{15}}$, l'observable réaction est la différentielle sur R_{15}.
- $\mathcal{R} = f \circ g$, avec :
  $f = \lightning$ processus d'injection de fautes et
  $g = E \circ (\oplus K_{16}) \circ S_i \circ P$
- $\mathcal{R}_m = f \circ g$, avec :
  $m = \oplus 2^i$, $i \in [\![0, 3]\!]$.
- Le distingueur est un crible combiné d'un algorithme de propagation de contraintes.

3 Implémentation et résultats

3.1 Réalisation de l'attaque en pratique

3.1.1 Représentation à l'aide de graphes

Pour simplifier les équations, nous considérons que pour chaque faute e nous avons la relation suivante avec j variant de 1 à 2 :

$$S_i(x_i) \oplus S_i(x_i^*) = \Delta_{y_i} \in \{\Delta_{z_j} \oplus e_j\} \quad . \tag{31}$$

Dans les s-boxes, les relations entre les entrées et les différentielles possibles en sortie, peuvent être représentées par 8 graphes disjoints G_i, un pour chaque s-box. Les nœuds d'un graphe représentent les 64 différentes valeurs d'entrée d'une s-box S_i. Deux nœuds sont reliés par une arrête pondérée, de sorte que leur poids soit égal au nombre de Δ_{y_i} possibles pour ces 2 entrées. Avant d'avoir commencé l'attaque, toutes les différentielles sont possibles. Le poids d'une arrête est alors égal à 16, ou 15 si l'arrête connecte deux entrées de la même ligne. L'attaque se termine lorsque le poids de chaque arrête est égale à 1. De manière plus générale, le poids total d'un tel graphe à la fin de l'attaque vaut : $\sum_{i=1}^{63} i = 2016$.

3.1.2 Algorithme

L'algorithme 3 : *Attaque* s'exécute jusqu'à ce que les s-boxes soient retrouvées *c.à.d* que les 8 graphes G_i, $i \in [\![1, 8]\!]$ aient un poids total de 2016. Chaque résultat d'une injection d'une faute e donne un ensemble d'équations de type (31) noté (eq_{e_i}), i est associé au numéro de la s-box et e à la faute. Ces équations (eq_{e_i}) sont déterminées grâce aux hypothèses de fautes possibles (ligne 7) trouvées en reprenant l'étude de la sous-section 2.2. Ces équations sont stockées dans la liste $\mathcal{L}$ (ligne 2). Puis, chaque graphe G_i, $i \in [\![1, 8]\!]$ est mis à jour avec (eq_{e_i}) grâce à l'algorithme 4 : *Mise à jour_graph* (ligne 12).

Algorithme 3 *Attaque* : Injecte des fautes mono-bit jusqu'à ce que les s-boxes soient définies à une translation près.

Sortie : 8 graphes G_i, $i \in [\![1, 8]\!]$ de poids total 2016, un pour chacune des 8 s-boxes.

1: Initialiser les 8 graphes G_i.
2: Créer une liste vide $\mathcal{L}$ d'ensemble d'équations (31).
3: **Tant que** tous les G_i n'ont pas un poids total de 2016 **faire :**
4: Injecter une faute mono-bit sur R_{14}.
5: Observer le nouveau couple (C, C^*) .
6: Calculer x, x^* avec équation (23) et Δ_z avec (26).
7: Lister les hypothèses sur la faute e_j.
8: Décrire les équations (31) : (eq_{e_i}) pour chaque s-box.
9: Ajouter les équations (eq_{e_i}) (31) à $\mathcal{L}$.
10: **Pour** $i = 1$ à 8 **faire :**
11: **Si** (eq_{e_i}) n'est pas déjà dans $\mathcal{L}$ **alors :**
12: *Mise à jour_graph*$((eq_{e_i}), G_i, \mathcal{L})$.
13: **Fin de si**
14: **Fin de boucle pour**
15: **Fin de boucle**
16: Retourne les 8 graphes G_i.

L'algorithme 4 travaille sur les relations (31) associées à une s-box S_i. Une relation (31) contient 2 entrées x_i et x'_i et une liste de Δ_{y_i} possibles. Si il n'y a qu'une seule différentielle Δ_{y_i} possible les propriétés (25) et (29) ou (30) peuvent être appliquées.

Par ailleurs nous donnons les précisions suivantes sur l'algorithme.

- (ligne 6) *Mettre à jour mutuellement* $\mathcal{L}$ *avec* (eq_{tmp}) signifie que : si il existe une autre équation dans $\mathcal{L}$ avec les mêmes entrées, le nombre de Δy_i possible peut être simplifié ; sinon (eq_{tmp}) doit juste être ajoutée à $\mathcal{L}$.
- (ligne 7) *Mettre à jour* G_i *avec* (eq_{tmp}) signifie que seules les Δ_{y_i} listées dans (eq_{tmp}) restent assignées à l'arrête reliant le nœud x_i au nœud x'_i. Le poids de l'arrête passe à 1 ou 2.
- (ligne 9) *Appliquer dans* G_i *la propriété* (25) signifie utiliser la propriété (25) pour trouver de nouvelle(s) relation(s) (31).
- (ligne 11 et 13)*Appliquer dans* G_i *la propriété* (29) *ou* (30) *:* ou (30) signifie utiliser la propriété (29) ou (30) pour trouver de nouvelles relations (31).

Algorithme 4 *Mise à jour_graph* : Stocke et traite les relations pour une s-box donnée

Entrée : 1 graphe G_i, $i \in [\![1, 8]\!]$
$\mathcal{L}$ liste des équations (31) connues.
eq_i une équation (31) pour la s-box S_i.
Sortie : Le graphe G_i, et la liste $\mathcal{L}$ mis à jour.

1: Créer une liste vide $\mathcal{L}'$ d'équations 31.
2: Ajouter eq_i à $\mathcal{L}'$.
3: **Tant que** $\mathcal{L}'$ n'est pas vide **faire :**
4: Prendre la première équation 31 de $\mathcal{L}'$ elle est notée (eq_{tmp}).
5: **Si** (eq_{tmp}) est une toute nouvelle relation dans $\mathcal{L}$ **alors :**
6: Mettre à jour mutuellement $\mathcal{L}$ et (eq_{tmp}).
7: Mettre à jour G_i avec (eq_{tmp}).
8: **Si** dans (eq_{tmp}), la différentielle Δ_{y_i} est connue **alors :**
9: Appliquer dans G_i la propriéte (25).
10: **Si** x_i et x'_i sont sur la même ligne **alors :**
11: G_i la propriété (29).
12: **Sinon**
13: Appliquer dans G_i la propriété (30).
14: **Fin de si**
15: **Fin de si**
16: Stocker les nouvelles relations (31) dans $\mathcal{L}'$.
17: **Fin de si**
18: Retirer (eq_{tmp}) dans $\mathcal{L}'$.
19: **Fin de boucle**

3.1.3 Injection des fautes en pratique

Cibler précisément R_{14} peut sembler difficile. En pratique, le plus souvent il est possible de savoir si la faute se produit dans le dernier ou l'avant dernier tour ou encore avant (voir le chapitre précédent IV). Si L_{16} a plus de 8 bits fautés, alors la faute a été injectée avant R_{14}, et doit être éliminée de notre campagne d'acquisitions. Sinon, L_{16} contient strictement moins de 9 bits fautés, la faute a bien été injectée dans R_{14}. Nous remarquons que si L_{16} contient seulement un bit fauté, la faute peut avoir été injectée dans R_{15}.

Bien qu'une telle faute apporte moins d'information qu'une dans R_{14} (permet d'attaquer moins de s-box à la fois), elle peut tout de même être exploitée.

Dans le but de valider le modèle de faute expérimentalement, l'attaque a été réalisée en pratique en suivant le protocole décrit en [103]. Le circuit choisi est une carte à puces en technologie CMOS 130 nm, fonctionnant à 66 MHz avec une tension d'alimentation de 1.3 V. Le DES implémenté dans ce circuit est la version classique (*c.à.d* il s'agit des s-boxes originales). Un tour de DES correspond à un cycle d'horloge. Les s-boxes sont implémentées en portes logiques et non en ROM. Il est à savoir que dans une telle conception, le chemin critique dépend des s-boxes, car elles sont composées de plusieurs couches de portes.

L'expérience consiste à réduire la source de courant pour fauter les calculs du DES. La campagne constituée de 1000 textes. Pour chaque valeur d'intensité réduite, le DES a donc été exécuté 1000 fois. A priori, plus l'intensité fournie est faible, plus il y a de fautes. Les résultats sont triés selon deux critères :

- la proportion de textes chiffrés fautés,
- la proportion de textes chiffrés fautés, issus d'une faute mono-bit.

Pour calculer ces métriques, une version logicielle (Matlab) du DES a été programmée, dans laquelle tous les bits des registres peuvent être inversés. Il y a donc 64×16 emplacements possibles, correspondant aux 64 bits des registres LR et aux 16 tours. Si pour une de ces positions, le DES en logiciel retourne les mêmes textes chiffrés que ceux obtenus expérimentalement, alors il est confirmé que la faute injectée est bien mono-bit. Bien sûr, la proportion de chiffrés incorrects est plus grande que la proportion de chiffrés incorrects causés par une faute mono-bit. Cette différence est appelée « multiple-errors ».

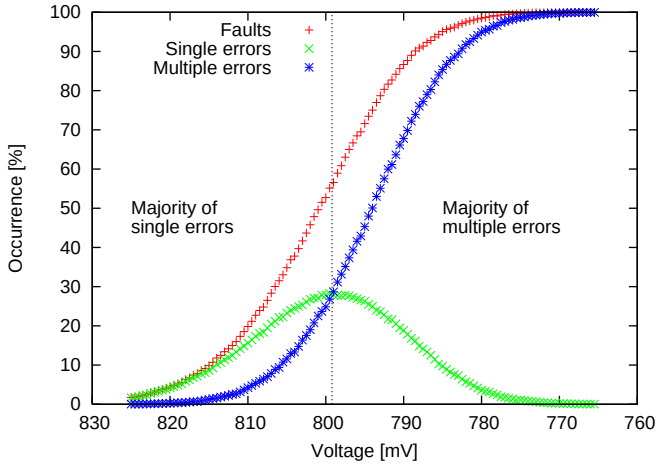


Figure 40 : Proportion de chiffrés fautés et proportion de chiffrés fautés dus à une faute mono-bit.

Les résultats sont illustrés en Figure 40. Le DES commence à retourner des chiffrés fautés avec une faible diminution de l'intensité. Cependant, le circuit reste fonctionnel, et ainsi l'intensité peut être diminuée petit à petit. Les fautes mono-bit apparaissent entre 825 et 820 mV.

3.2 Résultats

3.2.1 Résultats généraux

Nous avons simulé 1000 attaques avec Matlab. Le tableau 22 donne les statistiques du nombre de fautes nécessaires pour retrouver les s-boxes à une translation près. Notre étude montre qu'on a besoin en moyenne de 234 fautes pour définir les s-boxes a une translation près. C'est $4,45$ fois moins de fautes que l'attaque précédente [100], qui a besoin de 130 fautes par s-box soit 1040 fautes au total. Les propriétés P1 et P3 sur les s-boxes permettent de réduire significativement le nombre de fautes. Sans elles, en moyenne, 423 fautes sont nécessaires. Ce qui est toujours $2,45$ fois mieux que la précédente attaque. La propriété P1 réduit de manière significative le nombre de fautes en réduisant le nombres d'hypothèses sur la faute injectée. La propriété P1 simplifie le nombre de différentielles possibles en sortie des s-boxes.

Outils statistiques	sans P1 et P3	avec seulement P3	avec P1 et P3
moyenne	$423,07$	$280,57$	$234,76$
écart type	$63,30$	$47,43$	$34,08$
médiane	413	272	231
minimum	313	187	168
maximum	654	511	394

Tableau 22 : Statistiques sur 1000 attaques, du nombre de fautes nécessaires pour retrouver les s-boxes à une translation près

De plus, en pratique, pour 10000 fautes injectées dans R_{14}, la valeur de Δ_y est retrouvée dans 56% des cas.

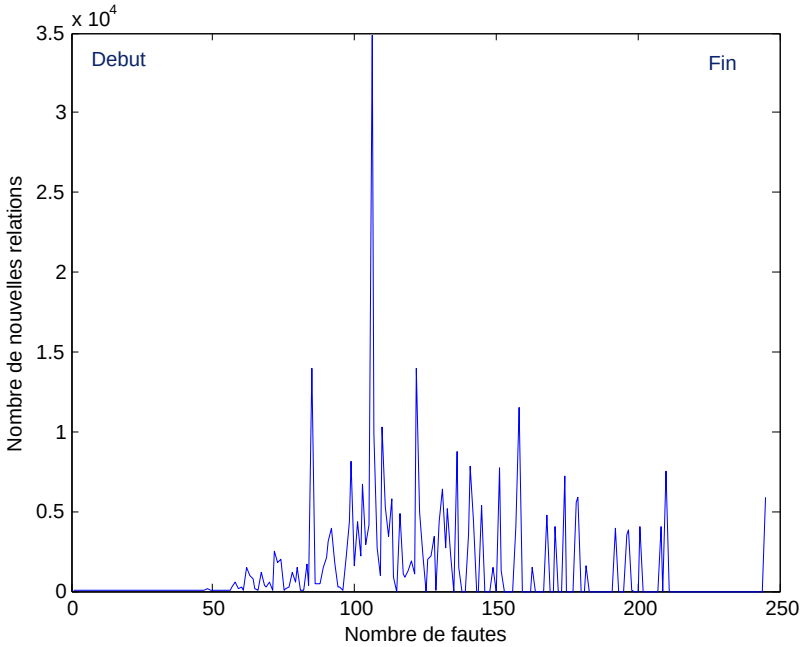


Figure 41 : Nombre de nouvelle(s) relation(s) (31) obtenue(s) à chaque faute injectée

Nous choisissons d'étudier le détail d'une attaque qui a abouti au bout de 244 fautes monobit. La Figure 41 donne le nombre de nouvelles relations (31) obtenues à chaque faute injectée. Au début ,il

y en a très peu, entre 2 et 8. Ce nombre commence à augmenter de manière significative autour de la 60ème faute. Le maximum est obtenu aux alentours de la 110ème faute. Les pics s'expliquent par l'utilisation des différentes propriétés. Nous remarquons que certaines fautes apportent beaucoup plus de relations que d'autres. Aux environs de la 130ème faute, certaines injections n'apportent plus aucune information. Puis, peu avant la 210ème faute plus aucune injection n'apporte de relations jusqu'à la dernière faute.

L'algorithme de l'attaque attend une faute pertinente par rapport à l'information qu'il a déjà.

3.2.2 Recherche exhaustive

Le paragraphe précédent montre que les dernières fautes apportent moins voir plus aucune information. Dans ce paragraphe, nous proposons une nouvelle idée : stopper l'attaque avant qu'elle ne se termine et finir en recherche exhaustive. L'idée est de limiter encore plus le nombre de fautes injectées.

Si l'Algorithme 3 *Attaque* est stoppé avant la fin, le poids total des graphes est différent de 2016. Certaines arrêtes ont un poids différents de 1. Il reste donc des hypothèses à faire pour retrouver les s-boxes à une translation près. Les propriétés (25), (29) et (30), permettent d'éliminer de nouvelles hypothèses à chaque hypothèse faite sur un Δ_{y_i}. Afin de parcourir le moins d'hypothèses l'utilisation de l'algorithme de Kruskal [104] est adéquate. Cet algorithme (de Kruskal) recherche l'arbre recouvrant de poids minimum. Ici, le poids représente le nombre d'hypothèses sur les Δ_{y_i} qu'il reste à tester. Ainsi nous espérons que le poids soit le plus petit possible. Cet algorithme optimise parfaitement notre recherche.

Le Tableau 23 présente les résultats pour un nombre donnés d'injections de fautes mono-bit. L'étude statistique a été menée en répétant 100 fois attaques. Le Tableau 23 donne pour commencer le nombre de s-boxes en moyenne qui sont définies à une translation près. La colonne suivante indique le nombre maximum de hypothèses qu'il reste pour retrouver les s-boxes à une translation près. Nous avons choisi la médiane plutôt que la moyenne car la présence de valeurs extrêmes altèrent cette dernière. La troisième et dernière colonne donne une estimation du nombre moyen d'hypothèses au total pour définir les s-boxes complètement. Nous rappelons que notre attaque définit les s-boxes à une translation près, il reste toujours 2^{32} hypothèses à tester à la fin.

Nombre de fautes	Moyenne du nombre de s-boxes qui sont définies à une translation près	Médiane du maximal du nombre d'hypothèses pour définir les s-boxes à une translation près	Nombre maximum d'hypothèses au total pour définir les s-boxes
120	$0,04$	$4,549 \cdot 10^{42}$	2^{174}
140	$0,89$	$9,5105 \cdot 10^{14}$	2^{82}
160	$2,76$	62208	2^{47}
180	$4,53$	16	2^{36}
200	$6,06$	8	2^{35}
220	$6,93$	4	2^{33}
240	$7,5$	0	2^{32}

Tableau 23 : Résultats pour 100 attaques avec différents nombres de fautes

Selon la puissance de calcul qu'un attaquant a à sa disposition, il peut avoir besoin d'injecter moins de 234 fautes. Par exemple, avec une puissance de calcule de 2^{47}, 160 suffisent à retrouver les s-boxes d'un pseudo DES. Nous rappelons que la recherche exhaustive des s-boxes sans l'attaque FIRE est impossible car cela représente : $2^{4\times 2^6\times 8} = 2^{2048}$ hypothèses.

C'est à l'attaquant de choisir le juste compromis entre le risque d'endommager son circuit en réalisant trop d'injections de fautes et le temps de calcul qui lui reste pour sa recherche finale en exhaustif.

4 Conclusion

Dans ce chapitre, nous avons présenté l'amélioration d'une attaque FIRE sur un pseudo-DES où les s-boxes sont modifiées, présentée à Chip-to-Cloud 2013 [31] puis publiée à FPS 2013 [32]. Notre attaque sélectionne un registre pertinent pour injecter des fautes mono-bit, ici R_{14} plutôt que R_{15}. Ce choix est en parfaite adéquation avec le chapitre IV. L'utilisation de certaines propriétés des s-boxes améliore nettement l'attaque.

Finalement nous avons besoin de 234 fautes pour retrouver les s-boxes, soit $4,44$ fois moins que l'attaque de référence. Il reste néanmoins une recherche exhaustive à mener à la fin de l'attaque. Par ailleurs, nous avons fait remarquer qu'un attaquant peut toujours finir en exhaustif même avec moins de 234 fautes. En effet, avec une puissance de calcul suffisante, nous pouvons encore réduire volontairement le nombre de fautes dans une attaque, en choisissant de s'arrêter à un nombre de fautes fixé.

De plus, l'étude d'une attaque par injection de fautes sur R_{13} a été évoquée. Cependant, la faute passe alors deux fois dans la fonction de Feistel, le nombre d'hypothèses à faire sur la faute injectée est beaucoup plus important. De plus, il devient plus difficile de reconnaître les fautes injectées sur R_{13} que celles injectées sur R_{14}, c'est pourquoi l'attaque complète n'a pas été menée.

Une perspective est d'étendre les travaux exposés en chapitre IV à la rétro-conception de tous les schémas de Feistel classiques et généralisés.

Partie VI

Conclusion

Le sujet principal de cette thèse porte sur l'élaboration d'un formalisme unifiant les attaques physiques matérielles. Les objectifs atteints, par notre formalisme sont :

- la présentation d'une unique méthode pour décrire les attaques physiques (chapitre II),
- l'élaboration d'un outil de comparaison, testé en pratique (chapitre II),
- la conception de nouvelles attaques (chapitres III et V),
- l'ébauche d'une étude générique des attaques potentielles sur un circuit (chapitre IV).

Dans le chapitre II, une unique méthode pour décrire les attaques physiques de manière structurée, dans un cadre mathématique a été exposée. Ainsi, la description d'une attaque physique repose sur une décomposition en trois étapes (campagne, prédictions, confrontation) et différents paramètres (deux observables, un chemin d'attaque, un ou des modèle(s) et un distingueur). Les avantages majeurs d'une telle décomposition sont :

- pouvoir étudier une attaque en détails et non uniquement dans sa globalité,
- remanier les paramètres d'attaques déjà existantes, pour en construire de nouvelles.

Dans ce même chapitre II, un outil de comparaison des attaques physiques, a été conceptualisé, puis testé en pratique sur un circuit. L'outil de comparaison repose sur les deux premières étapes de la description des attaques.

La description des attaques actuelles via le formalisme a permis de constater que les attaques n'utilisant que des mesures sont relativement rares. Ainsi, le but du chapitre III est de retrouver la clé de chiffrement d'un AES, sans utiliser ni texte clair, ni texte chiffré, ni template. Seules des mesures de consommation de courant ou EM, aux instants stratégiques sont exploitées.

Les difficultés de ce type d'attaque se définissent en deux points :

1. la détection des points d'intérêt,
2. le passage des mesures aux poids de Hamming.

Dans ce chapitre, deux approches portant sur la prise en compte des erreurs liées à l'estimation des poids de Hamming, ont été présentées :

1. une approche par acceptation des erreurs,
2. une approche par inférence Bayésienne.

L'attaque pouvant se faire à n'importe quel tour, il est possible d'affiner les résultats obtenus en utilisant les relations du KeyExpansion.

Dans l'idée d'élaborer un générateur systématique d'attaques, au chapitre IV, le cas particulier des schémas de Feistel généralisés a été étudié. L'étude porte sur des attaques par injection de faute mono-bit pour retrouver la clé de chiffrement. Il a été mis en évidence que certains blocs des GFN, sont plus pertinents que d'autres. Une méthode pour identifier ces blocs a été présentée. Le caractère générique de celle-ci implique que l'évaluation de la vulnérabilité n'est pas optimale. Cependant, elle peut être affiné par des connaissances précises sur l'algorithme ciblé. Cette méthode a été implémentée en magma et ainsi testée sur différents algorithmes GFN : DES, TWINE, MIBS et CLEFIA. Les résultats obtenus ont montré que notre étude est en adéquation avec l'état de l'Art des attaques par injection de fautes sur ces algorithmes. Ces travaux ont été publiés à FDTC 2014.

Pour finir, dans le chapitre V, une nouvelle attaque utilisant les résultats du chapitre IV pour faire de la rétro-conception est présentée. Il s'agit de l'amélioration d'une attaque de type FIRE sur un pseudo-DES dont les s-boxes sont modifiées. L'idée est d'injecter des fautes mono-bit dans R_{14} plutôt que R_{15}. L'utilisation de certaines propriétés des s-boxes améliore nettement l'attaque. Finalement, nous avons besoin de 234 fautes pour retrouver les s-boxes soit $4,44$ fois moins que l'attaque de référence. Il reste néanmoins une recherche exhaustive à mener à la fin de l'attaque. Par ailleurs, avec une puissance de calcul importante, il est possible de réduire volontairement le nombre de fautes. Cette attaque a été présentée à Chip-to-Cloud 2013 [31] et publiée à FPS 2013 [32].

Finalement le formalisme a été présenté à Chip-to-Cloud 2013 [29] et a été publié sur ePrint [30]. Il a également fait l'objet d'une publication à PROOFS 2014.

L'un des objectifs importants du formalisme est la recherche systématique d'attaques. Dans les chapitres III et V, des attaques issues du formalisme ont été présentées. Par ailleurs, le chapitre IV présente une étude systématique d'attaques. La création d'un générateur de toutes les attaques reste en perspective.

D'autres perspectives sont l'adaptation du formalisme à la cryptographie asymétrique ; mais également l'écriture formalisée des contre-mesures existantes.

Communications :

- Le sixième colloque du GDR SOC-SIP du CNRS 2012, session poster :
  Formalisme des attaques physiques.
  Hélène Le Bouder, Bruno Robisson et Assia Tria.

- CRYPTO'PUCES 2013 :
  Formalisme des attaques physiques.
  Hélène Le Bouder, Bruno Robisson et Assia Tria.

- Chip-to-Cloud 2013 :
  - *A Unified Formalism for Physical Attacks.* [29]
    Hélène Le Bouder, Bruno Robisson and Assia Tria.
  - *Fault Injection to Reverse Engineer DES-like Cryptosystems.* [31]
    Hélène Le Bouder, Sylvain Guilley, Bruno Robisson and Assia Tria.

- FPS 2013 Foundations & Practice of Security :
  Fault Injection to Reverse Engineer DES-like Cryptosystems. [32]
  Hélène Le Bouder, Sylvain Guilley, Bruno Robisson and Assia Tria.

- Cryptology ePrint : *A Unified Formalism for Physical Attacks.* [30]
  Hélène Le Bouder, Ronan Lashermes, Yanis Linge, Bruno Robisson, Assia Tria.

- PROOFS : Security Proofs for Embedded Systems 2014 :
  Physical functions : the common factor of side-channel and fault attacks ?
  Bruno Robisson, Hélène Le Bouder and Jean Yves Emmanuel Zie.

- FDTC 2014 Fault Diagnosis and Tolerance in Cryptography :
  On Fault Injections in Generalized Feistel Networks.
  Hélène Le Bouder, Gaël Thomas, Yanis Linge and Assia Tria.

Annexes

Sommaire

A Précisions sur les algorithmes de chiffrement

i Tableaux du DES

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

Tableau 24 : Permutation Initiale IP

S_1	32	1	2	3	4	5
S_2	4	5	6	7	8	9
S_3	8	9	10	11	12	13
S_4	12	13	14	15	16	17
S_5	16	17	18	19	20	21
S_6	20	21	22	23	24	25
S_7	24	25	26	27	28	29
S_8	28	29	30	31	32	1

Tableau 25 : Expansion E

16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

Tableau 26 : Tableau de Permutation P

15	1	8	14	6	11 3	4	9	7	2	13	12	0	5	10	
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

Tableau 27 : S-box 2 du DES

10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

Tableau 28 : S-box 3 du DES

7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

Tableau 29 : S-box 4 du DES

2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

Tableau 30 : S-box 5 du DES

12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

Tableau 31 : S-box 6 du DES

4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

Tableau 32 : S-box 7 du DES

13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Tableau 33 : S-box 8 du DES

ii Précision sur l'AES

$$\begin{pmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{pmatrix}$$

Figure 42 : Matrice du **MixColumn** MC

x	y	x	y	x	y	x	y	x	y
0	99	1	124	2	119	3	123	4	242
5	107	6	111	7	197	8	48	9	1
10	103	11	43	12	254	13	215	14	171
15	118	16	202	17	130	18	201	19	125
20	250	21	89	22	71	23	240	24	173
25	212	26	162	27	175	28	156	29	164
30	114	31	192	32	183	33	253	34	147
35	38	36	54	37	63	38	247	39	204
40	52	41	165	42	229	43	241	44	113
45	216	46	49	47	21	48	4	49	199
50	35	51	195	52	24	53	150	54	5
55	154	56	7	57	18	58	128	59	226
60	235	61	39	62	178	63	117	64	9
65	131	66	44	67	26	68	27	69	110
70	90	71	160	72	82	73	59	74	214
75	179	76	41	77	227	78	47	79	132
80	83	81	209	82	0	83	237	84	32
85	252	86	177	87	91	88	106	89	203
90	190	91	57	92	74	93	76	94	88
95	207	96	208	97	239	98	170	99	251
100	6	101	77	102	51	103	133	104	69
105	249	106	2	107	127	108	80	109	60
110	159	111	168	112	81	113	163	114	64
115	143	116	146	117	157	118	56	119	245
120	188	121	182	122	218	123	33	124	16
125	255	126	243	127	210	128	205	129	12
130	19	131	236	132	95	133	151	134	68
135	23	136	196	137	167	138	126	139	61
140	100	141	93	142	25	143	115	144	96
145	129	146	79	147	220	148	34	149	42
150	144	151	136	152	70	153	238	154	184
155	20	156	222	157	94	158	11	159	219
160	224	161	50	162	58	163	10	164	73
165	6	166	36	167	92	168	194	169	211
170	172	171	98	172	145	173	149	174	228
175	121	176	231	177	200	178	55	179	109
180	141	181	213	182	78	183	169	184	108
185	86	186	244	187	234	188	101	189	122
190	174	191	8	192	186	193	120	194	37
195	46	196	28	197	166	198	180	199	198
200	232	201	221	202	116	203	31	204	75
205	189	206	139	207	138	208	112	209	62
210	181	211	102	212	72	213	3	214	246
215	14	216	97	217	53	218	87	219	185
220	134	221	193	222	29	223	158	224	225
225	248	226	152	227	17	228	105	229	217
230	142	231	148	232	155	233	30	234	135
235	233	236	206	237	85	238	40	239	223
240	140	241	161	242	137	243	13	244	191
245	230	246	66	247	104	248	65	249	153
250	45	251	15	252	176	253	84	254	187
255	22								

Tableau 34 : Le tableau représentant la sbox de l'AES où x est une entrée et y une sortie

iii Précision sur CLEFIA

.0	.1	.2	.3	.4	.5	.6	.7	.8	.9	.a	.b	.c	.d	.e	.f	
0.	57	49	d1	c6	2f	33	74	fb	95	6d	82	ea	0e	b0	a8	1c
1.	28	d0	4b	92	5c	ee	85	b1	c4	0a	76	3d	63	f9	17	af
2.	bf	a1	19	65	f7	7a	32	20	06	ce	e4	83	9d	5b	4c	d8
3.	42	5d	2e	e8	d4	9b	0f	13	3c	89	67	c0	71	aa	b6	f5
4.	a4	be	fd	8c	12	00	97	da	78	e1	cf	6b	39	43	55	26
5.	30	98	cc	dd	eb	54	b3	8f	4e	16	fa	22	a5	77	09	61
6.	d6	2a	53	37	45	c1	6c	ae	ef	70	08	99	8b	1d	f2	b4
7.	e9	c7	9f	4a	31	25	fe	7c	d3	a2	bd	56	14	88	60	0b
8.	cd	e2	34	50	9e	dc	11	05	2b	b7	a9	48	ff	66	8a	73
9.	03	75	86	f1	6a	a7	40	c2	b9	2c	db	1f	58	94	3e	ed
a.	fc	1b	a0	04	b8	8d	e6	59	62	93	35	7e	ca	21	df	47
b.	15	f3	ba	7f	a6	69	c8	4d	87	3b	9c	01	e0	de	24	52
c.	7b	0c	68	1e	80	b2	5a	e7	ad	d5	23	f4	46	3f	91	c9
d.	6e	84	72	bb	0d	18	d9	96	f0	5f	41	ac	27	c5	e3	3a
e.	81	6f	07	a3	79	f6	2d	38	1a	44	5e	b5	d2	ec	cb	90
f.	9a	36	e5	29	c3	4f	ab	64	51	f8	10	d7	bc	02	7d	8e

Tableau 35 : s-box 1 de CLEFIA en hexadécimal

.0	.1	.2	.3	.4	.5	.6	.7	.8	.9	.a	.b	.c	.d	.e	.f	
0.	6c	da	c3	e9	4e	9d	0a	3d	b8	36	b4	38	13	34	0c	d9
1.	bf	74	94	8f	b7	9c	e5	dc	9e	07	49	4f	98	2c	b0	93
2.	12	eb	cd	b3	92	e7	41	60	e3	21	27	3b	e6	19	d2	0e
3.	91	11	c7	3f	2a	8e	a1	bc	2b	c8	c5	0f	5b	f3	87	8b
4.	fb	f5	de	20	c6	a7	84	ce	d8	65	51	c9	a4	ef	43	53
5.	25	5d	9b	31	e8	3e	0d	d7	80	ff	69	8a	ba	0b	73	5c
6.	6e	54	15	62	f6	35	30	52	a3	16	d3	28	32	fa	aa	5e
7.	cf	ea	ed	78	33	58	09	7b	63	c0	c1	46	1e	df	a9	99
8.	55	04	c4	86	39	77	82	ec	40	18	90	97	59	dd	83	1f
9.	9a	37	06	24	64	7c	a5	56	48	08	85	d0	61	26	ca	6f
a.	7e	6a	b6	71	a0	70	05	d1	45	8c	23	1c	f0	ee	89	ad
b.	7a	4b	c2	2f	db	5a	4d	76	67	17	2d	f4	cb	b1	4a	a8
c.	b5	22	47	3a	d5	10	4c	72	cc	00	f9	e0	fd	e2	fe	ae
d.	f8	5f	ab	f1	1b	42	81	d6	be	44	29	a6	57	b9	af	f2
e.	d4	75	66	bb	68	9f	50	02	01	3c	7f	8d	1a	88	bd	ac
f.	f7	e4	79	96	a2	fc	6d	b2	6b	03	e1	2e	7d	14	95	1d

Tableau 36 : s-box 2 de CLEFIA en hexadécimal

B Autres exemples d'attaques décrites avec le formalisme

i Autres exemples sur l'AES

i.a Attaque par injection de fautes sur K_{10}

- $\mathcal{K} = K_9^{l,c}$, la cible est un octet de la clé du tour 9.
- $O_S = C$, l'observable stimulus est le texte chiffré.
- $O_R = C^*$, l'observable réaction est le chiffré fauté.
- $\mathcal{R} = g_4 \circ (f \circ g_3) \circ g_2 \circ f \circ g_1$, avec :
  $f = \lightning$ processus d'injection de fautes
  $g_1 = SB^{-1} \circ SR^{-1} \circ \oplus$
  $g_2 = SR \circ SB$
  $g_3 =$ identité, si le cible n'est pas un octet de la première colonne, sinon $g_3 = SB$
  et $g_4 = \oplus$.
- $\mathcal{R}_m = g \circ m_i$, avec :
  $m_i = \oplus 2^i, i \in [\![0, 7]\!]$.
- Le distingueur est un crible.

i.b Attaque hybride : la DBA

La DBA (Differential Behaviour Analysis) présentée en [20] est une attaque hybride comme la FSA (chapitre II section 3.3).

- $\mathcal{K} = K_{10}^{l,c}$, la cible est un octet ou un bit de la clé du tour 10.
- $O_S = C$, l'observable stimulus est le texte clair.
- $O_R =$ comportement, l'observable réaction est la variable binaire le résultat est fauté ou non.
- $\mathcal{R} = g_2 \circ f \circ g_1$, avec :
  $f = \lightning$ processus d'injection de fautes,
  $g_1 = SB^{-1} \circ SR^{-1} \circ \oplus$ et $g_2 = \oplus \circ SR \circ SB$.
- $\mathcal{R}_m = g_2 \circ m \circ g_1$, avec :
  $m =$ test $\circ$ (collage à a), $a \in [\![0, 255]\!]$.
  avec *test* la fonction booléenne de test d'égalité entre C et C^*.
- Le distingueur est une corrélation.

i.c Attaque SCARE sur pseudo-AES

L'attaque est présentée dans [24]. Elle a pour cible, la s-box de l'AES fonction booléenne $b_{8\to 8}$.

- $\mathcal{K} = b$, la cible est une fonction booléenne $b_8 \to 1$. Il y a donc 256 valeurs possibles
- $O_S = T$, l'observable stimulus est le texte clair.
- $O_R = Conso$, l'observable réaction est la consommation de courant.
- $\mathcal{R} = f \circ g$, avec :
  $f = conso$
  et $g = \oplus K_0$.
- $\mathcal{R}_m = m \circ g$, avec :
  $m = HW$.
- Le distingueur est une corrélation.

ii Autres exemples sur le DES

ii.a Attaque de type DFA le DES

La DFA classique sur le DES est présenté en [11].

- $\mathcal{K} = K_{16}$, la cible est 6 bits de la clé du dernier tour.
- $O_S = (L_{16}, L^*_{16})$, l'observable stimulus est la partie de gauche du couple chiffré, chiffré fauté, après IP.
- $O_R = R^*_{16} \oplus R_{16}$, l'observable réaction est la différentielle de la partie de droite après IP.
- $\mathcal{R} = g \circ f$, avec :
  $f = \lightning$ processus d'injection de fautes et
  $g = P \circ S_i \circ \oplus \circ E$.
- $\mathcal{R}_m = g \circ m$, avec :
  $m_i = \oplus 2^i$, $i \in [\![0, 5]\!]$.
- Le distingueur est un crible.

ii.b Attaque SCA le DES

- $\mathcal{K} = K_1$, la cible est un fragment de 6 bits de la clé du tour 1 (une entrées de s-box S_i.
- $O_S = T$, l'observable stimulus est le texte clair.
- $O_R = Conso$, l'observable réaction est la consommation de courant en sortie des s-boxes.
- $\mathcal{R} = f \circ g$, avec :
  $f = conso$, consommation de courant et
  $g = S_i \circ \oplus \circ E \circ IP$.
- $\mathcal{R}_m = m \circ g$, avec :
  $m = HW$.
- Le distingueur est une corrélation ou une différence de moyenne.

ii.c Attaque de rétro-conception à deux niveaux d'étapes

Cet exemple est un exemple de rétro-conception combinant une attaque FIRE classique avec une attaque SCARE. Ces attaques sont présentées en détails dans le chapitre V. Pour résumer, le but est de retrouver les s-boxes d'un pseudo-DES. L'attaque FIRE permet uniquement de retrouver les différentielles en sortie $\Delta_y = S(x \oplus x^*)$. Ainsi l'attaque doit se finir de manière exhaustive. Cependant, un autre choix est possible utiliser une attaque SCARE. L'attaque présentée ici, est donc une attaque à deux niveaux (voir chapitre II section 2.4).

1. Première étape :
   - $\mathcal{K}^1 = \Delta_y$, où $\Delta_y = S(x \oplus x^*)$.
     Il y a 15 hypothèses k.
   - $O^1_S = (L_{16}, L^*_{16})$, l'observable stimulus est la partie de gauche du couple chiffré, chiffré fauté, après IP.
   - $O^1_R = R^*_{16} \oplus R_{16}$, l'observable réaction est la différentielle de la partie de droite après IP.
   - $\mathcal{R} = g \circ f$, avec :
     $f = \lightning$, processus d'injection de fautes et
     $g = P \circ (\oplus K_{16}) \circ E$.
     Le chemin d'attaque est illustré en Figure 35.
   - $\mathcal{R}^1_m = g \circ m_i$ avec :
     $m_i = \oplus 2^i$, $i \in [\![0, 5]\!]$.
   - Le distingueur est un crible.

2. Deuxième étape :
   - $\mathcal{K}^2 = y$ la cible est une valeur de sortie d'une s-box pour une entrée x fixée. Il y a 16 hypothèses possibles.
   - $O_S^2 = (R_0, \Delta_y)$, l'observable stimulus est le premier registre de droite et la différentielle correspondante.
   - $O_R^2 = Conso$, l'observable réaction est la consommation de courant.
   - $R^2 = f \circ g$, avec :
     $f = conso$
     et $g = \oplus\Delta_y(\oplus K_0) \circ E$,
     où $\Delta_y = S(x \oplus x') = \mathcal{K}^1$ avec x' les bits concernés de $K_0 \oplus E(R_0)$.
   - $\mathcal{R}_m^2 = m \circ g$ avec :
     $m = HW$.
   - Le distingueur est une corrélation.

C Rappels mathématiques

i Rappels d'algèbre

Définition 46 : $(D, \times, a)$, est un **monoïde** si :

1. $\forall (x, y) \in D^2, x \times y \in D$ (stabilité) ;
2. $\forall (x, y, z) \in D^3, x \times (y \times z) = (x \times y) \times z$ (associativité) ;
3. $\exists\ a \in D, \forall x \in D, x \times a = a \times x = x$ (existence d'un élément neutre).

Définition 47 : On dit que $(D, +, \times, 0, 1)$ est un **dioïde** si :

1. $(D, +, 0)$ est un monoïde commutatif ;
2. $(D, \times, 1)$ est un monoïde ;
3. $\times$ est distributif par rapport à $+$;
4. 0 est un élément absorbant pour $\times$, c'est-à-dire que $x \times 0 = 0 \times x = 0$ pour tout $x \in D$;
5. La relation $\leq$ définie pour tout $(x, y) \in D^2$ par $x \leq y$ s'il existe $z \in D$ tel que $x + z = y$, est une relation d'ordre.

Si l'on omet le dernier point, la structure définie est un demi-anneau.

Définition 48 : Un dioïde $(D, +, \times, 0, 1)$ est **idempotent** si : $\forall x \in D\ x + x = x$.

ii Rappels de probabilité

ii.a Notion d'indépendance

Définition 49 : X et Y sont **indépendants** $\Leftrightarrow Pr(X \cap Y) = Pr(X) \cdot Pr(Y)$.

Définition 50 : En statistique, des variables **indépendantes et identiquement distribuées** (iid) sont des variables aléatoires qui ont toutes la même loi de probabilité, mais sont indépendantes.

ii.b Formules diverses

• Soit X une variable aléatoire discrète, pouvant prendre n_x valeurs et de moyenne μ_x. L'espérance de X est alors :

$$\mathbb{E}(X) = \sum_{x=1}^{n_x} x \cdot Pr(X = x) \quad .$$

La variance de X est alors :

$$\mathrm{Var}(X) = \frac{1}{n_x} \cdot \sum_{x=1}^{n_x} (x - \mu_x)^2 \quad .$$

• Soit X une variable aléatoire continue de densité de probabilité F_x. L'espérance de X est alors :

$$\mathbb{E}(X) = \int x F_x(x) dx \quad .$$

La variance de X est alors :

$$\mathrm{Var}(X) = \int (x - \mathbb{E}(X))^2 \cdot F_x(x) dx \quad .$$

• Soient deux variables aléatoires X et Y la covariance est :

$$\mathrm{Cov} = \mathbb{E}(X - \mathbb{E}(X)) \cdot \mathbb{E}(Y - \mathbb{E}(Y)) \quad .$$

ii.c Propriétés de l'écart type

Propriété P7 : L'écart type est toujours positif ou nul, celui d'une constante est nul. L'écart type d'une variable aléatoire X à laquelle a été ajoutée une constante est égal à l'écart type de la variable X. Cette propriété est nommée **invariance par translation**.

Propriété P8 : L'écart type d'une variable aléatoire X multipliée par une constante est égal à la valeur absolue de la constante multipliée par l'écart type de la variable. Cette propriété est nommée **invariance par dilatation**.

ii.d Loi de probabilité totale

Pour un système exhaustif (fini ou dénombrable) d'événements (Y_i) , et si quel que soit i, $Pr(Y_i) \neq 0$, alors on a les formules de probabilités totales suivantes :

$$Pr(X) = \sum_i Pr(X|Y_i) \cdot Pr(Y_i) \quad . \tag{32}$$

$$Pr(X|Z) = \sum_i Pr(X|Y_i) \cdot Pr(Y_i|Z) \quad . \tag{33}$$

La formule adaptée à Z continue donne :

$$Pr(X|Z) = \sum_i Pr(X|Y_i) \cdot F_Y(Y_i|Z) \quad . \tag{34}$$

ii.e Formules de Bayes

Les formules de Bayes pour des variables discrètes sont les suivantes :

$$Pr(X|Y) = \frac{Pr(X \cap Y)}{Pr(Y)} \quad . \tag{35}$$

$$Pr(X|Y) = \frac{Pr(Y|X)Pr(X)}{Pr(Y)} \quad . \tag{36}$$

La formule de Bayes si les variables sont continues est la suivante :

$$F_X(x|Y = y) = \frac{F_Y(y|X = x)\, F_X(x)}{F_Y(y)} \quad . \tag{37}$$

Si X est discrète et Y continue, les formules de Bayes donnent :

$$Pr(X = x|Y = y) = \frac{F_Y(y|X = x)\, Pr(X = x)}{F_Y(y)} \quad . \tag{38}$$

$$F_Y(y|X = x) = \frac{Pr(X = x|Y = y)\, F_Y(y)}{Pr(X = x)} \quad . \tag{39}$$

D Inférence Bayésienne et détection des *PoI*

Nous proposons une autre méthode pour la détection de *PoI* : une méthode par inférence bayésienne. Cette analyse permet de retrouver les *PoI* pour une entrée et sortie de s-box. Elle est donc incomplète car il manque un point dépendant de la clé (celui avant le AddRoundKey). Ici X est le l'entrée de la s-box et Y la sortie. Soit une campagne dont les courbes sont composées de n points.

Pour chaque couple d'instant de mesures (i_1, i_2) avec $i_1 < i_2$ (soit $\frac{n(n-1)}{2}$ possibilités), les couples de mesures $(h'_1, h'_2)_{0 \leq i < n}$ sont extraits avec h'_1 mesuré au temps i_1 et h'_2 mesuré au temps i_2.

Soit la fonction booléenne $b_{\text{s-box}}$ retournant 1 si les instants (i_1, i_2) correspondent à l'entrée et à la sortie d'une s-box, 1 sinon. Nous cherchons à évaluer la probabilité de l'évènement $b_{\text{s-box}} = 1$.

Pour cela, il est possible d'utiliser l'inférence bayésienne.

$$Pr(b_{\text{s-box}} = 1|\{(h'_1, h'_2)_i\} = \frac{\prod_i F_{H'_X, H'_Y}((h'_1, h'_2)_i|b_{\text{s-box}} = 1) \cdot Pr(b_{\text{s-box}} = 1)}{F_{HX', HY'}(\{(h'_1, h'_2)_i\}} \quad . \tag{40}$$

E Résultats bruts de l'étude sur les GFN

i TWINE

$$
\begin{aligned}
V &= (0,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty)\\
\mathbf{r}-(d+1) &= 8\\
V_{\mathcal{F}} &= (8,5,6,3,6,7,4,5,4,5,6,3,6,3,4,5)\\
W_{\mathcal{F}} &= (8,9,6,7,6,7,4,5,4,5,6,7,6,7,4,5)\\
V &= (0,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty)\\
\mathbf{r}-(d+1) &= 7\\
V_{\mathcal{F}} &= (7,4,5,2,5,6,3,4,3,4,5,2,5,2,3,4)\\
W_{\mathcal{F}} &= (7,8,5,6,5,6,3,4,3,4,5,6,5,6,3,4)\\
V &= (0,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty)\\
\mathbf{r}-(d+1) &= 6\\
V_{\mathcal{F}} &= (6,3,4,1,4,5,2,3,2,3,4,1,4,-\infty,2,3)\\
W_{\mathcal{F}} &= (6,7,4,5,4,5,2,3,2,3,4,5,4,5,2,3)\\
V &= (0,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty)\\
\mathbf{r}-(d+1) &= 5\\
V_{\mathcal{F}} &= (5,2,3,-\infty,3,4,1,-\infty,-\infty,2,3,0,3,-\infty,1,2)\\
W_{\mathcal{F}} &= (5,6,3,4,3,4,1,2,-\infty,2,3,4,3,4,1,2)\\
V &= (0,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty)\\
\mathbf{r}-(d+1) &= 4\\
V_{\mathcal{F}} &= (4,-\infty,2,-\infty,-\infty,3,-\infty,-\infty,-\infty,1,2,-\infty,2,-\infty,0,1)\\
W_{\mathcal{F}} &= (4,5,2,3,-\infty,3,-\infty,-\infty,-\infty,1,2,3,2,3,0,1)\\
V &= (0,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty)\\
\mathbf{r}-(d+1) &= 3\\
V_{\mathcal{F}} &= (3,-\infty,-\infty,-\infty,-\infty,2,-\infty,-\infty,-\infty,-\infty,1,-\infty,1,-\infty,-\infty,0)\\
W_{\mathcal{F}} &= (3,4,-\infty,-\infty,-\infty,2,-\infty,-\infty,-\infty,-\infty,1,2,1,2,-\infty,0)\\
V &= (0,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty)\\
\mathbf{r}-(d+1) &= 2\\
V_{\mathcal{F}} &= (2,-\infty,-\infty,-\infty,-\infty,1,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,0,-\infty,-\infty,-\infty)\\
W_{\mathcal{F}} &= (2,3,-\infty,-\infty,-\infty,1,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,0,1,-\infty,-\infty)\\
V &= (0,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty)\\
\mathbf{r}-(d+1) &= 1\\
V_{\mathcal{F}} &= (1,-\infty,-\infty,-\infty,-\infty,0,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty)\\
W_{\mathcal{F}} &= (1,2,-\infty,-\infty,-\infty,0,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty)\\
V &= (0,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty)\\
\mathbf{r}-(d+1) &= 0
\end{aligned}
$$

$$
\begin{aligned}
V_{\mathcal{F}} &= (0,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty)\\
W_{\mathcal{F}} &= (0,1,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty)\\
V &= (-\infty,-\infty,0,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty)\\
\mathbf{r}-(d+1) &= 8\\
V_{\mathcal{F}} &= (6,3,8,5,4,5,6,3,6,7,4,5,4,5,6,3)\\
W_{\mathcal{F}} &= (6,7,8,9,4,5,6,7,6,7,4,5,4,5,6,7)\\
V &= (-\infty,-\infty,0,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty)\\
\mathbf{r}-(d+1) &= 7\\
V_{\mathcal{F}} &= (5,2,3,4,3,4,5,2,5,2,7,4,3,4,5,6)\\
W_{\mathcal{F}} &= (5,6,3,4,3,4,5,6,5,6,7,8,3,4,5,6)\\
V &= (-\infty,-\infty,0,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty)\\
\mathbf{r}-(d+1) &= 6\\
V_{\mathcal{F}} &= (4,-\infty,2,3,2,3,4,5,4,1,2,3,6,3,4,1)\\
W_{\mathcal{F}} &= (4,5,2,3,2,3,4,5,4,5,2,3,6,7,4,5)\\
V &= (-\infty,-\infty,0,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty)\\
\mathbf{r}-(d+1) &= 5\\
V_{\mathcal{F}} &= (3,4,-\infty,2,5,2,3,-\infty,3,-\infty,1,2,1,-\infty,3,0)\\
W_{\mathcal{F}} &= (3,4,-\infty,2,5,6,3,4,3,4,1,2,1,2,3,4)\\
V &= (-\infty,-\infty,0,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty)\\
\mathbf{r}-(d+1) &= 4\\
V_{\mathcal{F}} &= (2,-\infty,4,-\infty,-\infty,1,2,-\infty,-\infty,3,-\infty,-\infty,0,1,2,-\infty)\\
W_{\mathcal{F}} &= (2,3,4,5,-\infty,1,2,3,-\infty,3,-\infty,-\infty,0,1,2,3)\\
V &= (-\infty,-\infty,0,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty)\\
\mathbf{r}-(d+1) &= 3\\
V_{\mathcal{F}} &= (1,-\infty,-\infty,-\infty,-\infty,0,-\infty,-\infty,1,-\infty,3,-\infty,-\infty,-\infty,-\infty,2)\\
W_{\mathcal{F}} &= (1,2,-\infty,-\infty,-\infty,0,-\infty,-\infty,1,2,3,4,-\infty,-\infty,-\infty,2)\\
V &= (-\infty,-\infty,0,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty)\\
\mathbf{r}-(d+1) &= 2\\
V_{\mathcal{F}} &= (0,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,1,-\infty,-\infty,-\infty,-\infty,2,-\infty,-\infty,-\infty)\\
W_{\mathcal{F}} &= (0,1,-\infty,-\infty,-\infty,-\infty,-\infty,1,-\infty,-\infty,-\infty,-\infty,2,3,-\infty,-\infty)\\
V &= (-\infty,-\infty,0,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty)\\
\mathbf{r}-(d+1) &= 1\\
V_{\mathcal{F}} &= (-\infty,0,-\infty,-\infty,1,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty)\\
W_{\mathcal{F}} &= (-\infty,0,-\infty,-\infty,1,2,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty)\\
V &= (-\infty,-\infty,0,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty)\\
\mathbf{r}-(d+1) &= 0\\
V_{\mathcal{F}} &= (-\infty,-\infty,0,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty)\\
W_{\mathcal{F}} &= (-\infty,-\infty,0,1,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty)
\end{aligned}
$$

$$
\begin{aligned}
V &= (-\infty,-\infty,-\infty,-\infty,0,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty)\\
\mathbf{r}-(d+1) &= 8\\
V_{\mathcal{F}} &= (6,7,4,5,8,5,6,3,6,3,4,5,4,5,6,3)\\
W_{\mathcal{F}} &= (6,7,4,5,8,9,6,7,6,7,4,5,4,5,6,7)\\
V &= (-\infty,-\infty,-\infty,-\infty,0,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty)\\
\mathbf{r}-(d+1) &= 7\\
V_{\mathcal{F}} &= (5,2,7,4,3,4,5,2,5,6,3,4,3,4,5,2)\\
W_{\mathcal{F}} &= (5,6,7,8,3,4,5,6,5,6,3,4,3,4,5,6)\\
V &= (-\infty,-\infty,-\infty,-\infty,0,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty)\\
\mathbf{r}-(d+1) &= 6\\
V_{\mathcal{F}} &= (4,1,2,3,2,3,4,1,4,-\infty,6,3,2,3,4,5)\\
W_{\mathcal{F}} &= (4,5,2,3,2,3,4,5,4,5,6,7,2,3,4,5)\\
V &= (-\infty,-\infty,-\infty,-\infty,0,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty)\\
\mathbf{r}-(d+1) &= 5\\
V_{\mathcal{F}} &= (3,-\infty,1,2,1,-\infty,3,4,3,0,-\infty,2,5,2,3,-\infty)\\
W_{\mathcal{F}} &= (3,4,1,2,1,2,3,4,3,4,-\infty,2,5,6,3,4)\\
V &= (-\infty,-\infty,-\infty,-\infty,0,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty)\\
\mathbf{r}-(d+1) &= 4\\
V_{\mathcal{F}} &= (-\infty,3,-\infty,1,4,-\infty,2,-\infty,2,-\infty,0,1,-\infty,-\infty,2,-\infty)\\
W_{\mathcal{F}} &= (-\infty,3,-\infty,1,4,5,2,3,2,3,0,1,-\infty,-\infty,2,3)\\
V &= (-\infty,-\infty,-\infty,-\infty,0,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty)\\
\mathbf{r}-(d+1) &= 3\\
V_{\mathcal{F}} &= (-\infty,-\infty,3,-\infty,-\infty,-\infty,1,-\infty,-\infty,2,-\infty,-\infty,-\infty,0,1,-\infty)\\
W_{\mathcal{F}} &= (-\infty,-\infty,3,4,-\infty,-\infty,1,2,-\infty,2,-\infty,-\infty,-\infty,0,1,2)\\
V &= (-\infty,-\infty,-\infty,-\infty,0,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty)\\
\mathbf{r}-(d+1) &= 2\\
V_{\mathcal{F}} &= (-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,0,-\infty,2,-\infty,-\infty,-\infty,-\infty,1)\\
W_{\mathcal{F}} &= (-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,0,1,2,3,-\infty,-\infty,-\infty,1)\\
V &= (-\infty,-\infty,-\infty,-\infty,0,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty)\\
\mathbf{r}-(d+1) &= 1\\
V_{\mathcal{F}} &= (-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,0,-\infty,-\infty,-\infty,-\infty,1,-\infty,-\infty,-\infty)\\
W_{\mathcal{F}} &= (-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,0,-\infty,-\infty,-\infty,-\infty,1,2,-\infty,-\infty)\\
V &= (-\infty,-\infty,-\infty,-\infty,0,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty)\\
\mathbf{r}-(d+1) &= 0\\
V_{\mathcal{F}} &= (-\infty,-\infty,-\infty,-\infty,0,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty)\\
W_{\mathcal{F}} &= (-\infty,-\infty,-\infty,-\infty,0,1,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty)\\
V &= (-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,0,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty)\\
\mathbf{r}-(d+1) &= 8\\
V_{\mathcal{F}} &= (4,5,6,3,6,3,8,5,4,5,6,3,6,7,4,5)\\
W_{\mathcal{F}} &= (4,5,6,7,6,7,8,9,4,5,6,7,6,7,4,5)
\end{aligned}
$$

$$
\begin{aligned}
V &= (-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,0,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty)\\
\mathbf{r}-(d+1) &= 7\\
V_{\mathcal{F}} &= (3,4,5,6,5,2,3,4,7,4,5,2,5,2,3,4)\\
W_{\mathcal{F}} &= (3,4,5,6,5,6,3,4,7,8,5,6,5,6,3,4)\\
V &= (-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,0,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty)\\
\mathbf{r}-(d+1) &= 6\\
V_{\mathcal{F}} &= (2,3,4,1,4,-\infty,6,3,2,3,4,1,4,5,2,3)\\
W_{\mathcal{F}} &= (2,3,4,5,4,5,6,7,2,3,4,5,4,5,2,3)\\
V &= (-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,0,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty)\\
\mathbf{r}-(d+1) &= 5\\
V_{\mathcal{F}} &= (-\infty,2,3,4,3,-\infty,1,2,5,2,3,-\infty,3,0,1,-\infty)\\
W_{\mathcal{F}} &= (-\infty,2,3,4,3,4,1,2,5,6,3,4,3,4,1,2)\\
V &= (-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,0,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty)\\
\mathbf{r}-(d+1) &= 4\\
V_{\mathcal{F}} &= (-\infty,-\infty,2,-\infty,2,-\infty,4,-\infty,0,1,2,-\infty,-\infty,3,-\infty,1)\\
W_{\mathcal{F}} &= (-\infty,-\infty,2,3,2,3,4,5,0,1,2,3,-\infty,3,-\infty,1)\\
V &= (-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,0,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty)\\
\mathbf{r}-(d+1) &= 3\\
V_{\mathcal{F}} &= (-\infty,-\infty,-\infty,2,-\infty,-\infty,-\infty,0,3,-\infty,1,-\infty,1,-\infty,-\infty,-\infty)\\
W_{\mathcal{F}} &= (-\infty,-\infty,-\infty,2,-\infty,-\infty,-\infty,0,3,4,1,2,1,2,-\infty,-\infty)\\
V &= (-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,0,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty)\\
\mathbf{r}-(d+1) &= 2\\
V_{\mathcal{F}} &= (-\infty,-\infty,-\infty,-\infty,0,-\infty,2,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,1,-\infty,-\infty)\\
W_{\mathcal{F}} &= (-\infty,-\infty,-\infty,-\infty,0,1,2,3,-\infty,-\infty,-\infty,-\infty,-\infty,1,-\infty,-\infty)\\
V &= (-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,0,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty)\\
\mathbf{r}-(d+1) &= 1\\
V_{\mathcal{F}} &= (-\infty,-\infty,-\infty,0,-\infty,-\infty,-\infty,-\infty,1,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty)\\
W_{\mathcal{F}} &= (-\infty,-\infty,-\infty,0,-\infty,-\infty,-\infty,-\infty,1,2,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty)\\
V &= (-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,0,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty)\\
\mathbf{r}-(d+1) &= 0\\
V_{\mathcal{F}} &= (-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,0,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty)\\
W_{\mathcal{F}} &= (-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,0,1,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty)\\
V &= (-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,0,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty)\\
\mathbf{r}-(d+1) &= 8\\
V_{\mathcal{F}} &= (4,5,6,7,6,3,4,5,8,5,6,3,6,3,4,5)\\
W_{\mathcal{F}} &= (4,5,6,7,6,7,4,5,8,9,6,7,6,7,4,5)\\
V &= (-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,0,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty)\\
\mathbf{r}-(d+1) &= 7\\
V_{\mathcal{F}} &= (3,4,5,2,5,2,7,4,3,4,5,2,5,6,3,4)\\
W_{\mathcal{F}} &= (3,4,5,6,5,6,7,8,3,4,5,6,5,6,3,4)
\end{aligned}
$$

$$
\begin{aligned}
V &= (-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,0,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty)\\
\mathbf{r}-(d+1) &= 6\\
V_{\mathcal{F}} &= (2,3,4,5,4,1,2,3,6,3,4,-\infty,4,1,2,3)\\
W_{\mathcal{F}} &= (2,3,4,5,4,5,2,3,6,7,4,5,4,5,2,3)\\
V &= (-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,0,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty)\\
\mathbf{r}-(d+1) &= 5\\
V_{\mathcal{F}} &= (1,-\infty,3,0,3,-\infty,5,2,1,2,3,-\infty,3,4,-\infty,2)\\
W_{\mathcal{F}} &= (1,2,3,4,3,4,5,6,1,2,3,4,3,4,-\infty,2)\\
V &= (-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,0,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty)\\
\mathbf{r}-(d+1) &= 4\\
V_{\mathcal{F}} &= (-\infty,1,-\infty,3,2,-\infty,0,1,4,-\infty,2,-\infty,2,-\infty,-\infty,-\infty)\\
W_{\mathcal{F}} &= (-\infty,1,-\infty,3,2,3,0,1,4,5,2,3,2,3,-\infty,-\infty)\\
V &= (-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,0,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty)\\
\mathbf{r}-(d+1) &= 3\\
V_{\mathcal{F}} &= (-\infty,-\infty,1,-\infty,1,-\infty,3,-\infty,-\infty,0,-\infty,-\infty,-\infty,2,-\infty,-\infty)\\
W_{\mathcal{F}} &= (-\infty,-\infty,1,2,1,2,3,4,-\infty,0,-\infty,-\infty,-\infty,2,-\infty,-\infty)\\
V &= (-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,0,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty)\\
\mathbf{r}-(d+1) &= 2\\
V_{\mathcal{F}} &= (-\infty,-\infty,-\infty,1,-\infty,-\infty,-\infty,-\infty,2,-\infty,0,-\infty,-\infty,-\infty,-\infty,-\infty)\\
W_{\mathcal{F}} &= (-\infty,-\infty,-\infty,1,-\infty,-\infty,-\infty,-\infty,2,3,0,1,-\infty,-\infty,-\infty,-\infty)\\
V &= (-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,0,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty)\\
\mathbf{r}-(d+1) &= 1\\
V_{\mathcal{F}} &= (-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,1,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,0,-\infty,-\infty)\\
W_{\mathcal{F}} &= (-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,1,2,-\infty,-\infty,-\infty,-\infty,-\infty,0,-\infty,-\infty)\\
V &= (-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,0,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty)\\
\mathbf{r}-(d+1) &= 0\\
V_{\mathcal{F}} &= (-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,0,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty)\\
W_{\mathcal{F}} &= (-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,0,1,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty)\\
V &= (-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,0,-\infty,-\infty,-\infty,-\infty,-\infty)\\
\mathbf{r}-(d+1) &= 8\\
V_{\mathcal{F}} &= (6,3,4,5,4,5,6,3,6,3,8,5,4,5,6,7)\\
W_{\mathcal{F}} &= (6,7,4,5,4,5,6,7,6,7,8,9,4,5,6,7)\\
V &= (-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,0,-\infty,-\infty,-\infty,-\infty,-\infty)\\
\mathbf{r}-(d+1) &= 7\\
V_{\mathcal{F}} &= (5,2,3,4,3,4,5,6,5,2,3,4,7,4,5,2)\\
W_{\mathcal{F}} &= (5,6,3,4,3,4,5,6,5,6,3,4,7,8,5,6)\\
V &= (-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,0,-\infty,-\infty,-\infty,-\infty,-\infty)\\
\mathbf{r}-(d+1) &= 6\\
V_{\mathcal{F}} &= (4,5,2,3,6,3,4,-\infty,4,1,2,3,2,3,4,1)\\
W_{\mathcal{F}} &= (4,5,2,3,6,7,4,5,4,5,2,3,2,3,4,5)
\end{aligned}
$$

$$
\begin{aligned}
V &= (-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,0,-\infty,-\infty,-\infty,-\infty,-\infty)\\
\mathbf{r}-(d+1) &= 5\\
V_{\mathcal{F}} &= (3,-\infty,5,2,-\infty,2,3,0,3,4,1,-\infty,1,2,3,-\infty)\\
W_{\mathcal{F}} &= (3,4,5,6,-\infty,2,3,4,3,4,1,2,1,2,3,4)\\
V &= (-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,0,-\infty,-\infty,-\infty,-\infty,-\infty)\\
\mathbf{r}-(d+1) &= 4\\
V_{\mathcal{F}} &= (2,-\infty,-\infty,-\infty,0,1,2,-\infty,2,-\infty,4,-\infty,-\infty,1,-\infty,3)\\
W_{\mathcal{F}} &= (2,3,-\infty,-\infty,0,1,2,3,2,3,4,5,-\infty,1,-\infty,3)\\
V &= (-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,0,-\infty,-\infty,-\infty,-\infty,-\infty)\\
\mathbf{r}-(d+1) &= 3\\
V_{\mathcal{F}} &= (1,-\infty,-\infty,0,-\infty,-\infty,-\infty,2,1,-\infty,-\infty,-\infty,3,-\infty,-\infty,-\infty)\\
W_{\mathcal{F}} &= (1,2,-\infty,0,-\infty,-\infty,-\infty,2,1,2,-\infty,-\infty,3,4,-\infty,-\infty)\\
V &= (-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,0,-\infty,-\infty,-\infty,-\infty,-\infty)\\
\mathbf{r}-(d+1) &= 2\\
V_{\mathcal{F}} &= (-\infty,1,-\infty,-\infty,2,-\infty,0,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty)\\
W_{\mathcal{F}} &= (-\infty,1,-\infty,-\infty,2,3,0,1,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty)\\
V &= (-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,0,-\infty,-\infty,-\infty,-\infty,-\infty)\\
\mathbf{r}-(d+1) &= 1\\
V_{\mathcal{F}} &= (-\infty,-\infty,1,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,0,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty)\\
W_{\mathcal{F}} &= (-\infty,-\infty,1,2,-\infty,-\infty,-\infty,-\infty,-\infty,0,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty)\\
V &= (-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,0,-\infty,-\infty,-\infty,-\infty,-\infty)\\
\mathbf{r}-(d+1) &= 0\\
V_{\mathcal{F}} &= (-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,0,-\infty,-\infty,-\infty,-\infty,-\infty)\\
W_{\mathcal{F}} &= (-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,0,1,-\infty,-\infty,-\infty,-\infty)\\
V &= (-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,0,-\infty,-\infty,-\infty)\\
\mathbf{r}-(d+1) &= 8\\
V_{\mathcal{F}} &= (6,3,4,5,4,5,6,7,6,3,4,5,8,5,6,3)\\
W_{\mathcal{F}} &= (6,7,4,5,4,5,6,7,6,7,4,5,8,9,6,7)\\
V &= (-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,0,-\infty,-\infty,-\infty)\\
\mathbf{r}-(d+1) &= 7\\
V_{\mathcal{F}} &= (5,6,3,4,7,4,5,2,5,2,3,4,3,4,5,2)\\
W_{\mathcal{F}} &= (5,6,3,4,7,8,5,6,5,6,3,4,3,4,5,6)\\
V &= (-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,0,-\infty,-\infty,-\infty)\\
\mathbf{r}-(d+1) &= 6\\
V_{\mathcal{F}} &= (4,1,6,3,2,3,4,1,4,5,2,3,2,3,4,-\infty)\\
W_{\mathcal{F}} &= (4,5,6,7,2,3,4,5,4,5,2,3,2,3,4,5)\\
V &= (-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,0,-\infty,-\infty,-\infty)\\
\mathbf{r}-(d+1) &= 5\\
V_{\mathcal{F}} &= (3,0,1,-\infty,1,2,3,-\infty,3,-\infty,5,2,-\infty,2,3,4)\\
W_{\mathcal{F}} &= (3,4,1,2,1,2,3,4,3,4,5,6,-\infty,2,3,4)
\end{aligned}
$$

$$
\begin{aligned}
V &= (-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,0,-\infty,-\infty,-\infty)\\
\mathbf{r}-(d+1) &= 4\\
V_{\mathcal{F}} &= (2,-\infty,0,1,-\infty,-\infty,-\infty,3,2,-\infty,-\infty,1,4,-\infty,2,-\infty)\\
W_{\mathcal{F}} &= (2,3,0,1,-\infty,-\infty,-\infty,3,2,3,-\infty,1,4,5,2,3)\\
V &= (-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,0,-\infty,-\infty,-\infty)\\
\mathbf{r}-(d+1) &= 3\\
V_{\mathcal{F}} &= (-\infty,2,-\infty,-\infty,3,-\infty,1,-\infty,-\infty,-\infty,-\infty,0,-\infty,-\infty,1,-\infty)\\
W_{\mathcal{F}} &= (-\infty,2,-\infty,-\infty,3,4,1,2,-\infty,-\infty,-\infty,0,-\infty,-\infty,1,2)\\
V &= (-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,0,-\infty,-\infty,-\infty)\\
\mathbf{r}-(d+1) &= 2\\
V_{\mathcal{F}} &= (-\infty,-\infty,2,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,1,-\infty,-\infty,-\infty,-\infty,0,-\infty)\\
W_{\mathcal{F}} &= (-\infty,-\infty,2,3,-\infty,-\infty,-\infty,-\infty,-\infty,1,-\infty,-\infty,-\infty,-\infty,0,1)\\
V &= (-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,0,-\infty,-\infty,-\infty)\\
\mathbf{r}-(d+1) &= 1\\
V_{\mathcal{F}} &= (-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,1,-\infty,-\infty,-\infty,-\infty,0)\\
W_{\mathcal{F}} &= (-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,1,2,-\infty,-\infty,-\infty,0)\\
V &= (-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,0,-\infty,-\infty,-\infty)\\
\mathbf{r}-(d+1) &= 0\\
V_{\mathcal{F}} &= (-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,0,-\infty,-\infty,-\infty)\\
W_{\mathcal{F}} &= (-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,0,1,-\infty,-\infty)\\
V &= (-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,0,-\infty)\\
\mathbf{r}-(d+1) &= 8\\
V_{\mathcal{F}} &= (4,5,6,3,6,3,4,5,4,5,6,7,6,3,8,5)\\
W_{\mathcal{F}} &= (4,5,6,7,6,7,4,5,4,5,6,7,6,7,8,9)\\
V &= (-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,0,-\infty)\\
\mathbf{r}-(d+1) &= 7\\
V_{\mathcal{F}} &= (3,4,5,2,5,2,3,4,3,4,5,6,5,2,7,4)\\
W_{\mathcal{F}} &= (3,4,5,6,5,6,3,4,3,4,5,6,5,6,7,8)\\
V &= (-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,0,-\infty)\\
\mathbf{r}-(d+1) &= 6\\
V_{\mathcal{F}} &= (2,3,4,-\infty,4,1,2,3,2,3,4,5,4,1,6,3)\\
W_{\mathcal{F}} &= (2,3,4,5,4,5,2,3,2,3,4,5,4,5,6,7)\\
V &= (-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,0,-\infty)\\
\mathbf{r}-(d+1) &= 5\\
V_{\mathcal{F}} &= (1,2,3,-\infty,3,0,-\infty,2,1,-\infty,3,4,3,-\infty,5,2)\\
W_{\mathcal{F}} &= (1,2,3,4,3,4,-\infty,2,1,2,3,4,3,4,5,6)\\
V &= (-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,0,-\infty)\\
\mathbf{r}-(d+1) &= 4\\
V_{\mathcal{F}} &= (0,1,2,-\infty,2,-\infty,-\infty,1,-\infty,-\infty,-\infty,3,2,-\infty,4,-\infty)\\
W_{\mathcal{F}} &= (0,1,2,3,2,3,-\infty,1,-\infty,-\infty,-\infty,3,2,3,4,5)
\end{aligned}
$$

$$
\begin{aligned}
V &= (-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,0,-\infty)\\
\mathbf{r}-(d+1) &= 3\\
V_{\mathcal{F}} &= (-\infty,0,1,-\infty,1,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,2,-\infty,-\infty,3,-\infty)\\
W_{\mathcal{F}} &= (-\infty,0,1,2,1,2,-\infty,-\infty,-\infty,-\infty,-\infty,2,-\infty,-\infty,3,4)\\
V &= (-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,0,-\infty)\\
\mathbf{r}-(d+1) &= 2\\
V_{\mathcal{F}} &= (-\infty,-\infty,0,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,1,-\infty,-\infty,2,-\infty)\\
W_{\mathcal{F}} &= (-\infty,-\infty,0,1,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,1,-\infty,-\infty,2,3)\\
V &= (-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,0,-\infty)\\
\mathbf{r}-(d+1) &= 1\\
V_{\mathcal{F}} &= (-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,0,-\infty,-\infty,1,-\infty)\\
W_{\mathcal{F}} &= (-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,0,-\infty,-\infty,1,2)\\
V &= (-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,0,-\infty)\\
\mathbf{r}-(d+1) &= 0\\
V_{\mathcal{F}} &= (-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,0,-\infty)\\
W_{\mathcal{F}} &= (-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,0,1)
\end{aligned}
$$

ii CLEFIA

$$
\begin{aligned}
V &= (0,-\infty,-\infty,-\infty)\\
\mathbf{r}-(d+1) &= 4\\
V_{\mathcal{F}} &= (4,1,2,3)\\
W_{\mathcal{F}} &= (4,5,2,3)\\
V &= (0,-\infty,-\infty,-\infty)\\
\mathbf{r}-(d+1) &= 3\\
V_{\mathcal{F}} &= (3,0,1,2)\\
W_{\mathcal{F}} &= (3,4,1,2)\\
V &= (0,-\infty,-\infty,-\infty)\\
\mathbf{r}-(d+1) &= 2\\
V_{\mathcal{F}} &= (2,-\infty,0,1)\\
W_{\mathcal{F}} &= (2,3,0,1)\\
V &= (0,-\infty,-\infty,-\infty)\\
\mathbf{r}-(d+1) &= 1\\
V_{\mathcal{F}} &= (1,-\infty,-\infty,0)\\
W_{\mathcal{F}} &= (1,2,-\infty,0)
\end{aligned}
$$

$$
\begin{aligned}
V &= (0, -\infty, -\infty, -\infty) \\
\mathbf{r} - (d+1) &= 0 \\
V_{\mathcal{F}} &= (0, -\infty, -\infty, -\infty) \\
W_{\mathcal{F}} &= (0, 1, -\infty, -\infty) \\
V &= (-\infty, -\infty, 0, -\infty) \\
\mathbf{r} - (d+1) &= 4 \\
V_{\mathcal{F}} &= (2, 3, 4, 1) \\
W_{\mathcal{F}} &= (2, 3, 4, 5) \\
V &= (-\infty, -\infty, 0, -\infty) \\
\mathbf{r} - (d+1) &= 3 \\
V_{\mathcal{F}} &= (1, 2, 3, 0) \\
W_{\mathcal{F}} &= (1, 2, 3, 4) \\
V &= (-\infty, -\infty, 0, -\infty) \\
\mathbf{r} - (d+1) &= 2 \\
V_{\mathcal{F}} &= (0, 1, 2, -\infty) \\
W_{\mathcal{F}} &= (0, 1, 2, 3) \\
V &= (-\infty, -\infty, 0, -\infty) \\
\mathbf{r} - (d+1) &= 1 \\
V_{\mathcal{F}} &= (-\infty, 0, 1, -\infty) \\
W_{\mathcal{F}} &= (-\infty, 0, 1, 2) \\
V &= (-\infty, -\infty, 0, -\infty) \\
\mathbf{r} - (d+1) &= 0 \\
V_{\mathcal{F}} &= (-\infty, -\infty, 0, -\infty) \\
W_{\mathcal{F}} &= (-\infty, -\infty, 0, 1)
\end{aligned}
$$

F Rappel des notations et acronymes par ordre alphabétique

i Acronymes

- AES : Advanced Encryption Standard.
- ALU : Arithmetic Logic Unit.
- CPU : Central Processing Unit.
- CPA : Correlation Power Analysis.
- DBA : Differential Behaviour Analysis.
- DES : Data Encryption Standard.
- DFA : Differential Fault Analysis.
- FIA : Fault Injection Analysis.
- FIRE : Fault Injection Reverse Engineering.
- FGPA : Field-Programmable Gate Array.
- FSA : Fault Sensitive Analysis.
- GFN : Generalized Feistel Network.
- NICV : Nomalized Inter-Class Variance.
- NIST : National Institute of Standards and Technology.
- *PoI* : Points d'intérêt
- SCA : Side Channel Analysis.
- SCARE : Side Channel Attack Reverse Engineering.

ii Notations

- A : variable ou constante.
- a : une constante.

- B : désigne un bloc.
- $\mathbf{b}$: les blocs d'un GFN sont numérotés de 0 à $\mathbf{b}$.
- b : désigne une fonction booléenne.

- C : texte chiffré.
- $\mathcal{C}$: consommation parfaite.
- $\mathcal{C}'$: consommation avec un bruit de mesure.
- c : indice de colonne du state de l'AES.
- clk : horloge.
- c_k : compteur associée à l'hypothèse k.
- $Conso$: mesure de consommation de courant.
- $conso$: consommation de courant, la fonction physique.
- Cor : corrélation.
- Cov : covariance.

- D : dioïde.
- $\mathbb{D} = \mathbb{N} \cup \{-\infty\}$
- d : délais de diffusion.

- E : fonction d'expansion du DES.
- $\mathbb{E}$: espérance mathématique.

- EM : émission d'ondes électromagnétiques.
- EM : mesure d'émission d'ondes électromagnétiques.
- em : émission d'ondes électromagnétiques (la fonction physique).
- erf : fonction d'erreur de Gauss.

- f : fonction physique.
- F : fonction.
- $\mathcal{F}$: fonction de tour ou de Feistel.

- G_i : graphe associée à la s-box S_i.
- g : fonction.

- H : entropie.
- HD : distance de Hamming.
- HW : poids de Hamming.
- $H_X = HW(X)$.
- H'_X : mesure de H_X.
- $h_x = HW(X = x)$.

- I : information mutuelle.
- i : désigne un indice.
- IP : permutation initiale du DES.

- J et j : désigne des indices.
- K : clé de chiffrement
- $\mathbb{K}$: ensemble des hypothèses k.
- $\mathcal{K}$: cible d'une attaque.
- k : hypothèse sur la cible.
- $\hat{k}$: valeur réelle de la cible.
- k_d : hypothèse retournée par le distingueur.
- $\mathbb{K}_{\{(h_x,h_y)_i\}_{i\in[\![1,n]\!]}}$: ensemble des hypothèses $\mathbb{K}$ intersecté des hypothèses de clé possibles pour les n couples de poids de Hamming $(h_x, h_y)_i$.
- L : registre gauche d'un schéma de Feistel.
- $\mathcal{L}$ et $\mathcal{L}'$: listes.
- l : indice de ligne.
- $\mathbf{l}$: nombre de lignes.

- M : nombre de valeurs différentes pour les O_R dans une campagne.
- M_i : une matrice
- m : désigne un ou un ensemble de modèles.
- $\mathcal{M}$: matrice de diffusion.
- MC : **MixColumn**.

- N : cardinal de $\mathbb{P}$
- $\mathcal{N}$: loi normale.
- n : désigne une taille de campagne ou d'échantillons.
- n_f nombre de fautes pour retrouver les fragments de clé.
- n_Δ : nombre de différentielles.
- n_γ : nombre de fragments de clé attaqués.

- n_λ : nombre de blocs de clé attaqués.

- $\hat{o} = o_J$: résultat de la première expérience.
- O_S : observable stimulus.
- O_R : observable réaction.
- o_J : valeurs possibles des O_R pour une campagne finie.

- P : permutation du DES.
- $\mathbb{P}$: ensemble des prédictions possibles.
- $\mathcal{P}$: probabilité d'obtenir θ succès.
- $\mathbf{p} = Pr(\rho_{i_2} = \rho_{i_1})$: probabilité d'erreur de Θ_p.
- p_{hw} : probabilité d'estimer correctement un poids de Hamming.
- $P_{m,k}$: prédiction pour l'ensemble de modèle(s) m et l'hypothèse k.
- Pr : désigne une probabilité.

- q et q' des nombres de bits.

- R : registre droit d'un schéma de Feistel.
- $\mathcal{R}$: chemin d'attaque.
- $\mathcal{R}_m$: chemin d'attaque théorique associé au(x) modèle(s) m.
- $\mathbf{r}$: les tours d'un algorithmes de chiffrement par blocs sont numérotés de 0 à $\mathbf{r}$.

- S : s-box.
- $\mathcal{S}_{u,v} = \{x \in \mathbb{F}_2^q / S(x) \oplus S(x \oplus u) = v\}$
- SB : **SubBytes**.
- SR : **ShiftRow**.

- T : texte clair.
- t : temps.
- t_{clk} : période d'horloge.
- t_{Max} : temps du bit qui a le chemin le plus long pour aller d'un registre à un autre.
- t_s : temps pendant lequel la donnée doit rester stable.

- U : vecteur de compteur.
- U" : vecteur de compteur « randomomisé ».
- U_k : vecteur de compteur théorique pour la clé k.
- Up_{hw} : vecteur adapté à l'erreur de mesure.
- u : différence appliqué en entrée d'une s-box.

- V : vecteur associée à un tour fauté d'un schéma de Feistel.
- $V_{\mathcal{F}} = \mathcal{M}^{\mathbf{r}-(r+1)} \cdot V$
- v : sortie de s-box avec différence u en entrée.
- Var : variance.

- W : variable aléatoire définissant le bruit.
- w : événement de W.
- $W_{\mathcal{F}} = \mathcal{M}^{\mathbf{r}-r} \cdot V$

- X : variable aléatoire.

- $\mathcal{X}$: univers de X.
- x : événement de X.

- Y : variable aléatoire.
- $\mathcal{Y}$: univers de Y.
- y : événement de Y.

- z : variables

- α : constante.
- β : constante.
- Γ : nombre de s-box dans un GFN.
- γ : indice de fragment de bloc de clé.
- Δ : différentielle entre fauté et non fauté, $\Delta_a = a \oplus a^*$.
- $\delta = \log_2 \max_{u \neq 0,\, v} \#\mathcal{S}_{u,v}$
- ϵ : une expérience.
- θ : nombre moyen d'appels à Θ_m pour retrouver $\hat{k}$.
- Θ_m : oracle associé à $\mathcal{R}_m$.
- Θ_p : oracle probabiliste.
- σ : écart type.
- ρ_i : valeurs possibles des $P_{m,\hat{k}}$ pour une campagne finie.
- $\hat{\rho} = \rho_I$: résultat de la deuxième expérience.
- μ : désigne une moyenne.

- $\oplus$: fonction xor.
- * : fauté.
- $\#$: cardinal.
- $\lightning$: injection de faute (fonction physique).
- $\propto$: proportionnel.

Bibliographie

[1] Auguste Kerckhoffs. La cryptographie militaire. *Journal des sciences militaires*, 9(1) :5–38, 1883.

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

[3] Claude E. Shannon . Communication Theory of Secrecy Systems. *Bell system technical journal*, 28(4) :656–715, 1949.

[4] U.S Departement of commerce / national Institute of standards and Technology. Data Encryption Standart DES. *Federal Information Processing Standards Publication*, 1999.

[5] Ernest F. Brickell,Judy H Moore and M.R. Purtill. Structure in the S-Boxes of the DES. In *Advances in Cryptology—CRYPTO'86*, pages 3–8. Springer, 1987.

[6] ClaudeCarlet . Vectorial Boolean functions for cryptography. *Boolean Models and Methods in Mathematics, Computer Science, and Engineering*, 134 :398–469, 2010.

[7] NIST. Specification for the Advanced Encryption Standard. *FIPS PUB 197*, 197, November 2001.

[8] Eli Biham and Adi Shamir. Differential cryptanalysis of DES-like cryptosystems. *Journal of CRYPTOLOGY*, 4(1) :3–72, 1991.

[9] Nicolas Courtois and Gregory V. Bard. Algebraic cryptanalysis of the data encryption standard. In *Cryptography and Coding*, pages 152–169. Springer, 2007.

[10] Frédéric Raynal. *Attaques et mesures de protection des SI*, volume base documentaire : 42313210. Editions T.I., 2014. fre.

[11] Eli Biham and Adi Shamir. Differential Fault Analysis of Secret Key Cryptosystems. In *CRYPTO*, pages 513–525, 1997.

[12] Christophe Giraud. DFA on AES. In H. Dobbertin and V. Rijmen and A. Sowa, editor, *Advanced Encryption Standard - AES*, volume 3373 of *Lecture Notes in Computer Science*, pages 27–41. Springer, 2005.

[13] Gilles Piret and Jean-Jacques Quisquater. A differential fault attack technique against SPN structures, with application to the AES and Khazad. In C.D. Walter, editor, *Cryptographic Hardware and Embedded Systems — CHES 2003*, volume 2779 of *Lecture Notes in Computer Science*, pages 77–88. Springer, 2003.

[14] Ronan Lashermes, Guillaume Reymond, Jean-Max Dutertre, Jacques Fournier, Bruno Robisson and Assia Tria. A DFA on AES Based on the Entropy of Error Distributions. In *FDTC*, pages 34–43, 2012.

[15] Michael Tunstall, Debdeep Mukhopadhyay and Subidh Ali. Differential Fault Analysis of the Advanced Encryption Standard Using a Single Fault, 2011.

[16] Mukhopadhyay, Debdeep. An Improved Fault Based Attack of the Advanced Encryption Standard. In Preneel, Bart, editor, *Progress in Cryptology AFRICACRYPT 2009*, volume 5580 of *Lecture Notes in Computer Science*, pages 421–434. Springer Berlin / Heidelberg, 2009.

[17] Christophe Clavier. Secret external encodings do not prevent transient fault analysis. *Cryptographic Hardware and Embedded Systems-CHES 2007*, pages 181–194, 2007.

[18] Matthieu Rivain. Differential Fault Analysis on DES Middle Rounds. In *CHES*, pages 457–469, 2009.

[19] Yang Li, Kazuo Sakiyama, Shigeto Gomisawa, Toshinori Fukunaga, Junko Takahashi and Kazuo Ohta. Fault Sensitivity Analysis. In *CHES*, volume 6225, pages 320–334, 2010.

[20] Bruno Robisson and Pascal Manet. Differential Behavioral Analysis. In *CHES*, pages 413–426, 2007.

[21] Julia Borghoff, Lars R. Knudsen, Gregor Leander and Krystian Matusiewicz. Cryptanalysis of C2. In *Advances in Cryptology-CRYPTO 2009*, pages 250–266. Springer, 2009.

[22] Christophe Clavier. An improved SCARE cryptanalysis against a secret A3/A8 GSM algorithm. *Information Systems Security*, pages 143–155, 2007.

[23] Christophe Clavier and Antoine Wurcker . Reverse Engineering of a Secret AES-like Cipher by Ineffective Fault Analysis. In *Fault Diagnosis and Tolerance in Cryptography (FDTC), 2013 Workshop on*, pages 119–128, 2013.

[24] Matthieu Rivain and Thomas Roche. SCARE of Secret Ciphers with SPN Structures. In *Advances in Cryptology - ASIACRYPT 2013 - 19th International Conference on the Theory and Application of Cryptology and Information Security, Bengaluru, India, December 1-5, 2013, Proceedings, Part I*, pages 526–544, 2013.

[25] Amine Dehbaoui. *Analyse Sécuritaire des Émanations Électromagnétiques des Circuits Intégrés*. PhD thesis, Montpellier 2, 2011.

[26] Driss Aboulkassimi, Michel Agoyan, Laurent Freund, Jacques J. A. Fournier, Bruno Robisson and Assia Tria. ElectroMagnetic analysis (EMA) of software AES on Java mobile phones. In *WIFS*, pages 1–6, 2011.

[27] Loïc Zussa, Jean-Max Dutertre, Jessy Clédière and Assia Tria. Power supply glitch induced faults on FPGA : An in-depth analysis of the injection mechanism. In *IOLTS*, pages 110–115, 2013.

[28] Loïc Zussa, Jean-Max Dutertre, Jessy Clédière and Bruno Robisson. Analysis of the fault injection mechanism related to negative and positive power supply glitches using an on-chip voltmeter. In *HOST*, pages 130–135, 2014.

[29] Hélène Le Bouder, Bruno Robisson and Assia Tria. A Unified Formalism for Side-Channel and Fault Attacks on Cryptographic Circuits. In *Chip-to-Cloud Security Forum*, 2013.

[30] Hélène Le Bouder , Ronan Lashermes , Yanis Linge , Bruno Robisson and Assia Tria. A Unified Formalism for Physical Attacks. Cryptology ePrint Archive, Report 2014/682, 2014. `http://eprint.iacr.org/`.

[31] Hélène Le Bouder, Sylvain Guilley , Bruno Robisson and Assia Tria. Fault Injection to Reverse Engineer DES-like Cryptosystems. In *Chip-to-Cloud Security Forum*, 2013.

[32] Hélène Le Bouder, Sylvain Guilley, Bruno Robisson and Assia Tria. Fault Injection to Reverse Engineer DES-Like Cryptosystems. In *FPS*, pages 105–121, 2013.

[33] Silvio Micali and Leonid Reyzin. Physically Observable Cryptography. Cryptology ePrint Archive, Report 2003/120, 2003.

[34] Stefan Mangard, Elisabeth Oswald and François-Xavier Standaert. One for All - All for One : Unifying Standard DPA Attacks. Cryptology ePrint Archive, Report 2009/449, 2009.

[35] François-Xavier Standaert, François Koeune and Werner Schindler . How to compare profiled side-channel attacks ? In *Applied Cryptography and Network Security*, pages 485–498. Springer, 2009.

[36] François-Xavier Standaert, Tal G Malkin, and Moti Yung. A formal practice-oriented model for the analysis of side-channel attacks. *IACR e-print archive*, 134, 2006.

[37] François-Xavier Standaert, Tal Malkin and Moti G Yung. A unified framework for the analysis of side-channel key recovery attacks. In *Advances in Cryptology-Eurocrypt 2009*, pages 443–461. Springer, 2009.

[38] Stefan Mangard, Elisabeth Oswald and François-Xavier Standaert. One for all–all for one : unifying standard differential power analysis attacks. *IET Information Security*, 5(2) :100–110, 2011.

[39] Abdelaziz M Elaabid and Sylvain Guilley. Practical improvements of profiled side-channel attacks on a hardware crypto-accelerator. In *Progress in Cryptology–AFRICACRYPT 2010*, pages 243–260. Springer, 2010.

[40] Carolyn Whitnall, Elisabeth Oswald and François-Xavier Standaert. The myth of generic DPA... and the magic of learning. *IACR Cryptology ePrint Archive*, 2012 :256, 2012.

[41] Julien Doget, Emmanuel Prouff, Matthieu Rivain and François Xavier Standaert. Univariate side channel attacks and leakage modeling. *Journal of Cryptographic Engineering*, 1(2) :123–144, 2011.

[42] Ingrid Verbauwhede, Dusko Karaklajic and Jörn-Marc Schmidt. The Fault Attack Jungle-A Classification Model to Guide You. In *Fault Diagnosis and Tolerance in Cryptography (FDTC), 2011 Workshop on*, pages 3–8. IEEE, 2011.

[43] Amir Moradi, Mohammad T Manzuri Shalmani and Mahmoud Salmasizadeh. A generalized method of differential fault attack against AES cryptosystem. In *Cryptographic Hardware and Embedded Systems-CHES 2006*, pages 91–100. Springer, 2006.

[44] Kazuo Sakiyama and Yang Li and Mitsugu Iwamoto and Kazuo Ohta. Information-Theoretic Approach to Optimal Differential Fault Analysis. *IEEE Transactions on Information Forensics and Security*, 7(1) :109–120, 2012.

[45] Emmanuel Prouff and Matthieu Rivain. Masking against Side-Channel Attacks : A Formal Security Proof. In *EUROCRYPT*, volume 7881, pages 142–159. Springer, 2013.

[46] Werner Schindler. Advanced stochastic methods in side channel analysis on block ciphers in the presence of masking. *Journal of Mathematical Cryptology*, 2(3) :291–310, 2008.

[47] Houssem Maghrebi, Olivier Rioul, Sylvain Guilley and Jean-Luc Danger. Comparison between Side-Channel Analysis Distinguishers. In *ICICS*, pages 331–340, 2012.

[48] Julien Doget, Emmanuel Prouff, Matthieu Rivain, and François-Xavier Standaert, François-Xavier. Univariate side channel attacks and leakage modeling. *Journal of Cryptographic Engineering*, 1(2) :123–144, 2011.

[49] Paul C. Kocher and Joshua Jaffe and Benjamin Jun. Differential Power Analysis. In *CRYPTO*, pages 388–397, 1999.

[50] Eric Brier, Christophe Clavier and Francis Olivier. Correlation Power Analysis with a Leakage Model. In *CHES*, pages 16–29, 2004.

[51] Benedikt Gierlichs, Lejla Batina and Pim Tuyls. Mutual Information Analysis - A Universal Differential Side-Channel Attack. *IACR Cryptology ePrint Archive*, 2007 :198, 2007.

[52] Lejla Batina, Benedikt Gierlichs, Emmanuel Prouff, Matthieu Rivain, François-Xavier Standaert and Nicolas Veyrat-Charvillon. Mutual Information Analysis : a Comprehensive Study. *J. Cryptology*, 24(2) :269–291, 2011.

[53] Emmanuel Prouff and Matthieu Rivain. Theoretical and Practical Aspects of Mutual Information Based Side Channel Analysis. In *ACNS*, pages 499–518, 2009.

[54] Daniel Pérez Palomar and Sergio Verdù. Gradient of Mutual Information in Linear Gaussian Channels. In *Information Theory, IEEE Transactions*, volume 52, 2006.

[55] Amir Moradi, Nima Mousavi, Christof Paar and Mahmoud Salmasizadeh. A Comparative Study of Mutual Information Analysis under a Gaussian Assumption. In *WISA*, pages 193–205, 2009.

[56] Annelie Heuser, Sylvain Guilley and Olivier Rioul. A Theoretical Study of Kolmogorov-Smirnov Distinguishers : Side-Channel Analysis vs. Differential Cryptanalysis. *IACR Cryptology ePrint Archive*, 2014 :8, 2014.

[57] Carolyn Whitnall, Elisabeth Oswald and Luke Mather . An exploration of the kolmogorov-smirnov test as a competitor to mutual information analysis. In *Smart Card Research and Advanced Applications*, pages 234–251. Springer, 2011.

[58] Youssef Souissi, Maxime Nassar, Sylvain Guilley, Jean-Luc Danger and Florent Flament. First Principal Components Analysis : A New Side Channel Distinguisher. In *ICISC*, pages 407–419, 2010.

[59] François-Xavier Standaert and Cédric Archambeau. Using Subspace-Based Template Attacks to Compare and Combine Power and Electromagnetic Information Leakages. In *CHES*, pages 411–425, 2008.

[60] Suresh Balakrishnama and Aravind Ganapathiraju. Linear Discriminant Analysis - A Brief Tutorial. *Institute for Signal and Information Processing, Mississippi State University*, 1998.

[61] Eric Briers, Christophe Clavier and Francis Olivier. Optimal Statistical Power Analysis. *IACR Cryptology ePrint Archive*, 2003 :152, 2003.

[62] Alessandro Barenghi, Luca Breveglieri, Israel Koren and David Naccache. Fault Injection Attacks on Cryptographic Devices : Theory, Practice, and Countermeasures. *Proceedings of the IEEE*, 100(11) :3056–3076, 2012.

[63] Carolyn Whitnall and Elisabeth Oswald. A fair evaluation framework for comparing side-channel distinguishers. *Journal of Cryptographic Engineering*, 1(2) :145–160, 2011.

[64] Stefan Dziembowski and Krzysztof Pietrzak . Leakage-resilient cryptography. In *Foundations of Computer Science, 2008. FOCS'08. IEEE 49th Annual IEEE Symposium on*, pages 293–302. IEEE, 2008.

[65] Stefan Mangard . Hardware countermeasures against DPA–a statistical analysis of their effectiveness. In *Topics in Cryptology–CT-RSA 2004*, pages 222–235. Springer, 2004.

[66] Shivam Bhasin, Jean-Luc Danger, Sylvain Guilley and Zakaria Najm. NICV : Normalized Inter-Class Variance for Detection of Side-Channel Leakage. Cryptology ePrint Archive, Report 2013/717, 2013. `http://eprint.iacr.org/`.

[67] Yanis Linge, Cécile Dumas and Sophie Lambert-Lacroix . Using the Joint Distributions of a Cryptographic Function in Side Channel Analysis. In *In preparation. See IACR E-print*. Citeseer, 2014.

[68] Cédric Archambeau, Eric Peeters, François-Xavier Standaert and Jean-Jacques Quisquater. Template attacks in principal subspaces. In *Cryptographic Hardware and Embedded Systems-CHES 2006*, pages 1–14. Springer, 2006.

[69] Mathieu Renauld and François Xavier Standaert. Algebraic side-channel attacks. In *Information Security and Cryptology*, pages 393–410. Springer, 2011.

[70] Mathieu Renauld, François-Xavier Standaert and Nicolas Veyrat-Charvillon. Algebraic side-channel attacks on the AES : Why time also matters in DPA. In *Cryptographic Hardware and Embedded Systems-CHES 2009*, pages 97–111. Springer, 2009.

[71] Frederik Armknecht and Matthias Krause. Algebraic attacks on combiners with memory. In *Advances in Cryptology-CRYPTO 2003*, pages 162–175. Springer, 2003.

[72] Yanis Linge. *Études cryptographiques et statistiques de signaux compromettants*. PhD thesis, 2014.

[73] Stefan Mangard, Elisabeth Oswald and Thomas Popp. *Power Analysis Attacks - Revealing the Secrets of Smart Cards*. Springer Verlag, 2007.

[74] David Barber. *Bayesian Reasoning and Machine Learning*. Cambridge University Press, 04-2011 edition, 2011. In press.

[75] Nicolas Veyrat-Charvillon, Benoît Gérard, Mathieu Renauld and François-Xavier Standaert. An Optimal Key Enumeration Algorithm and Its Application to Side-Channel Attacks. In *Selected Areas in Cryptography*, pages 390–406, 2012.

[76] Robert G Gallager . Low-density parity-check codes. *Information Theory, IRE Transactions on*, 8(1) :21–28, 1962.

[77] Michel Agoyan, Jean-Max Dutertre, David Naccache, Bruno Robisson and Assia Tria. When Clocks Fail : On Critical Paths and Clock Faults. In *CARDIS*, pages 182–193, 2010.

[78] Michel Agoyan,Jean-Max Dutertre, Amir-Pasha Mirbaha, David Naccache, Anne-Lise Ribotta and Assia Tria. How to flip a bit ? In *IOLTS*, pages 235–239, 2010.

[79] Amine Dehbaoui and Jean-Max Dutertre, Bruno Robisson and Assia Tria. Electromagnetic transient faults injection on a hardware and a software implementations of AES. In *Fault Diagnosis and Tolerance in Cryptography (FDTC), 2012 Workshop on*, pages 7–15. IEEE, 2012.

[80] Yuliang Zheng, Tsutomu Matsumoto and Hideki Imai. On the Construction of Block Ciphers Provably Secure and Not Relying on Any Unproved Hypotheses. In *Advances in Cryptology - CRYPTO '89*, volume 435 of *LNCS*, pages 461–480. Springer, 1989.

[81] Thierry Berger P., Marine Minier and Gaël Thomas. Extended Generalized Feistel Networks using Matrix Representation. In *Selected Areas in Cryptography - SAC 2013*, volume 8282 of *LNCS*. Springer, 2013.

[82] Maryam Izadi, Babak Sadeghiyan, Seyed Saeed Sadeghian and Hossein Arabnezhad Khanooki. MIBS : A New Lightweight Block Cipher. In Juan A. Garay, Atsuko Miyaji and Akira Otsuka, editor, *CANS*, volume 5888 of *LNCS*, pages 334–348. Springer, 2009.

[83] Tomoyasu Suzaki, Kazuhiko Minematsu, Sumio Morioka and Eita Kobayashi. TWINE : A Lightweight Block Cipher for Multiple Platforms. In *Selected Areas in Cryptography - SAC 2012*, volume 7707 of *LNCS*, pages 339–354. Springer, 2012.

[84] Tomoyasu Suzaki and Kazuhiko Minematsu. Improving the Generalized Feistel. In *FSE*, volume 6147 of *LNCS*, pages 19–39. Springer, 2010.

[85] Taizo Shirai, Kyoji Shibutani, Toru Akishita, Shiho Moriai and Tetsu Iwata. The 128-Bit Blockcipher CLEFIA (Extended Abstract). In Alex Biryukov, editor, *FSE*, volume 4593 of *LNCS*, pages 181–195. Springer, 2007.

[86] Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks and Louis Wingers. The SIMON and SPECK Families of Lightweight Block Ciphers. *IACR Cryptology ePrint Archive*, page 404, 2013.

[87] Kyoji Shibutani, Takanori Isobe, Harunaga Hiwatari, Atsushi Mitsuda, Toru Akishita and Taizo Shirai. Piccolo : An Ultra-Lightweight Blockcipher. In *Cryptographic Hardware and Embedded Systems - CHES 2011*, volume 6917 of *LNCS*, pages 342–357. Springer, 2011.

[88] Kim, Jongsung and Hong, Seokhie and Sung, Jaechul and Lee, Sangjin and Lim, Jongin and Sung, Soohak. Impossible differential cryptanalysis for block cipher structures. In *Progress in Cryptology-INDOCRYPT 2003*, pages 82–96. Springer, 2003.

[89] Anne Canteaut. *Attaques de cryptosystèmes à mots de poids faible et construction de fonctions t-résilientes*. PhD thesis, 1997.

[90] *Contribution à la recherche de fonctions booléennes hautement non linéaires, et au marquage d'images en vue de la protection des droits d'auteur*. PhD thesis.

[91] Kaisa Nyberg . On the construction of highly nonlinear permutations. In *Advances in Cryptology—EUROCRYPT'92*, pages 92–98. Springer, 1993.

[92] Kaisa Nyberg . Differentially uniform mappings for cryptography. In *Advances in cryptology—Eurocrypt'93*, pages 55–64. Springer, 1994.

[93] Florent Chabaud and Serge Vaudenay . Links between differential and linear cryptanalysis. In *Advances in Cryptology—EUROCRYPT'94*, pages 356–365. Springer, 1995.

[94] Gregor Leander and Axel Poschmann. On the Classification of 4 Bit S-Boxes. In *WAIFI*, volume 4547 of *LNCS*, pages 159–176. Springer, 2007.

[95] Markku-Juhani O Saarinen. Cryptographic Analysis of All 4 x 4 - Bit S-Boxes. Cryptology ePrint Archive, Report 2011/218.

[96] Hua Chen, Wenling Wu and Dengguo Feng. Differential fault analysis on CLEFIA. In *Information and communications security*, pages 284–295. Springer, 2007.

[97] Junko Takahashi, and Toshinori Fukunaga. Improved differential fault analysis on CLEFIA. In *Fault Diagnosis and Tolerance in Cryptography, 2008. FDTC'08. 5th Workshop on*, pages 25–34. IEEE, 2008.

[98] Rémy Daudigny, Hervé Ledig, Frédéric Muller and Frédéric Valette :. SCARE of the DES. In *Applied Cryptography and Network Security*, pages 19–33. Springer, 2005.

[99] Donald E. Knuth. *The Art of Computer Programming, Volume 4A : Combinatorial Algorithms, Part 1*. Pearson Education India, 2011.

[100] Manuel San Pedro, Mate Soos and Sylvain Guilley. FIRE : fault injection for reverse engineering. *Information Security Theory and Practice. Security and Privacy of Mobile Devices in Wireless Communication*, pages 280–293, 2011.

[101] Tang Ming, Qiu Zhenlong, Deng Hui, Liu Shubo and Zhang Huanguo. Reverse Engineering Analysis Based on Differential Fault Analysis Against Secret S-boxes. *China Communications*, 9 :10, 2012.

[102] Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, Chris Hall and Niels Ferguson. Twofish : A 128-bit block cipher. *NIST AES Proposal*, 15, 1998.

[103] Nidhal Selmane, Sylvain Guilley and Jean-Luc Danger. Practical setup time violation attacks on AES. In *Dependable Computing Conference, 2008. EDCC 2008. Seventh European*, pages 91–96. IEEE, 2008.

[104] Joseph B. Kruskal. On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem. *Proceedings of The American Mathematical Society*, 7 :48–48, 1956.

École Nationale Supérieure des Mines
de Saint-Étienne

NNT : *Communiqué le jour de la soutenance*

Hélène LE BOUDER

A FORMALISM FOR PHYSICAL ATTACKS ON CRYPTOGRAPHIC DEVICE AND ITS EXPLOITATION TO COMPARE AND RESEARCH NEWS ATTACKS

Speciality : Microelectronics

Keywords : physical attacks, formalism, side channel, faults injection, power analysis, symmetric cryptography, generalized Feistel network, FIRE ,reverse-engineering...

Abstract : This PhD thesis is about physical cryptanalysis of blocks ciphers.
A cryptographic algorithm is thinking securely designed, it may be vulnerable to physical attacks. Physical attacks are a real threat, even for cryptographic algorithms proved secure mathematically. There are divided in two families: the side channel attacks which are are based on observations of the circuit behaviour during the computation, and the fault injection attacks which consist in disturbing the circuit behaviour in order to alter the correct progress of the algorithm. There are used to target the cipher key or reverse engineer the algorithm. The improvement of our formalism is to unify the two families Unifying the different attacks under a same formalism allows to deal with them with common tools; and compare them its a success of the formalism. A generic method to assess the vulnerability to differential fault analysis of generalized Feistel networks are presented. And an extension of this work is used to improve a FIRE attack on DES-like cryptosystem with customized s-boxes.

École Nationale Supérieure des Mines
de Saint-Étienne

NNT : *Communiqué le jour de la soutenance*

Hélène LE BOUDER

UN FORMALISME UNIFIANT LES ATTAQUES PHYSIQUES SUR CIRCUITS CRYTOGRAPHIQUES ET SON EXPLOITATION AFIN DE COMPARER ET RECHERCHER DE NOUVELLES ATTAQUES.

Spécialité: Microélectronique

Mots clefs : Attaques physiques, Formalisme, Attaque par observation, Injection de faute, Consommation de courant, Cryptographie symétrique, Schémas de Feistel généralisés, rétro-conception, FIRE.

Résumé :

Cette thèse se situe dans la cryptanalyse physique des algorithmes de chiffrement par blocs.
Un algorithme cryptographique est conçu pour être mathématiquement robuste. Cependant, une fois implémenté dans un circuit, il est possible d'attaquer les failles de ce dernier.
Par opposition à la cryptanalyse classique, on parle alors d'attaques physiques. Celles-ci ne permettent pas d'attaquer l'algorithme en soi, mais son implémentation matérielle. Il existe deux grandes familles d'attaques physiques différentes : les attaques par observation du circuit durant le chiffrement, et les attaques par injections de fautes, qui analysent l'effet d'une perturbation intentionnelle sur le fonctionnement du circuit. Les attaques physiques ont deux types d'objectifs : rechercher la clé ou faire de la rétro-conception (retrouver une partie d'un algorithme de chiffrement privé, ex : s-boxes modifiées).

Bien que leurs principes semblent distincts, cette thèse présente un formalisme qui permet d'unifier toutes ces attaques. L'idée est de décrire les attaques physiques de façon similaire, afin de pouvoir les comparer. De plus, ce formalisme a permis de mettre en évidence de nouvelles attaques. Des travaux novateurs ayant pour objet de retrouver la clé de chiffrement d'un AES, uniquement avec la consommation de courant ont été menés.
Une nouvelle attaque de type FIRE (Fault Injection for Reverse Engineering) pour retrouver les s-boxes d'un pseudo DES est également présentée dans la thèse.
Ce travail a abouti sur une réflexion plus générale, sur les attaques par injections de fautes dans les schémas de Feistel classiques et généralisés.