Algorithms for differential cryptanalysis

Victor Mollimard

DOCTORAT BRETAGNE LOIRE / MATHSTIC

UNIVERSITÉ DE RENNES 1

THÈSE DE DOCTORAT DE

L'UNIVERSITÉ DE RENNES 1

ÉCOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : *Informatique*

Par
Victor MOLLIMARD

Algorithmes pour la cryptanalyse différentielle

Algorithms for differential cryptanalysis

Thèse présentée et soutenue à Rennes, le 11 janvier 2022
Unité de recherche : IRISA, EMSEC

Rapporteurs avant soutenance :

Maria NAYA-PLASENCIA	Directrice de Recherche à l'INRIA Paris
Thomas PEYRIN	Professeur à la Nanyan Technological University of Singapore

Composition du Jury :

Examinateurs :	Maria NAYA-PLASENCIA	Directrice de Recherche à l'INRIA Paris
	Thomas PEYRIN	Professeur à la Nanyan Technological University of Singapore
	Christina BOURA	Maîtresse de conférences à l'Université de Versailles-Saint-Quentin-en-Yvelines
	Louis GOUBIN	Professeur des Universités à l'Université de Versailles-Saint-Quentin-en-Yvelines
	Marine MINIER	Professeure des Universités à l'Université de Lorraine
Dir. de thèse :	Pierre-Alain FOUQUE	IRISA, Professeur des Universités à l'Université de Rennes 1
Co-dir. de thèse :	Patrick DERBEZ	IRISA, Maître de conférences à l'Université de Rennes 1

Remerciements

Je tiens tout d'abord à remercier Pierre-Alain et Patrick de m'avoir accompagné pendant toutes les étapes de cette thèse, et ce, jusqu'à la rédaction et la soutenance, périodes hautes en couleurs. J'exprime également ma reconnaissance à l'École Normale Supérieure de Lyon qui a financé mon travail.

Je remercie aussi Maria et Thomas pour avoir accepté de rapporter mon manuscrit ainsi que tous les autres membres du jury, Christina, Louis et Marine qui siègent en présentiel et en distanciel malgré une petite pandémie inopportune et la distance à parcourir jusqu'à Rennes.

Je n'oublie pas les autres co-auteurs des publications : en particulier Stéphanie, pour m'avoir guidé dans l'organisation de la réunion d'équipe pendant un an, mais aussi Charles et Paul.

J'associe à mes remerciements tous les membres de l'équipe, EMSEC, puis des équipes, SPICY et CAPSULE, à commencer par mes anciens camarades hobbits, Baptiste, Pauline et Thomas. J'exprime aussi ma gratitude à mes camarades doctorants : Gauthier, Katharina et Solène ainsi que Céline et Adina qui ont partagé mon bureau. Je ne saurais oublier les « anciens » déjà partis vers d'autres horizons : Alexandre (le premier), Claire, Xavier, Florent et Angèle, Alban, Chen, Raghav, Weigqiang et tous les autres. Je garde aussi en mémoire tous les autres membres de l'équipe, les permanents : Adeline, Barbara, Gildas, Alexandre, Mohamed, Joseph, Annelie, David, Clémentine et Tristan mais aussi les doctorants et post-doctorants arrivés après moi : Daniel, Thomas, Gwendal qui me supportent depuis longtemps, Guillaume, notre expert en train, Joshua dit Jojo parti enseigner, Alexandre (le dernier) rencontré dans un stage précédent, Olivier et Olivier, Phuc et Nicolas avec leur aide au dépannage d'ordinateur un 23 décembre..., Adela, Damien, Agathe, Sadia et Diane, ces trois dernières ayant assisté avec moi à Cyber in Toulouse, Arthur qui prend en quelque sorte ma succession de doctorant en cryptanalyse symétrique, Corentin et tous ceux que j'ai pu croiser (la liste commence à être longue...). Et, pour leur aide précieuse, je remercie les quatre assistantes d'équipe dans l'ordre chronologique de nos interactions : Fabienne, Aurélie, Gaëlle et Hélène.

Je loue enfin le soutien indéfectible de ma famille et remercie en particulier mes parents, Christine et Vincent, pour leurs relectures actives du manuscrit et leurs visites à Rennes ainsi que mes sœurs, Adélie et Lorraine, pour leurs encouragements nourris.

Introduction et Résumé en français

La cryptographie du grec *kryptos*, « secret, caché » et *graphein*, « écrire » est un vieux sujet d'étude concernant en premier lieu la manière de cacher le sens d'un message à un tiers écoutant les communications. La première utilisation (en un certain sens) remonte à 1900 av. J.-C. dans une inscription de la tombe de Khnumhotep II où des hiéroglyphes inhabituels sont utilisés. Environ 1850 ans plus tard, Jules César utilise pour ses affaires militaires un chiffrement auquel il a donné son nom. Le chiffrement de `César` consiste à transformer un *message* (aussi appelé *clair*) en une séquence inintelligible de lettres appelée un *chiffré* en changeant chaque lettre du message en la n-ième (fixe) lettre suivante dans l'alphabet. Le receveur peut alors retrouver le message originel en appliquant la transformation inverse à chaque lettre du chiffré. Au cours du temps, les chiffrements se sont complexifiés avec par exemple l'utilisation de plusieurs alphabets et sur le plan pratique avec des mécanismes propres aux schémas comme des règles de conversions. Au XVI^{e} siècle, Giovan Battista Bellasi crée le chiffrement de Vigenère (faussement attribué au diplomate Blaise de Vigenère). Le chiffrement de Vigenère peut être compris comme l'application de plusieurs chiffrements de César (avec différentes translations) sur le message. Plus précisément, une séquence secrète de n lettres appelée une *clé* connue uniquement par l'envoyeur et le receveur est utilisée pour chiffrer le message préalablement découpé en blocs de taille n. Pour chaque bloc, la i-ème lettre du bloc est translatée dans l'alphabet de r lettres où r est la position dans l'alphabet de la i-ème lettre de la clé. Un exemple de chiffrement est donné en Figure 1.

À la fin du XIX^{e} siècle, la cryptographie a aussi été affectée par la vague de formalisme logique présente dans les différents milieux universitaires. En particulier, en 1883, Kerckhoff dans **La Cryptographie Militaire** proposa une liste de principes qu'un bon chiffrement devrait suivre. Les principes de Kerchkoff sont toujours utilisés de nos jours dans le design de primitives cryptographiques particulièrement au regard de la sécurité : le système devrait être en pratique indéchiffrable (sans connaissance de la clé) ou dit autrement, le système devrait garantir la *confidentialité* du message ; le seul paramètre

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Clé	L	A	T	I	N	L	A	T	I	N	L	A	T	I	N	L	A	T	I	N	L	A
Clair	L	O	R	E	M	I	P	S	U	M	D	O	L	O	R	S	I	T	A	M	E	T
Chiffré	W	O	K	M	Z	T	P	L	C	Z	O	O	E	W	E	D	I	M	I	Z	P	T

FIGURE 1 – Exemple de chiffrement de Vigenère. La table haute donne la correspondance entre les lettres en clair (colonne), les lettres de la clé (ligne) et le chiffré (entrée). La table basse donne le chiffrement de « LOREM IPSUM DOLOR SIT AMET » avec la clé « LATIN ».

secret du système devrait être la *clé* (petite et facilement changeable).

Depuis cette formalisation par Kerchkoff, deux révolutions suivirent en cryptographie. La première se passe durant la Deuxième Guerre mondiale avec la construction du premier ordinateur électronique programmable, le *Colossus*. Avec les puissances modernes de calcul, la cryptographie a largement adopté le modèle binaire pour les *clairs, clés* et *chiffrés* qui sont des séquences binaires de tailles arbitraires notées $\mathbb{F}_2^*$. L'autre révolution en cryptographie a été l'invention de la cryptographie asymétrique avec le chiffrement `RSA` [RSA78]. L'adoption de la cryptographie moderne avec les ordinateurs dans la vie civile à petits pas depuis les années 60 avec les réseaux bancaires et les communications téléphoniques. Au début des années 70, le gouvernement américain identifie un besoin pour un algorithme de chiffrement standardisé pour les informations sensibles mais non-classifiées. Ce besoin fut comblé par le `Data Encryption Standard (DES)` ouvert au pu-

blic en 1977 jusqu'à son remplacement par le fameux `Advanced Encryption Standard (AES)` en 2002. Pendant son utilisation, la sécurité du `DES` fut mise en doute notamment par le rôle qu'ont joué certains services secrets dans sa création (en particulier la NSA). Dans une optique naturelle vers plus de confiance pour les acteurs extérieurs, le processus de développement a complètement changé depuis et de nos jours existe une collaboration publique entre différents acteurs gouvernementaux, étrangers, scientifiques... Quand un nouveau schéma est jugé nécessaire, les agences de standardisation lancent un appel à contributions envers la communauté de la recherche cryptographique. Les candidats sont alors étudiés par la communauté pour aider à une première sélection par l'organisme standardisateur. Différentes sélections sont réalisées (selon des critères différents) jusqu'à obtenir un finaliste ou une liste de finalistes publiés avec des recommandations d'utilisation. Actuellement, deux compétitions majeures par le *National Institute of Science and Technology (NIST)* des États-Unis sont en cours : la compétition pour le chiffrement léger adapté aux objets connectés et la compétition post-quantique qui cherche à préparer la dépréciation de nombreux chiffrements asymétriques avec l'arrivée des ordinateurs quantiques.

Durant cette thèse, nous nous sommes focalisés sur l'étude des primitives symétriques (certaines intervenant dans la compétition légère du NIST) selon différents points de vue nous permettant de voir la cryptanalyse de différents objets en allant d'une étude complémentaire d'une attaque générale (présenté en Chapitre 1) à l'étude d'un critère dans une construction générale (Chapitre 2) et à l'étude précise d'une primitive (Chapitre 3). Pour introduire aussi clairement que possible les travaux de cette thèse, nous donnons une définition générale d'un chiffrement (Définition 0.1) en gardant le vocabulaire déjà utilisé dans cette introduction. Le reste de cette introduction donne plus de détails à propos des constructions étudiées et leur sécurité.

Définition 0.1 (Chiffrement)**.** *Une fonction de chiffrement est une fonction* $E : \mathbb{F}_2^* \times \mathbb{F}_2^* \to \mathbb{F}_2^*$ *telle que pour toute clé dans* $\mathbb{F}_2^*$*, la fonction* $E(\text{clé}, \cdot)$ *est une surjection et son inverse* $D(\text{clé}, \cdot)$ *peut être efficacement calculé lorsque la clé est connue.*

Idéalement, en plus d'assurer la *confidentialité*, un bon chiffrement symétrique devrait aussi garantir l'*intégrité*, c'est à dire la capacité à détecter un changement sur le chiffré et à le corriger si nécessaire ainsi que l'*authenticité*, c'est à dire le fait d'avoir une preuve de la provenance du chiffré. Essayer de construire directement des chiffrements ayant ces trois propriétés est un vrai défi demandant de nombreuses compétences pour chaque propriété

mais aussi pour toutes leurs interactions afin de garder un schéma sécurisé. En pratique, pour simplifier la création de schéma en cryptographie symétrique, les chiffrements sont découpés en plusieurs couches. La première couche consiste à construire une *primitive*, une fonction cryptographique qui assure la *confidentialité* du message. Les deux autres propriétés voulues (ou d'autres propriétés plus mineures) sont obtenues en intégrant une primitive dans une construction plus grosse appelée un *mode d'opération*. Cependant, la cryptographie moderne a tendance à ne pas considérer l'*intégrité* comme une propriété que doit apporter cette deuxième couche car elle est obtenue par d'autres biais loin du sujet de cette thèse. Dans ce manuscrit, nous présentons nos travaux concernant différentes primitives cryptographiques et continuons cette introduction avec la présentation de différentes constructions classiques de primitives dans la partie I, en donnant des éléments pour comprendre leur sécurité dans la partie II et nous terminons cette introduction avec la partie III par un résumé des différents chapitres correspondant aux travaux de cette thèse.

Avant d'introduire les différentes primitives, voici un bref aparté sur le `chiffrement à masque jetable`, un très célèbre schéma qui peut s'assimiler à un chiffrement de `Vigenère` où la clé est exactement de la même taille que le message et qui sert souvent de base de comparaison ou d'inspiration à d'autres schémas. Dans le `chiffrement à masque jetable`, pour chaque clair $m \in \mathbb{F}_2^*$, une clé k jetable (à utilisation unique) de la même taille que m est générée en tirant uniformément une valeur dans $\mathbb{F}_2^{|m|}$ où $|m|$ est la taille de m. Le chiffré c est calculé simplement de la manière suivante : $c = m \oplus k$ où $\oplus$ représente le ou exclusif (XOR) bit à bit entre m et k. En 1949, Shannon [Sha49] prouva que le chiffrement à masque jetable a une confidentialité parfaite (pour être précis, il prouve que le schéma a une sécurité qui est plus forte que la sécurité calculatoire utilisée maintenant) c'est à dire que même avec une puissance et un temps de calcul infinis, il n'est pas possible de retrouver le message depuis le chiffré sans connaissance de la clé. Malheureusement, le `chiffrement à masque jetable` a plusieurs défauts à commencer par le fait que la clé doit être de la même taille que le message et utilisée seulement une fois, ce qui le rend inadapté dans la vie réelle. Pour le remplacer, les cryptographes ont inventé plusieurs constructions reprenant certains aspects du *chiffrement à masque jetable* tout en introduisant des petites clés (composées de 2^n bits avec $5 \leq n \leq 10$) réutilisables. Bien que la sécurité sémantique de ces constructions ne soit pas prouvée (aucune méthode n'est connue pour l'obtenir dans le cas symétrique), la sécurité calculatoire qu'elles mettent en jeu font d'elles les principaux outils utilisés dans les communications modernes. Dans la

partie suivante, nous introduisons les constructions rencontrées lors des travaux de cette thèse : les chiffrements par bloc et les chiffrements par flot.

I Chiffrement symétrique

Pour se rapprocher de la confidentialité apportée par le chiffrement à masque jetable, les cryptographes utilisent principalement deux familles de constructions imitant différents aspects de celui-ci. Nous réintroduisons dans cette partie ces deux familles en commençant par les *chiffrements par bloc* en sous-partie I.i et en continuant avec les *chiffrements par flot* en sous-partie I.ii. Pour compléter cette introduction sur les schémas de chiffrements symétriques, nous terminons cette partie avec un bref paragraphe (sous-partie I.iii) sur les *modes d'opération*.

I.i Chiffrement par bloc

Dans une construction de chiffrement par bloc, le clair n'est plus de taille arbitraire mais précise notée n : on parle alors de blocs qui contiennent le clair. Il en va de même pour la clé qui a une taille k et pour le chiffré de taille m. En général, les chiffrés et les clairs sont de même taille ($m = n$) et souvent la clé est aussi de même taille que le clair. Dans ce cas ($m = n$), la fonction de chiffrement est donc une famille de permutations sur les blocs paramétrées par la clé comme formellement donné en Définition 0.2.

Définition 0.2 (Chiffrement par bloc)**.** *Un chiffrement par bloc est une fonction* $E : \mathbb{F}_2^k \times \mathbb{F}_2^n \to \mathbb{F}_2^n$ *telle que pour tout* $\kappa \in \mathbb{F}_2^k$*, les fonctions* $E(\kappa, \cdot)$ *sont des permutations.*

Plus récemment, [LRW02] propose une formalisation d'une variante des chiffrements par bloc appelée *chiffrement par bloc paramétrable* qui prennent un paramètre public supplémentaire appelé naturellement un *paramètre*. Ce *paramètre* est utilisé, dans certains cas, pour obtenir l'*authenticité* plus facilement dans certains modes d'opération ou pour réduire les coûts de chiffrements lorsque le paramètre peut être changé plus facilement que la clé comme dans le chiffrement d'un disque dur. Nous donnons une définition formelle de cette construction que nous avons rencontrée dans nos travaux avec la Définition 0.3.

Définition 0.3 (Chiffrement par bloc paramétrable)**.** *Un chiffrement par bloc paramétrable est une famille de fonctions* $E : \mathbb{F}_2^t \times \mathbb{F}_2^k \times \mathbb{F}_2^n \to \mathbb{F}_2^n$ *telle que pour tout* $\tau \in \mathbb{F}_2^t$*, les fonctions* $E(\tau, \cdot, \cdot)$ *sont des chiffrements par bloc.*

La construction de chiffrements par bloc comme fonction prenant une entrée et renvoyant une sortie est rarement donnée dans la littérature en faveur des constructions itératives (comme illustré dans la Figure 2) qui transforment en plusieurs *tours* le clair s_0 en chiffré s_r. Dans une construction itérative, la clé est utilisée en premier lieu pour générer une séquence $k_0, \ldots, k_r$ de clés de tour qui seront mélangées une à une (par tour) à l'état i.e. le clair partiellement transformé à ce tour. L'autre composant d'une construction itérative est la *fonction de tour* qui va comme son nom l'indique, transformer le bloc à chaque tour. La première itération calcule $s_1 = f(s_0 \oplus k_0)$, la deuxième calcule $s_2 = f(s_1 \oplus k_1)$ et ainsi de suite pour les r tours. Dans la littérature, le tampon qui va contenir au tour i la valeur s_i est appelé *état interne* de la construction. Deux objectifs nécessaires à la sécurité sont réalisés par la fonction de tour à savoir la *diffusion*, le mélange en position dans l'état interne des bits de clair et de chiffrés et la *confusion*, le mélange en complexité (du point de vue du calcul de l'inverse) des bits de clé et de clair. Plusieurs constructions concrètes adoptent cette forme itérative et les deux plus courantes sont présentes dans les travaux de cette thèse, à savoir les *réseaux de permutation-substitution* et les *constructions de Feistel*.

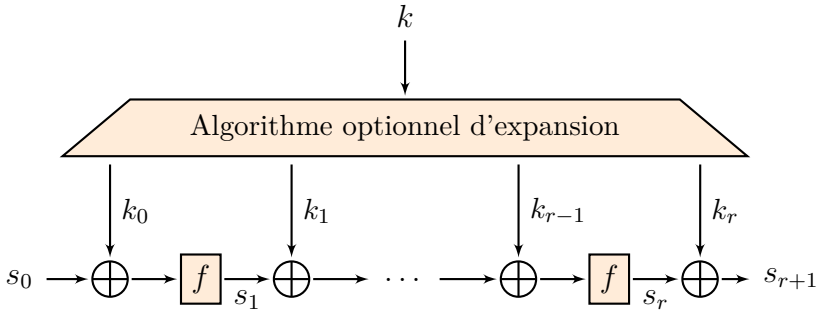

FIGURE 2 – Construction itérative d'un chiffrement par bloc de [Jea16]

Réseaux de Permutation-Substitution. Les réseaux de Permutation-Substitution sont peut-être les constructions itératives pour les chiffrements par bloc les plus connues par le standard `AES` proposé par Daeman et Rijmen [DR99] et consistent à découper la fonction de tour en deux parties : la couche de substitution apportant la confusion et la couche de permutation apportant la diffusion. Il n'y a pas de méthode connue pour créer une bonne couche de substitution pour un état de 64 ou 128 bits. Pour combler ce manque, la stratégie adoptée dans la littérature est de morceler l'état en plus petites cellules de $4, 8$ bits généralement et d'appliquer sur chaque cellule une bonne substitution appelée une

boite-S. La couche de permutation mélange l'information entre les cellules de l'état. Deux paradigmes sont possibles pour cette couche : le premier consiste à permuter et appliquer une opération linéaire aux cellules et est utilisé dans `AES` ou `SKINNY` [Bei+16]. Le second paradigme possible est de permuter directement les bits de l'état comme dans `PRESENT` [Bog+07] ou `Gift` [Ban+17]. Un exemple de chiffrement (jouet) orienté au niveau des bits est donné en Figure 3.

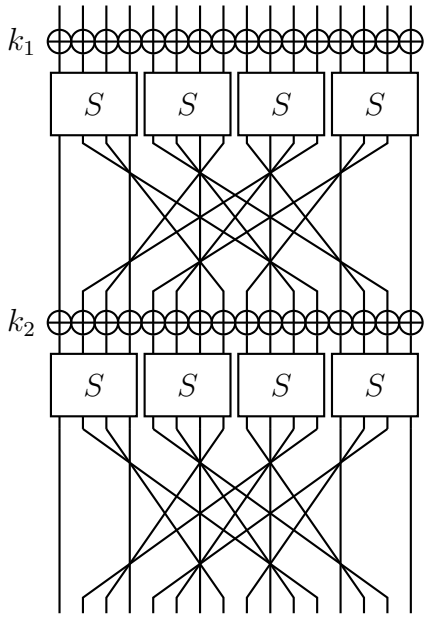

FIGURE 3 – Chiffrement jouet avec une permutation orientée au niveau des bits de [Jea16]

Constructions de Feistel. Les constructions de Feistel sont l'autre construction générique utilisée pour les chiffrements par bloc itératifs que nous avons étudiés lors de cette thèse. Dans la plus simple construction de Feistel (Figure 4), le message est séparé en deux moitiés (égales) appelées parfois *branches*. La branche de droite passe par une fonction F paramétrée par la clé et est ajoutée à la branche de gauche puis les deux branches sont échangées. La fonction F est souvent une couche de boites-S suivie d'un ajout de la clé (de tour). Plusieurs généralisations de cette construction ont été proposées dans la littérature et en particulier l'une d'elle sera plus précisément étudiée pour l'un des travaux de cette thèse dans la partie 2.

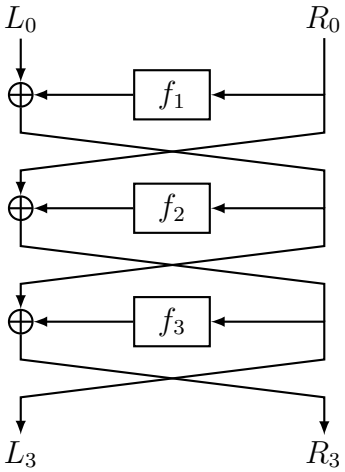

FIGURE 4 – Trois tours d'un réseau de Feistel de [Jea16] où les fonctions f_1, f_2, f_3 sont obtenues à partir de la fonction F avec les clés de tours une, deux et trois.

I.ii Chiffrement par flot

L'autre grande famille de chiffrements symétriques est celle des chiffrements par flot. À la différence des chiffrements par bloc, il n'y a pas un bloc de taille fixe mais un flot de bits à chiffrer. L'idée pour ce faire est de générer un flot de clé de même taille que le flot de message et d'ajouter (XOR) les deux (voir Figure 5). Dit autrement, un chiffrement par flot consiste à créer l'équivalent d'une clé de bonne taille pour un chiffrement à masque jetable à partir d'une clé de taille fixe. La confidentialité pour un chiffrement par flot passe donc par la difficulté à retrouver la valeur de l'état interne ou la clé originale depuis le flot de clé. En pratique, le flot de clé est généré bit à bit à mesure que le flot de message est reçu. On parle de *top d'horloge* du chiffrement à flot pour chaque bit de flot de clé généré.

Registre à décalage à rétroaction linéaire (LFSR). Si les boites-S étaient l'un des éléments les plus élémentaires des chiffrements par bloc, son remplacement dans le contexte des chiffrements par flot est le registre à décalage à rétroaction linéaire (LFSR). Un LFSR peut être vu comme un registre organisé en cases contenant des bits (souvent un bit). À chaque top d'horloge, le registre est décalé d'une position vers la droite et la première case prend comme valeur le résultat d'une fonction linéaire, la *fonction de rétroaction* appliquée à l'état (avant le décalage). Un schéma général de LFSR est donné dans la Figure 6.

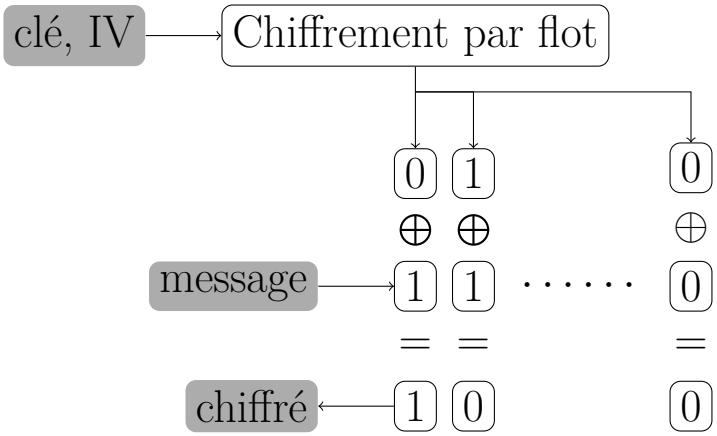

FIGURE 5 – Schéma d'un chiffrement par flot générique

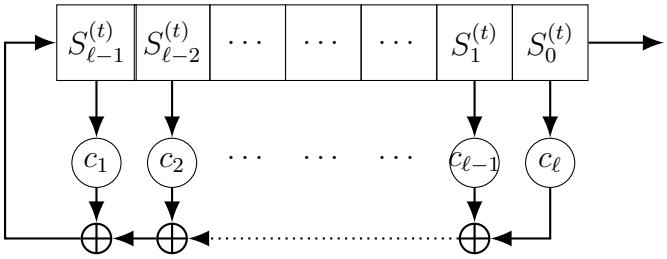

FIGURE 6 – Description générale d'un LFSR de [Jea16]

Délinéariser. Les LFSRs sont très pratiques dans la création de chiffrement à flot avec le top d'horloge du LFSR s'étendant naturellement au top d'horloge du chiffrement. Cependant, comme la fonction de rétroaction est linéaire, si elle est la seule composante utilisée, le système est facilement inversible, ce qui n'est pas idéal pour la sécurité. Deux stratégies différentes existent dans la littérature pour ajouter une opération non-linéaire aux LFSRs : transformer une fonction de rétroaction linéaire en fonction de rétroaction non-linéaire comme dans `Grain` [HJM07] ou `Trivium` [DP08]. L'autre solution est de désynchroniser le top d'horloge de plusieurs LFSRs avec une fonction non-linéaire dépendant des états de chaque LFSR comme dans `A5/1` [BGW99] ou `MICKEY` [BD08].

I.iii Mode d'opération

Une primitive peut être placée dans une plus grande construction appelée *mode d'opération* pour obtenir de nouvelles propriétés. Le premier usage des modes d'opération est d'étendre l'utilisation des chiffrements par bloc à des messages de taille supérieure à celle du bloc. Le mode d'opération le plus naïf pour les chiffrements par bloc est le mode *Elec-*

tronic CodeBook (ECB) où le message est simplement découpé en blocs, chacun chiffré individuellement avec la même clé. Malheureusement, bien que très simple de conception, ce mode n'offre pas de sécurité contre la *manipulation* de chiffré pour ne nommer que cette attaque : échanger la position de deux blocs crée un chiffré valide sans connaissance de la clé.

Dans la plupart des utilisations actuelles, la seule *confidentialité* est souvent insuffisante et l'*authenticité* est ajoutée avec le mode d'opération contrairement à l'*intégrité* qui vient d'autres procédés. Les modes d'opération qui, en plus de la confidentialité, apportent l'authenticité sont appelés *chiffrement authentifié (AE)* ou *chiffrement authentifié avec données associées (AEAD)*, si des données publiques sont associées au message. C'est ce type de chiffrement qui est demandé par la compétition de cryptographie symétrique légère du NIST.

En général, lorsqu'elle existe, une preuve de sécurité ou portant sur une autre propriété est donnée sur le mode d'opération. Dans ces preuves, la primitive considérée est une primitive idéale. Dans la réalité, les primitives pratiques utilisées introduisent des biais dans la preuve qui affaiblissent la sécurité de la construction. Les travaux de cette thèse portent en partie sur l'étude de certains de ces biais trouvés à partir de méthodes générales. Pour mieux les comprendre, dans la partie suivante, nous introduisons les notions de sécurité nécessaires à la présentation de nos travaux.

II Analyse et sécurité des chiffrements symétriques

II.i Modèle de sécurité

Pour comprendre la notion de sécurité dans les chiffrements symétriques, il est d'abord nécessaire de comprendre ce que l'attaquant recherche et ce qu'il peut faire. Trivialement, si un attaquant peut directement demander la valeur de la clé et l'obtenir, il pourra lire les messages à partir des chiffrés mais la primitive n'est pas, dans ce cas, considérée comme cassée. En pratique, le pouvoir donné à l'attaquant est modélisé par sa capacité à demander le chiffrement ou déchiffrement à un *oracle*. Un *oracle* peut être vu comme une boîte noire qui contient une clé adaptée au chiffrement qu'il représente et répond aux demandes de chiffrements et de déchiffrements envoyées par l'attaquant. Son but, la plupart du temps, est de retrouver la clé cachée dans l'oracle en utilisant seulement les demandes permises par celui-ci.

Dans la littérature, il y a plusieurs types d'oracles en fonction des demandes considérées comme permises. Le plus faible des oracles permet uniquement la requête de chiffrés décidés sans connaitre les clairs associés. Dans ce cas, le modèle d'attaque appelé *chiffrés connus* est le niveau le plus bas de sécurité demandé par les chiffrements à tel point qu'il est souvent négligé en faveur de preuves ou de raisonnements avec des modèles plus forts. Dans ces derniers, le pouvoir de l'attaquant est augmenté en demandant des paires clairs/chiffrés décidées, puis en demandant pour le chiffrement de clairs choisis (par l'attaquant) et finalement le chiffrement et déchiffrement de clairs et chiffrés arbitraires. Ces modèles sont appelés respectivement *clairs connus* (KP), *clairs choisis* (CPA) et *chiffrés choisis* (CCA).

Dans la littérature, l'autre facteur nécessaire à la description des attaques est la définition et la quantification des buts et succès de l'attaquant. Dans le formalisme actuel, l'attaquant va jouer à un jeu où il gagne lorsqu'il récupère une information précise qu'il devrait obtenir avec une probabilité faible. Une attaque peut être interprétée comme une stratégie pour l'attaquant telle que la probabilité de gagner est meilleure (par un facteur polynomial en la taille du secret) que la probabilité de choisir au hasard sa réponse. Un des jeux utilisés dans la littérature consiste à initialiser l'oracle uniformément et aléatoirement entre deux protocoles d'opération : le premier correspondant au fait de choisir correctement une clé et de l'utiliser normalement dans le schéma de chiffrement ; le deuxième au fait de renvoyer uniformément et aléatoirement une réponse de forme correcte. Le but dans ce jeu pour l'attaquant est de déterminer le protocole d'opération utilisé par l'oracle. Lorsqu'aucun attaquant ne peut gagner à ce ce jeu avec un avantage conséquent, le chiffrement est qualifié d'*indistinguable* (IND) et il est utilisé avec les précédents modèles d'attaquants pour définir complètement la sécurité d'un schéma en parlant par exemple de schéma *IND-CCA*.

Dans la cryptographie symétrique, de telles preuves de sécurité sont rarement faites pour les primitives et par extension, la sécurité des primitives est conceptualisée par un point de vue très pratique : elles sont sûres si aucune attaque pratique n'existe contre elles. La sécurité peut, dans ce cas, être comprise comme l'effort continu pour casser des primitives avec des techniques classiques ou de nouvelles attaques jusqu'à leur défaillance et remplacement par de nouvelles primitives plus sûres. Lors de leur travail, les cryptographes en symétrique ne cherchent pas toujours des attaques contre des primitives mais des composants de certaines attaques appelés *distingueurs* qui peuvent être compris comme des tests qui permettent de casser efficacement l'*indistinguabilité* des primitives.

II.ii Distingueur

Un distingueur est un algorithme qui casse (avec une bonne probabilité) le caractère d'*indistinguabilité* de chiffrements avec l'accès seul à des paires clairs/chiffrés en décidant si les paires sont générées honnêtement ou aléatoirement. Les distingueurs sont cherchés essentiellement en observant dans les chiffrements des comportements qui ne devraient pas exister dans des fonctions aléatoires. Un exemple de propriétés observées pour trouver des distingueurs est la propagation de différences dans le chiffrement dont nous parlons plus en détail dans la sous-partie II.iii. Pour les constructions itératives, un distingueur sur $r-1$ tours peut être étendu facilement en une attaque sur r tours, ce qui explique pourquoi une fois que les distingueurs sont obtenus lors d'une étude, les attaques correspondantes sont rarement données dans la littérature. En effet, en assumant que la clé de tour est plus petite que la clé, un attaquant peut demander un ensemble de chiffrés (sur r tours) et deviner la dernière sous-clé. La clé de tour devinée est utilisée pour déchiffrer un tour et le distingueur peut être testé sur l'ensemble des chiffrés partiellement déchiffrés avec l'idée que si la clé de tour est incorrecte, le distingueur devrait considérer l'ensemble testé comme généré par de l'aléatoire et non par une clé. Ce processus peut être répété avec un tour de moins jusqu'à obtenir assez de sous-clés pour reconstruire la clé maître. En conséquence, une bonne justification pour la sécurité d'une primitive est de montrer qu'aucune méthode classique comme des analyses différentielles ou linéaires ne permettent d'obtenir de bons distingueurs. Dans la sous-partie suivante, nous introduisons la cryptanalyse différentielle, la théorie principale que nous avons utilisée pour nos travaux dans cette thèse.

II.iii Cryptanalyse différentielle

À CRYPTO'90, Biham et Shamir [BS91a] présentaient leur analyse du Data Encryption standard, le standard de chiffrement américain avec une nouvelle méthode, la *cryptanalyse différentielle* où une différence est observée dans son passage par le système. À la même période, des attaques différentielles sont montées contre d'autres chiffrements par bloc comme `FEAL` [BS91b]. En 1998, Vaudenay *et al.* proposent un nouveau cadre appelé théorie de la décorrélation qui permet a priori (entre autres) de prévenir les attaques différentielles (directes), elle est appliquée pour la première fois pour construire les chiffrements `COCONUT` et `PEANUT` [Vau98]. La même année Wagner, [Wag99] prouve que la théorie de la décorrélation n'est pas suffisante pour esquiver toutes les attaques différentielles en exhibant une nouvelle attaque, les *boomerangs*. Depuis, de nouvelles attaques

et variantes n'ont cessé d'apparaître dans la littérature comme les attaques *sandwich* [DKS10], *rectangle* [BDK01] ou les attaques *différentielles impossibles* [Knu98]. La cryptanalyse différentielle est maintenant une méthode classique pour l'étude de nouvelles primitives qui, pour être sérieusement considérées, doivent avoir prouvé leur résistance contre différentes attaques différentielles.

La cryptanalyse différentielle étudie la transformation de différences au cours du chiffrement. Pour une paire de messages p_0 et p_1, la différence Δp entre les deux messages est définie par $\Delta p = p_0 \oplus p_1$. Trivialement, une différence est facilement propagée à travers une opération linéaire L puisque $L(\Delta p) = L(p_0) \oplus L(p_1)$. Cependant, on ne peut pas dire la même chose du passage par des opérations non-linéaires où cette égalité devient en général fausse. Pour décrire les diverses différences en sortie possibles pour une différence en entrée après une opération non-linéaire, la communauté parle de probabilité d'observer une différence de sortie précise calculée avec le dénombrement du nombre de paires p_0, $p_0 \oplus \Delta p$ qui atteignent cette différence (divisé par le nombre total de paires). Un modèle d'attaquant plus fort peut en plus des différences dans le message ou état interne introduire des différences dans la clé pour les constructions itératives, on parle alors de modèle d'attaque à clés reliées. Si dans tous ces cas, le chiffrement se comporte comme une fonction aléatoire au regard des différences, alors la probabilité d'observer une valeur particulière de différence en sortie devrait être de 2^{-n} où n est la taille du message. Sinon, nous venons de décrire un distingueur particulier contre le chiffrement qui peut être utilisé pour monter une attaque.

Faire une analyse différentielle complète d'une primitive est difficile et plusieurs sous-problèmes ont été proposés pour la simplifier dans la littérature. Pour les chiffrements itératifs, la différence peut être suivie tour par tour pour obtenir une *caractéristique différentielle* qui peut se décrire par la séquence de différences $(\Delta_0, \ldots, \Delta_r)$ observées au début de chaque tour plus la différence finale en sortie. La probabilité d'une caractéristique est compliquée à calculer directement mais peut être approximée en considérant que toutes les transitions d'un tour à l'autre sont indépendantes les unes des autres. Dans ce cas, la probabilité de la caractéristique est $\prod_{i=1}^{r} \mathbb{P}(\Delta_{i-1}, \Delta_i)$ où $\mathbb{P}(\Delta_{i-1}, \Delta_i)$ est la probabilité que la différence Δ_{i-1} devienne Δ_i après un tour. Le calcul de la probabilité d'une caractéristique peut aussi être simplifié en utilisant la structure de la fonction de tour, comme nous allons le montrer pour les SPNs. En effet, comme dit précédemment, la couche linéaire et l'addition de clé sont des couches où les différences vont se propager de manière déterministe (i.e. avec une probabilité 1). Il ne reste donc que la couche de substitution

à prendre en compte pour le calcul de la probabilité. Or, la couche de substitution est composée d'applications parallèles de boites-S qui sont donc les objets à considérer pour l'étude de différences à travers le tour. Puisque les boites-S sont bien plus petites que le bloc, la probabilité exacte d'observer une paire de différences d'entrée sortie $(\Delta_{\text{in}}, \Delta_{\text{out}})$ peut être calculée et l'information est souvent condensée dans une table appelée *table de distribution des différences* (DDT). La DDT est une table à double entrée qui pour une entrée correspondant à la paire de différences $(\Delta_{\text{in}}, \Delta_{\text{out}})$ donne la probabilité d'observer $\Delta_{\text{out}} = S(x) \oplus S(x \oplus \Delta_{\text{in}})$ pour les différentes valeurs de x et où S correspond à la boîte-S.

Pour simplifier une nouvelle fois la recherche de caractéristiques différentielles dans les SPNs, Knudsen [KRW99] a proposé d'ignorer la valeur exacte des différences pour ne garder que la présence (active) ou l'absence (inactive) de différence pour une boîte-S à un tour donné. Le terme adopté par la communauté pour décrire cet objet est *caractéristique différentielle tronquée*. Elles se sont révélées très utiles pour le construction de primitives en particulier pour donner des preuves simples pour la résistance contre les attaques différentielles. Pour être plus précis, pour une caractéristique différentielle tronquée donnée avec n boites-S actives, et pour la propagation à travers la boite-S avec la meilleure probabilité p, une borne supérieure facile à calculer pour la probabilité d'une caractéristique ayant la forme tronquée (pour un nombre donné de tours) est donc p^n. Le même raisonnement peut être appliqué pour les caractéristiques ayant le minimum de boites-S actives pour obtenir une borne supérieure sur les caractéristiques différentielles pour un nombre de tours donné. Si cette borne supérieure est plus petite que 2^{-m} où m est la taille de bloc, alors le chiffrement semble au premier abord être résistant à la cryptanalyse différentielle. Dans cette dernière décennie (depuis 2009 pour être exact), de nouvelles méthodes fondées sur les solveurs ont été présentées pour la recherche de caractéristiques différentielles tronquées [SNC09 ; MP13 ; Mou+12 ; Sun+14 ; KLT15]. Les solveurs sont des algorithmes génériques qui prennent en entrée la description d'un problème et cherche automatiquement une solution à celui-ci. Dans cette thèse, nous avons aussi utilisé des solveurs pour calculer de nouvelles caractéristiques différentielles. Des détails supplémentaires sur nos travaux concernant ces aspects sont présentés au Chapitre 3.

III Résumé des chapitres

Pendant cette thèse, nos travaux ont essentiellement porté sur de la cryptanalyse différentielle grâce à des algorithmes particuliers à travers différentes perspectives allant d'une étude détaillée d'une attaque récente avec des expériences algorithmiques à l'analyse d'un critère particulier des réseaux de Feistel avec un algorithme constructif ainsi que l'étude d'une primitive particulière `SKINNY` avec des solveurs. Dans cette partie, nous donnons un résumé de chacun de ces travaux présentés un à un dans les différents chapitres.

Dans le Chapitre 1, nous revisitons les attaques rapides par collision proche ("Fast Near Collision Attack" (FNCA)), un nouveau type d'attaque proposé par Zhang *et al.* [ZXM18]. Les FNCAs ont été appliquées à deux chiffrements : `Grain v1` (dans le papier original) et `A5/1` [Zha19] pour obtenir de nouvelles attaques plus rapides que les meilleures connues jusqu'alors. Cependant, nous avons trouvé des incohérences dans ces attaques et avons en conséquence publié des versions corrigées prouvant que les FNCAs ne sont pas des attaques efficaces pour les exemples donnés.

Dans le Chapitre 2, nous résolvons un problème ouvert depuis 10 ans et nous présentons les nouveaux résultats sur les réseaux de Feistel généralisés que nous avons obtenus. Plus précisément, nous avons étudié en particulier une propriété des permutations des réseaux de Feistel généralisés appelée *tour de diffusion*.

Pour finir, dans le Chapitre 3, nous présentons nos travaux sur les caractéristiques différentielles du chiffrement par bloc `SKINNY`. Nous utilisons une approche très classique de séparation de la recherche de caractéristiques différentielles. Grâce à des outils adaptés, nous avons trouvé des et parfois les meilleures caractéristiques différentielles de `SKINNY`.

III.i Revisiter les attaques rapides par collision proche

En 2018, Zhang *et al.* dans [ZXM18] proposent une nouvelle attaque générique, les attaques rapides par collision proche, et la première application de celle-ci sur le chiffrement à flot `Grain v1`. Puis, dans [Zha19] Zhang *et al.* l'appliquent une nouvelle fois avec succès contre `A5/1` pour obtenir une meilleure attaque de récupération de clé. Les FNCAs sont des attaques "diviser pour régner" pour retrouver l'état interne du chiffrement à flot depuis le flot de clé. Elles séparent l'état interne en deux parties : la partie cruciale (CP) et la partie restante (RP). Une fois la CP connue, il est facile de retrouver la RP. La partie cruciale est retrouvée grâce à la partie collision proche de l'attaque à partir d'un petit échantillon de flot de clé.

Lors de cette thèse, nous avons essayé d'implémenter l'attaque contre `A5/1` et après une étude du processus pour confirmation, nous avons trouvé plusieurs incohérences du point de vue de la théorie de l'information dans la description de l'attaque et des erreurs dans le code donné par Zhang *et al.* pour tester des composants de leur attaque. Nous nous sommes donc attelés à la réévaluation des complexités en corrigeant les théorèmes utilisés pour leurs calculs et en vérifiant expérimentalement qu'aucun biais existant ne pouvait expliquer les complexités annoncées par Zhang *et al.*. Nous donnons aussi les complexités corrigées des FNCAs prouvant donc que Zhang *et al.* ont sérieusement sous-estimé le coût réel de leur attaque. Ces travaux ont été publiés à FSE :
Patrick Derbez, Pierre-Alain Fouque et Victor Mollimard, "Fake Near Collisions Attacks", in : *IACR Transactions on Symmetric Cryptology* 2020.*4* (déc. 2020), p. 88-103, doi : `10.46586/tosc.v2020.i4.88-103`, url : `https://tosc.iacr.org/index.php/ToSC/article/view/8749`

III.ii Recherche efficace de couche de diffusion pour les réseaux de Feistel généralisés

Les Feistels sont une des constructions les plus classiques pour les chiffrements par bloc. À CRYPTO'89, Zheng *et al.* [ZMI89b] proposèrent différentes généralisations de la construction de base des Feistels. L'une d'entre elles consiste à appliquer en parallèle plusieurs constructions de Feistel de base puis à glisser toutes les branches vers la droite. Comme remarqué dans [Nyb96 ; SM10], la translation vers la droite n'est pas la seule permutation qui peut être employée ici. La construction où la permutation utilisée est arbitraire est maintenant appelée *réseau de Feistel généralisé* (GFN).

Cependant, lors de l'introduction des GFNs en 1996, aucune famille de permutations à préférer n'est donnée pour des constructions a priori plus sûres. En 2010 à FSE, Suzumaki et Minematsu [SM10] proposent un nouveau critère, le *tour de diffusion*, qui peut être utilisé pour informer dans le choix d'une permutation à privilégier. Le tour de diffusion peut être compris comme une mesure de la vitesse nécessaire en nombre de tours pour qu'une différence sur n'importe quelle branche se diffuse sur toutes les branches du réseau. Avec leur travail, Suzumaki et Minematsu sont capables de manière exhaustive d'évaluer toutes les permutations jusqu'à 16 branches pour trouver toutes les permutations optimales (selon ce critère). De plus, après avoir remarqué que pour toutes les tailles de réseau qu'ils testent, il y a une permutation optimale qui est *paire-impaire*, c'est-à-dire qui transforme tous les pairs en impairs et inversement. Ils proposent à partir de cette

observation une méthode de construction générale pour trouver de bonnes permutations paires-impaires lorsque le nombre de branches est une puissance de deux. En particulier, ils ont trouvé une bonne permutation paire-impaire avec 32 branches et un tour de diffusion de 10. En 2019, Cauchois *et al.* [CGT19] publient une nouvelle méthode pour la construction de permutations optimales paire-impaires leur permettant d'en trouver jusqu'à 26 branches.

Pendant cette thèse, nous avons étudié le problème de la recherche de permutations optimales pour le tour de diffusion pour les GFNs. Nous avons trouvé une nouvelle caractérisation du tour de diffusion qui permet de développer un algorithme efficace pour la recherche de permutations paires-impaires optimales entre 28 et 42 branches en testant si le tour de diffusion est à 9. En l'utilisant, nous avons exhibé toutes les permutations paires-impaires optimales pour 28, 30, 32, 34 et 36 branches ainsi que des bons candidats pour l'optimalité pour 38, 40 et 42 branches. Ces travaux ont été publiés à FSE :
Patrick Derbez *et al.*, "Efficient Search for Optimal Diffusion Layers of Generalized Feistel Networks", in : *IACR Transactions on Symmetric Cryptology* 2019.*2* (juin 2019), p. 218-240, DOI : 10.13154/tosc.v2019.i2.218-240, URL : https://tosc.iacr.org/index.php/ToSC/article/view/8321

III.iii Méthodes efficaces pour la recherche des meilleures caractéristiques différentielles de SKINNY

SKINNY est une famille de SPNs paramétrables créée en 2016 par Beierle *et al.* [Bei+16]. Six versions de SKINNY existent groupées en deux familles dépendentes de la taille de l'état interne : SKINNY-64 et SKINNY-128 utilisant respectivement des états internes de 64 et 128 bits. Pour donner une première idée de la sécurité contre la cryptanalyse différentielle, ses créateurs ont inclus dans le papier des spécifications [Bei+16] un modèle pour chercher les caractéristiques différentielles tronquées ayant le minimum de boites-S actives (avec différents modèles d'attaques où des différences peuvent être introduites dans les clés paramétrées).Le modèle qu'ils proposent est résolu par un solveur et plus précisément un solveur de programme linéaire ("mixed integer linear programming" MILP). En MILP, les variables sont des entiers et les contraintes à considérer sont des inéquations linéaires en ces variables.

Depuis la publication des spécifications, peu de résultats ont été publiés concernant les caractéristiques différentielles de SKINNY. Dans les limites de nos connaissances, seulement deux autres articles ont des résultats sur ce sujet : Liu *et al.* [LGS17] avaient besoin de

quelques caractéristiques pour étudier la résistance de SKINNY contre les attaques boomerangs et rectangles dans le cas des clés paramétrées reliées. En utilisant un modèle MILP pour SKINNY-64 et un outil ad hoc pour SKINNY-128, ils ont trouvé les meilleures caractéristiques différentielles jusqu'à 13 tours pour SKINNY-64 et de meilleures caractéristiques pour SKINNY-128 pour un petit nombre de tours (mais sans l'optimalité).

L'autre publication concernant les caractéristiques de SKINNY que nous avons trouvée dans la littérature a été publiée par Abdelkhalek *et al.* dans [Abd+17]. Ils construisent les meilleures caractéristiques différentielles de SKINNY-128 dans le modèle de clé simple jusqu'à 13 tours avec un modèle MILP contenant une description efficace jugée inexistante pour les boites-S de 8 bits. Ils montrent en plus qu'aucune caractéristique différentielle avec une meilleure probabilité que 2^{-128} n'existe pour 14 tours (et plus) dans le modèle de la clé simple.

Dans nos travaux, nous complétons les bornes sur les caractéristiques différentielles de SKINNY en utilisant une approche très classique qui sépare la recherche en deux étapes. La première étape consiste à chercher des caractéristiques différentielles tronquées ayant un nombre de boites-S actives égal ou inférieur à une cible donnée. La seconde étape prend les résultats de la première et essaye d'instancier en valeur les caractéristiques tronquées. Notre contribution dans ces travaux, outre la publication de meilleures bornes, est l'utilisation adaptée d'outils pour chacune des étapes. La première étape est réalisée avec un outil ad hoc fondé sur de la programmation dynamique, la seconde utilise un modèle de programmation par contrainte, un autre type de solveur que le MILP qui permet une description bien plus directe dans ce cas que ce dernier où nous avons raffiné les stratégies de résolution. Avec ces choix de méthode, nous retrouvons et terminons l'analyse sur les caractéristiques différentielles de SKINNY-64 en quelques secondes. Concernant SKINNY-128, nous retrouvons les résultats de [Abd+17] en quelques minutes, au lieu de 16 jours à l'origine. Nous trouvons aussi les meilleures caractéristiques différentielles de SKINNY-128 dans les différents modèles de clés reliées pour certains tours ainsi que des meilleures caractéristiques différentielles (sans optimalité) pour un plus grand nombre de tours et des preuves qu'aucune caractéristique différentielle n'existe (avec une bonne probabilité) pour certains nombres de tours. Les résultats détaillés sont dans la Sous-partie 3.5.3 du Chapitre 3. Ces travaux ont été publié à ACNS :
Stéphanie Delaune *et al.*, "Efficient Methods to Search for Best Differential Characteristics on SKINNY", in : *International Conference on Applied Cryptography and Network Security*, Springer, 2021, p. 184-207

Table of Contents

INTRODUCTION

Cryptography from the Greek *kryptos*, "secret, hidden" and *graphein*, "to write" is an old field of study concerning way to hide secret messages to a third eavesdropping party. Its first known use (in a sense) can be dated to 1900 BC with an inscription in the tomb of Khnumhotep II where non-standard hieroglyphs were used. Approximately 1850 years later, Julius Caesar used a cipher for his military affairs named now after him. The `Caesar` cipher consists in transforming a *message* into a gibberish sequence of letters called a *ciphertext* by changing each letter of the message into the n-th (fix) next letter in the alphabet. The receiver can then de-transform the *ciphertext* into the original *message* using the inverse operation on the letter of the *ciphertext*. Over time, cipher gained in design complexity, with the use for example of multiple alphabets and in practical complexity, with the use of helping mechanisms like slide rules. During the 16th century, Giovan Battista Bellaso created the `Vigenère` cipher (misattributed to the diplomat Blaise de Vigenère). The `Vigenère` cipher can be understood as multiple uses of the `Caesar` cipher on the message. More precisely, a secret sequence of letters of size n that is now called a *key* known only by the sender and receiver is used in the scheme. To encrypt, the message is split in block of size n. For each block, the i-th letter is shifted to the right by r in the alphabet where r is the position in the alphabet of the i-th letter of the *key*. An example of encryption following this scheme is given in Figure 7. In practice, another permutation table can be used and only the key must be kept secret, the table can be published.

At the end of the 19th century, cryptography was also changed by the formalism that swept the different intellectual disciplines. In particular, in 1883 Kerckhoff in *La Cryptography Militaire* [Ker83] proposed a list of principles that a cipher should follow. Kerckhoff's principles are still being followed nowadays in the design of cryptographic primitives particularly regarding the security: the system should be practically indecipherable or said in an another manner the system should guarantee the *confidentiality* of the message and the only secret should be the *key*.

Since this formalization, two important revolutions happened in cryptography. The first being the computing revolution during the second World War with the construction

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Key	L	A	T	I	N	L	A	T	I	N	L	A	T	I	N	L	A	T	I	N	L	A
Plaintext	L	O	R	E	M	I	P	S	U	M	D	O	L	O	R	S	I	T	A	M	E	T
ciphertext	W	O	K	M	Z	T	P	L	C	Z	O	O	E	W	E	D	I	M	I	Z	P	T

Figure 7 – Example of encryption with the `Vigenère` cipher. The upper table gives the correspondence between the plaintext letter (column), the key letter (row) and the ciphertext (entry). The lower table gives the ciphertext when the key is "LATIN" and the plaintext is "LOREM IPSUM DOLOR SIT AMET".

of the first programmable electronic and digital computer, the *Colossus*. With modern computational power appearing, cryptography has completely adopted the binary framework: the *message, key* and *ciphertext* are now arbitrary length bitstrings in $\mathbb{F}_2^*$. The other breakthrough in cryptography was the invention of asymmetric cryptography with the `RSA` cipher by Rivest, Shamir and Adleman in [RSA78]. In asymmetric cryptography, the secret key is replaced by a pair of keys: the public key and the private key. Adopting modern cryptography on computer in civilian life begins little by little in the 60's with banking and communication networks. In the early 70's, the US government identified a need for a standard encryption algorithm for unclassified but still sensitive information. The `Data Encryption Standard` (`DES`) was then opened to public uses in 1977 and kept as a standard up to its replacement in 2002 by the famous `Advance Encryption Standard` (`AES`). During its uses, the security of `DES` was doubted from its conception in

relation to various US agencies (most famously the NSA) involved in the design process. As a natural evolution towards more confidence, the way new primitives are developed changed completely (like `AES` for example) and nowadays it is a sort of public collaboration between several actors. When a new design is wished for, standardization agencies call for public contributions from the cryptographer community. Candidates are then studied by the community to inform a first sieving of the list which is studied again in the same way up to the point where one finalist or a portfolio of finalists are published with recommendations of uses. The organizers select during each round some candidate in light of these studies and justify their choices with public reports on the process of selection. Right now, two major competitions by the National Institute of Science and Technology (NIST) of the USA are taking places: the lightweight competition for new symmetric primitives adapted to IoT and the post-quantum competition to prepare the replacement of RSA necessary from the rapidly approaching quantum computer.

During this thesis, we focus essentially on analyzing the existing primitives with different algorithms and focusing on different aspects from properties that could be considered in their designs to studying criteria informing on potential attacks and to reviewing attacks against them. To introduce as clearly as possible the works of this thesis, we now give Definition 0.1, a general definition of the primitives we studied, the encryption ciphers by keeping the vocabulary already used here about ciphers. We will give more details regarding their constructions and security in the rest of this introduction.

Definition 0.1 (Encryption)**.** *An encryption function is a function* $E : \mathbb{F}_2^* \times \mathbb{F}_2^* \to \mathbb{F}_2^*$ *such that for any key in* $\mathbb{F}_2^*$*, the function* $E(key, \cdot)$ *is a surjection and its inverse function* $D(key, \cdot)$ *is efficiently computable when the key is known.*

Ideally in addition to ensuring *confidentiality*, a good symmetric cipher should give *integrity* meaning it should be possible to detect a tempering of the message and *authenticity* meaning there should be a process to verify that the source (of the ciphertext) is who it pretends to be. Trying to design directly a cipher offering this three properties is hard and asks for numerous competences, knowledge and various security proofs for each property but also for their interactions. To simplify the conception and verification of new designs, a layered strategy of conception is widely used in symmetric cryptography. The first layer consists in constructing, designing *primitives*, ciphers that ensure *confidentiality* for the messages. The missing two properties (or other properties for particular cases) are obtained by integrating a good primitive in a bigger construction although

modern cryptography is centered essentially on *confidentiality* and *authenticity*, *integrity* being granted from other processes. In this thesis, we will present our works concerning different primitives and for greater clarity, we will begin by reviewing classical ways to construct primitives in Section I, we give some elements to understand their resistance and security in Section II and introduce the works presented in each chapter in Section III.

Before introducing the different ways of constructing primitives, we review right now the `one time pad`, a well known scheme which can be assimilated to a `Vigenère` cipher where the key has the same size as the message that will serve as a basis to explain the other constructions. In the `one time pad`, for each message $m \in \mathbb{F}_2^*$, a key k of the same size as m is generated by uniformly taking a value in $\mathbb{F}_2^{|m|}$ where $|m|$ is the size of m and computing the corresponding ciphertext c with $c = m \oplus k$. In 1949, Shannon [Sha49] proved that the one time pad cipher has perfect confidentiality (to be precise this is called perfect secrecy which can be opposed to computational security or to semantic security, weaker security notions) that is to say that even given infinite time and computational power, it would be impossible to retrieve the message from the ciphertext without knowledge of the key. Unfortunately, the `one time pad` presents multiple defaults beginning with the fact that the key must be of the same size as the message and unique for each communication. As a result, cryptographers invented different constructions to approach the `one time pad` while keeping in a reasonable measure reusable small keys of 2^n bits with $5 \leq n \leq 10$. Although, the semantic security of those constructions is not proven, the computational security they exhibit made them the primary symmetric tools in use in modern communications. In the next Section, we review well known constructions that we encounter during the works of this thesis.

I Symmetric Cipher

To approach the one time pad in term of confidentiality, cryptographers now use principally two families of constructions miming different aspects of the one time pad. We review these two families of constructions beginning with the block ciphers in Subsection I.i and following in Subsection I.ii with stream ciphers. Finally, to complete this presentation, we briefly introduce in Subsection I.iii modes of operation corresponding to the general way to complete primitives that is used to obtain the *integrity* and *authenticity* or more peculiar properties in certain use cases.

I.i Block cipher

In block cipher construction, the message is not anymore of arbitrary length but limited to a block of size n. The same can be said about the key with a fix size k. The encryption is then a family of permutations of the block of size n parameterized by the key as can be seen in Definition 0.2.

Definition 0.2 (Block cipher)**.** *A block cipher is a function $E : \mathbb{F}_2^k \times \mathbb{F}_2^n \to \mathbb{F}_2^n$ such that for all κ, the function $E(\kappa, \cdot)$ is a permutation.*

More recently, in [LRW02] a slightly different construction was proposed: the *tweakable block ciphers* which contain an additional public parameter called a tweak (or tweaks if multiple are possible). The tweak can be used in different scenarios to add more easily some new properties like *integrity* or *authenticity*. We give a formal description of this construction in Definition 0.3.

Definition 0.3 (Tweakable block cipher)**.** *A tweakable block cipher is a family of functions $E : \mathbb{F}_2^t \times \mathbb{F}_2^k \times \mathbb{F}_2^n \to \mathbb{F}_2^n$ such that for all $\tau \in \mathbb{F}_2^t$, the functions $E(\tau, \cdot, \cdot)$ are block ciphers.*

Since constructing directly a block cipher is hard, modern constructions mostly adopt an iterative process, see Figure 8. In an iterated construction, the key is used to first generate a sequence $(k_0, \ldots, k_r)$ of *round keys* that will be added little by little to the message at each round. The other component of an iteration is the *round function* f that transforms the message at each round. The first iteration computes $s1 = f(s_0 \oplus k_0)$, the second iteration iteration corresponds to $s_2 = f(s_1 \oplus k_1)$ and the process is repeated for r rounds. In the literature, we talk about the internal state of a construction for a buffer s that will contain in order the values $s_0, s_1, \ldots, s_r$. Two objectives are realized by the round function namely the *diffusion*, shuffling every bits of message and key together and the *confusion*, making the inversion of the shuffle of key and message bits hard to inverse in a mathematical sense. Multiple generic constructions adopt this strategy of repeating rounds. During this thesis, we work with two of them: the Substitution-Permutation Networks and the Feistel constructions.

Substitution-Permutation Network (SPN). The Substitution-Permutation network is maybe the most well known iterative construction of block cipher from its use in the `AES` proposal by Daeman and Rijmen [DR99] and consists in applying a substitution layer followed by a permutation layer to the internal state. There is no well known method for

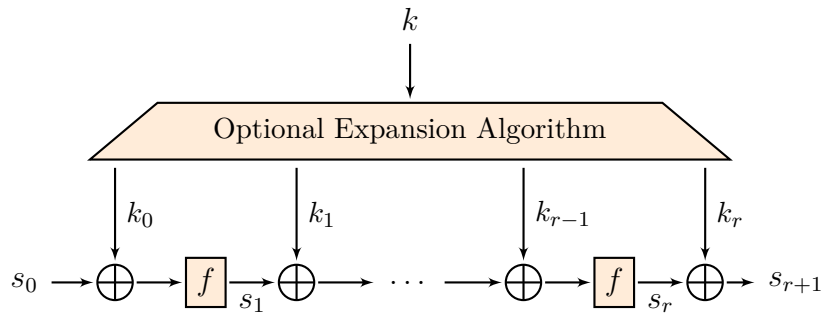

Figure 8 – Iterative construction of a block cipher from [Jea16]

finding a good substitution for a state of 64 or 128 bits. As a result, the strategy adopted is to split the state in smaller cells of $4, 8$ bits and applied in parallel to each of them a good substitution called a *S-box*. The permutation layer is the linear layer that will shuffle the bits of each cell in the state between different S-boxes for the next iteration. Two different shuffles are used in the literature: the first consists in a cell shuffle where the position of the cell will be permuted and a linear operation is applied to a set of cells to internally shuffle the bits presented in them. This is the solution adopted in `AES` and `SKINNY` [Bei+16]. The second shuffle possible for a SPN is a direct shuffle of the bits of the internal state which sends some bits going from a S-box to another S-box in the next round like the block ciphers `PRESENT` [Bog+07] or `Gift` [Ban+17]. An illustration of a bit-wise toy cipher is given in Figure 9.

Feistel constructions. The Feistel construction is the other generic construction of iterated block ciphers we encounter during this thesis. In the most basic Feistel construction, the message is split in two halves sometimes called branches. The right branch goes through a keyed function F and is xored to the left branch then the two branches are swapped. This keyed function is generally obtained by using a layer S-boxes followed by a xor with a round key. From this basic construction, multiple generalizations were proposed and one in particular is studied further in Chapter 2 for one of the works of this thesis.

I.ii Stream cipher

Apart from block ciphers, the other generic ciphers in symmetric cryptography are the stream ciphers. Whereas block ciphers imitate the one time pad from a permutation

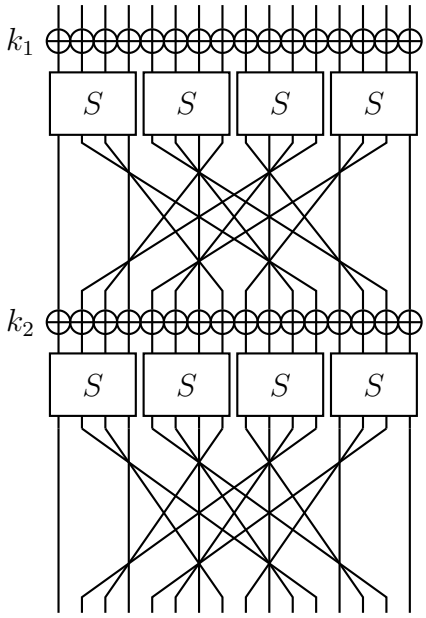

Figure 9 – Toy SPN with a bit oriented permutation layer. From [Jea16]

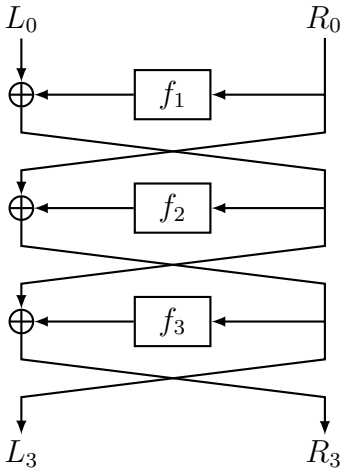

Figure 10 – 3 rounds of the basic Feistel construction from [Jea16] where the f_1, f_2, f_3 are obtained with the function F with respectively the first round key, the second and the third.

point of view, the stream cipher imitates it from its structure. More precisely, the key of a stream cipher is used to generate an arbitrary long sequence of bits called the *keystream* which will play the role of the secret key in an one time pad construction (see Figure 11). As a result, confidentiality for a stream cipher is ensured either by the difficulty from

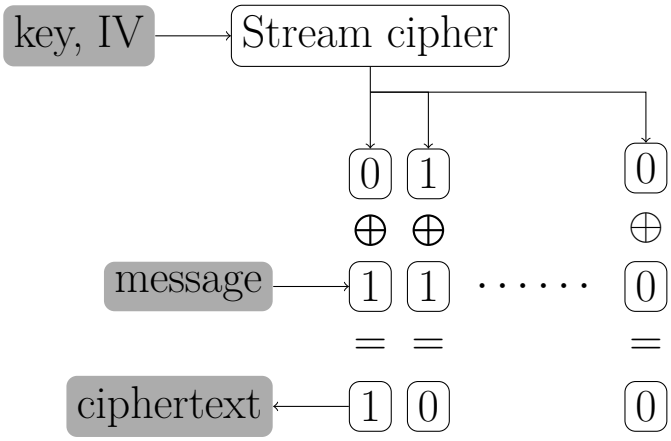

Figure 11 – Generic stream cipher

retrieving the key from the keystream sequence or by retrieving the internal state from the keystream.

In practice, the keystream is generated as needed as bit (or byte or other given quantity...) of the message need to be encrypted. When a new keystream bit is generated, the stream cipher is said to be clocked one time.

Linear Feedback Shift Register (LFSR). If the S-boxes and permutations were the basic building blocks for the SPN, the most basic object that composed stream ciphers are Linear Feedback Shift Registers. A LFSR can be seen as a rolling register containing a finite number of bits in each of its cell. At each clocking, a fix linear combination of some cells of the register called the *feedback function* is applied to the register to compute a new cell x. Then, the register is shifted discarding the last value of the register and freeing the first cell of the register which will take the value x. An example of a general LFSR is given in Figure 12.

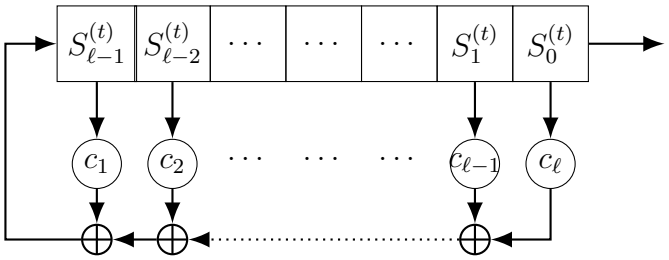

Figure 12 – General LFSR from [Jea16]

Non-linearization. LFSRs are quite practical to use for constructing stream cipher with the clocking of the LFSR extending well to the clocking of the stream cipher. However, since their operations are linear only the *diffusion* of the information exists if used alone and as a result, the system can easily be inverted. Two strategies were adopted to add *confusion* to the system and preventing an easy inversion: adding a non-linear function to a feedback function of a LFSR like Trivium [DP08] or Grain [HJM07]. The other solution is to change the clocking mechanism of multiple LFSRs to be a non-linear function of the state of the registers like in A5/1 [BGW99] or MICKEY [BD08].

I.iii Modes of operation

A mode of operation takes a primitive and uses it in a bigger construction to obtain a bigger cipher that can have more properties. The first use of mode of operation for block ciphers is to encrypt messages of greater size than the block. The most basic mode of operation for block ciphers is the *Electronic CodeBook (ECB)* consisting in splitting the message in blocks of the size considered by the block cipher. Each block is then encrypted with the same key through the block cipher and the resulting ciphertext are concatenated to obtain the whole ciphertext. Unfortunately, while being quite simple it does not offer security against multiple attacks like to name only this one *tempering*: switching the position of two ciphertext blocks creates a valid ciphertext.

In most uses nowadays, only *confidentiality* is not enough and *authenticity* is added with the mode of operation unlike *integrity* which is ensured with other methods that developed as an independent research topic from cryptography. The lightweight symmetric cryptographic competition of the NIST asked for AEAD primitives for this reason. Ciphers (with their mode of operation) which give *authenticity* in addition to *confidentiality* are called *authenticated encryption* (AE) or if a further public associated data is given, *authenticated encryption with associated data* (AEAD).

In general, a proof of the security or regarding other properties is given when a new mode of operation is published. Those proofs of security suppose that the primitives pasted in their modes have good properties regarding their securities. In the next section of this introduction, we revisit the necessary notions to understand the security of the primitives.

II Analysis and Security of symmetric ciphers

II.i Model of security

To understand the security of a symmetric cipher it is important to define the power of an attacker. Trivially, an attacker could ask for the key directly to one of the participants but obtaining it this way is not considered a break of the primitive.

In practice, the power given to the attacker is modeled as its capacity to query encryption or decryption requests to an *oracle*. An *oracle* can be understood as a black box that contains a random key for the targeted cipher and answers requests corresponding to the encryption or decryption of given or known messages or ciphertexts. The goal of the attacker is to retrieve the key using only the authorized queries of the oracle.

In the literature, there are multiple levels of oracle with varying types of queries that give varying security. The weakest oracle (with respect to the power of the attacker) consists in being able for the attacker to ask for ciphertexts selected by the oracle without knowing the corresponding plaintexts. This model is called the *known ciphertexts* oracle and is such a basic requirement for the security that it is most of the time skipped in favor of more powerful attacker models. The power of the attacker is increased with in order giving access to known plaintext ciphertext pairs, choosing arbitrary plaintexts to encrypt and finally choosing in addition arbitrary ciphertext to decrypt. Those models are called in order the *known plaintexts*, *chosen plaintexts* and *chosen ciphertexts* models.

In the literature, after defining the power of the attacker, the other necessary parameter needed to understand security are to define and quantify the goal and success of the attacker. The attacker will play a game where it can query the oracle (in function of the model) and try to win the game by finding the goal like the key. An attack can be understood as a strategy in this game for the attacker such that its probability of winning is greater by a polynomial term (in the security parameter) than a random selection of the answer.

One of the basic game of this type in the literature consists at the beginning of the game to have the oracle chooses secretly between two modes: a first corresponding to what was described previously and a second where the oracle answer to query with random strings (of the good form). The goal of the attacker is to find which of these cases is running for the oracle. When no attacker can win with a good advantage, the cipher is said to be *indistinguishable* and with the previous model for the power of the attacker, it is defined for example as *IND-CCA*.

In symmetric cryptography, such proofs of security are rarely done for primitives and their security is conceptualized from a practical point of view: no known practical attacks exist against them. Security can then be understood as a continuous effort in breaking primitives with classical or sometimes new techniques. When studied, cryptographers don't search first for attacks against ciphers but for a smaller component that exists in most attacks called a *distinguisher* that is used to build a general attack scheme.

II.ii Distinguishers

A distinguisher is an algorithm that breaks (with a good advantage) the *indistinguishibility* property of ciphers with only access to plaintexts ciphertexts pairs by deciding if the pairs are randomly generated or generated from a cipher. Distinguishers are mostly searched by observing ciphers and finding behaviors that should not exist in random functions.

For iterative constructions, a general extension of distinguishers over $r-1$ rounds into an attack over r rounds is well known in the community and explains why the last step (to obtain the attack) is rarely detailed in research papers. Indeed, if you assume that the last roundkey is smaller than the master key, an attacker can ask for a set of ciphertexts (over r rounds) and can guess the last roundkey. Using the guessed roundkey for a one round decryption, the distinguisher can be tested on the obtained partially decrypted set with the idea that if the roundkey is incorrect, the distinguisher should consider the set as originating from randomness. This process can be repeated (with one less round) up to the point where enough roundkeys have been found to retrieve the master key. By extension, a good justification used by the designers of primitives for the security of their constructions is to argue that no classical methods like linear or differential cryptanalysis produced a good distinguisher. In the following subsection, we introduce differential cryptanalysis, the principal theory we used to realized the different works of this thesis.

II.iii Differential cryptanalysis

At CRYPTO'90, Biham and Shamir [BS91a] presented an analysis of the Data Encryption Standard, the encryption standard of the standardization organism of the USA. In the same period, differential attacks were mounted against other block ciphers like `FEAL` [BS91b] to obtain new attacks. In 1998, Vaudenay *et al.* proposed a new framework called the decorrelation theory to prevent differential attacks (among others) applied for the first time in the ciphers `COCONUT` and `PEANUT` [Vau98]. The next year, Wagner in

[Wag99] proved that decorrelation theory is not enough to ensure security against differential attacks by exhibiting a new type of differential attack, the *boomerang attack* [Wag99].Since then new variants of differential attacks appeared in the literature like the *sandwich* [DKS10], *rectangle* [BDK01] or *impossible differential* [Knu98] attacks. Differential cryptanalysis is now a classical way to study ciphers and new primitives to be seriously considered must have been analyzed against the classical differential attacks.

Differential cryptanalysis studies the transformation of differences through a cipher. Given a pair of messages p_0 and p_1, the difference Δp between the two messages is $\Delta p = p_0 \oplus p_1$. Trivially, a difference is easily propagated through a linear transformation L since $L(\Delta p) = L(p_0) \oplus L(p_1)$. However, the same cannot be said about the passage through the non-linear operations where this equation becomes false. To describe the multiple outputs that can be obtained after a non-linear operation, the community talks about the probability of observing an output difference for a given input difference computed by counting the number of input pairs p_0 and $p_0 + \Delta p$ that reached the targeted output difference. In addition to differences in the message/state, more powerful attackers who can add (or know) differences in the key are sometimes considered for iterative constructions. If the cipher behaves like a random function regarding differential analysis then the probability to observe a particular output difference should be 2^{-n} where n is the size of the message. If it is not the case, we just described a particular distinguisher against this cipher allowing us to mount an attack.

Doing a complete differential analysis of primitives is hard and to simplify this studies for different primitives multiple sub-problems have been proposed in the literature. For iterative block ciphers, the difference can be followed through each round to obtain a *differential characteristics*. A differential characteristic can be described as the sequence $(\Delta_0, \Delta_1, \ldots, \Delta_r)$ of differences at the beginning of each round and the output difference. The exact probability of a differential characteristic is hard to compute but can be decently (most of the time) approximated by considering that the transition of each round is independent from the others. In this case, the probability of the differential is $\prod_{i=1}^{r} \mathbb{P}(\Delta_{i-1}, \Delta_i)$ where $\mathbb{P}(\Delta_{i-1}, \Delta_i)$ is the probability that difference Δ_{i-1} becomes difference Δ_i through the round function. Computing differential characteristics can also be simplify be leveraging the structure of the round function essentially for SPN constructions. Indeed, as said previously, the linear layer and the key addition are layers where the differences will deterministically propagate with probability 1. It leaves only the substitution layer to take into consideration to compute the probability through the round. Or the substitu-

tion layer is composed of parallel applications of the S-box which can then be the size considered for the study of the difference in the round. Since S-boxes are usually far smaller than block sizes, the exact probability for every pair $(\Delta_{\text{in}}, \Delta_{\text{out}})$ can be computed and the information can be regrouped in a table called the *differential distribution table* (DDT). The DDT is a double entry table which gives for entry $(\Delta_{\text{in}}, \Delta_{\text{out}})$ the probability of observing $\Delta_{\text{out}} = S(x) \oplus S(x \oplus \Delta_{\text{in}})$ over every possible value of x.

To simplify further the task of finding differential characteristics in SPN, Knudsen [KRW99] proposed to ignore the precise value of the differences to only keep the presence (active) or absence (inactive) of differences for S-boxes at a given round. The term adopted by the community for this object is *truncated differential characteristic*. They show themselves to be very useful for a design perspective and more precisely to argue in favor of the resistance of new designs against differential cryptanalysis. More precisely, consider a truncated differential characteristic with n active S-boxes and the propagation with the best probability p for the S-box, an easy upper bound on the probability of differential characteristics following this truncated form is then p^n. The same reasoning can be done for the truncated differential characteristics with the minimum number of active S-boxes to obtain an upper bound on the probability of differential characteristics on a given number of rounds. If this upper bound is lower than 2^{-m} where m is the block size, the cipher should be at first sight resistant against differential cryptanalysis. In the last decade (since 2009 to be exact), new methods based on solvers were presented to find truncated differential characteristics [SNC09; MP13; Mou+12; Sun+14; KLT15]. Solvers are algorithms taking as input a description of a problem and searching automatically a solution to this problem. In this thesis, we also use solvers to compute differential characteristics. More details concerning this part of our work can be found in Chapter 3.

III Overview of the manuscript

During the thesis, our works mainly focus on differential cryptanalysis over different perspectives from the study of one differential criteria in one type of Feistel construction, the study of a precise primitive `SKINNY` and revisiting some attacks that use differential cryptanalysis. Nevertheless, in this three works we always developed algorithms to construct or check our results. In this section, we give a brief review of each chapter of this thesis. In Chapter 1, we revisit the *Fast Near Collision Attack* (FNCA), a new type of attack proposed by Zhang *et al.* [ZXM18]. The FNCA was applied to two ciphers:

`Grain v1` [ZXM18] and `A5/1` in [Zha19] yielding a new key recovery attack against `Grain v1` and a new attack against `A5/1` that should be faster than the state of the art up to their publications. However, we found incoherences in the attack and proposed a correct version of the FNCA that proves the attack is not useful.

In Chapter 2, we solve a 10-year old open problem and give new results concerning Generalized Feistel networks. We especially studied a particular property called the *diffusion round* concerning the permutation of Generalized Feistel networks.

Finally, in Chapter 3, we studied the differential characteristics of the block cipher `SKINNY`. We used the very classical approach of splitting the search of differential characteristics in two parts. By using this classical approach with carefully selected tools we were able to find new best and good differential characteristics for `SKINNY`.

III.i Revisiting Fast Near Collision Attacks (FNCA)

In 2018, Zhang *et al.* in [ZXM18] proposed a new general attack, the Fast Near Collision Attack, and gave as a first application an attack the stream cipher Grain v1. Then, in [Zha19] Zhang *et al.* applied successfully the FNCA to A5/1 to obtain a better key recovery attack against this primitive. The FNCA is a divide-and-conquer attack to retrieve the internal state of a stream cipher from the keystream. It splits the internal state in two parts the crucial part (CP) and the rest part (RP). Once the crucial part is known, the rest part can efficiently be recovered. The crucial part is retrieved with a near collision attack based on a small sample of the keystream.

During the PhD, we tried to implement the FNCA against A5/1 and after a careful review of the process, we found incoherences from an information theoretical point of view in the description of the attack and some error in the code that Zhang used to test component of his attack on A5/1. As a result, we reevaluated the complexity of the FNCA and verified experimentally that no bias exists in the two targets that could explain the complexities announced by Zhang *et al.*. We also give the corrected complexity of the FNCA and prove that Zhang *et al.* severely underestimated the real cost of their attacks. This work is presented in Chapter 1 and was published at FSE:
Patrick Derbez, Pierre-Alain Fouque, and Victor Mollimard, "Fake Near Collisions Attacks", *in*: *IACR Transactions on Symmetric Cryptology* 2020.*4* (Dec. 2020), pp. 88–103, DOI: 10.46586/tosc.v2020.i4.88-103, URL: https://tosc.iacr.org/index.php/ToSC/article/view/8749

III.ii Efficient Search for Optimal Diffusion Layers of Generalized Feistel Networks

Feistels are one of the most classical way to construct block ciphers. At CRYPTO'89, Zheng *et al.* [ZMI89b] proposed different generalizations of the basic Feistel construction. One of them consists in applying in parallel multiple Feistel constructions then shifting all branches to the right. As remarks in [Nyb96; SM10], the shift to the right is not the only permutation that can be used. They named the construction with an arbitrary permutation as *Generalized Feistel Networks*.

However, no family of permutations to use for a better scheme was given. In 2010 at FSE, Suzumaki and Minematsu [SM10] proposed one criteria, the *diffusion round*, which can help for the choice of a particular permutation. The diffusion round of a permutation can be understood as a measure of the speed necessary in number of rounds for a difference on any branches to diffuse to every branches of the Feistel construction. In their work, Suzumaki and Minematsu were able to exhaust and evaluate every permutations for up to 16 branches to find all optimal permutations. Furthermore, after remarking that for every number of branches they studied, there was always an optimal even-odd permutation, defined as a permutation that transforms even numbers in odd numbers and the converse. As a result, they proposed a generic construction to find a good even-odd permutation when the number of branches is a power of two. In particular, they found a good even-odd permutation for 32 branches with a diffusion round of 10. In 2019, Cauchois *et al.* [CGT19] presented a new method to construct optimal even-odd permutations allowing them to find them up to 26 branches.

During this PhD, we studied the problem of finding optimal permutations for GFN. We presented a new characterization of the diffusion round and we used it in an efficient algorithm searching for optimal even-odd permutation with a diffusion round of 9 for GFN of 28 to 42 branches. As such, we were able to find all optimal even-odd permutations for 28, 30, 32, 34 and 36 branches as well as found good candidate permutations for 38, 40 and 42 branches.

This work is presented in Chapter 2 and was published at FSE:
Patrick Derbez *et al.*, "Efficient Search for Optimal Diffusion Layers of Generalized Feistel Networks", *in*: *IACR Transactions on Symmetric Cryptology* 2019.*2* (June 2019), pp. 218–240, DOI: 10.13154/tosc.v2019.i2.218-240, URL: https://tosc.iacr.org/index.php/ToSC/article/view/8321

III.iii Efficient Methods to Search for Best Differential Characteristics on SKINNY

SKINNY is a family of tweakey SPNs designed in 2016 by Beierle *et al.* [Bei+16]. There are six versions grouped in terms of size in two families: SKINNY-64 and SKINNY-128 using respectively blocks of 64 and 128 bits. To give a first idea of its security against differential cryptanalysis, its designer included in its specification [Bei+16] a model to search for the minimum number of active S-boxes in a truncated differential characteristics. The model is solved by a constraint solver and more precisely a MILP solver. A constraint solver is a generic algorithm that takes in input a description of a problem, containing variables and relations between them and searches a solution of this problem. In MILP, the variables' values are numbers and the constraint are linear inequalities between those variables.

Since then few results were published concerning the differential characteristics of SKINNY. To the best of our knowledge, only two papers present results on them: Liu *et al.* in [LGS17] needed some good differential characteristics to study the resistance of SKINNY against boomerang and rectangle attacks in the tweakey settings. Using a MILP modeling of SKINNY-64 and an ad-hoc tools for SKINNY-128, there were able to find in the different tweakey settings the best differential characteristics up to 13 rounds for SKINNY-64 and some good differential characteristics (but on a small number of rounds) for SKINNY-128.

The second result we found concerning the differential characteristics of SKINNY was published by Abdelkhalek *et al.* in [Abd+17]. They found the best differential characteristics of SKINNY-128 in the single key settings up to 13 rounds using a MILP model containing an efficient description up to now thought impossible of the 8-bit S-box of SKINNY-128. Moreover, they showed that no differential characteristic with a better probability than 2^{-128} exist for 14 rounds (and more) in the single key setting.

We complete the knowledge concerning the differential characteristics of SKINNY. In this work, we adopted a very classical approach of splitting the search in two steps. The Step 1 consists in searching truncated differential characteristics and the Step 2 consists in instantiating in value the solutions found in the first step. The contributions of this work other than the new results are the way the two steps are solved. We proposed a new tool to complete the Step 1 search based on dynamic programming. For the Step 2 process, we used a constraint programming solver, another type of constraint solver that allows more types of constraints than the MILP and obtained a faster resolution of the Step 2 process than the MILP modeling of [Abd+17]. In this way, we were able to

compute every optimal differential characteristics with a probability greater than 2^{-64} in the round reduced version of SKINNY-64. Concerning SKINNY-128, we retrieved the results of [Abd+17] within of few minutes (compared to 16 days) and found new best differential related-tweakey characteristics in the different related-tweakey settings. Furthermore, we found in some of those tweakey settings good differential characteristics on a greater round reduced versions and proved that none could exist for some given numbers of rounds.
This work is presented in Chapter 3 and was published at ACNS:
Stéphanie Delaune *et al.*, "Efficient Methods to Search for Best Differential Characteristics on SKINNY", *in*: *International Conference on Applied Cryptography and Network Security*, Springer, 2021, pp. 184–207

Chapter 1

Revisiting the Fast Near Collision Attack

1.1 Introduction

Recently, the fast near collision attack (FNCA), a new general attack was applied to both `Grain` [ZXM18] and `A5/1` [Zha19], two stream ciphers. During the PhD to study in more details `A5/1`, we try to implement completely this new attack against `A5/1` but we were not able to do so. Then, we did a review of the attack and found some incoherences in the published results. By extension, we also reviewed the FNCA against `Grain` and found the same incoherences. As a result, we decided to formally review and checked the results of these two papers using some algorithms containing elements of information theory.

`A5/1` was designed in 1987 for the *Global System for Mobile Communication* (GSM) and became public in 1994 with a leak of the general design before being completely reverse engineer in 1999 by Marc Briceno. Ross Anderson reported a "terrific row between NATO signal intelligence agencies in the mid-1980 over whether GSM encryption should be strong or not". Germany wanted a strong design to protect its communication against possible interception by the Warsaw Pact, the other allies wanted a weaker design. Finally, a French design was adopted. Without surprise, attacks and vulnerabilities were found beginning with Anderson in 1994 [And94] based on guessing majority of the state. In 1997, Golič [Gol97] proposed an attack with a complexity of $2^{41.16}$ where some guesses on the clock are used to generate a linear system of equations. A more efficient version of Golič attack was proposed by Biryukov *et al.* [BSW00] in 2000 with a well adapted time-memory trade-off. In the same year, Biham *et al.* published [BD00], another attack needing $2^{39.91}$ clockings and $2^{20.8}$ plaintexts. In 2003 Ekdhal *et al.* proposed a new attack [EJ03] against the initialization process of A5/1. It is improved a first time in 2004, by Maximov *et al.* in [MJB04] and a second time by Barkan *et al.* in [BB05] to yield an

attack needing a few minutes. Although multiples vulnerabilities were found, counter measures were adopted in the GSM standard to ensure security. Nevertheless, A5/1 is still an important target of study with its preponderant role in some legacy systems (G2).

`Gain v1` is another stream cipher and is part of the eSTREAM portfolio [RB08] with the rest of the Grain stream cipher family. As a more modern stream cipher, a considerable effort to cryptanalyse `Grain v1` exists in the literature. Since 2009 and its invention by Dinur *et al.* in [DS09], cube attacks were increasingly successfully applied to `Grain` to cite only some of those works [DS11; Wan+13; Ban14; Rah+16]. This last cube attack is a key recovery attack on reduced version of `Grain v1` where initialization is reduced to 100 rounds with a complexity reduced from the exhaustive search by a 2^{32} factors. Other methods of attacks against `Grain` in the literature are among others an algebraic attack [AM08], internal state recovery attacks [Mih+12b; Mih+12a] or a near collision attack [Zha+13]. Nevertheless, the complete `Grain` is still seen as a secure stream cipher for the numerous keystream bits necessary for the attacks. In this context, the attack proposed by Zhang *et al.* in [ZXM18] that we revisit in this Chapter is even more impressive as an efficient key recovery attack on the whole scheme.

Checking results is in some sciences such as experimental physics as important as the result itself. In these research domains, results have to be validated by two separate and independent teams before being published. In some computer sciences areas where results can depend on the input data set, it is also highly important to give access to these data and to the code. In data mining for example the reproducibility of results has been acknowledged as mandatory before publishing work in order to ease the checking and/or comparison of this work with further research works.

In symmetric cryptography, where usually the complexities of attacks and distinguishers can be out of reach with experiments, a well-known method consists in experimentally checking only some parts of the attack and/or by targeting a toy cipher. Indeed attacks can usually be split in two parts: the adversary has to guess some bits and then he evaluates some distinguishers. The evaluation of the distinguisher cannot be exhaustive since it would have been tested for all guess bits. If we checked that the distinguisher is working for random guess, we declare that the attack is validated. However, it is the authors' accountability to check carefully the experiments and reviewers usually verified the fact that the authors seem to have correctly performed their results. Nevertheless, sometimes it is not sufficient to ensure the correctness of some proposed attacks and it is up to the community to revisit and discuss previous works to offer new insights on their contribu-

tions. For instance in [Gra01] Granboulan showed that differential attacks on SKIPJACK proposed in [KRW99] were flawed because the probabilities of some differential characteristics were not correctly evaluated. In 2007, Wang, Keller and Dunkelman [WKD07] caught a similar error for an impossible differential used in several attacks on SHACAL-1. Such errors may also come from hypothesis which do not hold for all ciphers as exemplified by Murphy [Mur11] with boomerangs on both DES and AES.

Symmetric cryptography is not the only place where mistakes can be made. In public-key cryptography and provable cryptography, it is also possible to discover errors as the famous bug in the OAEP paper [BR94], which has been corrected in [Sho01; Fuj+01]. The same kind of problems appeared in proofs in symmetric cryptography for the equivalence between the random oracle model and ideal cipher model [CPS08] corrected in [HKT11] and more recently in the security proof of the OCB-2 mode of operation [Ino+19]. Consequently, Barthe *et al.* have developed tools to verify these proofs as in [Bar+11b; Bar+11a] and even on the corrected proofs they have been able to spot some errors or imprecisions since these tools do not accept unclear arguments or logical flaws. As a consequence, they design the EasyCrypt tool to help the verification of cryptographic proofs to reason about code-based proofs as these tools were first developed to verify programs. There is no such tool to check symmetric-key cryptanalysis. The verification of these attacks boils down to checking the complexity analysis of the cryptanalytic algorithm. The main difficulty is that some parts are heuristic and the verification of these heuristics are not easy to automatize and to perform rigorously. Moreover, understanding the problems is not always an easy task since it requires to reverse engineer the experiments performed which are subject to statistical effects and it is less easy than reading a proof.

Contributions. In this Chapter, we look at the recent fast near collision attacks proposed by Zhang, Xu and Meier against the Grain v1 [ZXM18] stream cipher and by Zhang against A5/1 [Zha19]. The main idea behind fast near collision attack consists in a divide-and-conquer partition of the full internal state into the crucial part (CP) and the rest part (RP). The latter part can be efficiently recovered using only the CP, while the former one is retrieved using a near collision attack based on a small number of bits of the keystream.

Our first goal was to implement the attack on the A5/1 stream cipher since the time and memory complexities seem within our reach and practical. However, during this process we discovered several issues in the claimed probabilities, leading to an overall complexity much worst than expected. In fact, we came up to implement a slower version of the attack proposed by Golić at Eurocrypt'97 [Gol97]. Consequently, we scrutinized

this article and decided to reevaluate the time complexity to 2^{28} calls to (the end of) Golić's attack, for an overall complexity around 2^{42}. Since this attack is a bit difficult as it is flooded with the details of the stream cipher under attack, we decided to present its basic ideas in a self-contained manner. The details can be found in Section 1.3. Finally, we decided to also verify the previous attack on `Grain v1` as proposed at Eurocrypt'18 and we discovered similar problems in the analysis. In particular, the correct overall complexity is 2^{113} and so the attack is less efficient than the naive exhaustive search in $2^{87.4}$ ticks on `Grain v1`. The complete analysis is presented in Section 1.4.

More importantly, we show in Section 1.2 that fast near collision attacks, as described in both [ZXM18] and [Zha19], are intrinsically erroneous. Replacing the refined self-contained method, which is the core of those attacks and the only algorithm relying on near collisions, by an algorithm outputting a random set (of fixed size) of pre-images would lead to the exact same complexities. Thus such attacks are illusive.

1.2 Fast Near Collision

At Eurocrypt'18, Zhang *et al.* described a new powerful cryptanalysis technique called *fast near collision attack*. This technique was specially designed to analyze stream ciphers and was successfully applied to both `Grain v1` [Hel+06] and `A5/1` [BGW99]. It combines both a divide-and-conquer approach and near collisions. The core idea is to use near collisions to restrict the possible values of some bits of the internal state.

1.2.1 The refined self-contained method

Let f be a public function from n to m bits, $\mathbf{x}_s$ be a secret n-bit word and $\mathbf{k}_s$ the output of $f(\mathbf{x}_s)$. A classical objective is to retrieve $\mathbf{x}_s$ from the knowledge of both f and $\mathbf{k}_s$. In the following we will explain how the fast near collision technique claims to restrict the search space for $\mathbf{x}_s$.

The process is composed of 3 procedures which aim at computing a set X containing $\mathbf{x}_s$ with a high enough probability.

Precomputation. The first step in a fast near collision attack is to construct a differential table T_d mapping each pair $(\Delta k, k)$ to all possible Δx such that:

- $|\Delta x| \leq d$

- there exists x such that $f(x) = k$ and $f(x \oplus \Delta x) = k \oplus \Delta k$.

In other words, the table T_d is a variant of the classical *differential distribution table* associated to an S-box. The number of times Δx is solution for $(\Delta k, k)$ is also stored as extra information. This allows for each value of k to select Δk to maximize the probability of $f(x \oplus \Delta x) = k \oplus \Delta k$ knowing both $f(x) = k$ and $\Delta x \in T_d[\Delta k, k]$.

Note that in case it would be too costly to fully compute T_d, x and Δx can be sampled.

Online. The second step of the procedure uses the precomputed table to generate a set X containing $\mathbf{x}_s$ with a good probability. The process is described in Algorithm 1. The idea is to randomly generate x, compute $k = f(x)$, look into $T_d[k \oplus \mathbf{k}_s, \mathbf{k}_s]$ for possible Δx's and check whether $f(x \oplus \Delta x) = \mathbf{k}_s$. If the last equality holds then $x \oplus \Delta x$ is added to the set X as a possible value for $\mathbf{x}_s$.

Algorithm 1 The refined self-contained method

```
1: Data: keystream k_s, difference Δk, table T_d,
2: Result: a set X such that x_s ∈ X has high probability
3: X ← ∅
4: for i = 0 to N do
5:     randomly generate x such that f(x) = k_s ⊕ Δk
6:     for all Δx ∈ T_d[Δk, k_s] do
7:         if f(x ⊕ Δx) = k_s then
8:             X ← X ∪ {x ⊕ Δx}
9:         end if
10:    end for
11: end for
12: return X
```

Amplifying phase. In order to increase the probability that X contains $\mathbf{x}_s$, Zhang *et al.* propose to run $N \times M$ times Algorithm 1 , each random invocation outputting a set denoted $X_{i,j}$ ($i = 1$ to N and $j = 1$ to M). Then a new set is outputted by computing

$$X = \bigcup_{i=1}^{N} \left(\bigcap_{j=1}^{M} X_{i,j} \right).$$

1.2.2 About probabilities

While we could discuss on the interest of this construction, we are only interested by the probability that $\mathbf{x}_s$ belongs to the constructed set X.

`Grain v1`. In [ZXM18], Zhang *et al.* used the fast near collision technique to mount an attack against `Grain v1`. They applied the refined self-contained method to a function f such that $n = 12$ and $m = 2$. They obtained a set X of size 848 and claimed the probability for $\mathbf{x}_s$ to belong to X is around 89.64% which is a bit higher than the $848/1024 = 82.81\%$ expected. Note that here the function f is such that $z = f(x)$ can be rewritten as $z = x_1 \oplus h(x_2)$ and thus, the refined self-contained method was applied on $h(x_2) = 0$. In particular this means that the search space is restricted without the knowledge of any bit of keystream.

`A5/1`. In [Zha19], the function f is such that $n = 15$ and $m = 2$. Zhang obtained a set X of size 7835 and claimed the probability for $\mathbf{x}_s$ to belong to X is around 99.09% which is higher than the $7835/8192 = 95.64\%$ expected.

We claim all those claimed probabilities are wrong or, more precisely, cannot be true without a big enough bias in the initialization phases of both `A5/1` and `Grain v1`. This is supported by the following theorem:

Theorem 1.1. *Let $\mathcal{A}$ be an algorithm which takes as input a function f and an element $\mathbf{k}_s$ and outputs a subset X of $f^{-1}(\mathbf{k}_s)$. Let $\mathbf{x}_s$ be an element of $f^{-1}(\mathbf{k}_s)$ drawn uniformly at random. The probability that $\mathbf{x}_s$ belongs to X is exactly*

$$|X|/|f^{-1}(\mathbf{k}_s)|.$$

The refined self-contained method fulfils the requirements of Theorem 1.1 but Zhang *et al.* claim the set X output by the algorithm contains the secret $\mathbf{x}_s$ which generated $\mathbf{k}_s$ with a good probability. Note that the algorithm can be run *before* the secret was actually generated and thus Zhang *et al.* claim can be invalidated by the following experiment:

1. randomly generate $\mathbf{k}_s$
2. run the refined self-contained method on f and $\mathbf{k}_s$ and obtain the subset X
3. draw $\mathbf{x}_s$ uniformly at random in $f^{-1}(\mathbf{k}_s)$
4. check whether $\mathbf{x}_s$ belongs to X

Hence, the probabilities given in both [ZXM18] and [Zha19], and by extension the complexity of corresponding attacks, are quite suspicious. Actually, they would hold if and only if it is not possible to draw $\mathbf{x}_s$ uniformly at random in $f^{-1}(\mathbf{k}_s)$ which would imply bias in the initialization process.

1.2.3 Several issues

We found several issues and unreproducible results in both [ZXM18] and [Zha19]. The first and most important one is about the set outputted by Algorithm 1 and, more precisely, about its average size and the average probability for the *right value* to belong to this set. For both `Grain v1` and `A5/1`, they were obtained experimentally from unspecified procedures and do not satisfy Theorem 1.1. Since Zhang *et al.* state to have conducted *extensive experiments*, either the whole experiments were flawed or the pseudo-random generators they used were biased.

Another issue lies in the amplifying phase. First the computations are all based on the wrong results regarding Algorithm 1 and so are unlikely to be correct. But there is another issue with this phase. Authors used two independent theorems to exhibit the claimed special behavior of the set X constructed in the amplifying phase: one to compute the size of X and one to compute the probability for the right value to belong to X. While using two different avenues to prove two properties on the same set is not important in regards to the truth of the statement, the theorem they used to compute the size of X (Statement 1.1 in this chapter) is flawed. As a consequence, there is a decorrelation between the computation of the probability that X contains the correct value and the computation of the size X, explaining again the incorrect complexities they found for their attacks.

Statement 1.1 (Theorem 3 of [ZXM18])**.** *Let V be a set and let draw uniformly at random a collection (U_i) of subsets of V. Let $F_i = \bigcup_{k \leq i} U_k$. Then on average the following relation holds:*

$$|F_{i+1}| = |F_i| + |U_{i+1}| - \sum_{j=0}^{|U_{i+1}|} \frac{\binom{|F_i|}{j} \cdot \binom{|F_{i+1}|-|F_i|}{|U_{i+1}|-j}}{\binom{|F_{i+1}|}{|U_{i+1}|}} \cdot j$$

The sum in the formula is expected to compute the average size of the intersection between both the sets F_i and U_{i+1} and this is where the error lies. The main idea is correct as they count the number of configurations such that j elements of U_{i+1} belong to F_i and $|U_{i+1}| - j$ elements do not. But actually, at this point, *not in F_i* does not mean *in $F_{i+1} - F_i$* but means *in $V - F_i$*. Indeed, U_{i+1} is drawn as a subset of V not as a subset of F_{i+1}. Hence the corrected version of the Statement 1.1 is proposed in Theorem 1.2.

Theorem 1.2 (Corrected version)**.** *Let V be a set and let draw uniformly at random a collection (U_i) of subsets of V. Let $F_i = \bigcup_{k \leq i} U_k$. Then on average the following relation*

holds:

$$|F_{i+1}| = |F_i| + |U_{i+1}| - \sum_{j=0}^{|U_{i+1}|} \frac{\binom{|F_i|}{j} \cdot \binom{|V|-|F_i|}{|U_{i+1}|-j}}{\binom{|V|}{|U_{i+1}|}} \cdot j$$

In particular, the formula used by Zhang *et al.* would always underestimate the average size of set F_i. This fully supports our claiming: to reach the probabilities announced in both [ZXM18] and [Zha19] the size of the set output by the refined self-contained method has to be bigger than they expected.

Finally there is a wrong assumption about the *right value*. More precisely, in both papers authors assume there is only one right value that will behave differently than the wrong ones. With enough keystream bits this is true that there is only one internal state solution. But the fast near collision only uses a *small* part of the known keystream bits and so the assumption of only one right value does not hold. For instance, for the attack against `A5/1`, the fast near collision technique is applied to only 5 keystream bits and we show Section 1.3.3 there are many more right values than only one.

In the next sections, we will show for both `Grain v1` and `A5/1`, the observed deviation in the probabilities is wrong and will give the corrected complexities of the corresponding attacks.

1.3 Fast Near Collision on A5/1

In this Section we carefully study the attack presented in [Zha19] on the A5/1 stream cipher. This attack is absed on a divide and conquer partition of the internal state. The first part of the internal state called the *crucial part* (CP) is The second part of the internal state called the *rest part* (RP) is obtained by leveraging the knowledge of the CP and the keystream with the attack proposed by Golić in [Gol97]. To be as complete as possible, we begin with recalling the design of `A5/1` and Golić attack. Then, we describe Zhang used of fnca on A5/1 and explain why its complexity was underestimated.

1.3.1 Description of `A5/1`

`A5/1` is a stream cipher underlined by a 64-bit internal state. The internal state is composed of three short linear feedback shift registers (LFSR) of length 19, 22 and 23 bits respectively. In the rest of the chapter we will refer to them as $R1$, $R2$, $R3$. As illustrated

in Figure 1.1, the feedback taps for each LFSR are positions 13, 16, 17 and 18 for $R1$, 20 and 21 for $R2$ and 7, 20, 21 and 22 for $R3$. Furthermore, each LFSR also possesses a clocking tap at position 8, 10, 10 for respectively $R1$, $R2$ and $R3$, represented with the red arrows in the figure.

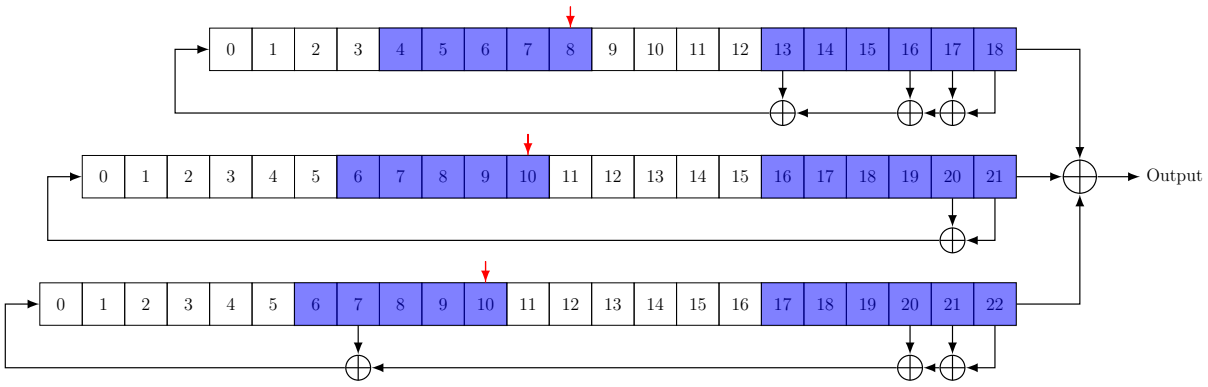

Figure 1.1 – Description of `A5/1` (source: [Jea16]). The 33 blue bits are the one required to compute the first 5 keystream bits.

`A5/1` uses an asynchroneous clocking regime for the LFSRs: at each clock tick, a LFSR is clocked if its clock tap value is the majority value between the three clocking taps (refer to Table 1.1 for the plain list of the clocking).

clock value of	R1	0	0	0	0	1	1	1	1
	R2	0	0	1	1	0	0	1	1
	R3	0	1	0	1	0	1	0	1
register clocked	R1	✓	✓	✓	-	-	✓	✓	✓
	R2	✓	✓	-	✓	✓	-	✓	✓
	R3	✓	-	✓	✓	✓	✓	-	✓

Table 1.1 – Clocking of `A5/1`

Finally, we review the utilisation of the `A5/1` stream cipher during a GSM conversation session with Algorithm 2, the pseudo code for the generation of the 228 bits of keystream of one GSM session.

1.3.2 An attack from Golić

In [Gol97], Golić introduces a clever memory-less attack against `A5/1`. It is a basic divide-and-conquer attack recovering the unknown initial state from a known keystream sequence (of 64 bits). Called the initial state S_0 and defined the sequence of internal states after n

Algorithm 2 The procedure used during a GSM session to generate 128 bits of keystream

$P = (F, K) = (P_{85}, \ldots, P_0)$ an 86-bit word composed of a 64-bit key K and a 22-bit frame F

for $i = 0$ to 85 **do** *Loading Parameters*
    $R1[0],\ R2[0],\ R3[0] \leftarrow R1[0] \oplus P_i,\ R2[0] \oplus P_i,\ R3[0] \oplus P_i$
    clock $R1$, $R2$, $R3$ simultaneously
end for
for $i = 0$ to 99 **do** *Initialization*
    clock $R1$, $R2$, $R3$ asynchronously
end for

for $i = 0$ to 227 **do** *Keystream generation*
    $z_i = R1[18] \oplus R2[21] \oplus R3[22]$
    clock $R1$, $R2$, $R3$ asynchronously
end for

Output $z = (z_{227}, \ldots, z_0)$ *the keystream*

clocking as S_{nn}. In his attack, Golič reconstructs the internal state S_{101} from the known keystream than uses it to determine S_0.

First, the main idea to recover S_{101} is quite simple: it is an elimination of the non-linear operations by guessing the clock bits. Remarks that in his paper, Golič proved that only $5 \cdot 2^{61} \approx 2^{63.32}$ states S_{101} were reachable from the inital state S_0. In the following, we will search S_{101} in this set of reachable states.

If, for each of the three LFSRs, one guesses the clock bit for n (asynchronous) clocks of the LFSRs, we can obtain $3n$ linear/affine equations. For instance, for $n = 10$ it means guessing on the initial state $R1[0..8]$, $R2[1..10]$, $R3[1..10]$ as well as $R1[13] \oplus R1[16] \oplus R1[17] \oplus R1[18]$. In this exemple, the equations obtained for the first register when guessed values are called $g_{1,i}, i \in [0..9]$ are:

$$\forall i \in [0..8], R1[i] = g_{1,i}$$
$$R1[13] \oplus R1[16] \oplus R1[17] \oplus R1[18] = g_{1,9}$$

Furthermore, from those $3n$ guesses we know the beginning of the clocking sequence and obtains on average $1+4n/3$ affine equations from the knowledge of the keystream bits

(the first keystream bit is computed before clocking with the guessed sequence). Indeed, at each step the probability for a register to be clocked is 3/4 (see Table 1.1) and as a consequence from the $3n$ guesses we know on average the clocking sequence for $4n/3$ rounds, leading to the equations $R1^i[18] \oplus R2^i[21] \oplus R3^i[22] = z_i$ for $0 \leq i \leq 4n/3$. Note that trivially these $1+4n/3$ equations are linearly independent: each one has at least two new variables appearing from the minimum two registers that are clocked while $n \leq 18$. They also are linearly independent of the first $3n$ equations if each of them contains at least one bit that was not guessed. This happen with high probability given that $n \leq 10$. For now, we will consider that the $1 + 4n/3$ equations are linearly independent of the $3n$ first. Hence, a naive solution would be to accumulate enough equations to solve the system by inverting a matrix. This would require n to be such that $1+4n/3+3n \geq 63.32$, so $n \geq 14.38$ (respecting effectively the condition for the $1+4n/3$ equations to be linearly independent). But actually, for $n > 10$, the equations are not linearly independent, and we need to increase the number of guesses to make.

To overcome this issue, Golić proposed a better algorithm close to the *early abort technique* [Lu+08]. The basic idea he proposed was to used redundant equation as a consistency tool. First, one uses the method previously described with $n = 10$ to obtain $1+3n+4n/3 \approx 44.3$ linearly independent equations on average. Than instead of guessing the remaining $m \approx \frac{(63.32-44.3)}{3}$ missing bits in each LFSR, Golìc proposed that at each step the adversary guesses/computes the majority bit, gets the corresponding equation from the corresponding keystream bit and checks whether it is consistent with the previously obtained equations. If the equation is consistent, the equation is added to the system, the missing clocking bits are guessed/computed from the majority bit and the already known clocking bits and the whole state is clocked. This process is repeated until the system uniquely determines the 64-bit state. Golić showed that the average complexity of the procedure is around $2^{41.16}$ simple operations.

In his paper, Golìc also presents a way to reconstruct the key and the frame number from the inital state S_0. In this sense, the attack proposed by Golìc in [Gol97] and by Zhang in [Zha19] are key recovery attacks against `A5/1`.

1.3.3 Fast near collision attack against `A5/1`

At Asiacrypt'19, Zhang proposed an improved memory-less attack against `A5/1`, claiming a time complexity around 2^{31} clocks [Zha19]. Given a sufficiently long sequence of

keystream bits (around 64), he proposed a 2-step procedure to recover the full internal state.

1. The main observation is that 2 consecutive bits of keystream only depend on 15 variables of the internal states. Using the technique described in Section 1.2.1, Zhang constructs a set containing approximately 7835 values for the 15 variables and claims that the probability the value we want is in it to be around 99.09%. Four such sets are constructed, one for each pair (z_i, z_{i+1}) of keystream bits, for i from 0 to 3. Then a sophisticated merge procedure is applied to construct a set of $2^{16.6}$ values for the 33 bits of the internal state leading to $z_0z_1z_2z_3z_4$. Furthermore, Zhang claims that the probability for the set to contain the right value is round $(0.9909)^4 = 96.41\%$. Note that $2^{16.6}$ possibilities is much lower than $2^{33-5} = 2^{28}$, which is what we would intuitively expect.

2. The 31 remaining state bits are recovered using the procedure of Golić described Section 1.3.2 with few refinements.

1.3.4 Complexity correction

In this section, we show the time complexity of the attack presented by Zhang at Asiacrypt'19 is actually much higher than announced in [Zha19]. More precisely, we show it is impossible to restrict the number of possible values for the 33 bits of the crucial part (CP) from 2^{33} to $2^{16.6}$ using only the 5 first keystream bits without drastically decreasing the probability of success of the attack. Hence, it turns out Zhang's attack has the same complexity than the one of Golić.

Theoretical analysis. As explained in Section 1.3.3, the attack proposed by Zhang begins by the recovery of the crucial part (CP) corresponding to 33 bits of the internal state of `A5/1`. Those bits are coloured in blue on Figure 1.1. The only information used in the procedure to do so is the first five bits of keystream generated from the internal state.

Let x be a randomly chosen value for the CP part and k its corresponding 5-bit keystream output. In his attack, Zhang claims that from k he can extract a set of $2^{16.6}$ CP configurations containing x with a very high probability. To invalidate this result we first make the following proposition:

Proposition 1.1. *Given a 5-bit keystream output k, there are* **exactly** 2^{28} *values for the 33 bits of the CP part leading to k.*

Proof. The 33 bits of the CP part can be divided into two groups: one composed of 15 bits ($R1[4...8]$, $R2[6..10]$ and $R3[6...10]$) used only to determine the clocking sequence and one composed of 18 bits ($R1[13..18]$, $R2[16...21]$, $R3[17..22]$) used to generate the keystream bits. Hence, once the 15 bits of the first group are fixed, the clocking sequence is known and so each of the five first keystream bits is computed as a linear combination of the 18 bits of the second groups. Furthermore, those 5 linear equations are independent since each of them depends on at least one bit that does not appear in the other ones (because at least two registers are clocked each round). Thus, for each possible value of k we have exactly $2^{18-5} = 2^{13}$ possible values for the 18 bits of the second groups. □

According to Proposition 1.1, the claim of Zhang would imply that over the 2^{33} possible values of the 33 bits of the CP part, only a subset of $2^{16.6+5} = 2^{21.6}$ values (a set of $2^{16.6}$ for each of the 2^5 possible keystream values) can be actually reached after A5/1 initialization, the remaining ones being reached with marginal probability. While it seems quite obvious that such a big bias would have already been observed, we ran several experiments to refute the claim made by Zhang.

Experimental results. We first experimentally verified Proposition 1.1. We count for each of the 2^5 5-bit keystream prefix the number of CP values that generate it. As expected, we found that for 5 given bits of keystream prefix, there are exactly 2^{28} CP combinations that generate it.

The second hypothesis we studied was a potential bias in reaching every CP configuration from the initialization phases of a GSM session. To test this hypothesis, we ran two experiments, sampling at random the 33-bit CP part after an `A5/1` initialization.

For the first one, we simply drawn uniformly at random 2^{36} 64-bit keys and 22-bit frame counters. For each of them we performed the initialization process of `A5/1` as detailed in Algorithm 2 and computed the corresponding value of the 33 bits of interest. To avoid any bias in the experiment we used `AES` in CTR mode as source of randomness. For the sake of clarity and to give more details about the random sampling, we give a pseudo-code description of the experiments in Algorithm 3.

The second experiment is exactly the same as the first one but the 64-bit key is composed of 54 random bits and 10 zeroes for its rightmost bits as it was traditionally done in some system, like comp128v2.

Algorithm 3 Experiment

```
sample a 128-bit word KEY from /dev/rand                                   Setup
initialize a 128-bit word COUNTER at 0
initialize a 2^33 array called CONFIGURATION;

for i = 0 to 2^36 do                                                       Experiment
    RANDOM = AES-CTR(KEY, COUNTER) and increase COUNTER
    extract from RANDOM one A5/1-key, KEYEXP and one A5/1-frame, FRAMEEXP
    do an A5/1 initialization with KEYEXP and FRAMEEXP
    select the 33-bit of the CP part of the obtained internal state
    increment the corresponding field in CONFIGURATION
end for

Output CONFIGURATION
```

We present in Figure 1.2 the distribution of occurrences of the 2^{33} possible CP values for respectively GSM session key of 54 random bits and 64 random bits and for randomly selected internal states of 64-bit in the form of histograms mapping a value n to the number of CP configurations (in log scale) that are sampled n times.

No bias as strong as the one presented in the complexity announced by Zhang can be observed on those representative histograms.

Finally, we provide a last experiment definitely showing the attack presented in [Zha19] is flawed.

1. Randomly generate a 5-bit word $\mathbf{k}_s$
2. Run the refined self-contained method to obtain a set X of size $2^{16.6}$. According to [Zha19], this set should contain the secret which generated $\mathbf{k}_s$ with probability 0.9641.
3. Do N times:
   (a) Randomly generate a key and a frame counter and run the initialization process
   (b) Check whether the first five keystream bits match $\mathbf{k}_s$. If not repeat the previous step.
   (c) Check whether the value of the CP part belongs to X.

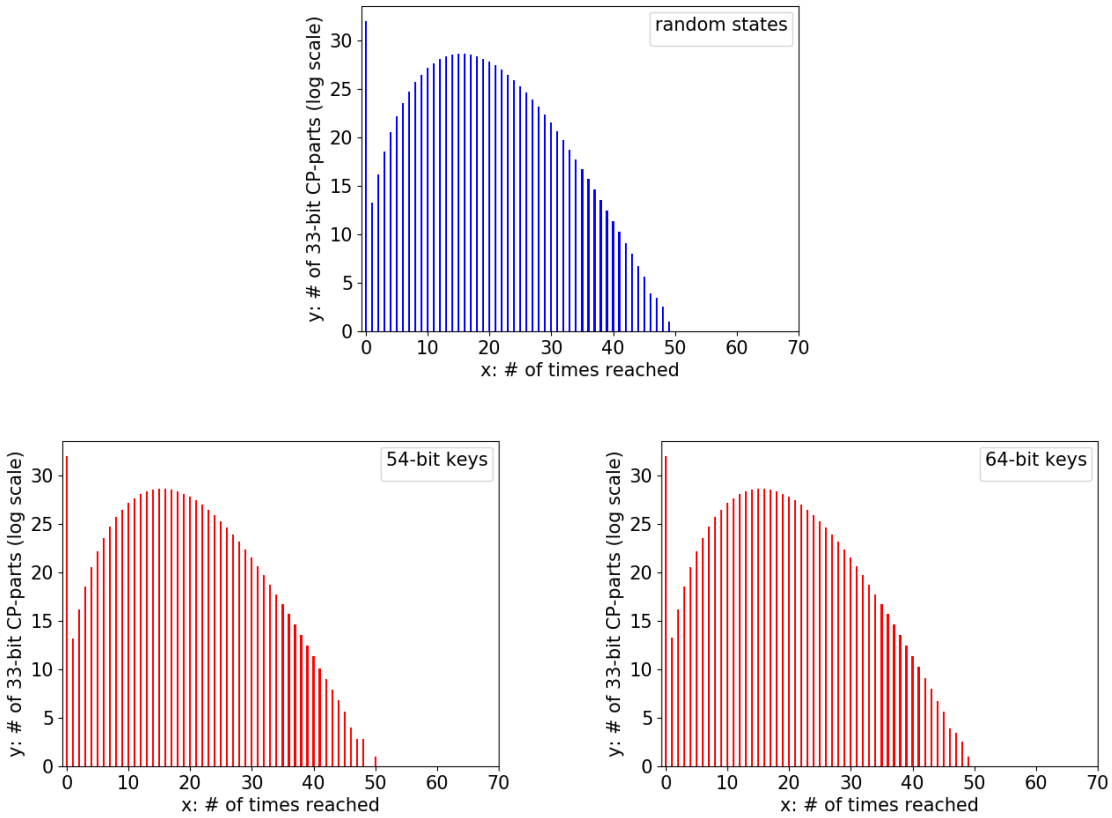

Figure 1.2 – One diagram represents the number of 33-bit CP part values (in log scale) from experimental data in ordinates that are reached exactly x times on the horizontal axe. The experimental data for the two red diagrams are generated as described in Algorithm 3 with 54-bit keys for the left one and 64-bit keys for the right one. The experimental data of the blue diagram comes from directly randomly generating the 33-bit CP part of the A5/1 internal state.

4. Check whether the experimental probability matches the expected one.

We ran this experiment[1] 10 times with $N = 2^{28}$ and found the experimental probability to be very close to $2^{-11.4}$, confirming the probability of 0.9641 claimed by Zhang to be far from the reality.

Corrected complexity. With the probability of success corrected, Zhang's attack becomes very similar to the one of Golić. The difference is that he would guess 18 extra bits while Golić would have 5 linear/affine equations between those 18 bits and the keystream.

1. Actually at each trial we took for X the set of the $2^{16.6}$ values reached the most. This highlights the refined self-contained method is irrelevant and can be replaced by any algorithm outputting a set of size $2^{16.6}$.

Hence in Zhang attack one would proceed less keystream bits before obtaining an invertible system of equations and thus more keystream bits should be checked *a posteriori*, leading to an attack which cannot be better than Golić one.

1.4 Fast Near Collision on Grain v1

In this section, we study the attack proposed at Eurocrypt'18 [ZXM18] in the same way we did in the previous section for `A5/1`.

1.4.1 Description of `Grain v1`

`Grain` is a family of stream ciphers that was retained in the eSTREAM portfolio [09]. In this chapter, we focus on `Grain v1` as specified in [HJM07]. This stream cipher is composed of one LFSR of 80 bits chained with a non-linear feedback shift register (NFSR) of 80 bits.

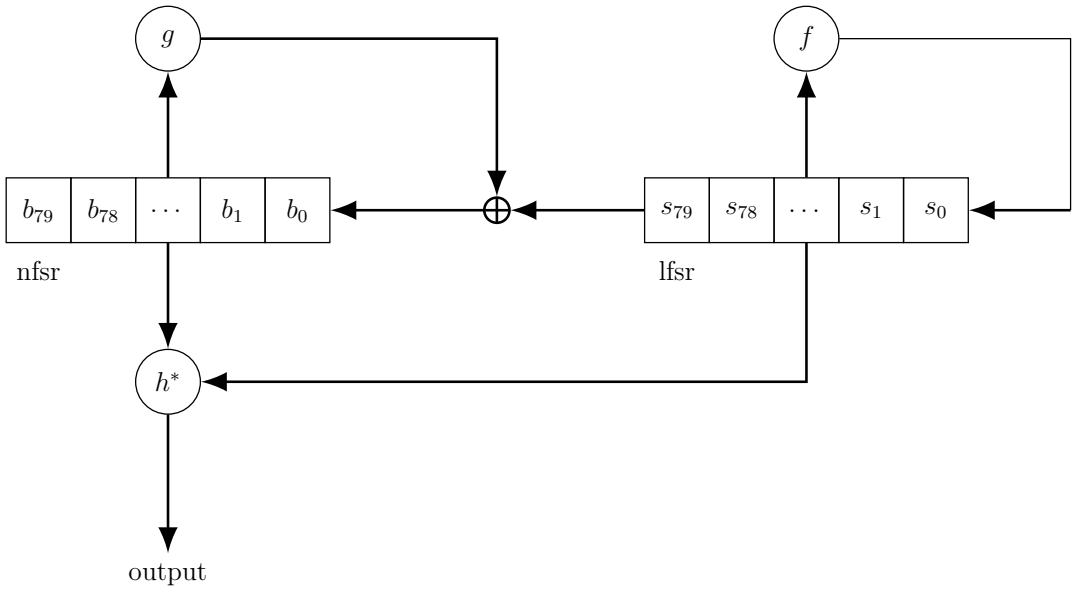

Figure 1.3 – Simple representation of the grain v1 cipher

At step i, the content of the LFSR is denoted by $s_i, s_{i+1}, \ldots, s_{i+79}$ and the content of the NFSR is denoted by $b_i, b_{i+1}, \ldots, b_{i+79}$.

The update function of the LFSR is defined as

$$s_{i+80} = s_{i+62} \oplus s_{i+51} \oplus s_{i+38} \oplus s_{i+23} \oplus s_{i+13} \oplus s_i,$$

and the one of the NFSR as

$$\begin{aligned} b_{i+80} &= s_i \oplus b_{i+62} \oplus b_{i+60} \oplus b_{i+52} \oplus b_{i+45} \oplus b_{i+37} \oplus b_{i+33} \oplus b_{i+28} \oplus b_{i+21} \oplus b_{i+14} \\ &\oplus b_{i+9} \oplus b_i \oplus b_{i+63}b_{i+60} \oplus b_{i+37}b_{i+33} \oplus b_{i+15}b_{i+9} \oplus b_{i+60}b_{i+52}b_{i+45} \\ &\oplus b_{i+33}b_{i+28}b_{i+21} \oplus b_{i+63}b_{i+45}b_{i+28}b_{i+9} \oplus b_{i+60}b_{i+52}b_{i+37}b_{i+33} \\ &\oplus b_{i+63}b_{i+60}b_{i+21}b_{i+15} \oplus b_{i+63}b_{i+60}b_{i+52}b_{i+45}b_{i+37} \oplus b_{i+33}b_{i+28}b_{i+21}b_{i+15}b_{i+9} \\ &\oplus b_{i+52}b_{i+45}b_{i+37}b_{i+33}b_{i+28}b_{i+21}. \end{aligned}$$

At each step, the output bit is computed from 8 bits of the NFSR and 4 bits of the LFSR as

$$z_i = h(s_{i+3}, s_{i+25}, s_{i+46}, s_{i+64}, b_{i+63}) \oplus \bigoplus_{k \in \mathcal{A}} b_{i+k},$$

where h is a boolean function of degree 3 and $\mathcal{A} = \{1, 2, 4, 10, 31, 43, 56\}$.

The initialization of `Grain v1` is described in Algorithm 4. First, the 80-bit key is loaded into the NFSR and the 64-bit IV into the 64 first bits of the LFSR. Remaining bits of the LFSR are set to 1. Then, the internal state is clocked 160 times with a re-injection of the output bits.

1.4.2 Zhang *et al.* attack

At Eurocrypt'18, Zhang *et al.* presented a fast near collisions attack against `Grain v1`, claiming a time complexity around $2^{75.7}$ ticks. Let x_i be $\bigoplus_{k \in \mathcal{A}} b_{i+k}$ so that $z_i = x_i \oplus h(s_{i+3}, s_{i+25}, s_{i+46}, s_{i+64}, b_{i+63})$. For each $0 \leq i < j \leq 19$, they applied the refined self-contained method together with the amplified phase to (z_i, z_j) and obtained a subset of the possible pre-images $X_{i,j}$ containing the right value with probability p. As x_i can be directly computed from the value keystream bit z_i and $s_{i+3}, s_{i+25}, s_{i+46}, s_{i+64}$ and b_{i+63}, they did not store x_i nor x_j in the $X_{i,j}$. As a result they claim that $X_{i,j}$ contains on average 848 elements (over 2^{10}) and $p = 89.64\%$.

The next step of the attack is to merge all those 190 sets to get a set X containing only values leading to the rightful first 20 bits of keystream. They claim that X would contain on average $2^{6.67}$ elements and the probability for the right value of the internal state to be in X would be around $p^{343} = (0.8964)^{343} = 2^{-54.09}$. The time complexity of the whole attack is then proportional to $|X| \times p^{-343}$.

Algorithm 4 The initialization procedure and keystream generation of `Grain v1`

KEY a 80-bit key *Parameters*
IV a 64-bit initial value

Load KEY in *Initialization*
Load IV in
for $i = 0$ to 15 **do**
    $s_i \leftarrow 1$
end for
for $i = 0$ to 159 **do**
    compute $o = h^*(s_{79}, s_{54}, s_{33}, s_{15}, b_{16}, b_{78}, b_{77}, b_{75}, b_{69}, b_{48}, b_{36}, b_{23})$
    clock the LFSR and the NFSR with the following respective feedback bits:
    $f(s_{79}, s_{66}, s_{56}, s_{41}, s_{28}, s_{17}) \oplus o$
    $g(b_{16}, b_{17}, b_{19}, b_{27}, b_{34}, b_{42}, b_{46}, b_{51}, b_{58}, b_{64}, b_{65}, b_{70}, b_{79}) \oplus s_{79} \oplus o$
end for

for i **do** *Keystream generation*
    compute $z_i = h^*(s_{79}, s_{54}, s_{33}, s_{15}, b_{16}, b_{78}, b_{77}, b_{75}, b_{69}, b_{48}, b_{36}, b_{23})$
    clock the LFSR and the NFSR with the following respective feedback bits:
    $f(s_{79}, s_{66}, s_{56}, s_{41}, s_{28}, s_{17})$
    $g(b_{16}, b_{17}, b_{19}, b_{27}, b_{34}, b_{42}, b_{46}, b_{51}, b_{58}, b_{64}, b_{65}, b_{70}, b_{79}) \oplus s_{79}$
end for

Output $\{z_i\}_i$ *the keystream*

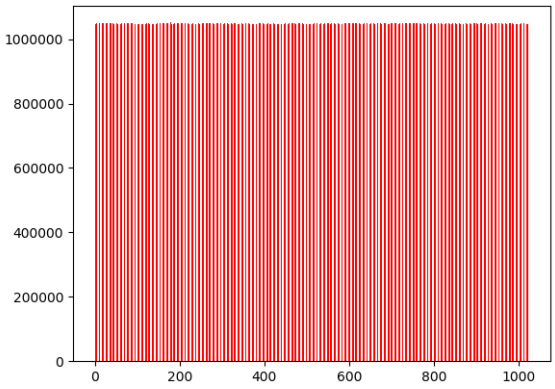

Figure 1.4 – Histogram plotting how many times each of the 2^{10} possible two successive input of the function h were reached after 2^{30} `Grain v1` initialization using random key and iv values.

1.4.3 Complexity correction

Experimental result on initialization. As for `A5/1`, we checked whether there are bias in the initialization phase of `Grain v1` which could explain the probability given by Zhang *et al.* in [ZXM18]. We have drawn uniformly at random 2^{30} keys and IVs, and for each them ran the initialization phase. We then looked at the 10 bits going through the function h to generate the 2 first keystream bits. As expected, we did not notice any bias in the distribution (see Figure 1.4).

Theoretical analysis. As explained Section 1.2.2, assuming all the 2^{160} possible internal states are equiprobable, and since the output function is balanced, the probability p for the right value to belong to $X_{i,j}$ has to be corrected to $848/1024 = 82.81\%$. In particular, the final probability becomes $(0.8281)^{343} = 2^{-93.32}$. But actually the whole attack is flawed. Indeed, since we merge 190 *independent* sets $X_{i,j}$ the probability for the right value to belong to X is p^{190} and not p^{343}. The mistake made by Zhang *et al.* lies in the merging process. First they construct the set $X_{0,1,2}$ by merging $X_{0,1}$, $X_{0,2}$ and $X_{1,2}$ claiming a probability of p^3 which is correct. Then they construct the set $X_{1,2,3}$ by merging $X_{1,2}$, $X_{1,3}$ and $X_{2,3}$ claiming a probability of p^3 which is also correct. But then they construct the set $X_{0,1,2,3}$ by merging $X_{0,1,2}$, $X_{1,2,3}$ and $X_{0,3}$ and claim a probability of $p^3 \times p^3 \times p = p^7$. This is wrong because $X_{0,1,2,3}$ is actually the merge of only 6 sets, $X_{1,2}$ being used twice, and the right probability is p^6. Thus the corrected probability for the right value to belong to X is $p^{190} = (0.8281)^{190} = 2^{-51.7}$. Surprisingly this is not so

far from the $2^{-54.09}$ claimed by Zhang *et al.*. But the 20 keystream bits $z_0 \dots z_{19}$ depend on 118 (linear combinations of) state bits (see Table 5 in [ZXM18]). Thus, according to Theorem 1.1, to reach such probability the set X has to contain $2^{118-20} \times 2^{-51.7} = 2^{47.3}$ elements and not only $2^{6.67}$. As a consequence, the overall complexity of the attack is increased by a factor $2^{47.3+51.7-6.67-54.09} = 2^{37.24}$, making it slower than an exhaustive search.

Experimental result on p. Finally, as for `A5/1` we provide the following experiment to support our claim regarding the correct complexity of the attack presented in [ZXM18].

1. Randomly generate a 3-bit word $\mathbf{k}_s$
2. Run the refined self-contained method to obtain a set X of size $2^{14.2}$ which should contain the secret which generated $\mathbf{k}_s$ with probability $(0.8964)^3 = 0.7203$ according to Zhang *et al.*.
3. Do N times:
   (a) Randomly generate a key and an IV and run the initialization process.
   (b) Check whether the first three keystream bits from the current internal state match $\mathbf{k}_s$. If not repeat the previous step.
   (c) Check whether the value of the internal state part belongs to X.
4. Check whether the experimental probability matches the expected one.

We ran this experiment[2] 10 times with $N = 2^{26}$ and found the experimental probability to be very close to $(0.8281)^3 = 0.5679$, confirming the inaccuracy of the probability 0.7203 claimed by Zhang *et al.*. To ensure that the initialization process does not introduce and/or remove any biases, we repeated 10 times the experiment for random states during the keystream generation phase too. In more details, we sampled $R < 2^{10}$ and before generating the three keystream bits, we updated by R rounds the internal state. As expected, none of those experiments supported Zhang *et al.* claimed probability.

2. As for `A5/1`, we simply took for X the set of size $2^{14.2}$ composed of the values reached the most during the experiment.

1.5 Conclusion

In the works about FNCA [ZXM18] and [Zha19], authors seem to have experimentally verified their claimed probabilities. They wrote they *have done a large number of experiments ... and almost all the experimental results conform to our theoretical predictions.* This statement is quite unlikely. Indeed, it was enough to add a loop in the publicly available C codes of their works to observe the deviation in the claimed probabilities. Furthermore, as shown in this Chapter, we presented simple algorithms to disprove their claim complexities. In more details, we found an error in the theory proposed and by correcting it and running some algorithmic experiments, we confirm our hypothesis that the complexities of the Fast Near Collision Attacks are seriously underestimated in the papers that introduce them.

To avoid this type of problem, we need to develop more tools to check and verify attacks or to find a completely new theory to explain the security of symmetric primitives like the decorrelation theory at the time of its publication. Nevertheless, it stays important and crucial to evaluate and correct manually previous scientific works.

Chapter 2

Efficient Search for Optimal Diffusion Layers of Generalized Feistel Networks

2.1 Introduction

The Feistel network is one of the main generic designs for building modern block ciphers. It was initially proposed in the data encryption standard DES [DES77], and is still used in more recent ciphers such as Twofish [Sch+98], Camellia [Aok+00] or SIMON [Bea+13]. The idea behind this construction is to split the plaintext into two halves x_0, x_1, and build the round function which sends (x_0, x_1) to $(x_1, x_0 \oplus F_i(x_1))$, where F_i is a non-linear function for the i-th round. One of the main advantage of this construction is that F_i does not need to be invertible, and thus it allows to transform a pseudorandom *function* (PRF) into a pseudorandom *permutation* (PRP). Moreover, there are theoretical arguments suggesting that it is a good method to construct block ciphers, as Luby and Rackoff proved in 1988 [LR88] that if each F_i is a pseudorandom function and all three are independent, then 3 rounds of the Feistel construction are enough to get a block cipher which is indistinguishable from a random permutation under the Chosen Plaintext Attack (CPA) model, and 4 rounds with 4 independent functions are enough in the Chosen Ciphertext Attack (CCA) model. This was later improved by Pieprzyk in 1990 [Pie90] : if one takes f as a pseudorandom function, 4 rounds of Feistel with $F_i = f$ for $i = 1, 2, 3$ and $F_4 = f^2$ are sufficient to obtain a block cipher that is indistinguishable from a random permutation in the CPA model. In 1989 at CRYPTO, Zheng *et al.* [ZMI89a] proposed some generalizations of the Feistel construction. Especially, they defined the *Type-2 Feistel*[1] construction, which splits the message into $2k$ blocks and uses a round

1. Note that some papers use the term Type-2 Generalized Feistel to denote this construction

function of the form

$$(x_0, \ldots, x_{2k-1}) \mapsto (x_{2k-1}, x_0 \oplus F_{i,0}(x_1), x_1, x_2 \oplus F_{i,1}(x_3), x_3, \ldots, x_{2k-2} \oplus F_{i,k-1}(x_{2k-1})),$$

where each $F_{i,j}$ is a pseudorandom function for the i-th round. This is essentially a parallel application of k Feistels followed by a cyclic shift of the blocks. They also showed that when all $F_{i,j}$ are pseudorandom functions, then $2k + 1$ rounds of such a construction provide a block cipher that is indistinguishable from a random permutation. Moreover, the Type-2 construction is inherently easier to compute in parallel, and the corresponding decryption function is basically the same except that the functions $F_{i,j}$ are applied in reverse order, i.e. for r rounds, the first round of decryption uses the functions $F_{r,j}$. Both of these properties make this construction very efficient in practice, both on hardware and software, e.g. TWINE [Suz+12] and Simpira [GM16a]. All of these arguments lead to some block ciphers based on this Type-2 Feistel construction, such as HIGHT [Hon+06] and CLEFIA [Shi+07].

At ASIACRYPT'96, Nyberg [Nyb96] studied a variant of the Type-2 Feistel construction using a different permutation than the cyclic shift, called Generalized Feistel Network. Such a construction was used to design block ciphers such as TWINE [Suz+12] and Piccolo [Shi+11]. However, Nyberg only focused on one specific permutation. Suzaki and Minematsu thus studied at FSE'10 [SM10] a more general case where the cyclic shift is replaced by any other permutation of the blocks. Their work was focused on finding permutations with the lowest *diffusion round*. The diffusion round is close to the concept of *diffusion* introduced by Shannon in 1949 [Sha49]. Essentially, a block cipher has *full diffusion* if every bit of the ciphertext depends on every bit of the plaintext. In the context of Generalized Feistel Network (GFN), [SM10] defined the diffusion round as the minimal number of rounds such that every *block* of the ciphertext depends on every *block* of the plaintext. Focusing on blocks instead of bits allows them to get rid of the precise specification of the functions $F_{i,j}$ as well as the exact size of the blocks, thus giving structural results. Especially, they tied the diffusion round of a given GFN to its resistance against Impossible Differential distinguishers [BBS99], proving that if a GFN has a diffusion round of DR, then it needs strictly more than $2DR + 1$ rounds to avoid any Impossible Differential distinguisher. Along with a lower bound on the diffusion round of a GFN of $2k$ blocks, they gave optimal permutations (w.r.t the diffusion round) for $2 \leq 2k \leq 16$. It is worthy to note that such an optimal permutation was then used to

design block ciphers such as TWINE [Suz+12]. At FSE'19, Cauchois *et al.* went further and gave optimal permutations for $18 \leq 2k \leq 26$, as well as good candidates for $2k = 32$ (which was already found in [SM10]), as well as for $2k = 64$ and 128 using a sophisticated technique that they called *Collision-free exhaustive search*. Note that these permutations are even-odd, i.e. the image of an even number is an odd number. On a side note, relaxing the condition that the permutation is the same in each round make the problem easier and in [Kal+17], Kales *et al.* give such a construction for any number of blocks.

Our contribution. In this chapter, we focus on even-odd permutations and we complete the work on the 10-year-old problem (introduced by [SM10]) of finding optimal even-odd permutations for 32 blocks, as well as finding optimal even-odd permutations for 28, 30 and 36 blocks which were not given in the previous literature. To do so, we propose a new characterization of a permutation reaching full diffusion after a given number of rounds. Using this characterization, we are able to create a very efficient algorithm, which on the previously mentioned cases yields all the permutations that achieve full diffusion in 9 rounds. Note that our algorithm essentially uses branch-and-bound techniques, and thus it is hard to evaluate the exact complexity. However, the size of the search space goes from 2^{43} for $2k = 28$ up to 2^{75} for $2k = 42$, but we were able to treat each of these cases in less than one hour for each value of k when using 72 threads. Moreover, this characterization has a very efficient implementation which allowed us to re-find all optimal even-odd permutations for up to 26 blocks with a basic exhaustive search in a few hours, showing that for these cases, there is no need for sophisticated techniques as in [CGT19]. Furthermore, for 34, 38, 40 and 42 blocks, we prove with this method that there is no even-odd permutation with a diffusion round of 9, which is the lower bound on the diffusion round for these sizes given in [SM10]. We were also able to find even-odd permutations with a diffusion round of 10 for $2k = 34$ (which is thus optimal), as well as even-odd permutations with diffusion round 11 for $2k = 38, 40, 42$. Finally, we evaluate the security of our constructed permutations against impossible differentials and differentials (by computing the minimum number of active S-boxes). In particular, for the 32 blocks case, and the impossible differentials, all our permutations have a one-round shorter longest impossible differential distinguisher compared to what was proposed by [CGT19], which brings it down to 17 rounds.

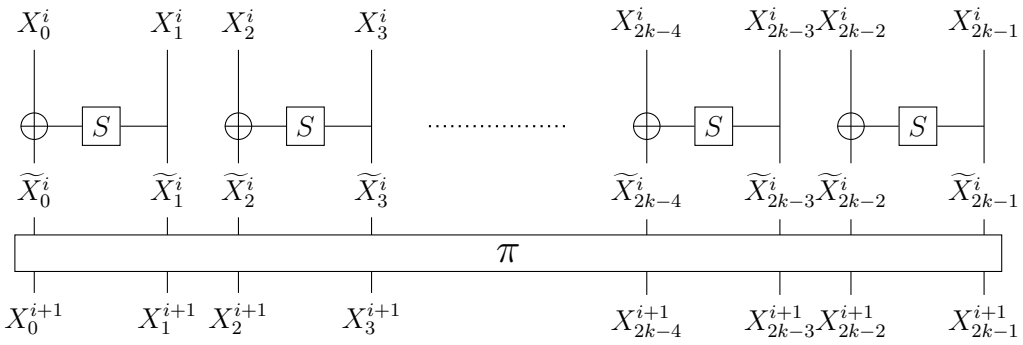

Figure 2.1 – Generalized Feistel Network

2.2 Preliminaries

2.2.1 Generalized Feistel Networks (GFN)

Zheng *et al.* [ZMI89a] introduced Type-2 Feistels as a generalization of the original Feistel construction. Given an even number $2k$ of blocks $(X_0, \ldots, X_{2k-1})$, it first applies the Feistel construction on the pairs of blocks which yields $(X_0 \oplus S_0(X_1),\ X_1, \ldots,\ X_{2k-2} \oplus S_{k-1}(X_{2k-1}), X_{2k-1})$. The blocks are then cyclically right shifted to obtain the result. Later, it was proposed to use another permutation than the cyclic shift in [Nyb96], leading to Generalized Feistel Networks.

Definition 2.1. *Let $2k$ be an even number, n, r be positive integers, and $\{F_{i,j}\}_{i\in\{1,\ldots,r\},j\in\{0,\ldots,k-1\}}$ be a set of cryptographic keyed functions from $\mathbb{F}_2^n$ to $\mathbb{F}_2^n$. Let π be a permutation over $2k$ elements. A Generalized Feistel Network (GFN) is a block cipher built as $\mathcal{R}_r \circ \cdots \circ \mathcal{R}_1$, where $\mathcal{R}_i$ is the round function*

$$\mathcal{R}_i :\ (X_0, \ldots, X_{2k-1}) \to \pi(X_0 \oplus F_{i,0}(X_1),\ X_1, \ldots,\ X_{2k-2} \oplus F_{i,k-1}(X_{2k-1}), X_{2k-1})$$

Note that for this chapter, neither the exact definition of the keyed functions $F_{i,j}$ nor their sizes are relevant. We can thus consider all of them as an arbitrary S-box S, leading to the framework depicted in Figure 2.1[2]. As the only variable parameters are thus k and π, we denote by GFN_π^k a GFN with $2k$ blocks that uses the permutation π.

2. In practice, one should carefully study the primitive if the same F-function is used, e.g. [GM16b]

2.2.2 Diffusion Round

We use the notations depicted in Figure 2.1. The input variables of the i-th round of a GFN are denoted by $(X_0^i, X_1^i, \ldots, X_{2k-1}^i)$. We also denote by $(\widetilde{X}_0^i, \widetilde{X}_1^i, \ldots, \widetilde{X}_{2k-1}^i)$ the variables which are at the input of the permutation π, i.e.

$$(X_0^{i+1}, X_1^{i+1}, \ldots, X_{2k-1}^{i+1}) = \pi(\widetilde{X}_0^i, \widetilde{X}_1^i, \ldots, \widetilde{X}_{2k-1}^i)$$

It is easy to see from Definition 2.1 that $X_{\pi(0)}^1$ depends on X_0^0 and X_1^0. More generally, any block $\widetilde{X}_j^r$ depends on a certain number of blocks from the round 0, i.e. computing $\widetilde{X}_j^r$ requires some blocks $\{X_{j_0}^0, \ldots, X_{j_l}^0\}$. Note that this does not depend on the size of the functions $F_{i,j}$ in the GFN. As in [SM10], we say in that case that any of these $X_{j_i}^0$ *diffuses* to $\widetilde{X}_j^r$, and we focus our study on the number of rounds needed to reach *full diffusion*.

Definition 2.2. *Let π be a permutation over $2k$ elements. We say that a block X_j^0 fully diffuses after r rounds if for all $i \in \{0, \ldots, 2k-1\}$, X_j^0 diffuses to $\widetilde{X}_i^r$. We say that π reaches full diffusion after r rounds if for all $j \in \{0, \ldots, 2k-1\}$, X_j^0 fully diffuses after r rounds. The smallest r that verifies this property for the block X_i^0 is called the diffusion round of the block X_i^0.*

Note that we need to study both the diffusion over the encryption *and* the decryption process. Indeed, there is no guarantee that an encryption function with good diffusion also keeps this property for its inverse. Since we have $(GFN_\pi^k)^{-1} = GFN_{\pi^{-1}}^k$, we need to study both the diffusion of π and π^{-1}. Naturally, we would like both π and π^{-1} to fully diffuse as quickly as possible, which leads to the following definition.

Definition 2.3. *Let π be a permutation over $2k$ elements. Denote by $DR_i(\pi)$ the minimum number of rounds r such that X_i^0 fully diffuses after r rounds in GFN_π^k.*

The diffusion round of a permutation π is:

$$DR_{max}(\pi) = \max_{0 \leq i \leq 2k-1} \left\{ DR_i(\pi), DR_i(\pi^{-1}) \right\} \tag{2.1}$$

This definition gives the same importance to the total diffusion of both π and π^{-1}. Definition 2.3 defines a natural partial order on the permutations: a permutation π_1 is better (at diffusing) than a permutation π_2 if $DR_{max}(\pi_1) \leq DR_{max}(\pi_2)$. Searching the best permutations (for the diffusion) directly can be difficult. As a result the methodology we adopt in this work is to search for permutations that diffuse totally in the forward direction and then check if their respective inverse also diffuses totally.

2.2.3 Even-odd Permutations

A naive way to search for optimal permutation would be to simply go through all of them and check the diffusion one permutation by one. However, there are $(2k)!$ permutations, which quickly grows beyond practical means. For example with $2k = 32$, approximately 2^{117} permutations should be checked. To reduce the number of permutations that will be tested, we will restrict ourselves to a specific class of permutations and give an equivalence relation which further reduces the number of permutations to be considered.

In [SM10], Suzaki and Minematsu did an exhaustive search for $1 \leq k \leq 8$, and made the observation that every optimal permutation (for such k) mapped even-number input blocks to odd-number output blocks and vice versa. We call such permutations even-odd. In the rest of this chapter, we will use the following notation for even-odd permutations. An even-odd permutation π of size $2k$ will be denoted by the pair of permutations (p, q) of size k verifying $\forall i \in [0, k-1],\ \pi(2i) = 2 \cdot p(i) + 1$ and $\pi(2i+1) = 2 \cdot q(i)$. The search space is now reduced to $(k!)^2$ permutations.

According to this, [SM10] gives the following lower-bound on the diffusion round of even-odd permutations (p, q).

Proposition 2.1. *Let $\mathcal{F}_i$ be the Fibonacci sequence, i.e. $\mathcal{F}_0 = 0, \mathcal{F}_1 = 1$ and $\mathcal{F}_i = \mathcal{F}_{i-1} + \mathcal{F}_{i-2}, i \geq 2$. Let $\pi = (p, q)$ be an even-odd permutation over $2k$ elements, and i be the smallest integer such that $\mathcal{F}_i \geq k$. Then $DR_{max}(\pi) \geq i + 1$.*

For a given permutation π, if the inequality is tight, we say that π is tight. A proof of this proposition already exists in both [SM10] and [CGT19]. According to our results, we will give another proof of this proposition in Section 2.3. We will also show in Section 2.3 that this bound is tight for the cases $2k = 28, 30, 32, 36$ and strict for $2k = 34, 38, 40, 42$.

2.2.4 Equivalence Classes of Even-odd Permutations

To further reduce the size of the search space, as in [CGT19], we use some equivalence classes, given by the following definition.

Definition 2.4. *Let π and π' be two even-odd permutations over $2k$ elements. We say that π and π' are equivalent if there exists a permutation φ over $2k$ elements such that*

$$\pi' = \varphi \circ \pi \circ \varphi^{-1} \text{ and } \forall i \in [0, k-1], \varphi(2i+1) = \varphi(2i) + 1.$$

From [CGT19], we can then give a set of permutations $\mathbb{P}_k$ such that for any equivalence class, there exists at least one $\pi \in \mathbb{P}_k$ which belongs to this class. This effectively gives us a set of class representatives (in which a few of them are redundant), and this set can be built from the following proposition, proven in [CGT19]. Recall that any permutation can be decomposed into a composition of cycles. We call *cycle structure* the unordered set of the length of these cycles, for example the permutation

$$(0\ 1\ 2\ 3)(4\ 5)(6\ 7)(8)$$

has a cycle structure of $\{4, 2, 2, 1\}$.

Proposition 2.2. *Let $\mathbb{P}_k$ be a set of even-odd permutations $\pi = (p, q)$ over $2k$ elements constructed as follows. For each possible cycle structure c of a permutation over k elements, pick one permutation p which has a cycle structure equal to c. Then, for every permutation q over k elements, add (p, q) in the set $\mathbb{P}_k$. By doing so, $\mathbb{P}_k$ contains at least one representative of each equivalence class induced by Definition 2.4. Moreover, $\mathbb{P}_k$ contains exactly $\mathcal{N}_k.k!$ elements, where $\mathcal{N}_k$ is the number of partitions of the integer k.*

This allows us to only consider $\mathcal{N}_k.k!$ permutations instead of $(k!)^2$. This is a significant improvement, as for example with $k = 16$, there are only $231 \times 16! \simeq 2^{52}$ permutations to go through, instead of $(16!)^2 \simeq 2^{88}$. However when k grows, it is still too big a number to try an exhaustive search. As such, we propose in Section 2.4 an efficient search algorithm to find all optimal even-odd permutations for a given k, without needing to do an exhaustive search.

2.3 Characterization of Full Diffusion

In this section, we will explain our strategy to search for a tight even-odd permutation, that is, a permutation with a diffusion round reaching the Fibonacci bound given in Proposition 2.1. We will first give an algebraic characterization for a permutation to have full diffusion, then give an algorithm to exploit this characterization and quickly search all such permutations. Note that here we only focus on the diffusion round of the permutation when considering encryption. That is, for a given permutation π, we focus only on $DR(\pi) = \max_{0 \leq i \leq 2k-1} \{DR_i(\pi)\}$. Then, once we found a permutation reaching the Fibonacci bound, we can easily check if π^{-1} also reaches this bound, and if that is the case, we found a tight permutation.

We describe here the main tools we used to design our search algorithm. Note that for two permutations p, q, we denote the composition $p \circ q$ by pq for better reading. We first begin by giving the following proposition.

Proposition 2.3. *Let $\pi = (p, q)$ be an even-odd permutation over $2k$ elements. Then π achieves full diffusion after r rounds if and only if each block X_j^0 is diffused to at least one block of each pair at the input of the $(r-1)$-th round, i.e. diffused to either $X_{2j'}^{r-1}$ or $X_{2j'+1}^{r-1}$ for each $j' \in \{0, \ldots, k-1\}$.*

Proof. Suppose that a given block X_i^0 has been fully diffused, i.e. to every block $\widetilde{X}_{2j}^r$ and $\widetilde{X}_{2j+1}^r, j \in \{0, \ldots, k-1\}$. Then X_i^0 must have diffused to at least X_{2j+1}^r for every j, as it is the only way to reach $\widetilde{X}_{2j+1}^r$. Thus, X_i^0 must have diffused to $\widetilde{X}_{2j'}^{r-1}$ with $j' = p^{-1}(j)$, which means that it has diffused to either $X_{2j'}^{r-1}$ or $X_{2j'+1}^{r-1}$.

On the other hand, suppose that a given block X_i^0 has diffused to an even block X_{2j}^{r-1}, then X_i^0 will be diffused to only $\widetilde{X}_{2j}^{r-1}$. If X_i^0 has diffused to an odd block X_{2j+1}^{r-1}, it will be diffused to both $\widetilde{X}_{2j}^{r-1}$ and $\widetilde{X}_{2j+1}^{r-1}$. In both cases, it will be diffused to $\widetilde{X}_{2j}^{r-1}$, then to $X_{2j'+1}^r$ with $j' = p(j)$, and finally to both $\widetilde{X}_{2j'}^r$ and $\widetilde{X}_{2j'+1}^r$. Thus, if for all $j \in \{0, \ldots, k-1\}$, X_i^0 is diffused to any block of the j-th pair at the input of the $(r-1)$-th round, it will be diffused to every block $\widetilde{X}_{2p(j)}^r$ and $\widetilde{X}_{2p(j)+1}^r$, and since p is a permutation, this means that we have full diffusion for X_i^0. □

Corollary 2.1. *Let $\pi = (p, q)$ be an even-odd permutation over $2k$ elements. Then π achieves full diffusion after r rounds if and only if each even block $X_{2j}^0, j \in \{0, \ldots, k-1\}$ diffuses to every even block $\widetilde{X}_{2j'}^{r-1}, j' \in \{0, \ldots, k-1\}$.*

Proof. For the proof of the previous theorem, we can easily see that a block X_j^0 diffuses to either $X_{2j'}^{r-1}$ or $X_{2j'+1}^{r-1}$ if and only if X_j^0 diffuses to $\widetilde{X}_{2j'}^{r-1}$. Moreover, we can easily see that if X_{2j}^0 is fully diffused, so is X_{2j+1}^0. Indeed, X_{2j}^0 being fully diffused is the same as $\widetilde{X}_{2j}^0$ being fully diffused, and X_{2j+1}^0 is always diffused to $\widetilde{X}_{2j}^0$. □

Thus we only need to focus on the diffusion of each block X_{2j}^0 to each block $\widetilde{X}_{2j}^{r-1}$. Now we can take a look a what would happen in an ideal scenario. Assume that we are studying the diffusion of a block X_{2j}^0. Then X_{2j}^0 is diffused to $\widetilde{X}_{2j_0^1}^0$ with $j_0^1 = j$. It is then diffused to both $\widetilde{X}_{2j_0^2}^1$ and $\widetilde{X}_{2j_0^2+1}^1$, with $j_0^2 = p(j_0^1)$. Then again :

- $\widetilde{X}_{2j_0^2}^1$ is diffused to both $\widetilde{X}_{2j_0^3}^2$ and $\widetilde{X}_{2j_0^3+1}^2$, with $j_0^3 = p(j_0^2)$.
- $\widetilde{X}_{2j_0^2+1}^1$ is diffused to $\widetilde{X}_{2j_1^3}^2$ with $j_1^3 = q(j_0^2)$

Assuming an ideal scenario, we would have $j_0^3 \neq j_1^3$, i.e. X_{2j}^0 has diffused to two different blocks after 4 rounds (minus the application of π on the fourth round). We can then keep going and get a series of j_ℓ^i which gives us the blocks on which X_{2j}^0 has diffused after $i+1$ rounds minus the last application of π, always assuming that we never have $j_\ell^i = j_{\ell'}^i$ for $\ell \neq \ell'$. The propagation for up to 7 rounds is given in Figure 2.2.

However, we cannot have $j_\ell^i \neq j_{\ell'}^i$ with $\ell \neq \ell'$ forever. Indeed, since we only have k blocks, we are bound at some point to have $j_\ell^i = j_{\ell'}^i$ and $\ell \neq \ell'$. However, we can easily compute the actual value of each j_ℓ^i. Indeed, if we take for example j_6^6 in Figure 2.2, then we know that

$$j_6^6 = (qppqp)(j_0^1) = (qppqp)(j).$$

Denote by $\mathbb{J}_j^i$ the set of equations obtained by expressing every j_ℓ^i that way. For example, we would have

$$\begin{aligned}\mathbb{J}_j^6 = \{&(ppppp)(j),\\ &(qpppp)(j),\\ &(pqppp)(j),\\ &(ppqpp)(j),\\ &(qpqpp)(j),\\ &(pppqp)(j),\\ &(qppqp)(j),\\ &(pqpqp)(j)\}\end{aligned}$$

According to this, we can give a generic way to compute $\mathbb{J}_j^i$. We start with $\mathbb{J}_j^1 = \{j\}$ and $\mathbb{J}_j^2 = \{p(j)\}$. To build $\mathbb{J}_j^i$ from $\mathbb{J}_j^{i-1}$, we begin by adding $p(x)$ to $\mathbb{J}_j^i$ for every term x in $\mathbb{J}_j^{i-1}$. Then, for every term x in $\mathbb{J}_j^{i-1}$ such that x can be written as $x = p(y)$ for some $y \in \mathbb{J}_j^{i-2}$, we also add $q(x)$ to $\mathbb{J}_j^i$.

We can justify this construction as follows. Suppose that a given j' belongs to $\mathbb{J}_j^{i-2}$ because X_{2j}^0 diffuses to $\widetilde{X}_{2j'}^{i-2}$. Then X_{2j}^0 diffuses to both $\widetilde{X}_{2j''}^{i-1}$ and $\widetilde{X}_{2j''+1}^{i-1}$ with $j'' = p(j')$. Thus for the next round, X_{2j}^0 will diffuse to both $X_{\widetilde{2j+1}}^i$ and $X_{\widetilde{2j'}}^i$, with $\widetilde{j} = p(j'')$ and $\widetilde{j'} = q(j'')$.

On the other hand, suppose that j' belongs to $\mathbb{J}_j^{i-2}$ because X_{2j}^0 diffuses to $\widetilde{X}_{2j'+1}^{i-2}$. In that case, X_{2j}^0 will only diffuse to $\widetilde{X}_{2j''}^{i-1}$ with $j'' = q(j')$. For the next round, X_{2j}^0 only diffuses to $X_{\widetilde{2j+1}}^i$ with $\widetilde{j} = p(j'')$.

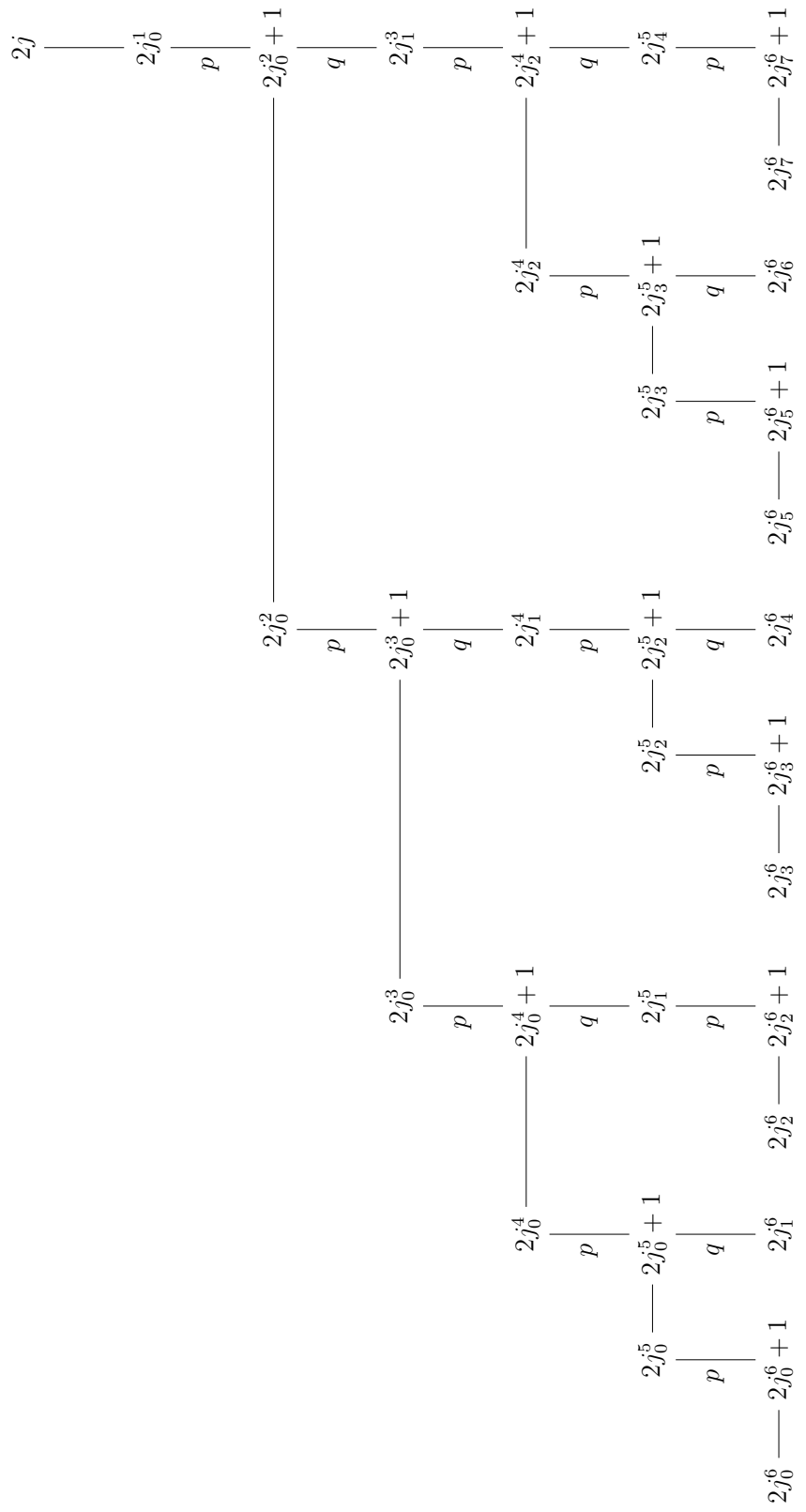

Figure 2.2 – Propagation tree for 7 rounds (minus the last application of π)

Thus in both cases, we need to have $\widetilde{j} = p(j'')$, but we only require $\widetilde{j'} = q(j'')$ in the first case, which corresponds exactly to the case where the previous term started with a composition by p.

Note that from this construction, we can deduce the following proposition.

Proposition 2.4. *The size of $\mathbb{J}_j^i$ is exactly $\mathcal{F}_i$ where $\mathcal{F}_i$ is the i-th term of the Fibonacci sequence.*

Proof. We can prove this by induction. Both $\mathbb{J}_j^1$ and $\mathbb{J}_j^2$ are of size 1, which corresponds to $\mathcal{F}_1$ and $\mathcal{F}_2$. We first add an element $p(x)$ to $\mathbb{J}_j^i$ for every $x \in \mathbb{J}_j^{i-1}$, thus $\mathcal{F}_{i-1}$ elements. Then, for every x in $\mathbb{J}_j^{i-1}$ such that $x = p(y)$ with $y \in \mathbb{J}_j^{i-2}$, we add $q(x)$ to $\mathbb{J}_j^i$. However, according to our construction, $\mathbb{J}_j^{i-1}$ contains such an element $x = p(y)$ for *every* term $y \in \mathbb{J}_j^{i-2}$. Thus, there are $\mathcal{F}_{i-2}$ such terms. In the end, $\mathbb{J}_j^i$ contains $\mathcal{F}_{i-1} + \mathcal{F}_{i-2} = \mathcal{F}_i$ elements, which concludes the induction. □

We can now use those sets $\mathbb{J}_j^i$ to fully characterize the fact that a block fully diffuses when using a given permutation.

Theorem 2.1. *Let $\mathbb{J}_j^{r-1}$ be the set of equations as defined above. Then for a given permutation $\pi = (p, q)$ over $2k$ blocks, X_{2j}^0 is fully diffused after r rounds if and only if $\mathbb{J}_j^{r-1}$ contains every number in $\{0, \ldots, k-1\}$ at least once.*

Proof. As $\mathbb{J}_j^{r-1} = \{j_0^{r-1}, \ldots, j_{\ell_{r-1}}^{r-1}\}$ is defined, it basically represents that X_{2j}^0 diffuses to every $\widetilde{X}_{2j_\ell^i}^i$. Thus, if $\mathbb{J}_j^i$ contains every number in $\{0, \ldots, k-1\}$ at least once, it exactly means that X_{2j}^0 diffuses to each block $\widetilde{X}_{2j'}^i, j' \in \{0, \ldots, k-1\}$. According to Corollary 2.1, this means that X_{2j}^0 achieves full diffusion after $i+1$ rounds. □

We can then easily deduce the following corollary.

Corollary 2.2. *Let $\pi = (p, q)$ be a permutation over $2k$ elements. Then we have $DR(\pi) = i+1$ if and only if i is the smallest integer such that for every $j \in \{0, \ldots, k-1\}$, $\mathbb{J}_j^i$ contains every number in $\{0, \ldots, k-1\}$ at least once.*

This gives us another proof for the Fibonacci bound given in Proposition 2.1. Indeed, for $\mathbb{J}_j^i$ to contain every number in $\{0, \ldots, k-1\}$ at least once, $\mathbb{J}_j^i$ must contain at least k terms. Thus, and since the size of $\mathbb{J}_j^i$ does not depend on j, the minimal number of rounds needed to have full diffusion for every block must be such that $\left|\mathbb{J}_j^i\right| = \mathcal{F}_i \geq k$. According to the previous corollary, if i is the smallest integer such that $\mathcal{F}_i \geq k$, this exactly means that $DR_{max}(\pi) \geq i + 1$

Note that from the construction of any $\mathbb{J}_j^i$, each term starts with a composition by p. Since p is a permutation, and we want full diffusion for every blocks, we can remove this first p from every term to get a smaller representation. Essentially, this means that we are considering the diffusion of the block $p^{-1}(j)$, but we will still write $\mathbb{J}_j^i$. As such, $\mathbb{J}_j^6$ for example is thus rewritten as

$$\begin{aligned}\mathbb{J}_j^6 = \{&(p^4)(j),\\ &(qp^3)(j),\\ &(pqp^2)(j),\\ &(p^2qp)(j),\\ &(qpqp)(j),\\ &(p^3q)(j),\\ &(qp^2q)(j),\\ &(pqpq)(j)\}\end{aligned}$$

To illustrate the previous characterization, we introduce what we call the *diffusion table (of rank i)* of an even-odd permutation (p, q) of size $2k$. The columns are indexed by the numbers from 0 to $k-1$ and the row are indexed by the products of p and q used to generate all sets $\mathbb{J}_j^i$. Each cell of the table is the value obtained by applying the permutation indexing the row to the value indexing the column of the cell. For example, the cell indexed by p^i and 0 contains $p^i(0)$. This provides a clear visualization of our characterization, as the j-th column is exactly $\mathbb{J}_j^i$.

Thus, we can easily illustrate Corollary 2.2 by verifying that every column of this table contains every possible values. We thus add one more row at the end of diffusion table called *diff* which contains the number of different values in a column. By construction, this is exactly the number of elements of $\mathbb{J}_j^i$ where j is the index of the column. In tables constructed as described, the full diffusion of a permutation corresponds to a *diff* row containing only the value k.

For example, we give in Table 2.1 the diffusion tables for the cyclical shift (i.e. $p = (7, 0, 1, 2, 3, 4, 5, 6)$ and $q = (0, 1, 2, 3, 4, 5, 6, 7)$) and one of the optimal permutation proposed by [CGT19] (i.e. $p = (6, 3, 7, 1, 0, 2, 4, 5)$ and $q = (3, 5, 1, 6, 4, 0, 2, 7)$) for $k = 8$ and $i = 7$, thus the optimal permutation clearly have a diffusion round of 8.

Finally, we can reformulate the problem of finding optimal even-odd permutations with

x	0	1	2	3	4	5	6	7
p^5	3	4	5	6	7	0	1	2
p^4q	4	5	6	7	0	1	2	3
p^3qp	4	5	6	7	0	1	2	3
p^2qp^2	4	5	6	7	0	1	2	3
pqp^3	4	5	6	7	0	1	2	3
qp^4	4	5	6	7	0	1	2	3
p^2qpq	5	6	7	0	1	2	3	4
pqp^2q	5	6	7	0	1	2	3	4
qp^3q	5	6	7	0	1	2	3	4
$pqpqp$	5	6	7	0	1	2	3	4
qp^2qp	5	6	7	0	1	2	3	4
$qpqp^2$	5	6	7	0	1	2	3	4
$qpqpq$	6	7	0	1	2	3	4	5
diff	4	4	4	4	4	4	4	4

x	0	1	2	3	4	5	6	7
p^5	4	3	5	1	6	7	0	2
p^4q	3	2	1	4	0	6	7	5
p^3qp	2	6	7	5	1	3	4	0
p^2qp^2	6	7	4	0	5	2	3	1
pqp^3	1	4	3	2	0	6	7	5
qp^4	2	5	7	6	3	1	4	0
p^2qpq	7	1	0	6	3	5	2	4
pqp^2q	4	5	2	1	7	0	6	3
qp^3q	5	0	6	2	4	3	1	7
$pqpqp$	5	0	6	3	2	4	1	7
qp^2qp	0	3	1	7	6	5	2	4
$qpqp^2$	3	1	2	4	7	0	5	6
$qpqpq$	1	6	4	3	5	7	0	2
diff	8	8	8	8	8	8	8	8

Table 2.1 – Diffusion tables for the cyclical shift (left table) and one optimal permutation proposed by [CGT19] (right table).

these tables. Indeed, it corresponds to finding the minimal i and even-odd permutations of size $2k$ such that their diffusion table have their *diff* row containing only k.

2.4 Searching for an Optimal Permutation over 9 Rounds

2.4.1 Efficient Search Algorithm

First, we can see that our characterization can be very efficiently implemented, as testing if $\pi = (p, q)$ has full diffusion mostly requires only a few table lookups. An example of an implementation for this test for 9 rounds is given in Appendix 2.B, and its efficiency allowed us to recover all optimal even-odd permutations for $k \leq 13$ with a basic exhaustive search. Especially, for $k = 13$, we were able to go through all $\mathcal{N}_{13}.13! \simeq 2^{39}$ permutations and check them in about 410 minutes on a single core. While these optimal permutations were already known, it shows that the sophisticated techniques introduced in [CGT19] were not necessary for these cases.

However for $k \geq 14$, it becomes too expensive to make this exhaustive search. We thus focus on finding optimal even-odd permutations for $14 \leq k \leq 21$, hence such permutations would have a diffusion round of 9. Given a cycle structure for p, we can easily find a

permutation p with such structure and thus we need to search q such that $\pi = (p, q)$ needs 9 rounds to reach full diffusion, i.e., such that each $\mathbb{J}_j^8$ contains all numbers from 0 to $k-1$.

Note that we cannot exploit $\mathbb{J}_j^8$ directly. Indeed, one might want to guess parts of q and check if $\mathbb{J}_j^8$ does not contains too many duplicates. However, to fully compute $\mathbb{J}_j^8$, we need to guess q in its entirety, which makes this strategy too expensive. We thus describe an efficient way to exploit this characterization to find optimal even-odd permutations.

First for a given j, if we take a look at $\mathbb{J}_j^6$, we can see that we need to make only 7 guesses over the images of q to fully compute $\mathbb{J}_j^6$. Indeed, we need to know

$$q(j), (qp)(j), (qp^2)(j), (qp^3)(j), (qpq)(j), (qp^2q)(j) \text{ and } (qpqp)(j).$$

Let $\mathbb{X}_j^6$ and $\mathbb{Y}_j^6$ be two subsets of $\mathbb{J}_j^6$, such that $\mathbb{X}_j^6 \cup \mathbb{Y}_j^6 = \mathbb{J}_j^6$, with

$$\mathbb{X}_j^6 = \{p^4(j), (pqp^2)(j), (p^2qp)(j), (p^3q)(j), (pqpq)(j)\}$$

$$\text{and } \mathbb{Y}_j^6 = \{(qp^3)(j), (qpqp)(j), (qp^2q)(j)\}.$$

According to the construction of $\mathbb{J}_j^8$, we can actually write

$$\mathbb{J}_j^8 = p^2(\mathbb{X}_j^6 \cup \mathbb{Y}_j^6) \cup (pq)(\mathbb{X}_j^6) \cup (qp)(\mathbb{X}_j^6 \cup \mathbb{Y}_j^6).$$

Assume that we made the 7 guesses mentioned above. In that case, we know the exact values in both $\mathbb{X}_j^6$ and $\mathbb{Y}_j^6$. Moreover, since p is known, we know exactly the values in $p^2(\mathbb{X}_j^6 \cup \mathbb{Y}_j^6)$. Finally, since we guessed 7 images of q, there might be some values in $(pq)(\mathbb{X}_j^6)$ and $(qp)(\mathbb{X}_j^6 \cup \mathbb{Y}_j^6)$ that are known.

Hence, we create three sets $\mathbb{K}_j, \widetilde{\mathbb{X}}_j^6$ and $\widetilde{\mathbb{Y}}_j^6$:

- $\mathbb{K}_j$ is the set of all known values of $\mathbb{J}_j^8$. Thus $p^2(\mathbb{X}_j^6 \cup \mathbb{Y}_j^6) \subset \mathbb{K}_j$ and there might be a few elements from $(pq)(\mathbb{X}_j^6)$ and $(qp)(\mathbb{X}_j^6 \cup \mathbb{Y}_j^6)$ in $\mathbb{K}_j$ too.

- $\widetilde{\mathbb{X}}_j^6$ is the subset of $\mathbb{X}_j^6$ such that for any $x \in \widetilde{\mathbb{X}}_j^6$, the value of $q(x)$ yet remains to be determined.

- In the same way, $\widetilde{\mathbb{Y}}_j^6$ is the subset of $p(\mathbb{X}_j^6 \cup \mathbb{Y}_j^6)$ such that for any $x \in \widetilde{\mathbb{Y}}_j^6$, the value of $q(x)$ is not determined.

For j to be fully diffused, we thus have the constraint

$$C_j : \left|\mathbb{K}_j \cup q(\widetilde{\mathbb{Y}}_j^6) \cup (pq)(\widetilde{\mathbb{X}}_j^6)\right| \geq k.$$

We then check if this constraint is valid, i.e. if there exist some guesses for the remaining images of q such that C_j holds, and this is described in the next section.

Now if we take a look at $\mathbb{J}_{j'}^6$ where $j' = p(j)$, we can see that we only need 3 more guesses to compute it, instead of 7 as before. Indeed, we already guessed

$$\begin{aligned}(qp)(j) &= q(j')\\ (qp^2)(j) &= (qp)(j')\\ (qp^3)(j) &= (qp^2)(j')\\ (qpqp)(j) &= (qpq)(j')\end{aligned}$$

and thus it only remains to guess

$$\begin{aligned}(qp^4)(j) &= (qp^3)(j')\\ (qp^2qp)(j) &= (qp^2q)(j')\\ (qpqp^2)(j) &= (qpqp)(j').\end{aligned}$$

By doing these guesses, we can build the sets $\mathbb{K}_{j'}, \mathbb{X}_{j'}^6$ and $\mathbb{Y}_{j'}^6$ as before, and thus get another constraint that needs to be checked

$$C_{j'} : \left|\mathbb{K}_{j'} \cup q(\widetilde{\mathbb{Y}}_{j'}^6) \cup (pq)(\widetilde{\mathbb{X}}_{j'}^6)\right| \geq k.$$

However by making those three new guesses, we might be able to compute new values in $\widetilde{\mathbb{X}}_j^6$ and $\widetilde{\mathbb{Y}}_j^6$. We thus need to update the constraint C_j according to these guesses, and then check again if C_j is valid.

This can be repeated until we have fully guessed q, in which case we have a solution, or show that no matter which guesses we made there is no solution which satisfies all constraints. This is the core of our algorithm, which is described from a high-level point of view in Algorithm 5.

Note however that the actual algorithm is a bit more sophisticated. Indeed, it might occur at some point that $p(j)$ was already processed, i.e. $C_{p(j)}$ is already a constraint we have. When this happens, we need to choose another starting block j, and re-apply the

Algorithm 5 Searching for optimal even-odd permutations over 9 rounds

function NEXTGUESS($p, q, j, \mathbb{C}$) $\mathbb{C}$ *is the list of known constraints*
  if q is fully determined **then**
    Print p, q
  else
    while all guesses are not made **do**
      Guess $(qp^3)(j), (qp^2q)(j)$ and $(qpqp)(j)$
      Update every constraints in $\mathbb{C}$ according to those guesses
      Deduce the new constraint C_j
      $\mathbb{C}' \leftarrow \mathbb{C} \cup \{C_j\}$
      if $\exists$ invalid constraint in $\mathbb{C}'$ **then**
        Make a new guess
      else
        NEXTGUESS$(p, q, p(j), \mathbb{C}')$
      end if
    end while
  end if
end function

$p \leftarrow$ chosen permutation with a given structure cycle
$j \leftarrow$ an element from the smallest cycle of p
while all guesses are not made **do**
  Guess $q(j), (qp)(j), (qp^2)(j), (qp^3)(j), (qpq)(j), (qp^2q)(j)$ and $(qpqp)(j)$
  Deduce the constraint C_j
  if C_j is a valid constraint **then**
    $\mathbb{C} \leftarrow \{C_j\}$
    NEXTGUESS$(p, q, p(j), \mathbb{C})$
  end if
end while

algorithm, while still keeping all previously computed constraints. In practice, we found that the most efficient strategy is to use an element from the shortest cycle of p as the first starting block. Then, if we need to choose another starting block, we pick an element in the next shortest cycle of p and so on. Moreover, when making some guesses for the images of q, it might happen that we already made this guess. This is not a problem, as this guess basically becomes free and does not add any more cost. Finally, except for the first seven guesses, we update and check all constraints after *each* guess.

2.4.2 Checking the Constraints

We first give a naive way to check if a constraint is valid. We are given three sets $\mathbb{K}, \mathbb{X}$ and $\mathbb{Y}$, resulting in the constraint

$$C : |\mathbb{K} \cup q(\mathbb{Y}) \cup (pq)(\mathbb{X})| \geq k.$$

We know the full permutation p, and for any $x \in \mathbb{X} \cup \mathbb{Y}$, $q(x)$ is still unknown. Let $\mathbb{A}$ denote the set of values a for which we still do not know the preimage of a through q, i.e. for any $a \in \mathbb{A}$, we do not know which x results in $q(x) = a$. Considering the guesses we already made on q, we always know this set $\mathbb{A}$, and thus have the following two relations $(pq)(\mathbb{X}) \subset p(\mathbb{A})$ and $q(\mathbb{Y}) \subset \mathbb{A}$. According to this, we can write

$$|\mathbb{K} \cup q(\mathbb{Y}) \cup (pq)(\mathbb{X})| \leq |\mathbb{K} \cup \mathbb{A} \cup p(\mathbb{A})|\,.$$

Hence if $|\mathbb{K} \cup \mathbb{A} \cup p(\mathbb{A})| < k$, we know that the constraint C cannot be valid. However, we can actually go further and get more precise information by doing the following.

We can formulate our problem in the following generic way. We are given three sets $\mathbb{K}, \mathbb{A}$, and $\mathbb{B}$ $(= p(\mathbb{A}))$, and we search for two sets $\widetilde{\mathbb{A}} \subset \mathbb{A}$ and $\widetilde{\mathbb{B}} \subset \mathbb{B}$ such that $|\mathbb{K} \cup \widetilde{\mathbb{A}} \cup \widetilde{\mathbb{B}}|$ is maximal, with $\widetilde{\mathbb{A}} = q(\mathbb{Y})$ and $\widetilde{\mathbb{B}} = (pq)(\mathbb{X})$. Note that, since p and q are permutations, we have $|\widetilde{\mathbb{A}}| = |\mathbb{X}|$ and $|\widetilde{\mathbb{B}}| = |\mathbb{Y}|$. Hence our idea is to determine whether there is at least one such pair $(\widetilde{\mathbb{A}}, \widetilde{\mathbb{B}})$ satisfying $|\mathbb{K} \cup \widetilde{\mathbb{A}} \cup \widetilde{\mathbb{B}}| \geq k$. Indeed if no such pair exists then constraint C does not hold. Note that if $\mathbb{X} \cap \mathbb{Y} \neq \emptyset$ then it is possible for such pair to exist while C does not hold. However we found this filter powerful enough for our need.

We can partition $\mathbb{K} \cup \mathbb{A} \cup \mathbb{B}$ into the following eight disjoint sets:

$$\begin{aligned}
\mathbb{S}_0 &= \mathbb{K} \cap \mathbb{A} \cap \mathbb{B} & \mathbb{S}_1 &= \mathbb{K}^c \cap \mathbb{A} \cap \mathbb{B} \\
\mathbb{S}_2 &= \mathbb{K} \cap \mathbb{A}^c \cap \mathbb{B} & \mathbb{S}_3 &= \mathbb{K} \cap \mathbb{A} \cap \mathbb{B}^c \\
\mathbb{S}_4 &= \mathbb{K}^c \cap \mathbb{A}^c \cap \mathbb{B} & \mathbb{S}_5 &= \mathbb{K}^c \cap \mathbb{A} \cap \mathbb{B}^c \\
\mathbb{S}_6 &= \mathbb{K} \cap \mathbb{A}^c \cap \mathbb{B}^c & \mathbb{S}_7 &= \mathbb{K}^c \cap \mathbb{A}^c \cap \mathbb{B}^c
\end{aligned}$$

Let k_A (resp. k_B) denote the cardinality of $\widetilde{\mathbb{A}}$ (resp. $\widetilde{\mathbb{B}}$), and k_A^i, k_B^i be such that

$$k_A^i = |\widetilde{\mathbb{A}} \cap \mathbb{S}_i| \leq \min(|\mathbb{S}_i|, k_A), \quad k_B^i = |\widetilde{\mathbb{B}} \cap \mathbb{S}_i| \leq \min(|\mathbb{S}_i|, k_B).$$

Since all $\mathbb{S}_i$ are disjoint, $\widetilde{\mathbb{A}} \subset \mathbb{A}$ and $\widetilde{\mathbb{B}} \subset \mathbb{B}$, notice that we have

$$\begin{aligned}
k_A^2 = k_A^4 = k_A^6 = k_A^7 = 0 \text{ and } k_A = k_A^0 + k_A^1 + k_A^3 + k_A^5 \\
k_B^3 = k_B^5 = k_B^6 = k_B^7 = 0 \text{ and } k_B = k_B^0 + k_B^1 + k_B^2 + k_B^4.
\end{aligned}$$

By selecting the two sets $\widetilde{\mathbb{A}} \cap \mathbb{S}_1$ and $\widetilde{\mathbb{B}} \cap \mathbb{S}_1$ as disjoint as possible we have:

$$\begin{aligned}
|\mathbb{K} \cup \widetilde{\mathbb{A}} \cup \widetilde{\mathbb{B}}| = |K| + k_A + k_B - k_A^0 - k_B^0 - k_B^2 - k_A^3 \\
- \max(k_A^1 + k_B^1 - |\mathbb{S}_1|, 0)
\end{aligned}$$

Indeed, first we have at most $|K| + k_A + k_B$ elements in $\mathbb{K} \cup \widetilde{\mathbb{A}} \cup \widetilde{\mathbb{B}}$. However among all those elements, some might be the same, which explains the remaining terms :

- Elements of $\widetilde{\mathbb{A}}$ and $\widetilde{\mathbb{B}}$ included in $\mathbb{S}_0$, $\mathbb{S}_2$ or $\mathbb{S}_3$ are duplicates since they all belong to $\mathbb{K}$.
- We need to take k_A^1 (resp. k_B^1) elements from $\mathbb{A}$ (resp. from $\mathbb{B}$), where all those elements belongs to $\mathbb{S}_1$. We thus have two cases. If $k_A^1 + k_B^1 \leq |\mathbb{S}_1|$, we can freely choose all those elements without having duplicates between $\widetilde{\mathbb{A}}$ and $\widetilde{\mathbb{B}}$. Indeed for example, if we have $k_A^1 = k_B^1 = 1$ and $\mathbb{S}_1 = \{0, 1, 2\}$, then we can put 0 in $\widetilde{\mathbb{A}}$ and 1 in $\widetilde{\mathbb{B}}$, thus resulting in no duplicates between $\widetilde{\mathbb{A}}$ and $\widetilde{\mathbb{B}}$. However if we have $k_A^1 + k_B^1 > |\mathbb{S}_1|$, then no matter what, we will have duplicates. Thus in the best case, we have $\max(k_A^1 + k_B^1 - |\mathbb{S}_1|, 0)$ duplicates that we need to count out.

Hence, maximizing $|\mathbb{K} \cup \widetilde{\mathbb{A}} \cup \widetilde{\mathbb{B}}|$ is straightforward as there is one specific order in which to find the values of k_A^i and k_B^i that always maximize the size of the union. We only give the way to optimally build $\widetilde{\mathbb{A}}$ since it is fully similar for $\widetilde{\mathbb{B}}$:

- First, using elements from $\mathbb{S}_5$ to build $\widetilde{\mathbb{A}}$ does not add any duplicate, thus we first pull elements from $\mathbb{S}_5$ and $k_A^5 = \min(k_A, |\mathbb{S}_5|)$.
- As mentioned above, using one element from $\mathbb{S}_1$ adds either zero or one duplicate, thus we then pull elements from $\mathbb{S}_1$ and $k_A^1 = \min(k_A - k_A^5, |\mathbb{S}_1|)$.
- Finally, elements from either $\mathbb{S}_0$ and $\mathbb{S}_3$ necessarily add duplicates, so we freely choose any $k_A^0 \leq |\mathbb{S}_0|$ and $k_A^3 \leq |\mathbb{S}_3|$ such that $k_A^0 + k_A^3 = k_A - k_A^5 - k_A^1$.

Finally, computing the maximal value for $|\mathbb{K} \cup \widetilde{\mathbb{A}} \cup \widetilde{\mathbb{B}}|$ only requires to compute $|\mathbb{S}_1|$, $|\mathbb{S}_4|$ and $|\mathbb{S}_5|$ and we then check how it compares to k.

2.4.3 Results

We ran our algorithm for every k such that we need at least 9 rounds to have full diffusion, according to Proposition 2.1. This corresponds to $14 \leq k \leq 21$, and we were able to find *all* optimal even-odd permutations for $k \in \{14, 15, 16, 18\}$. For $k \in \{17, 19, 20, 21\}$, our algorithm allowed us to prove that there is no even-odd permutation leading to a full diffusion after 9 rounds. Since 9 rounds correspond to the Fibonacci bound, we know that for these cases, we need at least 10 rounds to have full diffusion, and we give later in this section an optimal solution for $k = 17$ reaching full diffusion in 10 rounds, as well as good permutations for $k = 19, 20, 21$ with a diffusion round of 11. We can thus give the following theorem to summarize our results.

Theorem 2.2. *To build a Generalized Feistel Network GFN_π^k with full diffusion where π is an even-odd permutation, we have :*

- *For $k = 14, 15, 16$ and 18, the optimal number of rounds for full diffusion is 9.*
- *For $k = 17$, the optimal number of rounds for full diffusion is 10.*
- *For $k = 19, 20$ and 21, the optimal number of rounds for full diffusion is at least 10 and at most 11.*

We give in Table 2.2 an overview of our results. The first column gives the total time needed for our algorithm to either exhaust all optimal even-odd permutations, or prove that no such permutation exists. Note that this is the total CPU time, i.e. when using a single CPU, however our algorithm is highly parallelizable and thus the real time can

k	Time	Structure of p	Structure of q	Number of solutions
14	180 min	(6, 6, 1, 1)	(6, 6, 2)	144
		(6, 6, 2)	(6, 6, 1, 1)	144
		(6, 3, 2, 2, 1)	(6, 3, 2, 2, 1)	144
		(12, 2)	(12, 1, 1)	24
		(12, 1, 1)	(12, 2)	24
15	480 min	(10, 2, 2, 1)	(10, 2, 2, 1)	160
16	1023 min	(6, 6, 3, 1)	(6, 6, 3, 1)	432
		(6, 6, 2, 2)	(6, 3, 3, 2, 1, 1)	288
		(6, 3, 3, 2, 1, 1)	(6, 6, 2, 2)	216
17	1700 min	-	-	0
18	2213 min	(8, 8, 1, 1)	(8, 8, 2)	256
		(8, 8, 2)	(8, 8, 1, 1)	256
19	1913 min	-	-	0
20	1116 min	-	-	0
21	400 min	-	-	0

Table 2.2 – Results for optimal permutations with $DR_{max}(\pi) = 9$

be drastically reduced.[3] This shows that our algorithm is extremely efficient, as it can quickly solve the case $k = 16$ for which [CGT19] were not able to give an optimal solution. The second (resp. third) column gives the possible cycle structures of p (resp. q) in an optimal permutation, and the last column gives the number of solutions which have this structure. We can notice that not only the number of solutions is quite low, but also that the number of possible cycle structures is also quite limited. Moreover, we *always* have a fixed point in either p or q.

The most important result in this table is that there are actually even-odd permutations which have full diffusion after 9 rounds for $k = 16$, while both [SM10] and [CGT19] could only find a permutation with full diffusion after 10 rounds, leaving open the question of whether the theoretical bound of 9 rounds (from Proposition 2.1) could be reached. Our results shows that it is indeed possible, and thus this proves that our permutations are optimal when considering even-odd permutations. We will see in the next section that we can further regroup these permutations into more precise equivalence classes, leading for the case $k = 16$ to four equivalence classes, given in Table 2.3.

3. Less than one hour for each k using 72 threads.

(p, q)
$p = (1, 2, 3, 4, 5, 0, 7, 8, 9, 10, 11, 6, 13, 14, 12, 15)$ $q = (2, 6, 12, 10, 1, 13, 4, 15, 7, 9, 14, 5, 8, 3, 11, 0)$
$p = (1, 2, 3, 4, 5, 0, 7, 8, 9, 10, 11, 6, 13, 14, 12, 15)$ $q = (5, 6, 13, 10, 4, 12, 9, 15, 2, 1, 14, 7, 11, 3, 8, 0)$
$p = (1, 2, 3, 4, 5, 0, 7, 8, 9, 10, 11, 6, 13, 12, 15, 14)$ $q = (2, 1, 12, 8, 7, 14, 5, 4, 13, 11, 10, 15, 9, 3, 6, 0)$
$p = (1, 2, 3, 4, 5, 0, 7, 8, 6, 10, 11, 9, 13, 12, 14, 15)$ $q = (7, 6, 14, 9, 11, 15, 1, 12, 2, 13, 5, 4, 10, 8, 3, 0)$

Table 2.3 – Optimal equivalence classes with $k = 16$

2.4.4 Security Analysis

Recall that our search space $\mathbb{P}_k$ defined in Proposition 2.2 contains *at least* one representative for each class. Hence, among all the permutations we found, some of them might actually be in the same equivalence class. We can thus go further and regroup all representatives that belong to the same class using the following proposition.

Proposition 2.5. *Let $\pi = (p, q)$ be a permutation over $2k$ elements. Then for any permutation r such that $r \circ p \circ r^{-1} = p$, (p, q) and $(p, r \circ q \circ r^{-1})$ are equivalent.*

Proof. Let $\pi = (p, q)$ and $\pi' = (p, r \circ q \circ r^{-1})$ where r is a permutation such that $r \circ p \circ r^{-1} = p$. Recall that we have $\pi(2i) = 2p(i) + 1$ and $\pi(2i + 1) = 2q(i)$, for all $i \in \{0, \ldots, k - 1\}$. Now let φ be the permutation over $2k$ elements defined as

$$\varphi(2i) = 2r(i),\ \varphi(2i + 1) = 2r(i) + 1,\ \forall i \in \{0, \ldots, k - 1\}.$$

Then we have $\pi' = \varphi \circ \pi \circ \varphi^{-1}$. Indeed, if we look at the image of an even number $2i$, we

have

$$\begin{aligned}\varphi \circ \pi \circ \varphi^{-1}(2i) &= \varphi \circ \pi(2r^{-1}(i)) \\ &= \varphi(2(p \circ r^{-1})(i) + 1) \\ &= 2(r \circ p \circ r^{-1})(i) + 1 \\ &= 2p(i) + 1 = \pi'(2i).\end{aligned}$$

In the same way, the image of an odd number $2i + 1$ is

$$\begin{aligned}\varphi \circ \pi \circ \varphi^{-1}(2i+1) &= \varphi \circ \pi(2r^{-1}(i) + 1) \\ &= \varphi(2(q \circ r^{-1})(i)) \\ &= 2(r \circ q \circ r^{-1})(i) \\ &= \pi'(2i+1)\end{aligned}$$

We thus have $\pi' = \varphi \circ \pi \circ \varphi^{-1}$. Hence, π and π' are conjugate and thus equivalent, according to Definition 2.4. □

This leads us to the equivalence classes given in Table 2.4 to 2.7 in Appendix 2.A for $k = 14, 15, 18$. The column (p, q) gives both permutations p and q. The column Imp. Diff. gives the number of rounds for the longest Impossible Differential distinguisher. Note that this is only considering *structural* Impossible Differentials, where we do not specify neither the size of the blocks nor the definition of the S-boxes, such that contradictions are obtained on blocks rather than bits. The columns $S_N^{s,\delta}$ give the minimal number of rounds to get at least N active S-boxes, where each S-box is of size s and the highest differential probability is $2^{-\delta}$. We chose to only consider three cases : $S_N^{4,2}$, $S_N^{8,6}$ and $S_N^{8,7}$. The first case represents the best case for 4-bit S-boxes. Indeed, we know that there is no APN bijective S-boxes of size 4 (which would lead to a highest differential probability of 2^{-3}). As such, the best case is when the highest differential probability is 2^{-2}. It is still unknown whether 8-bit APN bijective S-boxes exist, so we consider both cases. If such an APN 8-bit S-box exists, the column $S_N^{8,7}$ is relevant, otherwise it would be $S_N^{8,6}$ (for example the AES S-box). The last thing is that N depends on the size of the key (as well as δ). Indeed, if we have a key of size λ, then we want N to verify $2^{-\delta N} < 2^{-\lambda}$, i.e. $N > \frac{\lambda}{\delta}$.

As the evaluation of the minimal number of rounds to get at least N S-boxes can be quite expensive, we limited ourselves to $\lambda = 2ks$, where k follows the notation in this chapter, i.e. we have $2k$ blocks of s bits and the key is of the same size as the state. Finally, the last column N_{20} shows the minimal number of active S-boxes for 20 rounds, as both [SM10] and [CGT19] also gave this metric for the permutations they found. It is worth mentioning that while our permutations are optimal (w.r.t the diffusion round), for the case $k = 16$, they have a minimal number of active S-boxes over 20 rounds which is lower than for the permutations given in [CGT19] in the same case, where those permutations have a diffusion round of 10 and at least 70 actives S-boxes over 20 rounds. However for all our permutations, the longest Impossible Differentials distinguisher we can build is over 17 rounds, which is at least one round lower than for the permutations with $k = 16$ given in [CGT19].

Note that we were still able to find the following optimal even-odd permutation for $k = 17$, which thus has a diffusion round of 10 :

$$p = (7, 1, 4, 13, 8, 16, 2, 3, 12, 5, 0, 9, 15, 14, 10, 11, 6)$$
$$q = (8, 0, 9, 10, 3, 2, 16, 6, 14, 11, 7, 4, 1, 12, 5, 15, 13)$$

For this permutation, the longest Impossible Differential distinguisher is over 19 rounds, and $S_{69}^{4,2}, S_{46}^{8,6}, S_{39}^{8,7}, N_{20}$ are respectively 20, 16, 15 and 20. For $k = 19, 20, 21$, we easily found permutations reaching full diffusion after 11 rounds with a random search, leaving open the question to find one permutation with a diffusion round of 10. We give an example for these cases below

$$k = 19 : \quad \begin{aligned} p &= (18, 3, 5, 9, 13, 15, 10, 16, 11, 8, 6, 1, 0, 2, 14, 7, 17, 12, 4) \\ q &= (9, 14, 2, 6, 3, 8, 16, 4, 0, 13, 18, 15, 5, 11, 7, 17, 12, 1, 10) \end{aligned}$$

$$k = 20 : \quad \begin{aligned} p &= (14, 5, 15, 1, 17, 3, 11, 8, 4, 0, 6, 13, 19, 10, 2, 9, 18, 12, 16, 7) \\ q &= (1, 17, 5, 18, 12, 2, 0, 16, 13, 6, 3, 10, 14, 8, 11, 19, 9, 15, 7, 4) \end{aligned}$$

$$k = 21 : \quad \begin{aligned} p &= (19, 10, 7, 17, 2, 16, 20, 9, 6, 0, 3, 12, 18, 1, 4, 11, 15, 13, 14, 8, 5) \\ q &= (20, 12, 0, 8, 7, 1, 4, 2, 10, 13, 5, 6, 11, 14, 19, 15, 9, 16, 3, 17, 18) \end{aligned}$$

2.5 Conclusion

We solved a 10-year-old problem which was to find an optimal (w.r.t diffusion round) even-odd permutation for a Generalized Feistel Network with 32 blocks. More specifically, we showed that there exist permutations which have a diffusion round of 9, while the best permutation found before had a diffusion round of 10. To do so, we give a precise characterization for the permutation to have full diffusion after a given number of rounds. This characterization allowed us to get a very efficient exhaustive search for $k \leq 13$. Even if optimal permutations were already known for these sizes, this shows that our characterization is powerful, thus we have no need to use the elaborated techniques from [CGT19] to treat all these cases. We then exploit this characterization to design a very efficient algorithm that allows us to exhibit *all* optimal even-odd permutations for 32 blocks, as well as for 28, 30 and 36 blocks, which also have an optimal diffusion round of 9 and were not given in the previous literature. For 34, 38, 40 and 42 blocks, our algorithm also allows us to prove that there is no even-odd permutation with a diffusion round of 9 (which is the lower bound), which is again a new result. However for these cases, we were able to give better optimality bounds when considering even-odd permutations, namely for $2k = 34$ the optimal number of rounds for full diffusion is exactly 10 rounds and for $2k = 38, 40, 42$, at most 11 rounds. We also give some security evaluation for Impossible Differentials and Differentials (through the minimum number of active S-boxes). Especially for Impossible Differentials, for the 32 blocks case, all our permutations have their longest impossible differential distinguishers over 17 rounds, which is at least one round lower than every permutation given in [CGT19] for this case.

2.A Results for Optimal Permutations

2.B Efficient Implementation to Test 9 Round Full Diffusion

We give an example of a `C++` implementation of the characterization for a permutation to have full diffusion over 9 rounds. This function takes `powerp` and `q` as parameters, which are respectively, the precomputed values of each power of p, i.e. `powerp[i][j] =` $p^i(j)$, and the permutation q.

(p, q)	Imp. Diff	$S_{57}^{4,2}$	$S_{38}^{8,6}$	$S_{33}^{8,7}$	N_{20}
$p = (1,2,3,4,5,0,7,8,9,10,11,6,12,13)$ $q = (10,7,13,11,9,8,4,1,12,5,3,2,6,0)$	17	19	14	13	66
$p = (1,2,3,4,5,0,7,8,9,10,11,6,12,13)$ $q = (8,6,13,10,7,9,1,12,5,2,4,3,11,0)$	17	19	14	13	66
$p = (1,2,3,4,5,0,7,8,9,10,11,6,13,12)$ $q = (9,1,13,5,2,10,3,7,12,11,8,4,6,0)$	17	19	14	13	66
$p = (1,2,3,4,5,0,7,8,9,10,11,6,13,12)$ $q = (4,1,13,5,10,9,2,11,8,12,6,3,7,0)$	17	19	14	13	66
$p = (1,2,3,4,5,6,7,8,9,10,11,0,13,12)$ $q = (3,5,2,13,0,10,9,11,8,12,6,4,7,1)$	17	22	14	13	52
$p = (1,2,3,4,5,6,7,8,9,10,11,0,13,12)$ $q = (3,1,13,11,8,10,9,7,12,5,2,4,6,0)$	17	22	14	13	52
$p = (1,2,3,4,5,6,7,8,9,10,11,0,12,13)$ $q = (3,11,8,13,6,10,9,5,2,12,0,4,7,1)$	17	22	14	13	52
$p = (1,2,3,4,5,6,7,8,9,10,11,0,12,13)$ $q = (3,7,13,5,2,10,9,1,12,11,8,4,6,0)$	17	22	14	13	52
$p = (1,2,3,4,5,0,7,8,6,10,9,12,11,13)$ $q = (4,9,6,11,13,12,10,2,8,1,5,3,7,0)$	17	23	19	18	46

Table 2.4 – Security evaluation for the best equivalence classes with $k = 14$

(p, q)	Imp. Diff	$S_{61}^{4,2}$	$S_{41}^{8,6}$	$S_{35}^{8,7}$	N_{20}
$p = (1,2,3,4,5,6,7,8,9,0,11,10,13,12,14)$ $q = (12,5,10,3,11,1,13,9,14,7,4,6,2,8,0)$	17	20	16	14	61
$p = (1,2,3,4,5,6,7,8,9,0,11,10,13,12,14)$ $q = (13,9,10,7,11,5,12,3,14,1,4,6,8,2,0)$	17	42	28	24	30

Table 2.5 – Security evaluation for the best equivalence classes with $k = 15$

```
bool checkDiffusion(vector<vector<unsigned int>> const & powerp,
                    vector<unsigned int> const & q){
```

(p, q)	Imp. Diff	$S_{65}^{4,2}$	$S_{43}^{8,6}$	$S_{37}^{8,7}$	N_{20}
$p = (1, 2, 3, 4, 5, 0, 7, 8, 9, 10, 11, 6, 13, 14, 12, 15)$ $q = (2, 6, 12, 10, 1, 13, 4, 15, 7, 9, 14, 5, 8, 3, 11, 0)$	17	33	22	19	40
$p = (1, 2, 3, 4, 5, 0, 7, 8, 9, 10, 11, 6, 13, 14, 12, 15)$ $q = (5, 6, 13, 10, 4, 12, 9, 15, 2, 1, 14, 7, 11, 3, 8, 0)$	17	33	22	19	40
$p = (1, 2, 3, 4, 5, 0, 7, 8, 9, 10, 11, 6, 13, 12, 15, 14)$ $q = (2, 1, 12, 8, 7, 14, 5, 4, 13, 11, 10, 15, 9, 3, 6, 0)$	17	50	33	29	26
$p = (1, 2, 3, 4, 5, 0, 7, 8, 6, 10, 11, 9, 13, 12, 14, 15)$ $q = (7, 6, 14, 9, 11, 15, 1, 12, 2, 13, 5, 4, 10, 8, 3, 0)$	17	50	33	29	26

Table 2.6 – Security evaluation for the best equivalence classes with $k = 16$

(p, q)	Imp. Diff	$S_{73}^{4,2}$	$S_{49}^{8,6}$	$S_{42}^{8,7}$	N_{20}
$p = (1, 2, 3, 4, 5, 6, 7, 0, 9, 10, 11, 12, 13, 14, 15, 8, 16, 17)$ $q = (10, 9, 14, 12, 15, 11, 13, 17, 2, 1, 6, 4, 7, 3, 5, 16, 8, 0)$	17	31	22	19	44
$p = (1, 2, 3, 4, 5, 6, 7, 0, 9, 10, 11, 12, 13, 14, 15, 8, 16, 17)$ $q = (14, 8, 12, 15, 13, 10, 9, 17, 7, 6, 16, 3, 5, 1, 4, 2, 11, 0)$	17	56	38	32	26
$p = (1, 2, 3, 4, 5, 6, 7, 0, 9, 10, 11, 12, 13, 14, 15, 8, 17, 16)$ $q = (2, 1, 6, 12, 15, 3, 13, 16, 10, 9, 14, 4, 7, 11, 5, 17, 8, 0)$	17	31	22	19	44
$p = (1, 2, 3, 4, 5, 6, 7, 0, 9, 10, 11, 12, 13, 14, 15, 8, 17, 16)$ $q = (11, 5, 9, 12, 2, 7, 6, 16, 3, 13, 1, 4, 10, 15, 14, 17, 8, 0)$	17	56	38	32	26

Table 2.7 – Security evaluation for the best equivalence classes with $k = 18$

```
auto const & p6 = powerp[6];
auto const & p5 = powerp[5];
auto const & p4 = powerp[4];
auto const & p3 = powerp[3];
auto const & p2 = powerp[2];
auto const & p = powerp[1];

unsigned int sizeperm = q.size();
for(unsigned int x = 0; x < sizeperm; x++){
        unsigned int qx = q[x];
        unsigned int qpq = q[p[qx]];
        unsigned int qp = q[p[x]];
        unsigned int qp2 = q[p2[x]];
        unsigned int qp3 = q[p3[x]];
        unsigned int qp2q = q[p2[qx]]];
        unsigned int qpqp = q[p[qp]];

        unsigned int indicator = (1 <<p6[x]);
        indicator |= (1 << p5[qx]);
        indicator |= (1 << p4[qp]);
        indicator |= (1 << p3[qp2]);
        indicator |= (1 << p2[qp3]);
        indicator |= (1 << p[q[p4[x]]]);
        indicator |= (1 << q[p5[x]]);
        indicator |= (1 << p3[qpq]);
        indicator |= (1 << p2[qp2q]);
        indicator |= (1 << p2[qpqp]);
        indicator |= (1 << p[q[p3[qx]]]);
        indicator |= (1 << p[q[p2[qp]]]);
        indicator |= (1 << p[q[p[qp2]]]);
        indicator |= (1 << q[p4[qx]]);
        indicator |= (1 << q[p3[qp]]);
        indicator |= (1 << q[p2[qp2]]);
        indicator |= (1 << q[p[qp3]]);
```

```
                indicator |= (1 << p[q[p[qpq]]]);
                indicator |= (1 << q[p2[qpq]]);
                indicator |= (1 << q[p[qp2q]]);
                indicator |= (1 << q[p[qpqp]]);
                if(__builtin_popcount(indicator) != sizeperm)
                        return false;
        }
        return true;
}
```

Chapter 3

Efficient Methods to Search for Best Differential Characteristics on SKINNY

Introduction

Differential cryptanalysis [BS91b] evaluates the propagation of an input difference $\delta X = X \oplus X'$ between two plaintexts X and X' through the ciphering process. Indeed, differential attacks exploit the fact that the probability of observing a specific output difference given a specific input difference is not uniformly distributed. Today, differential cryptanalysis is public knowledge, and block ciphers such as AES have proven bounds against differential attacks. A classical extension of differential cryptanalysis is the so called related-key differential cryptanalysis [Bih93] that allows an attacker to inject differences not only between the plaintexts X and X' but also between the keys K and K' (even if the secret key K stays unknown from the attacker). This attack has been recently extended to tweakable block ciphers [Bei+16]. Those particular ciphers allow in addition to the key, a public value called a tweak. Thus, related-tweakey differential attacks allow related-key differences but also related-tweak differences (*i.e.* differences in a pair of tweaks (T, T')). In differential attacks, two notions are considered: first, differentials where only the input and the output differences are known; and differential characteristics where each difference after each round is completely specified. A classical approach to evaluate the resistance against differential attacks is to compute the probability of the best differential characteristic of the cipher.

Finding optimal (related-tweakey) differential characteristics is a highly combinatorial problem that hardly scales. To limit this explosion, a common solution consists in using a truncated representation [Knu95] for which cells are abstracted by single bits that indicate whether sequences contain differences or not. Typically, each cell (*i.e.* byte or nibble) is

abstracted by a single bit (or, equivalently, a Boolean value). In this case, the goal is no longer to find the exact input and output differences, but to find the positions of these differences, *i.e.*, the presence or absence of a difference for every cell. When a difference is present at the input of an S-box, we talk about an active S-box or an active byte/nibble. However, some truncated representations may not be valid (*i.e.*, there do not exist actual byte values corresponding to these difference positions) because some constraints at the byte level are relaxed when reasoning on difference positions.

Hence, the optimal (related-tweakey) differential characteristic problem is usually solved in two steps [BN10; Abd+17]. In the first one, every differential byte is abstracted by a Boolean variable, denoted by Δ, that indicates whether there is a difference or not at this position, and we search for all truncated representations of low weight as the less differences passing through S-boxes there are, the more the probability is increased. Then, for each of these low weight truncated representations, the second step aims at deciding whether it is valid (*i.e.*, whether it is possible to find actual cell values, denoted δ, for every Boolean variable) and, if it is valid, at finding the actual cell values that maximize the probability of obtaining the output difference given the input difference.

Related Work. Many techniques have been proposed to search for the Step 1 solutions using automatic tools such as Boolean satisfiability (SAT) [SNC09; MP13], [SWW17] or Mixed Integer Linear Programming (MILP) [Sun+14; ST17; Bei+16] and Satisfiability Modulo Theories (SMT) [KLT15]. Dedicated solutions have also been proposed [Mat94].

Regarding the search of the best instantiation of a truncated characteristic, most of the approaches were ad-hoc and dedicated to a precise cipher [Laf18; SWW18; FJP13b; BN10; Gér+18; ENP19]. Concerning the use of SAT solvers, [SWW18] implements a SAT model for differential cryptanalysis based on `Cryptominisat5` [SNC09] for `Midori64` and `LED64`. This model implies a sufficiently small number of clauses to model the non-zero values of the `DDT` and to be applicable. However, no result concerning 8-bit S-boxes are given. As SAT uses Boolean formulas, it seems that the same problem than for MILP appears for modeling S-box: a huge number of Boolean formulas will be necessary to correctly model this step even if dedicated tools as Logic Friday or the Expresso algorithm [Abd+17] are used. In [Abd+17], 16 days are needed to find the best related tweakey differential characteristics on `SKINNY`-128 for the **SK** model. Recently, in [Ger+20; Gér+18], the authors introduce Constraint Programming (CP) models for Step 2 and the performance results are really promising regarding AES-192 and AES-256.

Our contribution. In this chapter, we refine the security bounds on the `SKINNY`-n tweak-

able block cipher regarding differential cryptanalysis for the four following attack models according to the size of the tweakey: the **SK** model focuses on single-key attack, the **TK1** model considers related-tweakey attack when the tweakey has only one component, the **TK2** model in the related-tweakey settings considers 2 components and the **TK3** model, 3 components.

To do so, we implement Step 1 using an ad-hoc method inspired from [FJP13b]. We also propose a CP model for Step 2 taking as input the solutions outputted by Step 1. Thus, we provide, for the first time, the best differential related-tweakey characteristics up to 14 rounds for the **TK1** model. We also consider the **TK2** and **TK3** models and we were able to find some differential characteristics up to 16 rounds for the **TK2** model and up to 17 rounds for the **TK3** model of SKINNY-128. However, we were not able to test all the solutions Step 1, and thus these differential characteristics are not necessarily optimal. This is an important improvement compared to previous results. For instance, in [LGS17] Liu *et al.* could only find the best differential characteristics up to 7 and 9 rounds for **TK1** and **TK2**. Finally we also show there is no differential characteristic with probability higher than 2^{-128} against 15 rounds in the **TK1** model, 19 rounds for **TK2** and 23 rounds for **TK3**. All those results clearly show that SKINNY is much more resistant to differential cryptanalysis than one would expect while counting the number of active S-boxes.

As a feedback, we also provide the time results we obtain when implementing the Step 1 using another tool, a MILP model for the 4 attack settings. As a result we show that MILP is not always the best choice. First, for Step 1, the ad-hoc method is able to surpass the MILP model. Second, the CP model proposed for Step 2 is incomparably much faster than the MILP model proposed in [Abd+17] that requires 16 days according their paper.

All the codes to reproduce these results can be found at [Del+21b].

Organization of the chapter. Section 3.1 gives a short description of SKINNY-n; Section 3.4 presents our Ad-Hoc tool and gives performance results comparing our Ad-Hoc model with a MILP one; Section 3.5 presents our dedicated modeling for Step 2 based on CP and analyzes the obtained results. Finally, some differential characteristics we found are given in the appendices (Appendix 3.A for SKINNY-64 and Appendix 3.B for SKINNY-128).

3.1 Cipher under study: SKINNY-n

In this section, we briefly review the tweakable block cipher SKINNY-n where n denotes the block size and can be equal to 64 or 128 bits. All the details that have been overlooked can be found in [Bei+16].

As its name indicates, it enciphers blocks of length 64 or 128 bits seen as a 4×4 matrix of cells (nibbles for $n = 64$ or bytes for $n = 128$). We denote $x_{i,j,k}$ the cell at row i and column j of the internal state at the beginning of round k (i.e. $0 \leq i, j \leq 3$ and $0 \leq k \leq r + 1$ where r is the number of rounds depending on the tweak length and on the key length). SKINNY-n follows the TWEAKEY framework from [JNP14]. SKINNY-n has three main tweakey size versions: the tweakey size can be equal to $t = 64$ or 128 bits, $t = 128$ or 256 bits and $t = 192$ or 384 bits and we denote $z = t/n$ the tweakey size to block size ratio. Then, the number of rounds is directly derived from the z value: between 32 rounds for the 64/64 version up to 56 for the 128/384 version.

The tweakey state is also viewed as a collection of z 4×4 square arrays of cells (nibbles for $n = 64$ or bytes for $n = 128$). We denote these arrays $TK1$ when $z = 1$, $TK1$ and $TK2$ when $z = 2$, and finally $TK1$, $TK2$ and $TK3$ when $z = 3$. We also denote by $TKk_{i,j}$ the nibble or the byte at position $[i, j]$ in TKk. Moreover, we define the associated adversarial model **SK** (resp. **TK1**, **TK2** or **TK3**) where the attacker cannot (resp. can) introduce differences in the tweakey state.

One encryption round of SKINNY is composed of five operations applied in the following order: SubCells (SC), AddConstants (AC), AddRoundTweakey (ART), ShiftRows (SR) and MixColumns (MC) (see Fig. 3.1).

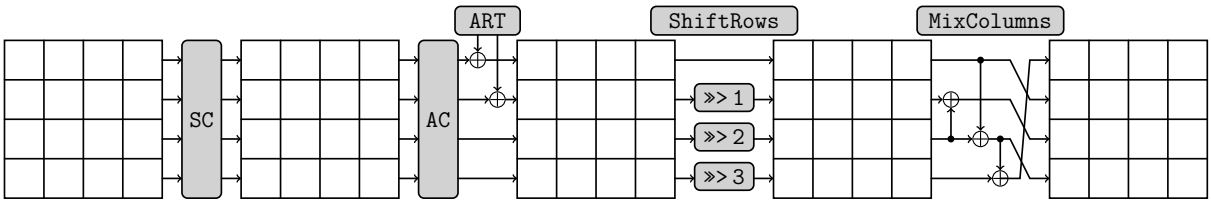

Figure 3.1 – the SKINNY round function with its five transformations [Jea16].

SubCells. A 4-bit ($n = 64$) or an 8-bit ($n = 128$) S-box is applied to each cell of the state. See [Bei+16] for the details of the S-boxes.

AddConstants. A 6-bit affine LFSR is used to generate round constants c_0 and c_1 that

are XORed to the state at position $[0, 0]$ and $[1, 0]$ whereas the constant $c_2 = \texttt{0x02}$ is XORed to the position $[2, 0]$.

AddRoundTweakey. The first and second rows of all tweakey arrays are extracted and bitwise exclusive-ored to the cipher internal state, respecting the array positioning. More formally, we have:

- $x_{i,j} = x_{i,j} \oplus TK1_{i,j}$ when $z = 1$,
- $x_{i,j} = x_{i,j} \oplus TK1_{i,j} \oplus TK2_{i,j}$ when $z = 2$,
- $x_{i,j} = x_{i,j} \oplus TK1_{i,j} \oplus TK2_{i,j} \oplus TK3_{i,j}$ when $z = 3$.

Then, the tweakey arrays are updated. First, a permutation P_T is applied on the cells positions of all tweakey arrays: if $\ell = 4 * i + j$ where i is the row index and j is the column index, then the cell ℓ is moved to position $P_T(\ell)$ where $P_T = [9, 15, 8, 13, 10, 14, 12, 11, 0, 1, 2, 3, 4, 5, 6, 7]$. Second, every cell of the first and second rows of $TK2$ and $TK3$ are individually updated with an LFSR on 4 bits (when $n = 64$) or on 8 bits (when $n = 128$) with a period equal to 15.

ShiftRows. The rows of the cipher state cell array are rotated to the right. More precisely, the second (resp. third and fourth) cell row is rotated by 1 position (resp. 2 and 3 positions).

MixColumns. Each column of the cipher internal state array is multiplied by the 4×4 binary matrix M:

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

Since 2016 and the birth of SKINNY-128, the cryptographic world never stopped trying to attack it. Among all the cryptanalysis results, we could cite the following ones in the related-tweakey settings and classified according the type of attacks. First, in [SQH19; Zha+20; LGS17], boomerang and rectangle related-tweakey attacks are considered. The best result is on 28 rounds with a complexity of 2^{315} in time based on a boomerang distinguisher of 23 rounds in the **TK3** scenario. Concerning impossible related-tweakey attack [Sun+17; LGS17], the best attack has 23 rounds using a distinguisher with 15 rounds in the **TK2** scenario. Even if the distinguishers presented here have less rounds, they do

not look at the same attack scenario. This work essentially goes further than [Abd+17] concerning the search of the best related-tweakey differential trails and aims at refining the best security bounds of SKINNY in this attack model.

3.2 Best known differential characteristics on SKINNY

Analysis concerning the differential characteristics of SKINNY already exist in the literature. The designers of SKINNY proposed the first analysis of security of the primitive done in their design paper. The first result concerning truncated differential characteristics were published in this occasion. The authors proposed a MILP model to compute the best truncated differential characteristic of SKINNY under the different tweakeys settings. In particular, they were able to reduce the search space of the differences in the tweakeys in the **TK2** and **TK3** settings. Details on this work are reviewed in Subsection 3.2.1.

In [LGS17], Liu *et al.* proposed a more targeted analysis of the security of SKINNY in the **TK** models against boomerang and rectangle attacks. For their analysis, they need to know some differential characteristics for SKINNY-64 and SKINNY-128 in the different **TK** models. They were able to find some optimal differential characteristics for SKINNY-64 and some goods for SKINNY-128 which to the best of our knowledge were the best known before the work we realized. We review briefly the part of their work concerning differential characteristics in Subsection 3.2.2.

Finally, [Abd+17] proposed a way to integrate large S-boxes in MILP models with the S-boxes of AES and SKINNY-128. More importantly for us, their idea allows them to construct and run a MILP model searching for differential characteristics of SKINNY-128. They found with it the best differential characteristic for SKINNY-128 in the **SK** models for 13 rounds and prove that no differential characteristics with a better probability than 2^{-128} exist for 14 rounds in the same model. Their work regarding SKINNY is explained in Subsection 3.2.3.

3.2.1 Best truncated differentail characteristics

To give a first idea of the security of SKINNY, its designer proposed a simple MILP model up to 30 rounds to search for the best truncated differential characteristics. Remarks that this model is the same for SKINNY-128 and SKINNY-64.

For a model searching the truncated characteristics, the only operations that need to

be represented are the ShiftRows, the MixColumn and the AddRoundTweakeys (if there are differences in the keys). In their models, Bierle *et al.* have three sets of variables:

- $\{x_{k,i,j} | i, j \in [0..3], k \in [0..r]\}$ corresponding to the activity pattern of the internal state round by round. $x_{k,i,j}$ is the activity of the cell of row i and column j at round k: it is equal to 1 if there is a difference in the cell and 0 otherwise.
- $\{y_{k,i,j} | i, jin[0..3], k \in [0..r]\}$ corresponds to the activity pattern after the tweakeys are added to the state.
- $\{\kappa_{i,j} | i, j \in [0..3]\}$ corresponds to the initial activity pattern of the tweakey state.

With these variables, it is easy to see that the ShiftRows operations can be modeled in another constraint by using the correct indexes between the different variables. In addition, the two others are only composed of XOR operations. As a result, Bierle *et al.* decomposed them to their xor levels and used a classical way to model the xor of binary variables to obtain a complete model of SKINNY.

For **SK** and **TK1**, the model can be run directly with an objective of minimizing the number of $x_{r,i,j}$ that are equal to 1. For **TK2** and **TK3**, Bierle *et al.* choose to not represent the activity patterns of each tweakeys but the one of their sum. This approach needs a lot less variables than modeling each activity pattern: going from 2 or 3 times 16r to 16r variables. Furthermore, they remark that cancellation could happened in the sums of the tweakeys but only a limited number of times (depending on the tweakey setting considered) for each cell (followed after the permutation of the tweakeys states). This limited number of cancellations appears from the updates by LFSRs of the second and third tweaks. For example, for **TK2**, if at round k the sums of the first and second tweakeys canceled, then at round $k+1$ it cannot cancel again: since the first tweakey is unchanged (up to a permutation of the cells) whereas the second one was updated by a LFSR (up to the permutation of the cells). Or as the LFSR has a cycle length of 15, up to two cancellations can happen for the up to 30 rounds considered. The only difference in this model between **TK2** and **TK3** is the maximum number of cancellations for each cell: 2 for **TK2** and 3 for **TK3** for each of the original active cells in κ. It is however not clear with this modeling if the found bounds will be tight for **TK1**, **TK2** and **TK3**. In our work, we keep the idea concerning the cancellations in our Step 1 process and build upon it (see details in Section 3.4). Running these models gave results to the designers of SKINNY that we give again here in Table 3.1.

# rounds	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
SK	1	2	5	8	12	16	26	36	41	46	51	55	58	61	66
TK1	0	0	1	2	3	6	10	13	16	23	32	38	41	45	49
TK2	0	0	0	0	1	2	3	6	9	12	16	21	25	31	35
TK3	0	0	0	0	0	0	1	2	3	6	10	13	16	19	24
# rounds	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
SK	75	82	88	92	96	102	108	(114)	(116)	(124)	(132)	(138)	(136)	(148)	(158)
TK1	54	59	62	66	70	75	79	83	85	88	95	102	(108)	(112)	(120)
TK2	40	43	47	52	57	59	64	67	72	75	82	85	88	92	96
TK3	27	31	35	43	45	48	51	55	58	60	65	72	77	81	85

Table 3.1 – Minimum number of active S-boxes, Table 7 in [Bei+16]. Minimum number of active S-boxes in a truncated differential characteristic. Numbers in parenthesis are upper bound on the minimum.

3.2.2 Result in related-tweakeys models

In their works on boomerang and rectangle attacks against `SKINNY`, they needed good differential characteristics for their analysis and proposed in their works to find them with different tools. The results they found are summarized in Table 3.2.

For `SKINNY-64` , Liu *et al.* repurposed the MILP models of [LGS17] and proceed in three steps:

1. Generate every truncated differential characteristics with exactly the minimum number $\mathrm{AS}_{\min}$ of active S-boxes for the given parameters;
2. Try to instantiate (with a MILP modeling of the 4-bit S-box) them and keep the best probability found p. If $p = 2^{-2\mathrm{AS}_{\min}-i}$ with $i \in \{0, 1, 2\}$ then the solutions found are necessary optimal. Otherwise, do the third step;
3. Find all truncated differential characteristics with less than $\left\lfloor \frac{\log_2(p)}{2} \right\rfloor$ active S-boxes. Then, try to instantiate them with a MILP model.

With this method, Liu *et al.* were able to find the optimal differential characteristics of `SKINNY`-64 up to 13 rounds in the different **TK**s settings. But they remark that it works as long as the number of truncated differential characteristics to consider is small (less than 5000).

For `SKINNY-128` , Liu *et al.* remark that modeling the 8-bit S-box is too heavy for MILP and proposed to replace it by a dedicated search algorithm used after computing

	# rounds	6	7	8	9	10	11	12	13
TK1	SKINNY-64	2^{-12}	2^{-20}	2^{-26}	2^{-32}	2^{-46}	2^{-64}	2^{-76}	2^{-82}
	SKINNY-128	2^{-12}	2^{-20}	2^{-54}					
TK2	SKINNY-64	2^{-4}	2^{-6}	2^{-12}	2^{-20}	2^{-28}	2^{-35}	2^{-48}	2^{-55}
	SKINNY-128			2^{-12}	$2^{-34.42}$				
TK3	SKINNY-64	2^{0}	2^{-2}	2^{-4}	2^{-6}	2^{-12}	2^{-20}	2^{-28}	2^{-38}
	SKINNY-128					2^{-12}	2^{-21}	$2^{-62.83}$	

Table 3.2 – Probability of the best differential characteristics found in [LGS17] (summary of Table 6 and Table 7 of their work).

with MILP truncated differential characteristics. The dedicated algorithm begins with fixing the values of the (few) active cells in the initial tweakeys of the truncated solutions. Then they compute the whole tweakey schedule. Once the tweakey schedule is known, it is easy to find differential characteristics or verify there is no differential. They run their methods only on a small number of truncated characteristics with a minimum of active S-boxes.

3.2.3 The best differential characteristic

As Liu *et al.* remark in [LGS17], the fact that MILP cannot model 8-bit S-boxes (because it would be too costly) was admitted in the cryptographer community. Abdelkhalek *et al.* in [Abd+17] proved that it was in fact possible. Their work is an improvement of the MILP model of Sun *et al.* in [Sun+14] which proposed to search for truncated differential characteristics but used an elegant way to remove truncated solutions that do not correspond to existing differential characteristics. They add as a criteria for the MILP the existence of an instantiation by modeling only possible input/output differences and ignoring probability: the DDT is truncated in terms of probability: every non-zero entry is replaced by 1. They called this truncated version of the DDT the *-DDT.

In [Abd+17], Abdelkhalek *et al.* used the classical approaches to first compute the truncated differential characteristics with less than a target number of active S-boxes with a MILP model. Then, they searched again with a MILP model instantiation of the truncated characteristics. For the model of this second step, they reused the *-DDT to build a MILP model able to treat 8-bit S-boxes. They split the DDT of the 8-bit S-box over the different probabilities in different *-DDT. More precisely, for each probability p in the DDT, they create one *-DDT corresponding to every entry that has probability p. The

non-zeroes entries being identical, they can be changed to 1. With this decomposition, they split the modeling of S-boxes transitions in value in two steps: a first constraint, representing the selection of a *-DDT or said in another manner, a constraint representing the choice of the probability of the transition and a second constraint selecting one precise transition with the selected probability. With this method, they were able to find the best differential characteristics (with a probability of 2^{-123}) for 13 rounds SKINNY-128 in the **SK** setting. Furthermore, they were able to prove no differential characteristic with a better probability than 2^{-128} exists for 14 rounds in the **SK** setting.

3.3 Overview of the search method

During this PhD, we studied the search of differential characteristics with a very classical approach. In Subsection 3.3.1, we review this approach and give details on how we treat the different steps in Section 3.4 and Section 3.5. Subsection 3.3.2 is a digression on other tools that could be used for this approach and a brief argumentation to explain why we did not use them.

3.3.1 A classical approach

As said previously, we adopt a very classical approach to search for the differential characteristics of SKINNY. Just like [LGS17; Abd+17] for SKINNY or [GMS16; Ger+20] for AES, we use a two step process that we called Step 1 and Step 2:

- **Step 1** consists in computing for a given number of rounds the truncated differential characteristics with a given number of active S-boxes. In our work, we proposed an ad-hoc tool to replace the MILP modeling done in previous works based on dynamic programming. We try other approaches to solve this step but keep only the ad-hoc tool to compute our results. Comments on the other approaches are found in Subsection 3.3.2. Details concerning our process are in Section 3.4.

- **Step 2** searches for instantiation of truncated differential characteristics with better probability than a target p. Such an instantiation of a truncated solution gives us a better upper bound for the probability. In particular since the best S-boxes transitions of SKINNY-128 have a probability of 2^{-2}, there were already rounds (represented with a lower opacity in Table 3.1) for which it is not necessary at all to search for a

differential characteristic. The same reasoning can be followed for SKINNY-64 with 32 active S-boxes.

It is possible to automatize completely the exchange of informations between the first and second steps. In reality, we run our process manually by calling generating some truncated solutions then trying to instantiate them and if necessary go back to the Step 1. With our method, we obtained new optimal differential characteristics for the **TK**s settings and some good differential characteristics for **TK2** and **TK3** settings. The precise results concerning them obtained from our works can be found in Section 3.5.3.

The major contribution of our work was the way we implement the search for each step. More precisely, in Section 3.4, we explained in details how we search the truncated differential characteristics. Then, in Section 3.5 we give details on the way we treat the instantiation of the truncated differential characteristics we found in the first step.

3.3.2 About other tools

When we began working on this project, one of the goas was to study the efficiency of different types of tools for searching differential characteristics. We dropped this idea quite fast because such a comparison would need other targets than SKINNY,

For Step 1. We tried different approaches to solve the Step 1 problem, including MILP, SAT and CP models.

Our SAT model is encoded through the high level modeling language MiniZinc while our CP model is based on the Choco-solver. Unfortunately, the results of both the SAT and the CP models are really bad: for example, for all instances greater than 16 rounds we were unable to obtain solutions in reasonable time. This is mainly due to the need to enumerate solutions for SAT, which implies to prohibit all solutions previously found. For CP, on the other hand, this has to do with the nature of the Boolean variables themselves where the Choco-solver can not efficiently propagate lower bounds and upper bounds on Boolean variables.

Our MILP model was much better than our SAT and CP ones. We started from the original model presented in [Bei+16] but made several optimizations. First, we added constraints in the **SK** model to obtain all solutions up to column shifts in order to remove symmetries. Moreover, as the original model only describes the way to find the minimal number of active S-boxes, we added a constraint in each model to set a lower bound on

the number of active S-boxes and thus, we were able to enumerate all the Step 1 solutions given a particular lower bound for the number of active S-boxes. Then, in the original MILP model all xor operations were modeled using dummy variables which is known to be inefficient. Thus we replaced the corresponding inequalities, using that $x \oplus y \oplus z = 0$ can be described with the three inequalities:

$$\{x + y \geq z\},\ \{x + z \geq y\},\ \{y + z \geq x\}.$$

Finally, regarding the resolutions of the MILP models, the parallelization were left to the Gurobi solver.[1]

We compared the MILP model to our ad-hoc tool and we found that our MILP model is much slower in most cases and actually too slow to output all the Step 1 solutions needed to perform Step 2. Running times are given in Table 3.3.

Rounds	Model	Obj_{Step1}	MILP	Ad-hoc
14	**TK1**	$45 \rightarrow 59$	$>$ 6h	5m
19	**TK2**	$52 \rightarrow 63$	$>$ 6h	19m
20	**SK**	96	342m	16s
20	**TK1**	70	38m	28s
20	**TK2**	57	745s	193s
20	**TK3**	45	998s	326s

Table 3.3 – Comparison of the running times required to generate all Step 1 solutions between our MILP and ad-hoc approaches.

Note that while our ad-hoc tool gave very good running times, it may require a lot of memory to store the array C. For instance, for 30 rounds in **TK3** mode, our tool required up to 500GB of RAM to finish the search. It is also important to note that it did not take fully advantage of the 128 cores of our server, and most often used less than 40 cores.

For Step 2. With preliminary results with CP solver proving to be far faster than what exists up to now with MILP and with the complexity of building an efficient ad-hoc tool for Step 2, we basically only explored the uses of CP solver for this step of the search. All details of our method for the resolution of Step 2 can be found in Section 3.5.

1. see: `https://www.gurobi.com/documentation/9.0/refman/threads.html`

3.4 Models and Results for Step 1

As explained in the introduction, in a first step called Step 1, we abstract each possible difference at cell (nibble or byte) level by a binary variable which symbolizes the presence/absence of a difference value at a given position of the cipher. The main concern regarding this step is the combinatorial explosion induced by the abstract XOR operation for which the sum of two non-zero values can lead to the presence or the absence of a difference.

3.4.1 Possible Transitions

Since the S-box is bijective and the `ShiftRows` operation only permutes cells, both those operations do not affect truncated differences. But for the `AddRoundTweakey` and `MixColumns` transformations we need to take care of the XOR operation. More precisely, given two truncated differences a and b we know that the possible values of $(a, b, a \oplus b)$ are:

$$(0,0,0), (0,1,1), (1,0,1), (1,1,0), (1,1,1)$$

However we have to pay attention to uninstantiable solutions. For instance, given three truncated differences a, b and c, $(1,1,1,0,0,1)$ is a possible value for $(a, b, c, a \oplus b, a \oplus c, b \oplus c)$ but it is impossible to instantiate it because if $a = b$ and $a = c$ then $b = c$.

Hence we rewrite the equation $y = \texttt{MixColumns} \circ \texttt{AddRoundTweakey}(x, k)$ to avoid such patterns:

- $y[1] = x[0] \oplus k[0]$,
- $y[3] = y[1] \oplus x[2]$,
- $y[0] = y[3] \oplus x[3]$,
- $y[2] = x[1] \oplus k[1] \oplus x[3]$

We experimentally verified that each truncated solution of this system can be instantiated.

Keyschedule. When looking at the key schedule of `SKINNY` at the cell level and for truncated differential characteristics it is mostly a simple cell permutation. In the model **SK**, there are no differences in the round keys. In the **TKx** models, differences in the round keys are possible. If the number of rounds targeted is at most 30, the rule for active

cells on the round keys is quite simple: either the cell is inactive for all round keys, either it is active for all round keys but one (**TK2**) or two (**TK3**).

3.4.2 Ad-hoc Models for Step 1

To the best of our knowledge, the most efficient algorithm to search for truncated differential characteristics on SPN ciphers is the one described in [FJP13b] by Fouque *et al.* which was applied on the 3 versions of AES. It is mostly dynamic programming as Round i is independent of the paths of rounds $0, 1, \ldots, i-1$ and at each step we only have to save, for each truncated state, the minimal number of active S-boxes to reach it. Hence, the complexity of this algorithm is exponential in the state size but linear in the number of rounds. The algorithm is specified in Algorithm 6. At the end of the algorithm we obtain an array C such that $C[r][s]$ contains the minimal number of active S-boxes required to reach state s after r rounds. Retrieving the truncated representations is then done quite easily using C, starting from the last state to the first. Let say we want to exhaust all truncated differential characteristics on R rounds with at most b active S-boxes ending with state s. From $C[R-1][s]$, we know whether such characteristic exists or not. If $C[R-1][s] \leq b$ we exhaust all states s' such that the transition $s' \rightarrow s$ through one round is possible and, for each of them, we now need to exhaust all truncated differential characteristics on $R-1$ rounds with at most $b - |s|$ active S-boxes ending with state s'.

The complexity of the algorithm in the single key model is very low, and we experimentally counted around $(R-1) \times 2^{20}$ simple operations for R rounds. A naive solution to search for truncated representations in the **TK1**, **TK2** and **TK3** models would be to apply the previous algorithm for each possible configuration of the key. While for **TK1** this would only increase the overall complexity by a factor 2^{16}, the search would not be practical for both the **TK2** and **TK3** models. Indeed, because of the possible cancellations occurring in the round keys, the number of configurations is very high:

$$\left(\sum_{k=0}^{8}\binom{8}{k}\left(\sum_{i=0}^{tk-1}\binom{\lfloor (R-1)/2 \rfloor}{i}\right)^k\right)\left(\sum_{k=0}^{8}\binom{8}{k}\left(\sum_{i=0}^{tk-1}\binom{\lceil (R-1)/2 \rceil}{i}\right)^k\right).$$

For instance, for $R = 30$, there are more than 2^{64} configurations in the **TK2** model.

In the following we present the first practical algorithm which tackles down the problem for the **TK** models without relying on a black box solver as MILP, SAT or CP solvers. Actually this is the only algorithm fast enough to generate all the Step 1 solutions required

Algorithm 6 Search for the best truncated representation (**SK**).

for each state s **do**
    $M[s] \longleftarrow$ list of states s' reachable from s through one round
end for

for each state s **do**
    $C[0][s] \longleftarrow$ number of active cells of s
end for
for $1 \leq r < R$ **do**
    foreach *state* s **do** $C[r][s] \longleftarrow \infty$
    for each state s **do**
        for each state s' in $M[s]$ **do**
            $c \longleftarrow C[r-1][s] +$ number of active cells of s'
            if $c < C[r][s']$ **then**
                $C[r][s'] \longleftarrow c$
            end if
        end for
    end for
end for
Return C

to perform the Step 2. Indeed, the best differential characteristic is rarely based on the truncated differential characteristics minimizing the number of active S-boxes and thus we need to generate a large number of truncated characteristics to find the one instantiating with the best probability. As we will explain in Section 3.3.2, all other approaches we tried to generate them failed.

The idea of our ad-hoc method is quite similar to the one used in the single key model. Actually, to compute the minimal number of active S-boxes at round $r+1$ we only need to know the minimal number of active S-boxes for each possible state at round r together with the number of cancellations for each key cell occurred so far. Indeed, we do not need to know at which rounds the cancellations occurred but only how many times they did. A simplified version of this algorithm is described in Algorithm 7. The most important part is related to the variable *cancelled* which count how many times each key cell is cancelled through the encryption. It is a vector of 16 cells, each cell taking values among $\{0, 1, \ldots, x-1, r\}$ for the **TKx** model. The main advantage of our representation is that at each step of the algorithm, $C[r][s]$ contains at most $(x+1)^{16}$ elements for the **TKx** model which is much lower than the number of possible sequences of round keys.

Finally we introduce a new improvement which greatly speeds up the search procedure. It is based on the so-called *early abort technique* principle and the idea is to handle the key cell by cell. Indeed, we expect that the best truncated differential characteristics do not involve many active cells in the round key and so we want to quickly cut those branches during the search. To do so we first pick a key cell and guess whether it is active or not. At this step we have not decided yet if any cancellations occur nor their positions but only if it is always 0 or at least once 1. Then we apply the algorithm partially and guess another key cell if and only if it seems possible to find a truncated differential characteristic with a small enough number of active S-boxes. More precisely, along the search we have the relation $y = x \oplus k$ where k is the round key. We introduce a new 16-bit variable g such that $g_i = 0$ if we made a choice for bit i of k and 1 otherwise. To compute the possible truncated transitions from x to y through k for all the possible key (according to g) we can restrict ourself at looking at the possible truncated transitions from $(x|g)$ to y through $(k|g)$ where $|$ is the bitwise OR. Indeed, we use the fact that in truncated setting $1 \oplus 1$ is 0 or 1 and thus our technique allows to handle all the possible keys by looking only at few transitions.

Algorithm 7 Search for the best truncated representation (**TK**).

for each state s, round key k **do**
    $M[k][s] \longleftarrow$ list of states s' reachable from s and k through one round
end for
for each state s **do**
    $C[0][s] \longleftarrow \{(\text{number of active cells of } s, 0)\}$
end for
for $r = 1$ **do**$R - 1$
    foreach *state* s **do** $C[r][s] \longleftarrow \emptyset$
    for each state s **do**
        for each (cost, cancelled) $\in C[r-1][s]$ **do**
            for each round key k compatible with cancelled **do**
                for state s' in $M[k][s]$ **do**
                    $c \longleftarrow$ cost + number of active cells of s'
                    $C[r][s'] \longleftarrow C[r][s'] \cup \{(c, \text{update}(\mathit{cancelled}, k))\}$
                end for
            end for
        end for
    end for
    foreach *state* s **do** keepOptimals($C[r][s]$)
end for
Return C

3.4.3 Results for Step 1

For Step 1, we run our ad-hoc tool on the four attack scenarios (**SK**, **TK1**, **TK2**, and **TK3**) when varying the number of rounds between 3 and 20. We conducted all our experiments on our server composed of 2× AMD EPYC 7742 64-Core and 1TB of RAM. In particular, we were able to complete the security analysis made in [Bei+16; Alf+18] and claim that the minimal number of active S-boxes in **TK1** for 28, 29 and 30 rounds are 105, 109 and 113 respectively (as shown in Table 3.4).

# Rounds	28	29	30
TK1	105	109	113

Table 3.4 – Lower bounds on the number of active S-boxes in SKINNY.

However, the optimal solution of Step 2, in terms of differential characteristic probability, could be obtained for a number of active S-boxes which is not the optimal one. Hereafter, we denote Obj_{Step1} the number of active S-boxes we consider when solving the problems. For example, assume that, when processing Step 2, one obtains a differential characteristic with the best probability equal to $2^{-3\times 6} = 2^{-18}$ with $Obj_{Step1} = 6$ and whereas the optimal differential probability of the S-box is 2^{-2}. It means that one has to test all solutions outputted by Step 1 until $Obj_{Step1} < 18/2 = 9$ to be sure that none has a better differential characteristic probability. This is exactly what happened for the case of SKINNY-128 in the **TK** models. We only want to stress here that computing the optimal bounds is often not enough and we need to go further. However, increasing the value of Obj_{Step1} induces an increase of the possible number of Step 1 solutions as illustrated in the third column of Table 3.6. As one can see, this number of solutions tends to grow exponentially when we increase v. For example, for SKINNY-128 with 14 rounds in the **TK1** model, for the optimal value $v^* = 45$, Step 1 outputs only 3 solutions; whereas we have 897 solutions for $v = v^* + 5 = 50$; 137 019 solutions for $v = v^* + 10 = 55$ and finally 7 241 601 solutions for $v = 59$. So, the time required to output all those Step 1 solutions and the time required for the Step 2 computations on 1 solution outputted by the Step 1 become the bottleneck of the overall process.

3.5 Modeling Step 2 with CP

The aim of Step 2 is to try to instantiate the abstracted solutions provided by Step 1 while maximizing the probability of the differential characteristic. Thus, Step 2 takes as input a solution of Step 1 with the objective function of maximizing the probability of the differential characteristic. However, some solutions of Step 1 could not be instantiated in Step 2 as refining the abstraction level of Step 2 will induce *non-consistent* solutions. In the literature, this step has been modeled using ad-hoc methods [BN10], MILP [Abd+17], SAT [SWW18] or CP [Ger+20]. As MILP [Abd+17] and SAT [SWW18] seem to hardly scale due to prohibitive computational times (linked with the size of the 8-bit S-boxes that must be represented in the form of linear inequalities or of clauses), we focus here on a dedicated CP method implemented using the Choco solver [PFL16]. We also provide, in the second part of this section, the results we obtain when instantiating the differential characteristics in the 4 attack scenarios.

3.5.1 Constraint Programming

Although less usual than MILP to tackle cryptanalytic problems, CP has already been used in e.g. [GMS16; ENP19]. We recall some basic principles of CP and we refer the reader to [RBW06] for more details.

CP is used to solve Constraint Satisfaction Problems (CSPs). A CSP is defined by a triple (X, D, C) such that $X = \{x_1, x_2, \ldots, x_n\}$ is a finite set of variables, D is a function that maps every variable $x_i \in X$ to its domain $D(x_i)$ and $C = \{c_1, c_2, \ldots, c_m\}$ is a set of constraints. $D(x_i)$ is a finite ordered set of integer values to which the variable x_i can be assigned to, whereas c_j defines a relation between some variables $vars(c_j) \subseteq X$. This relation restricts the set of values that may be assigned simultaneously to $vars(c_j)$. Each constraint is equipped with a filtering algorithm which removes from the domains of $vars(c_j)$, the values that cannot satisfy c_j.

In CP, constraints are classified in two categories. *Extensional constraints*, also called *table constraints*, explicitly define the allowed (or forbidden) tuples of the relation. *Intentional constraints* define the relation using mathematical operators. For instance, in a CSP with $X = \{x_1, x_2, x_3\}$ such that $D(x_1) = D(x_2) = D(x_3) = \{0, 1\}$, a constraint ensuring that the sum of the variables in X is different from 1 can be either expressed in extension (1) or in intention (2):

1. $\textsc{Table}(\langle x_1, x_2, x_3 \rangle, \langle (0,0,0), (0,1,1), (1,0,1), (1,1,0), (1,1,1) \rangle)$

2. $x_1 + x_2 + x_3 \neq 1$

Actually, any intentional constraint can be encoded with an extensional one provided enough memory space, and conversely [Dem+16b]. However, they may offer different performances.

The purpose of a CSP is to find a *solution*, i.e. an assignment of all variables to a value from their respective domains such that all the constraints are simultaneously satisfied. When looking for a solution, a two-phase mechanism is operated: the *search space exploration* and the *constraint propagation*. The exploration of the search space is processed using a *depth-first search*. At each step, a decision is taken, i.e. a non-assigned variable is selected and its domain is reduced to a singleton. This modification requires to check the satisfiability of all the constraints. This is achieved thanks to constraint propagation which applies each constraint filtering algorithm. Any application may trigger modifications in turn; the propagation ends when either no modification occurs and all constraints are satisfied or a failure is thrown, *i.e.*, at least one constraint cannot be satisfied. In the former case, if all variables are assigned, a solution has been found. Otherwise a new decision is taken and the search is pursued. In the latter case, a backtrack to the first refutable decision is made and the search is resumed.

Turning a CSP into a Constrained Optimisation Problem (COP) is done by adding an objective function. Such a function is defined over variables of X, the purpose is then to find the solution that optimizes the objective function. Finding the optimal solution is done by repeatedly applying the two-phase mechanism above, and by adding a *cut* on the objective function that prevents from finding a same cost solution in the future.

3.5.2 Modeling Step 2 with CP

Given a Boolean solution for Step 1, Step 2 aims at searching for the byte-consistent solution with the highest (related-tweakey) differential characteristic probability (or proving that there is no byte-consistent solution). In this section, Model 1 describes the CP model we used for `SKINNY`-128 (**SK**). Actually, the ones used to model the other variants, as well as `SKINNY`-64 are rather similar.

For each Boolean variable $\Delta X_{r,i,j}$ of Step 1, we define an integer variable $\delta X_{r,i,j}$. The domain of this integer variable depends on the value of the Boolean variable in the Step 1 solution: If $\Delta X_{r,i,j} = 0$, then the domain is $D(\delta X_{r,i,j}) = \{0\}$ (*i.e.*, $\delta X_{r,i,j}$ is also assigned to 0); otherwise, the domain is $D(\delta X_{r,i,j}) = [1, 255]$. For each byte that passes through

Model 1 Formulation of **SK** Step2.

Minimize

$$Obj_{Step2} = \sum_{r=1}^{R}\sum_{i=1}^{4}\sum_{j=1}^{4} P_{r,i,j} \tag{3.1}$$

subject to

$$20 \times n \leq \sum_{r=1}^{R}\sum_{i=1}^{4}\sum_{j=1}^{4} P_{r,i,j} \leq \min(70 \times n, O^*) \tag{3.2}$$

$$\forall r \in 1..R, \forall i \in 1..4, \forall j \in 1..4$$
$$\begin{cases} \delta X_{r,i,j} = 0 \wedge \delta SB_{r,i,j} = 0 \wedge P_{r,i,j} = 0 & \text{if } \Delta X_{r,i,j} = 0 \\ \delta X_{r,i,j} \geq 1 \wedge \delta SB_{r,i,j} \geq 1 \wedge P_{r,i,j} \geq 20 & \text{otherwise} \end{cases} \tag{3.3}$$

$$\forall r \in 1..R, \forall i \in 1..4, \forall j \in 1..4$$
$$\textsc{Table}(\langle \delta X_{r,i,j}, \delta SB_{r,i,j}, P_{r,i,j} \rangle, \langle \text{SBox} \rangle) \quad \text{if } \Delta X_{r,i,j} \neq 0 \tag{3.4}$$

$$\forall r \in 1..R-1, \forall j \in 1..4 \qquad \delta SB_{r,0,j} = \delta X_{r+1,1,j} \tag{3.5}$$

$$\forall r \in 1..R-1, \forall j \in 1..4$$
$$\begin{cases} \delta SB_{r,2,(2+j)\%4} = \delta X_{r+1,2,j} & \text{if } \Delta SB_{r,1,(3+j)\%4} = 0 \\ \delta SB_{r,1,(3+j)\%4} = \delta X_{r+1,2,j} & \text{if } \Delta SB_{r,2,(2+j)\%4} = 0 \\ \delta SB_{r,1,(3+j)\%4} = \delta SB_{r,2,(2+j)\%4} & \text{if } \Delta X_{r+1,2,j} = 0 \\ \textsc{Table}(\left\langle \delta SB_{r,1,(3+j)\%4}, \delta SB_{r,2,(2+j)\%4}, \delta X_{r+1,2,j} \right\rangle, \langle \text{XOR} \rangle) & \text{otherwise} \end{cases} \tag{3.6}$$

$$\forall r \in 1..R-1, \forall j \in 1..4$$
$$\begin{cases} \delta SB_{r,2,(2+j)\%4} = \delta X_{r+1,3,j} & \text{if } \Delta SB_{r,0,j} = 0 \\ \delta SB_{r,0,j} = \delta X_{r+1,3,j} & \text{if } \Delta SB_{r,2,(2+j)\%4} = 0 \\ \delta SB_{r,0,j} = \delta SB_{r,2,(2+j)\%4} & \text{if } \Delta X_{r+1,3,j} = 0 \\ \textsc{Table}(\left\langle \delta SB_{r,0,j}, \delta SB_{r,2,(2+j)\%4}, \delta X_{r+1,3,j} \right\rangle, \langle \text{XOR} \rangle) & \text{otherwise} \end{cases} \tag{3.7}$$

$$\forall r \in 1..R-1, \forall j \in 1..4$$
$$\begin{cases} \delta X_{r+1,0,j} = \delta X_{r+1,3,j} & \text{if } \Delta SB_{r,3,(1+j)\%4} = 0 \\ \delta SB_{r,3,(1+j)\%4} = \delta X_{r+1,3,j} & \text{if } \Delta X_{r+1,0,j} = 0 \\ \delta SB_{r,3,(1+j)\%4} = \delta X_{r+1,0,j} & \text{if } \Delta X_{r+1,3,j} = 0 \\ \textsc{Table}(\left\langle \delta SB_{r,3,(1+j)\%4}, \delta X_{r+1,0,j}, \delta X_{r+1,3,j} \right\rangle, \langle \text{XOR} \rangle) & \text{otherwise} \end{cases} \tag{3.8}$$

where $\forall r \in R..n$, $\forall i \in 1..4$, $\forall j \in 1..4$,

$$\delta X_{r,i,j} \in 0..255,\ \delta SB_{r,i,j} \in 0..255,\ P_{r,i,j} \in \{0, 20, .., 70\},$$

and $\langle \text{XOR} \rangle$ encodes $\oplus$ relation and $\langle \text{SBox} \rangle$ the S-box constraint.

an S-box, we define an integer variable $\delta SB_{r,i,j}$ which corresponds to the difference after the S-box. Its domain is $\{0\}$ if $\Delta X_{r,i,j}$ is assigned to 0 in the Step 1 solution; otherwise, it is $D(\delta SB_{r,i,j}) = [1, 255]$. This is expressed in (3.3) of Model 1.

Finally, as we look for a byte-consistent solution with maximal probability, we also add an integer variable $P_{r,i,j}$ for each byte in an S-box: this variable corresponds to the absolute value of the base 2 logarithm of the probability of the transition through the S-box. Actually, a factor 10 has been applied to avoid considering floats. Thus we define a TABLE constraint (3.4) composed of valid triplets of the form $(\delta X_{r,i,j}, \delta SB_{r,i,j}, P_{r,i,j})$. Note that these triplets only contain non-zero values and that $P_{r,i,j}$ takes only 2 different values for the 4-bit S-box (`SKINNY`-64) and 7 different values for the 8-bit S-box (`SKINNY`-128). There are roughly 2^{14} triplet elements in the Table constraint for the `SKINNY`-128 case. As the S-box layer is the only non-linear layer, the other operations could be directly implemented in a deterministic way at the cell level. The associated constraints thus follow the `SKINNY`-128 linear operations. When possible, i.e. when one element is known to be zero, we replace XOR constraints (encoded using TABLE constraints) by a simple equality constraint. This corresponds to TABLE constraints (3.5), (3.6), (3.7) and (3.8) in Model 1.

The overall goal is finally to find a byte-consistent solution which maximizes differential characteristic probability. Thus, we define an integer variable Obj_{Step2} to minimize the sum of all $P_{r,i,j}$ variables (3.1). This value mainly depends on the number of S-boxes outputted by Step1 Obj_{Step1} and can be bounded to $[\![20 \cdot Obj_{Step1}, 70 \cdot Obj_{Step1}]\!]$ (3.2).

The differences for the models **TK1**, **TK2** and **TK3** are the modeling of the XORs induced by the lanes of the tweakey through XOR table constraints. Each XOR constraint depicted in Model 1 provides high quality filtering but requires 65536 tuples to be stored which results in prohibitive memory usage. This may limit the number of threads that can be used for the resolution, which was the case for **TK2** and **TK3**. To get around this issue, we encoded the XOR constraint in intention (by defining filtering rules), providing a more memory efficient algorithm, at the expense of filtering strength (see Model 2). It can be sum up as using the two most constraint variables to sieve the reachable values for the third variable. This last choice was applied for **TK2** and **TK3** (`SKINNY`-128 only). We also rely on TABLE constraints to model the LFSRs applied on TK2 and TK3.

Concerning the search space strategy, for the **TK2** and the **TK3** attack settings, the Step 1 only outputs the truncated value of the sum of the TKi. Thus, the search space strategy first looks at the cancellation places of the sum of the TKi and then instantiates

Model 2 The XOR constraint in intention

Model for $a \oplus b \oplus c = 0$ with $a, b, c \in X_A, X_B, X_C$
where a and b are more constraint than c.

$$\begin{aligned} X'_C &= \varnothing \\ \forall x \in X_A, \forall y \in X_B & \\ X'_C &= X'_C \bigcup \{x \oplus y\} \\ X_C &= X_C \bigcap X'_C \end{aligned} \tag{3.9}$$

the TKi values according to those positions. For the **TK1** setting, we simply apply the default Choco-solver strategy.

Concerning the parallelization, we affect one solution outputted by Step 1 per thread and we share between the threads the value of Obj_{Step2}.

3.5.3 Step 2 Performance Results

We run our Step 2 model on the two versions of `SKINNY` (`SKINNY`-64 and `SKINNY`-128) using our CP models written in Choco-solver. We conduct all our experiments on our server composed of 2× AMD EPYC 7742 64-Core and 1TB of RAM. All the reported times are `real` system times.

Up to our knowledge, we only found [Abd+17] that gives time results concerning finding the best **SK** differential characteristic probability on `SKINNY`-128 using a MILP tool based on Gurobi.

More precisely, the authors say: "In our experiments, we used Gurobi Optimizer with Xeon Processor E5-2699 (18 cores) in 128 GB RAM." and, for 13 rounds, "in our environment, the test of 6 classes [Step 1 solutions with 58 active S-boxes without symmetries] finished in 16 days. Finally, it is proven that the tight bound on the probability of differential characteristic for 13 rounds is 2^{-123}" in the **SK** model.

Regarding the **TK** models, the best known results were obtained by Liu *et al.* also using MILP models [LGS17]. They could only find the best differential characteristics up to 7, 9 and 13 rounds for **TK1**, **TK2** and **TK3** respectively.

Results for `SKINNY-64`.

We sum up in Table 3.5 all the results we obtain for `SKINNY`-64 in the four different attack models (**SK**,**TK1**,**TK2** and **TK3**). The overall time, in this case, is not a bottleneck.

We only give results concerning number of rounds that are at the limit (just under and just upper) when regarding the number of active S-boxes which is equal to 32 in the case of SKINNY-64 as the state size is 64 bits and as the best differential probability of the S-box is equal to 2^{-2}. Thus, the best overall differential characteristic probability must be under 2^{-64}.

Note that sometimes, we need to browse several Obj_{Step1} bounds to find the optimal differential characteristic probability when the number of rounds is fixed. Indeed, we need to proactively adapt the probability bound we found. For example, in the case of **TK2** SKINNY-64 with 13 rounds, the optimal Obj_{Step1} is equal to 25 and when providing the Step 2 process with this Obj_{Step1} bound, we find a best differential characteristic probability equal to 2^{-55}. Thus, we need to enumerate all the Step 1 solutions with $Obj_{Step1} = 26$ and $Obj_{Step1} = 27$ to be sure that the previous probability is really the best one. Then, before running again Step 2 on those new results we adapt the best probability to the new bound equal to 2^{-55} instead of the old bound equal to 2^{-64}.

We also provide in Appendix 3.A the details of the best found differential characteristics.

	Nb Rounds	Obj_{Step1}	Nb sol. Step 1	Step 2 time	Best Pr
SK	7	26	2	1s	2^{-52}
SK	8	36	17	1s	$< 2^{-64}$
TK1	10	23	1	1s	2^{-46}
TK1	11	32	2	1s	$= 2^{-64}$
TK2	13	$25 \rightarrow 27$	10	1s	2^{-55}
TK2	14	31	1	1s	$< 2^{-64}$
TK3	15	$24 \rightarrow 26$	46	2s	2^{-54}
TK3	16	$27 \rightarrow 31$	87	4s	$= 2^{-64}$
TK3	17	31	2	1s	$< 2^{-64}$

Table 3.5 – Overall results concerning SKINNY-64 in the four attack models. Step 2 time corresponds to the Step 2 time taken over all Step 1 solutions when Obj_{step1} takes the values precise in the first column. Best Pr corresponds to the best found probability of a differential characteristic.

Results for SKINNY-128.

In the same way, we provide in Table 3.6 the best differential characteristic probability with the total time required for this search for the 4 different attack models. As one

can see, we also verify all the possible values for Obj_{Step1} for a given number of rounds, depending on the probability value previously found. Thus, this time, the number of solutions outputted by Step 1 could be huge when we move away from the optimal Step 1 value v^*. However, as the time spent to solve one solution is reasonable (at least when considering **SK** and **TK1**), our model scales reasonably well: the worst case requires 25 days of `real` time on our server on 8 threads and 31 GB of RAM[2]. Our **TK2** and **TK3** models are based on XOR constraints encoded in intention (and not using tables) and these experiences have been launched using the 128 threads of our server.

Concerning **TK2** and **TK3**, we were not able to perform all the computations due to the huge number of Step 1 solutions. Hence we decided to handle only the Step 1 solutions with exactly one active byte in the round keys in order to limit the number of truncated characteristics to instantiate. Those results are given in Table 3.7. We provide in Appendix 3.B the best **TK2** differential characteristic we found for 16 rounds, and the best **TK3** differential characteristic we found for 17 rounds. Note that we do not know if these differential characteristics are optimal in terms of probability as we were not able to test all the solutions Step 1.

Lessons learnt.

The overall gap is not to find the optimal value of $Obj_{Step1} = v^*$ for a given number of rounds and to enumerate the corresponding overall solutions if the Step 1 model is sufficiently tight. The real gap is if the value obtained for Obj_{Step2} (here equal to $2 \times v^*$ as the best differential probability for the S-box is equal to 2^{-2}) is far from the optimal bound then we have to increase Obj_{Step1} up to the bound $\lfloor Obj_{Step2}/2 \rfloor$. Further we are from v^* in the Step 1 resolution, more numerous are the Step 1 solutions (in fact this number grows exponentially as could be seen in Table 3.6). Thus, the time for the Step 2 resolution becomes the bottleneck.

Conclusion

In this chapter, we improve the security bounds regarding differential characteristics search on the block cipher `SKINNY`. As usually done, we have divided the search procedure into two steps: Step 1 which abstracts the difference values into Boolean variables

2. It seems that the use of the 128 threads was prohibited by the memory usage of XOR tables (i.e. XOR in extension).

and finds the truncated characteristics with the smallest number of active S-boxes; and Step 2 which inputs the results of Step 1 to output the best possible probability instantiating the abstract solutions outputted by Step 1. Of course, each solution of Step 1 could not always be instantiated in Step 2.

For Step 1, an ad-hoc method which heavily uses the structure of the problem is proposed. For solving Step 2, we have implemented a Choco-solver model. Regarding Step 2, our Choco-solver model is much faster than any other approaches. It allowed us to find, for the first time, the best (related-tweakey) differential characteristics in the **TK1** model up to 14 rounds for SKINNY-128 and to show there is no differential trail on 15 rounds with a probability better than 2^{-128}. Regarding the **TK2** model, we were able to find the best differential trails up to 16 rounds. For **TK3**, we are able to exhibit a differential characteristic up to 17 rounds. Note that in [LGS17] Liu *et al.* were only able to reach 7 and 9 rounds in the **TK1** and **TK2** model respectively. Our approach is thus an important improvement.

3.A Best (related-tweakey) differential characteristics for SKINNY-64

The best **SK** differential characteristics on 7 rounds of SKINNY-64 with probability equal to 2^{-52} is given in Table 3.8. The best **TK1** differential characteristics on 10 rounds of SKINNY-64 with probability equal to 2^{-46} is given in Table 3.9. The Best **TK2** differential characteristics on 13 rounds of SKINNY-64 with probability equal to 2^{-55} is given in Table 3.10. Best **TK3** differential characteristics on 15 rounds of SKINNY-64 with probability equal to 2^{-54} is given in Table 3.11.

	Nb Rounds	Obj_{step1}	Nb sol. Step 1	Step 2 time	Best Pr
SK	9	$41 \rightarrow 43$	52	16s	2^{-86}
SK	10	$46 \rightarrow 48$	48	11s	2^{-96}
SK	11	$51 \rightarrow 52$	15	4s	2^{-104}
SK	12	$55 \rightarrow 56$	11	6s	2^{-112}
SK	13	$58 \rightarrow 61$	18	2m27s	2^{-123}
SK	14	$61 \rightarrow 63$	6	21s	$\leq 2^{-128}$
TK1	8	$13 \rightarrow 16$	14	4s	2^{-33}
TK1	9	$16 \rightarrow 20$	6	3s	2^{-41}
TK1	10	$23 \rightarrow 27$	6	4s	2^{-55}
TK1	11	$32 \rightarrow 36$	531	37s	2^{-74}
TK1	12	$38 \rightarrow 46$	186 482	213m	2^{-93}
TK1	13	$41 \rightarrow 53$	2 385 482	2 days	$2^{-106.2}$
TK1	14	$45 \rightarrow 59$	11 518 612	20 days	2^{-120}
TK1	15	$49 \rightarrow 63$	7 542 053	25 days	$\leq 2^{-128}$
TK2	9	$9 \rightarrow 10$	7	3s	2^{-20}
TK2	10	$12 \rightarrow 17$	132	11s	$2^{-34.4}$
TK2	11	$16 \rightarrow 25$	4203	6m	$2^{-51.4}$
TK2	12	$21 \rightarrow 35$	1 922 762	512m	$2^{-70.4}$
TK2	19	$52 \rightarrow 63$	530 693	280m	$\leq 2^{-128}$
TK3	10	6	3	3s	2^{-12}
TK3	11	10	3	10s	2^{-21}
TK3	12	$13 \rightarrow 17$	373	1h	$2^{-35.7}$
TK3	13	$16 \rightarrow 25$	34 638	85h	$2^{-51.8}$
TK3	23	$55 \rightarrow 63$	47 068	11h	$\leq 2^{-128}$

Table 3.6 – Overall results concerning SKINNY-128 in the four attack models. Step 2 time corresponds to the Step 2 time taken over all solutions of *Step1-enum* when Obj_{step1} takes the values precise in the first column. Best Pr corresponds to the best found probability of a differential characteristic.

	Nb Rounds	Obj_{step1}	Best Pr
TK2	13	$25 \rightarrow 44$	$2^{-86.2}$
TK2	14	$31 \rightarrow 54$	$\geq 2^{-105.8}$
TK2	15	$35 \rightarrow 56$	$\geq 2^{-113.8}$
TK2	16	$40 \rightarrow 63$	$\geq 2^{-127.6}$
TK3	14	$19 \rightarrow 33$	2^{-67}
TK3	15	$24 \rightarrow 40$	2^{-81}
TK3	16	$27 \rightarrow 48$	2^{-98}
TK3	17	$31 \rightarrow 54$	2^{-110}
TK3	19	$43 \rightarrow 63$	$\leq 2^{-128}$
TK3	20	$45 \rightarrow 63$	$\leq 2^{-128}$
TK3	21	$48 \rightarrow 63$	$\leq 2^{-128}$
TK3	22	$51 \rightarrow 63$	$\leq 2^{-128}$

Table 3.7 – Overall results concerning `SKINNY`-128 with exactly one active cell in the tweakey.

Round	$\delta X_i = X_i \oplus X'_i$ (before `SB`)	δSBX_i (after `SB`)	Pr(States)
$i = 1$	0040 4444 4440 4400	0020 2222 2220 2200	$2^{-2 \cdot 10}$
2	0000 0020 0200 2002	0000 0010 0100 1001	$2^{-2 \cdot 4}$
3	0010 0000 0000 0001	0080 0000 0000 0008	$2^{-2 \cdot 2}$
4	0000 0080 0000 0080	0000 0040 0000 0040	$2^{-2 \cdot 2}$
5	0400 0000 0004 0000	0200 0000 0002 0000	$2^{-2 \cdot 2}$
6	0000 0200 0200 0000	0000 0100 0100 0000	$2^{-2 \cdot 2}$
7	0001 0000 0011 0001	0008 0000 0088 0008	$2^{-2 \cdot 4}$

Table 3.8 – The Best **SK** differential characteristics on 7 rounds of `SKINNY-64` with probability equal to 2^{-52}. The four words represent the four rows of the state and are given in hexadecimal notation.

Round	$\delta X_i = X_i \oplus X'_i$ (before `SB`)	δSBX_i (after `SB`)	$\delta TK1_i$	Pr(States)
$i = 1$	0000 0002 0020 0200	0000 0001 0010 0100	1000 0000 0B80 0000	$2^{-2 \cdot 3}$
2	1000 1000 0000 0000	B000 8000 0000 0000	B000 8000 1000 0000	$2^{-2 \cdot 2}$
3	0000 0000 0000 0000	0000 0000 0000 0000	0010 0000 B000 8000	–
4	0010 0010 0000 0010	00B0 00A0 0000 00B0	00B0 0080 0010 0000	$2^{-2 \cdot 3}$
5	0B00 0000 0002 0000	0100 0000 0001 0000	0000 1000 00B0 0080	$2^{-2 \cdot 2}$
6	0000 0100 0000 0000	0000 0800 0000 0000	0000 B800 0000 1000	$2^{-2 \cdot 1}$
7	0000 0000 0B00 0000	0000 0000 0100 0000	0000 0010 0000 B800	$2^{-2 \cdot 1}$
8	0001 0000 0000 0001	0008 0000 0000 0008	0008 00B0 0000 0010	$2^{-2 \cdot 2}$
9	0080 0000 000B 0000	0040 0000 0001 0000	0000 0100 0008 00B0	$2^{-2 \cdot 2}$
10	0140 0040 0110 0140	0820 0020 0880 0820	0000 0B08 0000 0100	$2^{-2 \cdot 7}$

Table 3.9 – The Best **TK1** differential characteristics on 10 rounds of `SKINNY-64` with probability equal to 2^{-46}. The four words represent the four rows of the state and are given in hexadecimal notation.

Round	$\delta X_i = X_i \oplus X'_i$ (before SB)	δSBX_i (after SB)	$\delta TK1_i$	$\delta TK2_i$	Pr(States)
$i = 1$	0000 8200 0080 0000	0000 4100 0040 0000	0000 0008 0502 0000	0000 000C 060C 0000	$2^{-2\cdot3}$
2	4000 0000 0410 4000	2000 0000 02A0 2000	5000 0002 0000 0008	D000 0008 0000 000C	$2^{-2\cdot4}$
3	0000 A000 0002 0002	0000 6000 0006 0003	0800 0000 5000 0002	0800 0000 D000 0008	$2^{-2\cdot3}$
4	0630 0000 0000 0600	03F0 0000 0000 0100	0250 0000 0800 0000	01A0 0000 0800 0000	$2^{-3\cdot3}$
5	1000 0000 0000 0000	9000 0000 0000 0000	8000 0000 0250 0000	1000 0000 01A0 0000	2^{-2}
6	0000 0000 0000 0000	0000 0000 0000 0000	2000 5000 8000 0000	2000 5000 1000 0000	–
7	0000 0000 0000 0000	0000 0000 0000 0000	0080 0000 2000 5000	0020 0000 2000 5000	–
8	00A0 00A0 0000 00A0	0060 0050 0000 0050	0020 0050 0080 0000	0040 00B0 0020 0000	$2^{-2\cdot3}$
9	0500 0000 000B 0000	0C00 0000 000C 0000	0000 8000 0020 0050	0000 4000 0040 00B0	$2^{-3\cdot2}$
10	0000 0C00 0000 0000	0000 0200 0000 0000	0000 2500 0000 8000	0000 9700 0000 4000	2^{-2}
11	0000 0000 0B00 0000	0000 0000 0100 0000	0000 0080 0000 2500	0000 0090 0000 9700	2^{-2}
12	0001 0000 0000 0001	000A 0000 0000 0008	0005 0020 0000 0080	000F 0030 0000 0090	$2^{-2\cdot2}$
13	0080 0000 0001 0000	0040 0000 0008 0000	0000 0800 0005 0020	0000 0300 000F 0030	$2^{-2\cdot2}$

Table 3.10 – The Best **TK2** differential characteristics on 13 rounds of SKINNY-64 with probability equal to 2^{-55}. The four words represent the four rows of the state and are given in hexadecimal notation.

Round	$\delta X_i = X_i \oplus X'_i$ (before SB)	δSBX_i (after SB)	$\delta TK1_i$	$\delta TK2_i$	$\delta TK3_i$	Pr(States)
$i = 1$	0000 0001 4000 0004	0000 0008 2000 0002	0000 080D 0000 0800	0000 0408 0000 0500	0000 0E0D 0000 0C00	$2^{-2\cdot3}$
2	0000 0000 0000 0020	0000 0000 0000 0010	0008 0000 0000 080D	000B 0000 0000 0408	000E 0000 0000 0E0D	2^{-2}
3	010D 000D 0000 000D	0A0E 0002 0000 0002	0D08 0000 0008 0000	0109 0000 000B 0000	060F 0000 000E 0000	$2^{-2\cdot3}2^{-3}$
4	0020 0000 2000 0000	0030 0000 3000 0000	0000 0008 0D08 0000	0000 0007 0109 0000	0000 000F 060F 0000	$2^{-2\cdot2}$
5	0000 0030 0030 0000	0000 00C0 00C0 0000	D000 0008 0000 0008	2000 0003 0000 0007	3000 0007 0000 000F	$2^{-3\cdot2}$
6	0000 C000 000C 0000	0000 2000 0002 0000	0800 0000 D000 0008	0F00 0000 2000 0003	0700 0000 3000 0007	$2^{-2\cdot2}$
7	0200 0000 0000 0200	0500 0000 0000 0300	08D0 0000 0800 0000	0640 0000 0F00 0000	0B90 0000 0700 0000	$2^{-2\cdot2}$
8	3000 0000 0000 0000	D000 0000 0000 0000	8000 0000 08D0 0000	E000 0000 0640 0000	B000 0000 0B90 0000	2^{-3}
9	0000 0000 0000 0000	0000 0000 0000 0000	8000 D000 8000 0000	D000 9000 E000 0000	5000 4000 B000 0000	–
10	0000 0000 0000 0000	0000 0000 0000 0000	0080 0000 8000 D000	00C0 0000 D000 9000	0050 0000 5000 4000	–
11	0010 0010 0000 0010	0080 0090 0000 00A0	0080 00D0 0080 0000	00A0 0030 00C0 0000	00A0 0020 0050 0000	$2^{-2\cdot3}$
12	0A00 0000 0005 0000	0A00 0000 000A 0000	0000 8000 0080 00D0	0000 8000 00A0 0030	0000 A000 00A0 0020	$2^{-2}2^{-3}$
13	0000 0A00 0000 0000	0000 0A00 0000 0000	0000 8D00 0000 8000	0000 5600 0000 8000	0000 D100 0000 A000	2^{-3}
14	0000 0000 0000 0000	0000 0000 0000 0000	0000 0080 0000 8D00	0000 0010 0000 5600	0000 00D0 0000 D100	–
15	0000 0000 0004 0000	0000 0000 0002 0000	000D 0080 0000 0080	000D 00B0 0000 0010	0008 0060 0000 00D0	2^{-2}

Table 3.11 – The Best **TK3** differential characteristics on 15 rounds of SKINNY-64 with probability equal to 2^{-54}. The four words represent the four rows of the state and are given in hexadecimal notation.

3.B Best (related-tweakey) differential characteristics for SKINNY-128

Concerning the best **SK** differential characteristics on 13 rounds of SKINNY-128, We obtain the same best **SK** differential characteristics on 13 rounds of SKINNY-128 with probability equal to 2^{-123} given in Table 11 of Appendix D of [Abd+17]. The best **TK1** differential characteristics on 14 rounds of SKINNY-128 with probability equal to 2^{-120} is given in Table 3.12. The best **TK2** differential characteristics on 16 rounds of SKINNY-128 with probability equal to $2^{-127.6}$ we found is given in Table 3.13. The best **TK3** differential characteristics on 17 rounds of SKINNY-128 with probability equal to 2^{-110} we found is given in Table 3.14.

Round	$\delta X_i = X_i \oplus X'_i$ (before SB)	δSBX_i (after SB)	$\delta TK1_i$	Pr(States)
$i = 1$	02000002 00000200 00020000 00020040	08000008 00000800 00080000 00080004	00000000 00000000 01000000 01000000	$2^{-2\cdot6}$
2	00000400 08000008 00000000 08000000	00000100 10000010 00000000 10000000	00000100 00000000 00000000 00000000	$2^{-2\cdot4}$
3	00000010 00000000 10100000 00000000	00000040 00000000 40400000 00000000	00000000 00000000 00000100 00000000	$2^{-2\cdot3}$
4	00004000 00000040 00004040 00004000	00000400 00000004 00000404 00000400	00000000 01000000 00000000 00000000	$2^{-2\cdot5}$
5	04000400 00000400 00050000 04040400	05000500 00000100 00050000 05050500	00000000 00000000 00000000 01000000	$2^{-3\cdot6}2^{-2}$
6	00050500 05000500 00000004 05000505	00050500 01000100 00000005 05000505	00000000 00000100 00000000 00000000	$2^{-3\cdot6}2^{-2\cdot2}$
7	00050005 00050500 00040000 00000500	00050005 00050500 00050000 00000500	00000000 00000000 00000000 00000100	$2^{-3\cdot6}$
8	00000000 00050005 00000500 00050000	00000000 00010005 00000500 00050000	00000000 00010000 00000000 00000000	$2^{-3\cdot3}2^{-2}$
9	00000000 00000000 00000000 05000000	00000000 00000000 00000000 05000000	00000000 00000000 00000000 00010000	2^{-3}
10	00000005 00000000 00000000 00000000	00000001 00000000 00000000 00000000	00000001 00000000 00000000 00000000	2^{-2}
11	00000000 00000000 00000000 00000000	00000000 00000000 00000000 00000000	00000000 00000000 00000001 00000000	–
12	00000000 00000000 00000000 00000000	00000000 00000000 00000000 00000000	00000000 00000001 00000000 00000000	–
13	00000000 00000000 01000000 00000000	00000000 00000000 20000000 00000000	00000000 00000000 00000000 00000001	2^{-2}
14	00002000 00000000 00002000 00002000	00008000 00000000 00008000 00008000	00010000 00000000 00000000 00000000	$2^{-2\cdot3}$

Table 3.12 – The Best **TK1** differential characteristics on 14 rounds of SKINNY-128 with probability equal to 2^{-120}. The four words represent the four rows of the state and are given in hexadecimal notation.

Round	$\delta X_i = X_i \oplus X'_i$ (before SB) δSBX_i (after SB)	$\delta TK1_i$ $\delta TK2_i$	Pr(States)
$i = 1$	00000000 00404010 40400000 40000000 00000000 00040440 04040000 04000000	00000000 00000000 00000000 00007700 00000000 00000000 00000000 00003900	$2^{-2 \cdot 6}$
2	00000400 00000000 40000000 00000404 00000500 00000000 04000000 00000101	00000000 00770000 00000000 00000000 00000000 00730000 00000000 00000000	$2^{-2 \cdot 3}\ 2^{-3}$
3	00010000 00000500 00000000 00000100 00200000 00000500 00000000 00002000	00000000 00000000 00000000 00770000 00000000 00000000 00000000 00730000	$2^{-2 \cdot 2}\ 2^{-3}$
4	00000000 00200000 00000005 00200000 00000000 00800000 00000005 00800000	00000077 00000000 00000000 00000000 000000E7 00000000 00000000 00000000	$2^{-2 \cdot 2}\ 2^{-3}$
5	80050090 00000090 00058000 00050090 03010002 00000002 00010200 00010003	00000000 00000000 00000077 00000000 00000000 00000000 000000E7 00000000	$2^{-2 \cdot 8}$
6	00010303 03010002 00000001 01010003 00202020 20200009 00000020 20200020	00000000 00000077 00000000 00000000 00000000 000000CE 00000000 00000000	$2^{-2 \cdot 6}\ 2^{-3 \cdot 4}$
7	20000000 00202020 B0002000 00002020 80000000 00808080 80008000 00009380	00000000 00000000 00000000 00000077 00000000 00000000 00000000 000000CE	$2^{-2 \cdot 6}\ 2^{-2.4}\ 2^{-3}$
8	00930000 80000000 00000080 00008000 00EA0000 03000000 00000003 00000300	00770000 00000000 00000000 00000000 009D0000 00000000 00000000 00000000	$2^{-2 \cdot 3}\ 2^{-6}$
9	00000000 00000000 00000000 00030000 00000000 00000000 00000000 00BC0000	00000000 00000000 00770000 00000000 00000000 00000000 009D0000 00000000	2^{-5}
10	BC000000 00000000 00000000 00000000 4C000000 00000000 00000000 00000000	77000000 00000000 00000000 00000000 3B000000 00000000 00000000 00000000	2^{-6}
11	00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000	00000000 00000000 77000000 00000000 00000000 00000000 3B000000 00000000	–
12	00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000	00007700 00000000 00000000 00000000 00007700 00000000 00000000 00000000	–
13	00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000	00000000 00000000 00007700 00000000 00000000 00000000 00007700 00000000	–
14	0000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000	00000000 77000000 00000000 00000000 00000000 EF000000 00000000 00000000	–
15	00000000 00000000 00980000 00000000 00000000 00000000 00420000 00000000	00000000 00000000 00000000 77000000 00000000 00000000 00000000 EF000000	2^{-5}
16	00000042 00000000 00000042 00000042 00000008 00000000 00000008 00000008	–	$2^{-2.4 \cdot 3}$

Table 3.13 – The Best **TK2** differential characteristics we found on 16 rounds of SKINNY-128 with probability equal to $2^{-127.6}$. The four words represent the four rows of the state and are given in hexadecimal notation.

Round	$\delta X_i = X_i \oplus X'_i$ (before SB) δSBX_i (after SB)	$\delta TK1_i$ $\delta TK2_i$ $\delta TK3_i$	Pr(States)
$i = 1$	00000200 00320000 08000000 00000808 00000800 00920000 18000000 00001010	00000000 00BA0000 00000000 00000000 00000000 00430000 00000000 00000000 00000000 00730000 00000000 00000000	$2^{-2\cdot3}$ $2^{-3\cdot2}$
2	00100000 00000800 00000000 00001000 00400000 00001000 00000000 00004000	00000000 00000000 00000000 00BA0000 00000000 00000000 00000000 00430000 00000000 00000000 00000000 00730000	$2^{-2\cdot3}$
3	00000000 00400000 00000010 00400000 00000000 00040000 00000040 00040000	000000BA 00000000 00000000 00000000 00000086 00000000 00000000 00000000 00000039 00000000 00000000 00000000	$2^{-2\cdot3}$
4	04400005 00000005 00400400 00400005 05040001 00000001 00040100 00040005	00000000 00000000 000000BA 00000000 00000000 00000000 00000086 00000000 00000000 00000000 00000039 00000000	$2^{-2\cdot6}$ $2^{-3\cdot2}$
5	00040505 05040001 00000004 04040005 00010101 01010028 00000001 01010001	00000000 000000BA 00000000 00000000 00000000 0000000D 00000000 00000000 00000000 0000009C 00000000 00000000	$2^{-2\cdot9}$ 2^{-3}
6	01000000 00010101 03000100 00000101 20000000 00202020 20002000 0000B320	00000000 00000000 00000000 000000BA 00000000 00000000 00000000 0000000D 00000000 00000000 00000000 0000009C	$2^{-2\cdot6}$ 2^{-3} 2^{-4}
7	00B30000 20000000 00000020 00002000 00EE0000 80000000 00000080 00008000	00BA0000 00000000 00000000 00000000 001A0000 00000000 00000000 00000000 004E0000 00000000 00000000 00000000	$2^{-2\cdot3}$ 2^{-7}
8	00000000 00000000 00000000 00800000 00000000 00000000 00000000 00030000	00000000 00000000 00BA0000 00000000 00000000 00000000 001A0000 00000000 00000000 00000000 004E0000 00000000	2^{-2}
9	03000000 00000000 00000000 00000000 29000000 00000000 00000000 00000000	BA000000 00000000 00000000 00000000 34000000 00000000 00000000 00000000 A7000000 00000000 00000000 00000000	2^{-4}
10	00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000	00000000 00000000 BA000000 00000000 00000000 00000000 34000000 00000000 00000000 00000000 A7000000 00000000	–
11	00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000	0000BA00 00000000 00000000 00000000 00006900 00000000 00000000 00000000 0000D300 00000000 00000000 00000000	–
12	00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000	00000000 00000000 0000BA00 00000000 00000000 00000000 00006900 00000000 00000000 00000000 0000D300 00000000	–
13	00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000	00000000 BA000000 00000000 00000000 00000000 D3000000 00000000 00000000 00000000 69000000 00000000 00000000	–
14	00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000	00000000 00000000 00000000 BA000000 00000000 00000000 00000000 D3000000 00000000 00000000 00000000 69000000	–
15	0000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000	00000000 0000BA00 00000000 00000000 00000000 0000A700 00000000 00000000 00000000 00003400 00000000 00000000	–
16	00000000 00000000 00000029 00000000 00000000 00000000 00000030 00000000	00000000 00000000 00000000 0000BA00 00000000 00000000 00000000 0000A700 00000000 00000000 00000000 00003400	2^{-3}
17	00300000 00000000 00300000 00300000 00400000 00000000 00400000 00400000	–	$2^{-2\cdot3}$

Table 3.14 – The Best **TK3** differential characteristics we found on 17 rounds of SKINNY-128 with probability equal to 2^{-110}. The four words represent the four rows of the state and are given in hexadecimal notation.

Conclusion

In this thesis, we have presented our works on the uses and constructions of different algorithms for differential cryptanalysis from the study of particular and general primitives and we have revisited an attack.

In Chapter 1, we revisit the fast near collision attack (FNCA), a new generic attack for stream ciphers applied successfully by their authors against `Grain v1` and `A5/1`. The FNCA is an internal state recovery attack based on divide-and-conquer strategy applied with a near collision property on the targeted stream ciphers. The divide-and-conquer strategy begins with splitting the internal state in two parts: the crucial part (CP) which is recovered by leveraging the near collision property with the new self-refined method proposed for the FNCA and the rest part (RP) which is efficiently computable when the CP is known. In our work as presented in Chapter 1, we show that the self-refined method has an incorrectly evaluated complexity by Zhang *et al.*. We proved this error doubly by first exhibiting the incorrect theorem in the work of Zhang *et al.* and proposed a corrected version and used it to compute the real complexity of the attacks proposed by Zhang *et al.*. The second way we proved the error in the FNCA is by using some information theory elements in an absurd reasoning: if the FNCA was correct, we should be able to observe some bias in the distribution of the internal state after the initialization phases of the stream ciphers. We ran several experiments with some simple codes to disprove the existence of those bias and so proved that the FNCA cannot be correct (regarding its claim complexity).

Chapter 2 presents our work concerning the diffusion round of Generalized Feistel networks of type 2. We built an efficient algorithm based on our new characterization of the diffusion round that constructs optimal even-odd permutations or gives better lower bounds for GFN of up to 42 branches (although our analysis is still valid for bigger cases, the computation cost becomes too great). More precisely, our characterization can be used to directly exhaust every optimal even-odd solutions up to 26 branches and the efficient algorithm is used for GFNs between 28 and 42 branches. For these values, we found optimal even-odd permutations for 28, 30, **32**, 34 and 36 branches which were unknown to the best of our knowledge with a diffusion round of 9 except for the 34 branches case

which has a diffusion round of 10. For the other cases, we prove that the best even-odd permutations have a diffusion round of at least 10 and found permutations with a diffusion round of 11 without ensuring the optimality of the results. However, the optimal permutations have a worse differential resistance than non-optimal (with a diffusion round of one more) permutations as remarked in the selection of the permutation in `WARP`, a new Feistel primitive [Ban+20].

In Chapter 3, we show how we were able to obtain better and sometimes the best differential characteristics for `SKINNY` in the different attacker settings namely single key, one tweakey, two tweakeys and three tweakeys. We used in our analysis a very classical approach of splitting the search of differential characteristics in two: computing good truncated differential characteristics for `SKINNY` (Step1) and trying to instantiate them in values (Step2). In addition to the new differential characteristics we found, this work offers two major contributions. For Step1, we developed a new algorithm to compute efficiently truncated differential characteristics with dynamic programming. More precisely, we leveraged that a differential characteristic can be seen as a smaller differential characteristic extended by one round and that not all information must be remembered for this smaller differential characteristic in this problem. The Step2 process used a Constraint Programming solver which allows for a straightforward modeling of `SKINNY` in particular for the S-box compared to previous works using MILP or SAT. Our contribution concerning Step2 is the precise search strategy given to the solver to be more efficient in the search. With the way we implement the two steps search, we were able to find again the best differential characteristics for `SKINNY`-64 in a few seconds and for `SKINNY`-128 in the **SK** setting and to find for the first time the best differential characteristic for `SKINNY`-128 in the **TK1** setting. We also gave new results for in **TK2** and **TK3** settings, optimal up to 12 and 13 rounds and better than previously known up to 16 and 17 rounds respectively. The next step for the automation of the search of differential characteristics is to give a description of the cipher and to have the whole analysis done by the computer in this work corresponding to the creation of the models. TAGADA [Lib+21], a Tool for Automatic Generation of Abstraction-based Differential Attacks proposes to do exactly this automatic generation of models but is still in development.

BIBLIOGRAPHY

[09] *The eSTREAM project*, https://www.ecrypt.eu.org/stream/project.html, 2009.

[Abd+17] Ahmed Abdelkhalek *et al.*, "MILP Modeling for (Large) S-boxes to Optimize Probability of Differential Characteristics", *in*: *IACR Trans. Symmetric Cryptol.* 2017.*4* (2017), pp. 99–129.

[Alf+18] Gianira N. Alfarano *et al.*, "ShiftRows Alternatives for AES-like Ciphers and Optimal Cell Permutations for Midori and Skinny", *in*: *IACR Trans. Symmetric Cryptol.* 2018.*2* (2018), pp. 20–47, DOI: 10.13154/tosc.v2018.i2.20-47, URL: https://doi.org/10.13154/tosc.v2018.i2.20-47.

[AM08] Mehreen Afzal and Ashraf Masood, "Algebraic cryptanalysis of a nlfsr based stream cipher", *in*: *2008 3rd International Conference on Information and Communication Technologies: From Theory to Applications*, IEEE, 2008, pp. 1–6.

[And94] Ross Anderson, "A5 (was: Hacking digital phones)", *in*: *Newsgroup Communication* (1994).

[Aok+00] Kazumaro Aoki *et al.*, "Camellia: A 128-Bit Block Cipher Suitable for Multiple Platforms - Design and Analysis", *in*: *Selected Areas in Cryptography, 7th Annual International Workshop, SAC 2000, Waterloo, Ontario, Canada, August 14-15, 2000, Proceedings*, 2000, pp. 39–56, DOI: 10.1007/3-540-44983-3_4, URL: https://doi.org/10.1007/3-540-44983-3%5C_4.

[Bac07] Fahiem Bacchus, "GAC Via Unit Propagation", *in*: *Principles and Practice of Constraint Programming - CP 2007, 13th International Conference, CP 2007*, vol. 4741, LNCS, Springer, 2007, pp. 133–147.

[Ban+17] Subhadeep Banik *et al.*, "GIFT: a small present", *in*: *International Conference on Cryptographic Hardware and Embedded Systems*, Springer, 2017, pp. 321–345.

[Ban+20] Subhadeep Banik *et al.*, "WARP: Revisiting GFN for Lightweight 128-bit Block Cipher.", *in*: *IACR Cryptol. ePrint Arch.* 2020 (2020), p. 1320.

[Ban14] Subhadeep Banik, "Dynamic cube attack on 105 round Grain v1", *in*: *Appl. Stat* 34.*2* (2014), pp. 49–50.

[Bar+11a] Gilles Barthe *et al.*, "Beyond Provable Security Verifiable IND-CCA Security of OAEP", *in*: *Topics in Cryptology - CT-RSA 2011 - The Cryptographers' Track at the RSA Conference 2011, San Francisco, CA, USA, February 14-18, 2011. Proceedings*, ed. by Aggelos Kiayias, vol. 6558, Lecture Notes in Computer Science, Springer, 2011, pp. 180–196.

[Bar+11b] Gilles Barthe *et al.*, "Computer-Aided Security Proofs for the Working Cryptographer", *in*: *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, ed. by Phillip Rogaway, vol. 6841, Lecture Notes in Computer Science, Springer, 2011, pp. 71–90.

[BB05] Elad Barkan and Eli Biham, "Conditional estimators: An effective attack on A5/1", *in*: *International Workshop on Selected Areas in Cryptography*, Springer, 2005, pp. 1–19.

[BBS99] Eli Biham, Alex Biryukov, and Adi Shamir, "Cryptanalysis of Skipjack Reduced to 31 Rounds Using Impossible Differentials", *in*: *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding*, 1999, pp. 12–23, DOI: 10.1007/3-540-48910-X_2, URL: https://doi.org/10.1007/3-540-48910-X%5C_2.

[BD00] Eli Biham and Orr Dunkelman, "Cryptanalysis of the A5/1 GSM stream cipher", *in*: *International Conference on Cryptology in India*, Springer, 2000, pp. 43–51.

[BD08] Steve Babbage and Matthew Dodd, "The MICKEY stream ciphers", *in*: *New Stream Cipher Designs*, Springer, 2008, pp. 191–209.

[BDK01] Eli Biham, Orr Dunkelman, and Nathan Keller, "The rectangle attack—rectangling the Serpent", *in*: *International Conference on the Theory and Applications of Cryptographic Techniques*, Springer, 2001, pp. 340–357.

[Bea+13] Ray Beaulieu *et al.*, "The SIMON and SPECK Families of Lightweight Block Ciphers", *in*: *IACR Cryptology ePrint Archive* 2013 (2013), p. 404, URL: `http://eprint.iacr.org/2013/404`.

[Bei+16] Christof Beierle *et al.*, "The SKINNY Family of Block Ciphers and Its Low-Latency Variant MANTIS", *in*: *Advances in Cryptology - CRYPTO 2016 Part II*, vol. 9815, LNCS, Springer, 2016, pp. 123–153.

[Bel+07] Nicolas Beldiceanu *et al.*, "Global Constraint Catalogue: Past, Present and Future", *in*: *Constraints An Int. J.* 12.*1* (2007), pp. 21–62.

[BGW99] Marc Briceno, Ian Goldberg, and David Wagner, *A pedagogical implementation of A5/1*, tech. rep., Available at `http://www.scard.org`, May 1999.

[BH03] Christian Bessière and Pascal Van Hentenryck, "To Be or Not to Be ... a Global Constraint", *in*: *Principles and Practice of Constraint Programming - CP 2003, 9th International Conference*, vol. 2833, LNCS, Springer, 2003, pp. 789–794.

[Bie14] Armin Biere, "Yet another local search solver and Lingeling and friends entering the SAT Competition 2014", *in*: *Sat competition* 2014.*2* (2014), p. 65.

[Bih93] Eli Biham, "New Types of Cryptanalytic Attacks Using related Keys (Extended Abstract)", *in*: *Advances in Cryptology - EUROCRYPT '93*, vol. 765, LNCS, Springer, 1993, pp. 398–409.

[Bir+10] Alex Biryukov *et al.*, "Key Recovery Attacks of Practical Complexity on AES-256 Variants with up to 10 Rounds", *in*: *Advances in Cryptology - EUROCRYPT 2010*, vol. 6110, LNCS, Springer, 2010, pp. 299–319.

[BK09] Alex Biryukov and Dmitry Khovratovich, "Related-Key Cryptanalysis of the Full AES-192 and AES-256", *in*: *Advances in Cryptology - ASIACRYPT 2009*, vol. 5912, LNCS, Springer, 2009, pp. 1–18.

[BKN09] Alex Biryukov, Dmitry Khovratovich, and Ivica Nikolic, "Distinguisher and Related-Key Attack on the Full AES-256", *in*: *Advances in Cryptology - CRYPTO 2009*, vol. 5677, LNCS, Springer, 2009, pp. 231–249.

[BN10] Alex Biryukov and Ivica Nikolic, “Automatic Search for Related-Key Differential Characteristics in Byte-Oriented Block Ciphers: Application to AES, Camellia, Khazad and Others”, *in*: *Advances in Cryptology - EUROCRYPT 2010*, vol. 6110, LNCS, Springer, 2010, pp. 322–344.

[BN11] Alex Biryukov and Ivica Nikolic, “Search for Related-Key Differential Characteristics in DES-Like Ciphers”, *in*: *Fast Software Encryption - FSE 2011*, vol. 6733, LNCS, Springer, 2011, pp. 18–34.

[Bog+07] Andrey Bogdanov *et al.*, “PRESENT: An ultra-lightweight block cipher”, *in*: *International workshop on cryptographic hardware and embedded systems*, Springer, 2007, pp. 450–466.

[Bou+04] Frédéric Boussemart *et al.*, “Boosting Systematic Search by Weighting Constraints”, *in*: *Proceedings of the 16th Eureopean Conference on Artificial Intelligence, ECAI'2004*, ed. by Ramón López de Mántaras and Lorenza Saitta, IOS Press, 2004, pp. 146–150.

[BR94] Mihir Bellare and Phillip Rogaway, “Optimal Asymmetric Encryption”, *in*: *Advances in Cryptology - EUROCRYPT '94, Workshop on the Theory and Application of Cryptographic Techniques, Perugia, Italy, May 9-12, 1994, Proceedings*, ed. by Alfredo De Santis, vol. 950, Lecture Notes in Computer Science, Springer, 1994, pp. 92–111.

[Bra+84] Robert King Brayton *et al.*, *Logic Minimization Algorithms for VLSI Synthesis*, Kluwer Academic Publishers, 1984.

[Bry86] Randal E. Bryant, “Graph-Based Algorithms for Boolean Function Manipulation”, *in*: *IEEE Trans. Computers* 35.*8* (1986), pp. 677–691.

[BS91a] Eli Biham and Adi Shamir, “Differential cryptanalysis of DES-like cryptosystems”, *in*: *Journal of CRYPTOLOGY* 4.*1* (1991), pp. 3–72.

[BS91b] Eli Biham and Adi Shamir, “Differential Cryptanalysis of Feal and N-Hash”, *in*: *Advances in Cryptology - EUROCRYPT '91*, vol. 547, LNCS, Springer, 1991, pp. 1–16.

[BSW00] Alex Biryukov, Adi Shamir, and David Wagner, “Real Time Cryptanalysis of A5/1 on a PC”, *in*: *International Workshop on Fast Software Encryption*, Springer, 2000, pp. 1–18.

[CGT19] Victor Cauchois, Clément Gomez, and Gaël Thomas, "General Diffusion Analysis: How to Find Optimal Permutations for Generalized Type-II Feistel Schemes", *in*: *IACR Trans. Symmetric Cryptol.* 2019.*1* (2019).

[Cid+18] Carlos Cid *et al.*, "Boomerang Connectivity Table: A New Cryptanalysis Tool", *in*: *Advances in Cryptology - EUROCRYPT 2018*, vol. 10821, LNCS, Springer, 2018, pp. 683–714.

[CPS08] Jean-Sébastien Coron, Jacques Patarin, and Yannick Seurin, "The Random Oracle Model and the Ideal Cipher Model Are Equivalent", *in*: *Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings*, ed. by David A. Wagner, vol. 5157, Lecture Notes in Computer Science, Springer, 2008, pp. 1–20.

[CS14] Geoffrey Chu and Peter J. Stuckey, *Chuffed solver description*, Available at `http://www.minizinc.org/challenge2014/description_chuffed.txt`, 2014.

[CY10] Kenil C. K. Cheng and Roland H. C. Yap, "An MDD-based generalized arc consistency algorithm for positive and negative table constraints and some global constraints", *in*: *Constraints* 15.*2* (2010), pp. 265–304.

[Del+21a] Stéphanie Delaune *et al.*, "Efficient Methods to Search for Best Differential Characteristics on SKINNY", *in*: *International Conference on Applied Cryptography and Network Security*, Springer, 2021, pp. 184–207.

[Del+21b] Stéphanie Delaune *et al.*, "SKINNY with Scalpel Comparing Tools for Differential Analysis", working paper or preprint, Apr. 2021, URL: `https://hal.archives-ouvertes.fr/hal-03040548`.

[Dem+16a] Jordan Demeulenaere *et al.*, "Compact-Table: Efficiently Filtering Table Constraints with Reversible Sparse Bit-Sets", *in*: *Principles and Practice of Constraint Programming - CP 2016*, vol. 9892, LNCS, Springer, 2016, pp. 207–223.

[Dem+16b] Jordan Demeulenaere *et al.*, "Compact-Table: Efficiently Filtering Table Constraints with Reversible Sparse Bit-Sets", *in*: *Principles and Practice of Constraint Programming - CP 2016*, vol. 9892, LNCS, Springer, 2016, pp. 207–223.

[Der+19] Patrick Derbez *et al.*, "Efficient Search for Optimal Diffusion Layers of Generalized Feistel Networks", *in*: *IACR Transactions on Symmetric Cryptology* 2019.*2* (June 2019), pp. 218–240, DOI: 10.13154/tosc.v2019.i2.218-240, URL: https://tosc.iacr.org/index.php/ToSC/article/view/8321.

[DES77] DES, "Data Encryption Standard", *in*: *FIPS PUB 46, Federal information processing standards publication 46* (1977).

[DFM20] Patrick Derbez, Pierre-Alain Fouque, and Victor Mollimard, "Fake Near Collisions Attacks", *in*: *IACR Transactions on Symmetric Cryptology* 2020.*4* (Dec. 2020), pp. 88–103, DOI: 10.46586/tosc.v2020.i4.88-103, URL: https://tosc.iacr.org/index.php/ToSC/article/view/8749.

[DKS10] Orr Dunkelman, Nathan Keller, and Adi Shamir, "A practical-time related-key attack on the KASUMI cryptosystem used in GSM and 3G telephony", *in*: *Annual cryptology conference*, Springer, 2010, pp. 393–410.

[DKV07] Debapratim De, Abishek Kumarasubramanian, and Ramarathnam Venkatesan, "Inversion Attacks on Secure Hash Functions Using satSolvers", *in*: *Theory and Applications of Satisfiability Testing - SAT 2007, 10th International Conference*, vol. 4501, LNCS, Springer, 2007, pp. 377–382.

[DP08] Christophe De Canniere and Bart Preneel, "Trivium", *in*: *New stream cipher designs*, Springer, 2008, pp. 244–266.

[DR13] Joan Daemen and Vincent Rijmen, *The design of Rijndael: AES-the advanced encryption standard*, Springer Science & Business Media, 2013.

[DR99] Joan Daemen and Vincent Rijmen, "AES proposal: Rijndael", *in*: (1999).

[DS09] Itai Dinur and Adi Shamir, "Cube attacks on tweakable black box polynomials", *in*: *Annual international conference on the theory and applications of cryptographic techniques*, Springer, 2009, pp. 278–299.

[DS11] Itai Dinur and Adi Shamir, "Breaking Grain-128 with dynamic cube attacks", *in*: *International Workshop on Fast Software Encryption*, Springer, 2011, pp. 167–187.

[EJ03] Patrik Ekdahl and Thomas Johansson, "Another attack on A5/1", *in*: *IEEE transactions on information theory* 49.*1* (2003), pp. 284–289.

[ENP19] Maria Eichlseder, Marcel Nageler, and Robert Primas, "Analyzing the Linear Keystream Biases in AEGIS", *in*: *IACR Trans. Symmetric Cryptol.* 2019.*4* (2019), pp. 348–368.

[ES06] Niklas Eén and Niklas Sörensson, "Translating Pseudo-Boolean Constraints into SAT", *in*: *JSAT* 2.*1-4* (2006), pp. 1–26.

[FIP01] FIPS 197, *Advanced Encryption Standard*, Federal Information Processing Standards Publication 197, U.S. Department of Commerce/N.I.S.T., 2001.

[FJP13a] Pierre-Alain Fouque, Jérémy Jean, and Thomas Peyrin, "Structural Evaluation of AES and Chosen-Key Distinguisher of 9-Round AES-128", *in*: *Advances in Cryptology - CRYPTO 2013 - Part I*, vol. 8042, LNCS, Springer, 2013, pp. 183–203.

[FJP13b] Pierre-Alain Fouque, Jérémy Jean, and Thomas Peyrin, "Structural Evaluation of AES and Chosen-Key Distinguisher of 9-Round AES-128", *in*: *Advances in Cryptology - CRYPTO 2013 - Part I*, vol. 8042, LNCS, Springer, 2013, pp. 183–203.

[Fuj+01] Eiichiro Fujisaki *et al.*, "RSA-OAEP Is Secure under the RSA Assumption", *in*: *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*, ed. by Joe Kilian, vol. 2139, Lecture Notes in Computer Science, Springer, 2001, pp. 260–274.

[Gec06] Gecode Team, *Gecode: Generic Constraint Development Environment*, Available from `http://www.gecode.org`, 2006.

[Ger+17] David Gerault *et al.*, *Combining Solvers to Solve a Cryptanalytic Problem*, CP/ICLP/SAT Doctoral Program - part of the conference CP 2017, Available at `http://cp2017.a4cp.org/downloads/dp-2017-proceedings.pdf`, 2017.

[Gér+17] David Gérault *et al.*, *Revisiting AES Related-Key Differential Attacks with Constraint Programming*, Cryptology ePrint Archive, Report 2017/139, Extended version of [Gér+18], `https://eprint.iacr.org/2017/139`, 2017.

[Gér+18] David Gérault *et al.*, "Revisiting AES related-key differential attacks with constraint programming", *in*: *Inf. Process. Lett.* 139 (2018), pp. 24–29.

[Ger+20] David Gerault *et al.*, "Computing AES related-key differential characteristics with constraint programming", *in*: *Artif. Intell.* 278 (2020).

[Gil14] Henri Gilbert, "A Simplified Representation of AES", *in*: *Advances in Cryptology - ASIACRYPT 2014 - Part I*, vol. 8873, LNCS, Springer, 2014, pp. 200–222.

[GL16] David Gérault and Pascal Lafourcade, "Related-Key Cryptanalysis of Midori", *in*: *Progress in Cryptology - INDOCRYPT 2016*, vol. 10095, LNCS, 2016, pp. 287–304.

[GM16a] Shay Gueron and Nicky Mouha, "Simpira v2: A Family of Efficient Permutations Using the AES Round Function", *in*: *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I*, 2016, pp. 95–125, DOI: 10.1007/978-3-662-53887-6_4, URL: https://doi.org/10.1007/978-3-662-53887-6%5C_4.

[GM16b] Shay Gueron and Nicky Mouha, "Simpira v2: A Family of Efficient Permutations Using the AES Round Function", *in*: *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I*, 2016, pp. 95–125, DOI: 10.1007/978-3-662-53887-6_4, URL: https://doi.org/10.1007/978-3-662-53887-6%5C_4.

[GMS16] David Gérault, Marine Minier, and Christine Solnon, "Constraint Programming Models for Chosen Key Differential Cryptanalysis", *in*: *Principles and Practice of Constraint Programming - CP 2016*, vol. 9892, LNCS, Springer, 2016, pp. 584–601.

[Gol97] Jovan Dj Golić, "Cryptanalysis of alleged A5 stream cipher", *in*: *International Conference on the Theory and Applications of Cryptographic Techniques*, Springer, 1997, pp. 239–255.

[GP10] Henri Gilbert and Thomas Peyrin, "Super-Sbox Cryptanalysis: Improved Attacks for AES-Like Permutations", *in*: *Fast Software Encryption - FSE 2010*, vol. 6147, LNCS, Springer, 2010, pp. 365–383.

[Gra01] Louis Granboulan, “Flaws in differential cryptanalysis of Skipjack”, *in*: *International Workshop on Fast Software Encryption*, Springer, 2001, pp. 328–335.

[Hel+06] Martin Hell *et al.*, “A Stream Cipher Proposal: Grain-128”, *in*: *Proceedings 2006 IEEE International Symposium on Information Theory, ISIT 2006, The Westin Seattle, Seattle, Washington, USA, July 9-14, 2006*, IEEE, 2006, pp. 1614–1618.

[HJM07] Martin Hell, Thomas Johansson, and Willi Meier, “Grain: a stream cipher for constrained environments”, *in*: *IJWMC* 2.*1* (2007), pp. 86–93.

[HKT11] Thomas Holenstein, Robin Künzler, and Stefano Tessaro, “The equivalence of the random oracle model and the ideal cipher model, revisited”, *in*: *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011*, ed. by Lance Fortnow and Salil P. Vadhan, ACM, 2011, pp. 89–98.

[Hon+06] Deukjo Hong *et al.*, “HIGHT: A New Block Cipher Suitable for Low-Resource Device”, *in*: *Cryptographic Hardware and Embedded Systems - CHES 2006, 8th International Workshop, Yokohama, Japan, October 10-13, 2006, Proceedings*, 2006, pp. 46–59, DOI: 10.1007/11894063_4, URL: https://doi.org/10.1007/11894063%5C_4.

[Ino+19] Akiko Inoue *et al.*, “Cryptanalysis of OCB2: Attacks on Authenticity and Confidentiality”, *in*: *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part I*, ed. by Alexandra Boldyreva and Daniele Micciancio, vol. 11692, Lecture Notes in Computer Science, Springer, 2019, pp. 3–31.

[Jea16] Jérémy Jean, *TikZ for Cryptographers*, https://www.iacr.org/authors/tikz/, 2016.

[JNP14] Jérémy Jean, Ivica Nikolic, and Thomas Peyrin, “Tweaks and Keys for Block Ciphers: The TWEAKEY Framework”, *in*: *Advances in Cryptology - ASIACRYPT 2014 - Part II*, vol. 8874, LNCS, Springer, 2014, pp. 274–288.

[Kal+17] Daniel Kales *et al.*, "Improvements to the Linear Layer of LowMC: A Faster Picnic", *in*: *IACR Cryptology ePrint Archive* 2017 (2017), p. 1148, URL: http://eprint.iacr.org/2017/1148.

[Ker83] Auguste Kerckhoff, "La cryptographie militaire", *in*: *Journal des sciences militaires* IX (Jan. 1883).

[Kho+16] Khoongming Khoo *et al.*, *Human-readable Proof of the Related-Key Security of AES-128*, 2016, URL: http://eprint.iacr.org/2016/025.

[KHP07] Jongsung Kim, Seokhie Hong, and Bart Preneel, "Related-Key Rectangle Attacks on Reduced AES-192 and AES-256", *in*: *Fast Software Encryption - FSE 2007*, vol. 4593, LNCS, Springer, 2007, pp. 225–241.

[KLT15] Stefan Kölbl, Gregor Leander, and Tyge Tiessen, "Observations on the SIMON Block Cipher Family", *in*: *Advances in Cryptology - CRYPTO 2015 - Part I*, vol. 9215, LNCS, Springer, 2015, pp. 161–185.

[Knu95] Lars R. Knudsen, "Truncated and Higher Order Differentials", *in*: *Fast Software Encryption: Second International Workshop - FSE.* Vol. 1008, LNCS, Springer, 1995, pp. 196–211.

[Knu98] Lars Knudsen, "DEAL-a 128-bit block cipher", *in*: *complexity* 258.*2* (1998), p. 216.

[KR07] Lars R. Knudsen and Vincent Rijmen, "Known-Key Distinguishers for Some Block Ciphers", *in*: *Advances in Cryptology - ASIACRYPT 2007*, vol. 4833, LNCS, Springer, 2007, pp. 315–324.

[KRW99] Lars R Knudsen, Matthew JB Robshaw, and David Wagner, "Truncated differentials and Skipjack", *in*: *Annual International Cryptology Conference*, Springer, 1999, pp. 165–180.

[Laf18] Frédéric Lafitte, "CryptoSAT: a tool for SAT-based cryptanalysis", *in*: *IET Information Security* 12.*6* (2018), pp. 463–474.

[Lec+07] Christophe Lecoutre *et al.*, "Nogood Recording from Restarts", *in*: *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence*, 2007, pp. 131–136, URL: http://ijcai.org/Proceedings/07/Papers/019.pdf.

[Lec+09] Christophe Lecoutre *et al.*, "Reasoning from last conflict(s) in constraint programming", *in*: *Artif. Intell.* 173.*18* (2009), pp. 1592–1614, DOI: 10.1016/j.artint.2009.09.002, URL: https://doi.org/10.1016/j.artint.2009.09.002.

[LGS17] Guozhen Liu, Mohona Ghosh, and Ling Song, "Security Analysis of SKINNY under Related-Tweakey Settings (Long Paper)", *in*: *IACR Trans. Symmetric Cryptol.* 2017.*3* (2017), pp. 37–72, DOI: 10.13154/tosc.v2017.i3.37-72, URL: https://doi.org/10.13154/tosc.v2017.i3.37-72.

[Lib+21] Luc Libralesso *et al.*, "Automatic Generation of Declarative Models for Differential Cryptanalysis", *in*: *27th International Conference on Principles and Practice of Constraint Programming (CP 2021)*, 2021.

[Liu+17] Fanghui Liu *et al.*, "A Tolerant Algebraic Side-Channel Attack on AES Using CP", *in*: *Principles and Practice of Constraint Programming - 23rd International Conference, CP 2017*, vol. 10416, LNCS, Springer, 2017, pp. 189–205.

[LL04] Yat Chiu Law and Jimmy H. M. Lee, "Global Constraints for Integer and Set Value Precedence", *in*: *10th International Conference on Principles and Practice of Constraint Programming (CP)*, Springer Berlin Heidelberg, 2004, pp. 362–376.

[LR88] Michael Luby and Charles Rackoff, "How to Construct Pseudorandom Permutations from Pseudorandom Functions", *in*: *SIAM J. Comput.* 17.*2* (1988), pp. 373–386, DOI: 10.1137/0217022, URL: https://doi.org/10.1137/0217022.

[LRW02] Moses Liskov, Ronald L Rivest, and David Wagner, "Tweakable block ciphers", *in*: *Annual International Cryptology Conference*, Springer, 2002, pp. 31–46.

[Lu+08] Jiqiang Lu *et al.*, "Improving the Efficiency of Impossible Differential Cryptanalysis of Reduced Camellia and MISTY1", *in*: *Topics in Cryptology - CT-RSA 2008, The Cryptographers' Track at the RSA Conference 2008, San Francisco, CA, USA, April 8-11, 2008. Proceedings*, ed. by Tal Malkin, vol. 4964, Lecture Notes in Computer Science, Springer, 2008, pp. 370–386.

[Lu15] Jiqiang Lu, "A methodology for differential-linear cryptanalysis and its applications", *in*: *Designs, Codes and Cryptography* 77.*1* (2015), pp. 11–48.

[Mat94] Mitsuru Matsui, "On Correlation Between the Order of S-boxes and the Strength of DES", *in*: *Advances in Cryptology - EUROCRYPT '94*, vol. 950, LNCS, Springer, 1994, pp. 366–375.

[McC56] E. J. McCluskey Jr., "Minimization of Boolean Functions*", *in*: *Bell System Technical Journal* 35.*6* (1956), pp. 1417–1444.

[Mih+12a] Miodrag Mihaljević *et al.*, "Generic cryptographic weakness of k-normal Boolean functions in certain stream ciphers and cryptanalysis of grain-128", *in*: *Periodica Mathematica Hungarica* 65.*2* (2012), pp. 205–227.

[Mih+12b] Miodrag J Mihaljević *et al.*, "Internal state recovery of Grain-v1 employing normality order of the filter function", *in*: *IET Information Security* 6.*2* (2012), pp. 55–64.

[MJB04] Alexander Maximov, Thomas Johansson, and Steve Babbage, "An improved correlation attack on A5/1", *in*: *International Workshop on Selected Areas in Cryptography*, Springer, 2004, pp. 1–18.

[Mou+12] Nicky Mouha *et al.*, "Differential and Linear Cryptanalysis Using Mixed-Integer Linear Programming", *in*: *Information Security and Cryptology - 7th International Conference, Inscrypt*, vol. 7537, LNCS, Springer, 2012, pp. 57–76.

[MP13] Nicky Mouha and Bart Preneel, "A Proof that the ARX Cipher Salsa20 is Secure against Differential Cryptanalysis", *in*: *IACR Cryptology ePrint Archive* 2013 (2013), p. 328, URL: `http://eprint.iacr.org/2013/328`.

[MS13] Pawel Morawiecki and Marian Srebrny, "A SAT-based preimage analysis of reduced Keccak hash functions", *in*: *Inf. Process. Lett.* 113.*10-11* (2013), pp. 392–397.

[MS77] Florence Jessie MacWilliams and Neil James Alexander Sloane, *The theory of error-correcting codes*, vol. 16, Elsevier, 1977.

[MSR14] Marine Minier, Christine Solnon, and Julia Reboul, "Solving a Symmetric Key Cryptographic Problem with Constraint Programming", *in*: *13th International Workshop on Constraint Modelling and Reformulation (ModRef), in conjunction with CP'14*, 2014, pp. 1–13.

[Mur11] Sean Murphy, “The return of the cryptographic boomerang”, *in*: *IEEE Transactions on Information Theory* 57.*4* (2011), pp. 2517–2521.

[MZ06] Ilya Mironov and Lintao Zhang, “Applications of SAT Solvers to Cryptanalysis of Hash Functions”, *in*: *Theory and Applications of Satisfiability Testing - SAT 2006, 9th International Conference*, vol. 4121, LNCS, Springer, 2006, pp. 102–115.

[Net+07] Nicholas Nethercote *et al.*, “MiniZinc: Towards a Standard CP Modelling Language”, *in*: *Principles and Practice of Constraint Programming - CP 2007*, vol. 4741, LNCS, Springer, 2007, pp. 529–543.

[Nyb96] Kaisa Nyberg, “Generalized Feistel Networks”, *in*: *Advances in Cryptology - ASIACRYPT ’96, International Conference on the Theory and Applications of Cryptology and Information Security, Kyongju, Korea, November 3-7, 1996, Proceedings*, 1996, pp. 91–104, DOI: 10.1007/BFb0034838, URL: https://doi.org/10.1007/BFb0034838.

[Opt18] Gurobi Optimization, *Gurobi Optimizer Reference Manual*, 2018, URL: http://www.gurobi.com.

[PFL16] Charles Prud’homme, Jean-Guillaume Fages, and Xavier Lorca, *Choco Documentation*, TASC, INRIA Rennes, LINA CNRS UMR 6241, COSLING S.A.S., 2016, URL: http://www.choco-solver.org.

[Pie90] Josef Pieprzyk, “How to Construct Pseudorandom Permutations from Single Pseudorandom Functions”, *in*: *Advances in Cryptology - EUROCRYPT ’90, Workshop on the Theory and Application of of Cryptographic Techniques, Aarhus, Denmark, May 21-24, 1990, Proceedings*, 1990, pp. 140–150, DOI: 10.1007/3-540-46877-3_12, URL: https://doi.org/10.1007/3-540-46877-3%5C_12.

[Qui55] W. V. Quine, “A Way to Simplify Truth Functions”, *in*: *The American Mathematical Monthly* 62.*9* (1955), pp. 627–631.

[Rah+16] Majid Rahimi *et al.*, “Dynamic cube attack on Grain-v1”, *in*: *IET Information Security* 10.*4* (2016), pp. 165–172.

[Ram+11] Venkatesh Ramamoorthy *et al.*, "The Design of Cryptographic S-Boxes Using CSPs", *in*: *Principles and Practice of Constraint Programming - CP 2011 - 17th International Conference, CP 2011*, vol. 6876, LNCS, Springer, 2011, pp. 54–68.

[RB08] Matthew Robshaw and Olivier Billet, *New stream cipher designs: the eSTREAM finalists*, vol. 4986, Springer, 2008.

[RBW06] Francesca Rossi, Peter van Beek, and Toby Walsh, *Handbook of Constraint Programming (Foundations of Artificial Intelligence)*, New York, NY, USA: Elsevier Science Inc., 2006, ISBN: 0444527265.

[RSA78] Ronald L Rivest, Adi Shamir, and Leonard Adleman, "A method for obtaining digital signatures and public-key cryptosystems", *in*: *Communications of the ACM* 21.*2* (1978), pp. 120–126.

[Sch+98] Bruce Schneier *et al.*, "Twofish: A 128-bit block cipher", *in*: *NIST AES Proposal* 15 (1998), p. 23.

[Sha49] Claude E Shannon, "Communication theory of secrecy systems", *in*: *Bell system technical journal* 28.*4* (1949), pp. 656–715.

[Shi+07] Taizo Shirai *et al.*, "The 128-Bit Blockcipher CLEFIA (Extended Abstract)", *in*: *Fast Software Encryption, 14th International Workshop, FSE 2007, Luxembourg, Luxembourg, March 26-28, 2007, Revised Selected Papers*, 2007, pp. 181–195, DOI: 10.1007/978-3-540-74619-5_12, URL: https://doi.org/10.1007/978-3-540-74619-5%5C_12.

[Shi+11] Kyoji Shibutani *et al.*, "Piccolo: An Ultra-Lightweight Blockcipher", *in*: *Cryptographic Hardware and Embedded Systems - CHES 2011 - 13th International Workshop, Nara, Japan, September 28 - October 1, 2011. Proceedings*, 2011, pp. 342–357, DOI: 10.1007/978-3-642-23951-9_23, URL: https://doi.org/10.1007/978-3-642-23951-9%5C_23.

[Sho01] Victor Shoup, "OAEP Reconsidered", *in*: *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*, ed. by Joe Kilian, vol. 2139, Lecture Notes in Computer Science, Springer, 2001, pp. 239–259.

[Sin06] R. Singleton, "Maximum Distance -nary Codes", *in*: *IEEE Trans. Inf. Theor.* 10.*2* (Sept. 2006), pp. 116–118, ISSN: 0018-9448, DOI: 10.1109/TIT.1964.1053661, URL: http://dx.doi.org/10.1109/TIT.1964.1053661.

[SM10] Tomoyasu Suzaki and Kazuhiko Minematsu, "Improving the generalized Feistel", *in*: *International Workshop on Fast Software Encryption*, Springer, 2010, pp. 19–39.

[SMB18] Sadegh Sadeghi, Tahereh Mohammadi, and Nasour Bagheri, "Cryptanalysis of Reduced round SKINNY Block Cipher", *in*: *IACR Trans. Symmetric Cryptol.* 2018.*3* (2018), pp. 124–162, DOI: 10.13154/tosc.v2018.i3.124-162, URL: https://doi.org/10.13154/tosc.v2018.i3.124-162.

[SNC09] Mate Soos, Karsten Nohl, and Claude Castelluccia, "Extending SAT Solvers to Cryptographic Problems", *in*: *Theory and Applications of Satisfiability Testing - SAT 2009, 12th International Conference, SAT 2009*, vol. 5584, LNCS, Springer, 2009, pp. 244–257.

[SQH19] Ling Song, Xianrui Qin, and Lei Hu, "Boomerang Connectivity Table Revisited. Application to SKINNY and AES", *in*: *IACR Trans. Symmetric Cryptol.* 2019.*1* (2019), pp. 118–141, DOI: 10.13154/tosc.v2019.i1.118-141, URL: https://doi.org/10.13154/tosc.v2019.i1.118-141.

[ST17] Yu Sasaki and Yosuke Todo, "New Impossible Differential Search Tool from Design and Cryptanalysis Aspects - Revealing Structural Properties of Several Ciphers", *in*: *Advances in Cryptology - EUROCRYPT 2017*, vol. 10212, LNCS, 2017, pp. 185–215.

[Sun+14] Siwei Sun *et al.*, "Automatic Security Evaluation and (Related-key) Differential Characteristic Search: Application to SIMON, PRESENT, LBlock, DES(L) and Other Bit-Oriented Block Ciphers", *in*: *Advances in Cryptology - ASIACRYPT 2014 Part I*, vol. 8873, LNCS, Springer, 2014, pp. 158–178.

[Sun+17] Siwei Sun *et al.*, "Analysis of AES, SKINNY, and Others with Constraint Programming", *in*: *24th International Conference on Fast Software Encryption*, 2017.

[Suz+12] Tomoyasu Suzaki *et al.*, "TWINE : A Lightweight Block Cipher for Multiple Platforms", *in*: *Selected Areas in Cryptography, 19th International Conference, SAC 2012, Windsor, ON, Canada, August 15-16, 2012, Revised Se-*

lected Papers, 2012, pp. 339–354, DOI: 10.1007/978-3-642-35999-6_22, URL: https://doi.org/10.1007/978-3-642-35999-6%5C_22.

[SWW17] Ling Sun, Wei Wang, and Meiqin Wang, "Automatic Search of Bit-Based Division Property for ARX Ciphers and Word-Based Division Property", *in*: *Advances in Cryptology - ASIACRYPT 2017 - Part I*, 2017, pp. 128–157.

[SWW18] Ling Sun, Wei Wang, and Meiqin Wang, "More Accurate Differential Properties of LED64 and Midori64", *in*: *IACR Trans. Symmetric Cryptol.* 2018.*3* (2018), pp. 93–123.

[Tod+17] Yosuke Todo *et al.*, "Cube Attacks on Non-Blackbox Polynomials Based on Division Property", *in*: *Advances in Cryptology - CRYPTO 2017*, vol. 10403, LNCS, Springer, 2017, pp. 250–279.

[Vau98] Serge Vaudenay, "Provable security for block ciphers by decorrelation", *in*: *Annual Symposium on Theoretical Aspects of Computer Science*, Springer, 1998, pp. 249–275.

[VLS18] Hélène Verhaeghe, Christophe Lecoutre, and Pierre Schaus, "Compact-MDD: Efficiently Filtering (s)MDD Constraints with Reversible Sparse Bit-sets", *in*: *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018*, 2018, pp. 1383–1389.

[Wag99] David Wagner, "The boomerang attack", *in*: *International Workshop on Fast Software Encryption*, Springer, 1999, pp. 156–170.

[Wal00] Toby Walsh, "SAT v CSP", *in*: *Principles and Practice of Constraint Programming - CP 2000, 6th International Conference*, vol. 1894, LNCS, Springer, 2000, pp. 441–456.

[Wan+13] Yongjuan Wang *et al.*, "The Improved Cube Attack on Grain-v1.", *in*: *IACR Cryptol. ePrint Arch.* 2013 (2013), p. 417.

[WKD07] Gaoli Wang, Nathan Keller, and Orr Dunkelman, "The delicate issues of addition with respect to XOR differences", *in*: *International Workshop on Selected Areas in Cryptography*, Springer, 2007, pp. 212–231.

[Zha+13] Bin Zhang *et al.*, "Near collision attack on the grain v1 stream cipher", *in*: *International Workshop on Fast Software Encryption*, Springer, 2013, pp. 518–538.

[Zha+20] Boxin Zhao *et al.*, "Generalized related-key rectangle attacks on block ciphers with linear key schedule: applications to SKINNY and GIFT", *in*: *Des. Codes Cryptogr.* 88.*6* (2020), pp. 1103–1126, DOI: 10.1007/s10623-020-00730-1, URL: https://doi.org/10.1007/s10623-020-00730-1.

[Zha19] Bin Zhang, "Cryptanalysis of GSM Encryption in 2G/3G Networks Without Rainbow Tables", *in*: *International Conference on the Theory and Application of Cryptology and Information Security*, Springer, 2019, pp. 428–456.

[ZK16] Neng-Fa Zhou and Håkan Kjellerstrand, "The Picat-SAT Compiler", *in*: *Practical Aspects of Declarative Languages - PADL 2016*, vol. 9585, LNCS, Springer, 2016, pp. 48–62.

[ZK17] Neng-Fa Zhou and Håkan Kjellerstrand, "Optimizing SAT Encodings for Arithmetic Constraints", *in*: *Principles and Practice of Constraint Programming - 23rd International Conference, CP 2017*, vol. 10416, LNCS, Springer, 2017, pp. 671–686.

[ZKF15] Neng-Fa Zhou, Hakan Kjellerstrand, and Jonathan Fruhman, *Constraint Solving and Planning with Picat*, Springer, 2015.

[ZMI89a] 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, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, 1989, pp. 461–480, DOI: 10.1007/0-387-34805-0_42, URL: https://doi.org/10.1007/0-387-34805-0%5C_42.

[ZMI89b] Yuliang Zheng, Tsutomu Matsumoto, and Hideki Imai, "On the construction of block ciphers provably secure and not relying on any unproved hypotheses", *in*: *Conference on the Theory and Application of Cryptology*, Springer, 1989, pp. 461–480.

[ZXM18] Bin Zhang, Chao Xu, and Willi Meier, "Fast Near Collision Attack on the Grain v1 Stream Cipher", *in*: *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part II*, ed. by Jesper Buus Nielsen and Vincent Rijmen, vol. 10821, Lecture Notes in Computer Science, Springer, 2018, pp. 771–802.

DOCTORAT BRETAGNE LOIRE / MATHSTIC

UNIVERSITÉ DE RENNES 1

Titre : Algorithmes pour la cryptanalyse différentielle

Mot clés : Cryptographie, Chiffrement symétrique, Cryptanalyse différentielle

Résumé : La sécurité en cryptographie symétrique semble être une notion très floue pour des non-spécialistes du domaine. Pour simplifier le raisonnement fait par les cryptanalystes, une primitive symétrique est sûre lorsqu'aucune attaque pratique n'est trouvée contre celle-ci. Une grande part de cette démonstration consiste à essayer des attaques classiques contre les différentes primitives existantes. Dans cette thèse, nous présentons nos travaux de cryptanalyse dans cette optique en utilisant différents algorithmes constructifs ou algorithmes de test.

Nous commençons par revisiter les attaques rapides par collision proche publiées en 2018. Nous prouvons avec des algorithmes inspirés de la théorie de l'information que la complexité de ces attaques était sérieusement sous-estimée et donnons la version corrigée.

Nous proposons ensuite une nouvelle caractérisation d'un aspect particulier des réseaux de Feistel. Elle nous permet de déduire un algorithme efficace pour trouver (construire) les permutations optimales en ce sens, apportant ainsi une solution à un problème vieux de 10 ans.

Nous terminons avec l'utilisation de solveurs, des algorithmes généraux prenant en entrée la description d'un problème et renvoyant en sortie une solution à celui-ci pour le calcul de meilleures caractéristiques différentielles du chiffrement par bloc SKINNY.

Title: Algorithms for differential cryptanalysis

Keywords: Cryptography, Symmetric cipher, Differential cryptanalysis

Abstract: Security in symmetric cryptography seems to be a vague notion for non specialists. To simplify the reasoning done by cryptanalysts, a symmetric primitive is secured when no practical attack have been found against it. A large part of the security demonstration of a primitive consists in trying every classical attack against the studied primitives. In this thesis, we review our cryptanalysis works with this idea by using different algorithms to construct or test our results.

We begin by revisiting the fast near collision attacks proposed in 2018. We proved with test algorithms inspired by information theory that the complexity of this attack is seriously underestimated and gave the correct estimation.

We then gave a new characterization of a particular property of Feistel networks. It allowed us to build a new efficient algorithm to find the optimal permutations (for this property) solving with the constructed permutations a 10 year old problem.

We end this document with the use of solvers, generic algorithms taking a description of a problem in input and returning as output a solution to this problem. In this work, they are used to compute better differential characteristics of the block cipher SKINNY.