

MYAGA403 – Cryptographie

Chiffrement par blocs : construction, critère de résistance et cryptanalyse

Jules Baudrin

jules.baudrin@uvsq.fr

Version du 27/09/26

Résumé

Ce polycopié est le support de 6h de cours/TD. Lors des premières séances, un aperçu général de la cryptographie a été dressé en mettant notamment l'accent sur les définitions de sécurité et sur les objets « idéaux » auxquels se comparer lorsqu'on souhaite mesurer le comportement aléatoire des primitives cryptographiques. Dans la suite, nous nous intéressons plus particulièrement aux primitives de cryptographie symétrique sans aucun doute les plus étudiées : les chiffrements par blocs. En premier lieu, en se concentrant sur l'étude du standard AES, nous présentons les grandes stratégies de conception. Nous nous tournons ensuite vers certains outils de mathématiques discrètes qui permettent à la fois de décrire, étudier et classer les fonctions cryptographiques. Enfin, nous présentons quelques idées de cryptanalyse et d'évaluation de la sécurité en cryptographie symétrique. *Ce cours et ce document sont amenés à évoluer, vos retours (par mail ou de vive voix) sont donc précieux.*

Table des matières

1	Principes de construction d'un chiffrement par blocs	3
1.1	Chiffrement par blocs	3
1.2	Construction itérative	5
1.3	Construction d'une fonction de tour	7
1.4	L'AES (Advanced Encryption Standard)	10
2	Représentations des fonctions booléennes	12
2.1	Table des valeurs	12
2.2	Forme normale algébrique	13
2.3	Représentation univariée	14
2.4	Transformée de Walsh	16
3	Quelques principes de cryptanalyse et de conception	19
3.1	Attaques algébriques	19
3.2	Attaques statistiques	22
3.3	Cryptanalyse différentielle	23
3.3.1	Principe	23
3.3.2	Différentielles et chemins différentiels	24
3.3.3	Analyse d'une fonction de tour	25
3.3.4	Choix des composants du point de vue différentiel	27
3.3.5	Quelques mots sur la résistance optimale aux attaques différentielles	28
3.4	Cryptanalyse linéaire	29

1 Principes de construction d'un chiffrement par blocs

Dans toute la suite, le corps à deux éléments $\{0, 1\}$ (muni de l'addition et la multiplication modulo 2) est noté $\mathbb{F}_2$.

1.1 Chiffrement par blocs

La construction d'un chiffrement est une tâche ardue. L'une des principales difficultés vient du fait qu'un tel algorithme doit être en mesure de traiter des messages de taille arbitraire (choisie à terme par l'utilisateur). Un sous-problème naturel est donc de construire un chiffrement opérant sur des messages de taille *fixe*. En cryptographie symétrique, où la même clé est utilisée pour chiffrer et déchiffrer, cette idée mène directement à la définition d'un *chiffrement par blocs*.

Définition 1.1 (Chiffrement par blocs). Soit n et κ deux entiers naturels non nuls. Un chiffrement par blocs $\mathcal{E}$ opérant sur des messages de taille n (on parle également d'un état de taille n) et des clés de taille κ est une famille de 2^κ bijections $E_k: \mathbb{F}_2^n \xrightarrow{\sim} \mathbb{F}_2^n$, où E_k est indicé par une clé $k \in \mathbb{F}_2^\kappa$, i.e. $\mathcal{E} := (E_k)_{k \in \mathbb{F}_2^\kappa}$.



Attention 1.2. Noter que la définition de chiffrement par blocs est purement mathématique et ne prend en compte aucune propriété cryptographique. Un chiffrement par blocs peut donc être sécurisé ou non-sécurisé.

Exercice 1. Donner une définition équivalente d'un chiffrement par blocs vu comme une fonction à deux entrées et une sortie.

Exercice 2 (Attaques génériques). Quelle attaque peut-on monter si n est trop petit ? Donner une borne sur n satisfaisante pour la plupart des cas d'usage ? Mêmes questions lorsque κ est trop petit.

Exercice 3 (Des ordres de grandeurs en temps).

1. Donner une approximation d'un million et d'un milliard en puissance de 2.
2. Exprimer le nombre de secondes dans une journée en puissance de 2.
3. L'âge de l'Univers est estimé à environ 13,9 milliards d'années. Combien cela représente de secondes ?
4. Sachant qu'un processeur moderne a une fréquence de 6 GHz, (c'est-à-dire qu'il effectue 6 milliards de cycles par seconde), combien d'âge de l'Univers sont nécessaires à un ordinateur pour retrouver par force brute une clé de 128 bits ?
5. La puissance de calcul mondiale est estimée à 2^{70} opérations en virgule flottante par seconde [Int25]. Combien de temps cela nous prendrait-il collectivement ?

À lui seul, un chiffrement par blocs ne permet pas de chiffrer de manière sécurisée, et ce, même si les messages à transmettre sont de taille fixe (*cf.* les définitions de IND-CPA et de chiffrement non-déterministe). En revanche, s'ils sont utilisés dans un *mode d'opération*

approprié, ils peuvent permettre de garantir bien plus que la simple confidentialité (il existe par exemple des chiffrements authentifiés qui garantissent en plus authenticité et intégrité des messages).

Dans l'étude des modes d'opérations, les chiffrements par blocs sont généralement idéalisés comme des familles de permutations pseudo-aléatoires (PRP). Il est donc primordial que les bijections E_k soient *calculatoirement indistinguables* d'une bijection tirée aléatoirement. Il est également nécessaire qu'à la donnée de couples clairs-chiffrés $(x, E_k(x))$, aucune information ne fuite sur la clé k , qui est le seul paramètre secret.

L'indistinguabilité est une définition plus forte de sécurité que l'incapacité de retrouver la clé puisque si la clé est retrouvée, la fonction de chiffrement E_k est complètement connue et donc distinguable d'une fonction aléatoire. À l'inverse, il n'est pas rare (mais ça n'est pas systématique) qu'une propriété distinguante d'un chiffrement par blocs permette dans un second temps de récupérer tout ou partie de la clé.

Exercice 4. Construire un chiffrement par blocs qui peut être distingué d'une permutation tirée aléatoirement avec un avantage presque maximal mais pour lequel le distingué n'apporte aucune information sur la clé.

Construire un chiffrement par blocs c'est donc construire une (gigantesque) famille de bijections (2^{128} bijections lorsque $\kappa = 128$) qui doit à peu de choses près ressembler au tirage de 2^κ bijections aléatoires. En pratique, un chiffrement par blocs se doit également d'être *implémentable efficacement*, c'est-à-dire admettre une description (matérielle ou logicielle) minimaliste. Faire coexister aléatoire et implémentation est intuitivement une tâche difficile : dans certains contextes, le caractère aléatoire d'un objet se mesure même par la taille du plus petit programme permettant de le générer (au sens de Kolmogorov [Wik26b]). Pourtant, comme nous allons le voir, il existe des constructions, relativement simples, qui permettent en pratique d'obtenir des chiffrements par blocs (et plus généralement des primitives symétriques) efficaces et aux propriétés cryptographiques satisfaisantes.

Exercice 5 (Des ordres de grandeur en espace). Soit F une fonction quelconque de $\mathbb{F}_2^n$ vers lui-même.

1. Combien coûte le stockage de F en mémoire ?
2. Supposons que l'on sache que F est bijective. Combien de bits coûte (au minimum) le stockage de F en mémoire ?
NB : Une telle représentation existe et est connue sous le nom de code de Lehmer [Wik26a, Lai88].
3. Un ZettaByte (ZB) correspond à 1000 milliards de Gigaoctets (GB). Sachant que le stockage mondial est estimé à 10,2 ZB en 2024 et (environ 19,3 ZB en 2029) [Hua25, Fig. 23 p. 50], quelle est la plus grande valeur de n pour laquelle on pourrait collectivement stocker une fonction $F: \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$?
4. À l'inverse, combien de fonctions $F: \mathbb{F}_2^{64} \rightarrow \mathbb{F}_2^{64}$ peut-on stocker collectivement ? Et par combien faudrait-il multiplier le stockage actuel pour stocker un chiffrement par blocs avec un état de 64 bits et une clé de 128 bits ?

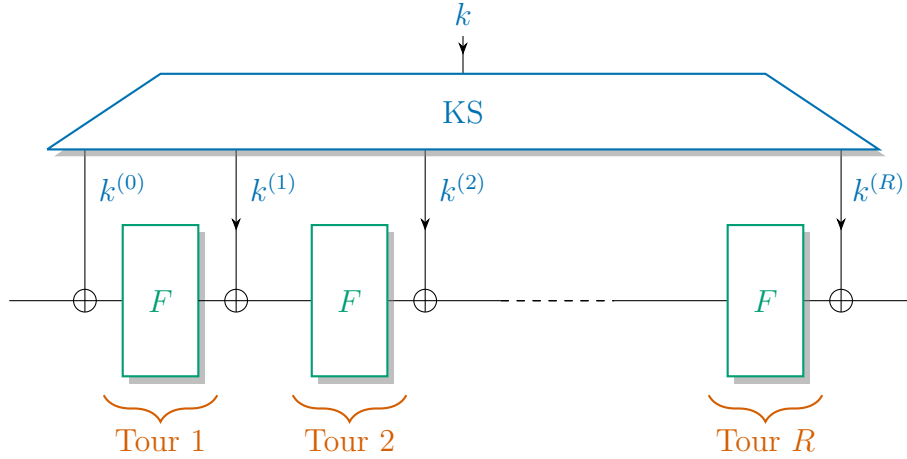


FIGURE 1 : Un chiffrement à clé alternée.

1.2 Construction itérative

Pour les raisons d'implémentation mentionnées plus haut, la quasi-totalité des primitives de cryptographie symétrique, et en particulier les chiffrements par blocs, est construite de manière itérative, c'est-à-dire comme la composition de plusieurs sous-routines bijectives appelées *fonctions de tour* : $E_k = F_{k_R}^{(R)} \circ \dots \circ F_{k_1}^{(1)}$. Chacune de ces fonctions dépend d'une quantité secrète k_i appelée *clé de tour* et dérivée de la *clé maître* k par un algorithme dit de *cadencement de clé*.

Pour réduire encore le coût d'implémentation mais aussi (et surtout) pour permettre une analyse de sécurité plus précise, les fonctions de tour F_i sont en pratique quasi identiques. En effet, elles sont généralement construites à partir d'une unique bijection indépendante de la clé $F: \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$, sous la forme $F_{k_i}^{(i)}(x) := F(x) \oplus k_i \oplus c_i$ où c_i est une constante (dépendante de l'indice de tour mais ni du message ni de la clé). La seule différence entre les fonctions de tour provient alors seulement des clés et constantes de tour. Quitte à modifier un peu la description de l'algorithme de cadencement de clé pour y intégrer l'ajout de constante, les fonctions de tour ont donc généralement la structure suivante $F^{(i)}(x) = F(x) \oplus k_i$. On parle alors de chiffrement à *clé alternée* dont la définition précise et le schéma sont donnés ci-dessous.

Dans la suite, on notera Add_c la fonction d'addition modulo 2 : $\text{Add}_c : x \mapsto x \oplus c$.

Définition 1.3 (Chiffrement itératif (à clé alternée)). Soit $\kappa \geq 1, n \geq 1, R \geq 1$. Un chiffrement itéré à R tours $\mathcal{E} = (E_k: \mathbb{F}_2^n \xrightarrow{\sim} \mathbb{F}_2^n)_{k \in \mathbb{F}_2^\kappa}$ est défini par la donnée :

- d'un algorithme de *cadencement de clé*, $\text{KS}: \mathbb{F}_2^\kappa \rightarrow (\mathbb{F}_2^n)^{R+1}$, et
- d'une fonction de tour, c'est-à-dire, d'une bijection $F: \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$.

Étant donné une clé maître $k \in \mathbb{F}_2^\kappa$, les clés de tour sont données par $\text{KS}(k) := (k^{(0)}, \dots, k^{(R)})$ et le chiffrement instancié E_k défini par :

$$E_k = \text{Add}_{k^{(R)}} \circ F \circ \text{Add}_{k^{(R-1)}} \circ F \circ \text{Add}_{k^{(R-2)}} \circ \dots \circ F \circ \text{Add}_{k^{(0)}}.$$

Les deux exercices qui suivent permettent de mieux cerner deux détails d'importance fournis sans preuve ci-dessus : d'une part l'utilisation de fonctions de tour « quasi identiques » mais non identiques et d'autre part l'apparition d'une clé $k^{(0)}$ dite de blanchiment dans la définition et la Figure 1.

Exercice 6 (Clé de blanchiment). Soit $F: \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ une bijection. On considère les deux chiffrements par blocs suivants :

$$E_k = \text{Add}_{k(R)} \circ F \circ \text{Add}_{k(R-1)} \circ F \circ \text{Add}_{k(R-2)} \circ \cdots \circ \text{Add}_{k(1)} \circ F \text{ et}$$

$$\tilde{E}_k = \text{Add}_{k(R)} \circ F \circ \text{Add}_{k(R-1)} \circ F \circ \text{Add}_{k(R-2)} \circ \cdots \circ \text{Add}_{k(1)} .$$

1. Dans un contexte d'attaques à clairs et chiffrés connus, montrer qu'il est aussi difficile d'attaquer le chiffrement E_k que le chiffrement $\tilde{E}_k$.
2. En déduire l'intérêt d'introduire une addition de clé initiale $k^{(0)}$, dite de blanchiment.

Exercice 7 (Slide attack). On considère un chiffrement à clé alternée dont les fonctions de tour sont toutes *identiques*. On note cette fonction $F_k = \text{Add}_k \circ F$ et le chiffrement :

$$E_k = (\text{Add}_k \circ F) \circ (\text{Add}_k \circ F) \circ \cdots \circ (\text{Add}_k \circ F) .$$

1. Supposons que l'on dispose de deux messages clairs m_1 et m_2 qui vérifient la propriété $m_2 = F_k(m_1)$, ainsi que leurs chiffrés respectifs c_1 et c_2 . Quelle relation en déduit-on sur c_1 et c_2 ? Sous quelle hypothèse a-t-on équivalence?
2. On considère maintenant $\tilde{E}_k = E_k \circ \text{Add}_k$ qui admet une clé de blanchiment en plus. On dit que (m_1, m_2) forme une paire « glissée » lorsque les entrées correspondantes de E_k sont décalées d'un tour, c'est-à-dire lorsque $m_2 = F(m_1 \oplus k)$. Quelles sont les relations obtenues à l'aide d'une paire glissée (m_1, m_2) ?
3. En déduire un moyen de détecter une paire glissée sans connaissance de la clé.
4. Décrire une attaque à clairs et chiffrés connus permettant de trouver une paire glissée puis de retrouver la clé k .

L'un des avantages des chiffrements itératifs est donc de réduire la construction d'une famille de 2^κ bijections à la seule définition des deux fonctions KS et F . L'intuition (fausse *a priori* sans analyse dédiée) derrière une telle construction est qu'elle permet d'atteindre un niveau de sécurité satisfaisant après suffisamment d'itérations, le nombre de tours étant un paramètre complètement dépendant des fonctions F et KS. Comme nous le verrons, les chiffrements itératifs ont également le grand avantage de permettre d'étudier des versions réduites des chiffrements afin de mesurer la *marge de sécurité* de ce dernier (si tant est qu'il soit plus facile d'attaquer un chiffrement réduit à moins de tours que la version initiale, ce qui, bien que vrai dans la plupart des cas, est faux en général). L'utilisation d'une seule et même fonction F permet également, sous des hypothèses précises, de déduire de l'étude de F ou de versions réduites du chiffrement, des propriétés sur un plus grand nombre de tours.

Remarque 1.4 (Fonctions de hachage et permutations cryptographiques). Dans certains contextes, d'autres primitives de cryptographie symétrique sont nécessaires. C'est par exemple le cas du hachage de mot qui nécessite une *fonction de hachage cryptographique*, ou encore de certains modes de chiffrement authentifié qui reposent sur l'utilisation d'une *permutation cryptographique*. Ces fonctions cryptographiques doivent également se rapprocher de fonctions tirées aléatoirement (dans d'autres espaces de fonctions que les fonctions bijectives). Elles sont également construites le plus généralement de manière itérative. À titre d'exemple, certaines permutations cryptographiques sont construites comme un chiffrement instancié E_k pour lequel la clé k est fixe et publiquement décrite dans les spécifications.

1.3 Construction d'une fonction de tour

Si les algorithmes de cadencement de clé sont primordiaux, leur analyse et leur construction suivent moins de règles établies que pour les fonctions de tour. De plus, la majeure partie des analyses de sécurité de chiffrements par blocs ignore (au moins dans un premier temps) les spécificités du cadencement tel que décrit par son concepteur. Dans la suite, nous nous intéresserons donc exclusivement aux fonctions de tour F et présentons deux des plus fameuses familles de constructions.

Réseau de Feistel. Historiquement, la première construction de fonction de tour est appelée *réseau de Feistel*. Les réseaux de Feistel sont utilisés dans de nombreux chiffrements, à l'instar de Lucifer, du standard historique DES [Nat77] ou encore des plus récents Simon [BSS⁺13] et Simeck [YZS⁺15]. De manière intuitive, un schéma de Feistel permet de réduire le problème de construire une *bijection* sur $2n$ bits à la construction d'une *fonction* sur n bits. Un chiffrement par blocs fondé sur un réseau de Feistel est illustré en Figure 2 et la fonction sur n bits y est notée S . Les réseaux de Feistel présentent des avantages certains :

- En termes d'implémentation matérielle comme logicielle, il est intéressant de se reposer sur des fonctions dont les circuits ou tables des valeurs sont aussi petits que possible.
- Comme nous le verrons, il est également plus simple de construire des *fonctions* cryptographiquement optimales (pour des définitions d'optimalité à définir plus tard) que des *bijections* optimales.

Réseau de substitution-permutation (SPN). L'approche suivie par les dénommés *réseaux de substitution-permutations* (souvent abrégés SPN) est légèrement différente. La stratégie est ici de réduire la construction d'une bijection sur n bits à la construction d'une bijection S opérant sur m bits, où m est un *petit* diviseur de n , et d'une bijection *linéaire* L opérant sur n bits. La fonction S est généralement appelée boîte-S (de l'anglais Sbox pour « substitution box »). Elle est utilisée $\frac{n}{m}$ fois en parallèle, tandis que la bijection linéaire L est appliquée à l'état entier de n bits. Un exemple de SPN est illustré en

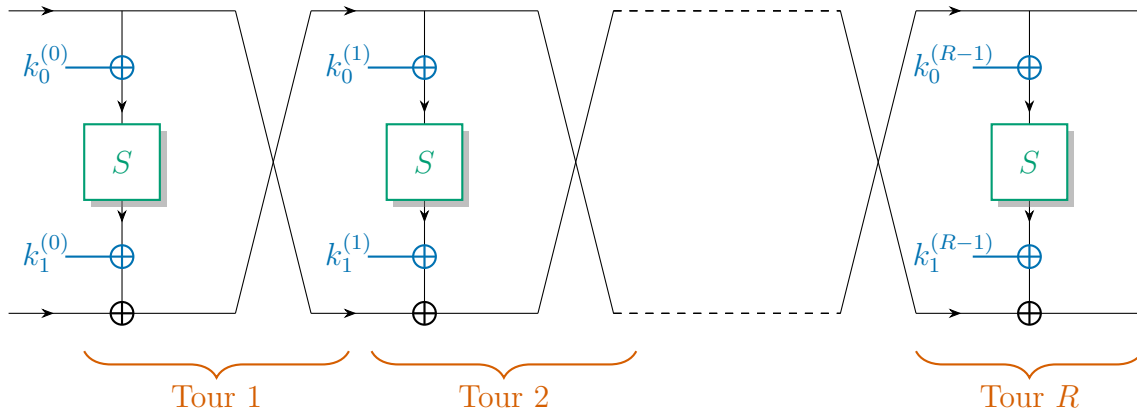


FIGURE 2 : Un réseau de Feistel.

Figure 3. La logique des SPN hérite des principes de Shannon [Sha49] sur la construction de fonctions cryptographiques qui se doivent de fournir :

- de la *confusion*, c'est-à-dire que les relations entre les bits de messages clairs, chiffrés et la clé se doivent d'être « très complexes et intriquées ». Comme nous le verrons, cela se traduit mathématiquement par des relations algébriques de haut degré et contenant de nombreux termes.
- de la *diffusion* visant à ce que chaque bit de sortie dépende de tous les bits du message clair et de la clé.

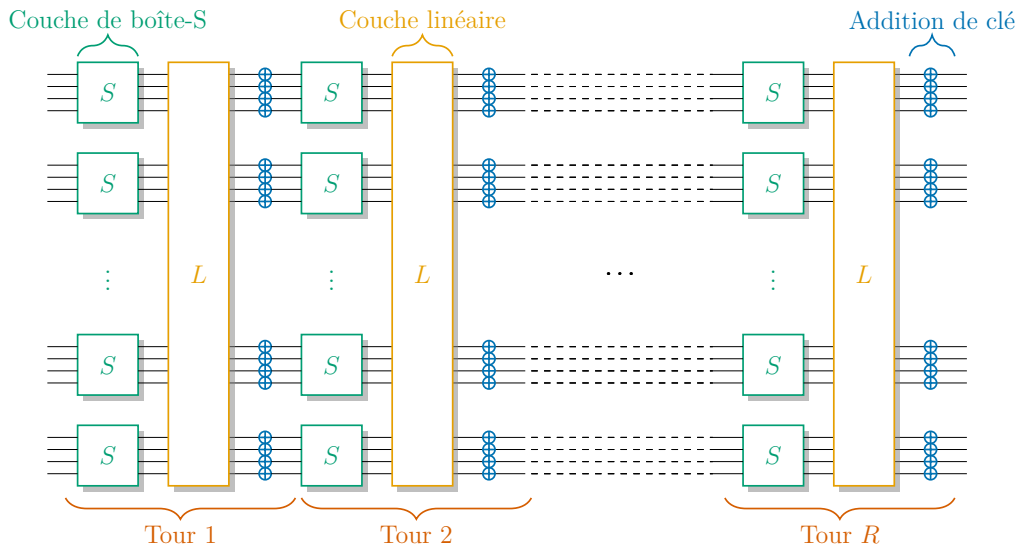


FIGURE 3 : Un réseau de substitution-permutation (SPN).

En particulier, la boîte-S doit nécessairement être non-linéaire sinon la fonction de tour F serait linéaire comme composition de fonctions linéaires et n'atteindrait donc pas le haut degré demandé par le critère de confusion. Les SPN sont également propices à des implémentations efficaces puisque le seul composant non-linéaire, la boîte-S, est réduit à une action locale sur peu de bits qui est donc comparativement moins coûteuse en termes de calculs, de taille de circuit ou de table de valeurs. La couche linéaire L est quant à

elle facile à implémenter et son action peut donc porter globalement sur tout l'état. Ces principes de confusion/diffusion à travers des opérations non-linéaires locales et opérations linéaires globales apparaissent également dans les réseaux de Feistel présentés plus haut.

Exercice 8 (Les boîtes-S sont nécessaires.). On considère un chiffrement itéré dont chaque tour est constitué uniquement d'une addition de clé suivie d'une bijection linéaire L . On note les états intermédiaires x_i , autrement dit $x_{i+1} \stackrel{\text{def}}{=} L(x_i \oplus k^{(i)})$ où x_0 est le message clair. Le chiffrement comporte r tours.

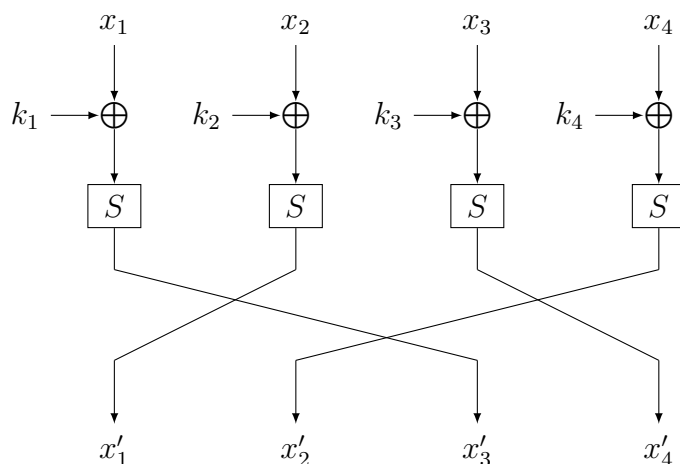
1. Exprimer le chiffré c en fonction du clair m , de L et des différentes sous-clés $k^{(i)}$.
2. Montrer qu'il existe une valeur K , indépendante du message, telle que $E_k(m) = L^r(m) \oplus K$.
3. En déduire une attaque nécessitant un couple clair/chiffré (m, c) . Que peut alors faire un tel adversaire ?

Pour remédier à cela, on propose de transformer le chiffrement. Les fonctions de tour sont linéaires et bijectives mais dépendent désormais de l'indice de tour et de la clé de tour, autrement dit $x_{i+1} \stackrel{\text{def}}{=} L_{k^{(i)}}^{(i)}(x_i)$.

4. Montrer qu'un tel chiffrement n'est toujours pas résistant en décrivant une attaque à clair et chiffré connus. Quelle est la complexité en temps/données/mémoire de cette attaque ?
5. Montrer que, même dans un modèle d'attaque à clairs choisis, la complexité en données de l'attaque ne diminue pas sensiblement.
6. Quel grand principe de construction déduit-on ?

Exercice 9 (Nécessité de la diffusion). Afin d'éviter les attaques de l'exercice précédent, on introduit maintenant une couche non linéaire. On considère un chiffrement par blocs de $n = tm$ bits. L'état est décomposé en t mots de m bits : $x = (x_1, \dots, x_t)$.

Chaque tour est constitué d'une addition de sous-clé, de l'application en parallèle d'une même S-box bijective $S : \mathbb{F}_2^m \rightarrow \mathbb{F}_2^m$ sur chacun des t mots, puis d'une couche linéaire L qui permute les t mots de l'état. Autrement dit, $L(x_1, \dots, x_t) = (x_{\pi(1)}, \dots, x_{\pi(t)})$ pour une certaine permutation π de $\{1, \dots, t\}$. La fonction de tour peut être représentée de la manière suivante :



1. Considérons deux messages clairs qui ne diffèrent que d'un seul mot de m bits, par exemple $m_0 = (x_1, x_2, \dots, x_t)$ et $m_1 = (x'_1, x_2, \dots, x_t)$. Que peut-on dire de leur chiffrés après un nombre quelconque de tours ?
2. En déduire une représentation plus simple du chiffrement.
3. Décrire une attaque (à clairs connus ou choisis) permettant à un adversaire de chiffrer et de déchiffrer sans connaître la clé.
4. Pour complexifier les choses, on décide en plus de rajouter un cadencement de clés. La clé de tour i est désormais calculée comme $k^{(i)} = (k_{\pi^i(1)}, \dots, k_{\pi^i(t)})$ à partir de la clé maître $k = (k_1, \dots, k_t)$. Décrire dans ce cas une attaque de recouvrement de clés.

Pour remédier à cela, on propose de transformer le chiffrement. Les permutations sont désormais différentes à chaque tour et peuvent dépendre de la clé de tour.

5. Montrer que cette modification ne résout pas le problème précédent mis en évidence aux questions 1 à 3.

On considère seulement dans la suite que $t = 4$. Dans un dernier élan d'espoir, on décide de remplacer les couches linéaires de permutation de mots par la fonction linéaire suivante : $L: (x_1, x_2, x_3, x_4) \mapsto (x_3, x_4, x_1 \oplus x_3, x_2 \oplus x_4)$.

6. Montrer que ce changement ne constitue toujours pas un choix sécurisé.
7. Quelle propriété doit donc posséder la couche linéaire d'un SPN ?

Dans la suite, nous présentons le standard international AES qui est de fait le chiffrement par blocs le plus utilisé aujourd'hui. Nous motivons également ses principes de conception et présentons quelques éléments de cryptanalyse.

1.4 L'AES (Advanced Encryption Standard)

L'Advanced Encryption Standard [AES01] (ou plus communément appelé AES) est un chiffrement par blocs standardisé en 2000 par l'agence américaine des standards (NIST) après un processus ouvert de propositions et d'analyse de potentiels candidats qui dura entre 1997 et 2000. Initialement appelé Rijndael [DR02] par ses concepteurs Daemen & Rijmen, l'AES est désormais le standard de chiffrement par blocs international le plus utilisé.

L'AES est un chiffrement opérant sur un état de $n = 128$ bits. Cet état est usuellement représenté comme une matrice 4×4 d'octets, dont les indices sont numérotés de haut en bas, puis de gauche à droite, cf. Figure 4.

La fonction de tour de l'AES, $F: \mathbb{F}_2^{128} \rightarrow \mathbb{F}_2^{128}$, suit une construction SPN. Elle se décompose de la façon suivante :

$$F := \text{MC} \circ \text{SR} \circ S,$$

où les opérations MC, SR, S sont respectivement appelées MixColumns, ShiftRows et couche de boîte-S.

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

FIGURE 4 : Représentation matricielle et numérotation des octets de l'état d'AES.

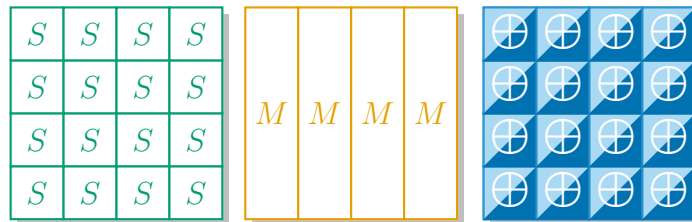


FIGURE 5 : Les couches de boîtes-S, MixColumns et d'addition de clé de la fonction de tour d'AES.

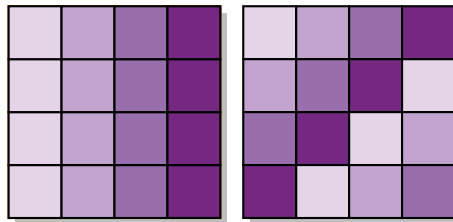


FIGURE 6 : L'état d'AES avant (gauche) et après (droite) l'application de ShiftRows.

Couche de boîte-S. Comme vu précédemment, la couche de boîte-S consiste en l'application d'une même boîte-S bijective $S: \mathbb{F}_2^8 \rightarrow \mathbb{F}_2^8$ sur chaque octet de l'état. La construction et le choix de cette boîte-S seront décrits plus loin.

ShiftRows. L'opération de ShiftRows est une simple réorganisation des octets de l'état : les octets de la ligne d'indice $i \in \llbracket 0, 3 \rrbracket$ (0 étant la ligne la plus haute) sont décalés circulairement de i positions vers la gauche.

MixColumns. La couche de MixColumns consiste en l'application en parallèle d'une application linéaire sur 32 bits sur chaque colonne de l'état. Plus précisément M est une application $\mathbb{F}_{2^8}$ -linéaire (et en particulier $\mathbb{F}_2$ -linéaire). Elle peut donc être vue comme une matrice $M \in \mathcal{M}_{4,4}(\mathbb{F}_{2^8})$ appliquée à chaque vecteur (colonne) d'octets. Le corps à 2^8 éléments est en pratique identifié avec les vecteurs binaires de taille 8 en utilisant $\mathbb{F}_{2^8} := \mathbb{F}_2[X]/(X^8 + X^4 + X^3 + X + 1)$. La matrice M est alors donnée par :

$$M := \begin{pmatrix} 0x2 & 0x3 & 0x1 & 0x1 \\ 0x1 & 0x2 & 0x3 & 0x1 \\ 0x1 & 0x1 & 0x2 & 0x3 \\ 0x3 & 0x1 & 0x1 & 0x2 \end{pmatrix} = \begin{pmatrix} \alpha & 1 + \alpha & 1 & 1 \\ 1 & \alpha & 1 + \alpha & 1 \\ 1 & 1 & \alpha & 1 + \alpha \\ 1 + \alpha & 1 & 1 & \alpha \end{pmatrix}$$

où α est une racine du polynôme irréductible $X^8 + X^4 + X^3 + X + 1$. La couche linéaire d'AES est ainsi la composition de **ShiftRows** suivie de **MixColumns** : $L := \text{MC} \circ \text{SR}$.

La fonction de tour est naturellement suivie d'une addition de clé de tour dérivée de la clé maître. Le nombre de tours, la taille de la clé maître et le cadencement de clé dépendent des instances d'AES. La version la plus utilisée et la plus largement étudiée est la version AES-128 avec une clé de taille $\kappa = 128$ et $R = 10$ tours. Il existe également AES-192 pour laquelle $\kappa = 192$, $R = 12$ et AES-256 pour laquelle $\kappa = 256$, $R = 14$.

Exercice 10. D'après les spécifications exactes de l'AES, l'opération **MixColumns** est omise au dernier tour (contrairement aux $R - 1$ premiers tours). Dit autrement,

$$\text{AES}_k(x) = (\text{Add}_{k(10)} \circ \tilde{F}) \circ (\text{Add}_{k(9)} \circ F) \cdots \circ (\text{Add}_{k(1)} \circ F) \circ \text{Add}_{k(0)},$$

où F est la fonction de tour décrite plus haut et $\tilde{F} = \text{SR} \circ \mathcal{S}$. On définit également une version modifiée d'AES avec un **MixColumns** final :

$$\widetilde{\text{AES}}_k(x) = (\text{Add}_{k(10)} \circ F) \circ (\text{Add}_{k(9)} \circ F) \cdots \circ (\text{Add}_{k(1)} \circ F) \circ \text{Add}_{k(0)}.$$

1. Décrire F en fonction de $\tilde{F}$ et trouver une relation entre **AES**, $\widetilde{\text{AES}}$ et **MixColumns**.
2. En déduire que **AES** est aussi sécurisé qu' $\widetilde{\text{AES}}$.
3. Peut-on encore simplifier le dernier tour sans modifier la sécurité ?

Pour l'instant, nous avons seulement survolé la présentation de l'AES mais aucun argument de sécurité n'a été présenté. Pour cela, nous avons besoin de quelques outils de mathématiques discrètes qui nous permettront de décrire certaines stratégies d'attaques et les contre-mesures pensées pour les contrer.

2 Représentations des fonctions booléennes

La très grande majorité des fonctions étudiées en cryptographie (symétrique) sont des fonctions dont les entrées et sorties sont des mots binaires. Une fonction de $\mathbb{F}_2^n$ vers $\mathbb{F}_2$ sera appelée *fonction booléenne* tandis qu'une fonction de $\mathbb{F}_2^n$ vers $\mathbb{F}_2^m$ sera appelée *fonction booléenne vectorielle*.

2.1 Table des valeurs

Comme pour toute fonction, la façon la plus naturelle de décrire une fonction $F: \mathbb{F}_2^n \rightarrow \mathbb{F}_2^m$ est de considérer sa *table de valeurs* ou son *graphe*, c'est-à-dire l'ensemble des couples $(x, F(x))$ pour tout $x \in \mathbb{F}_2^n$. Il s'agit donc d'un tableau à 2^n entrées de m bits donc son exploration coûte 2^n opérations et son stockage de l'ordre de $m2^n$.

Ce coût en mémoire implique que la table des valeurs n'est utilisée dans une implémentation logicielle (comme en cryptanalyse) que lorsque n et m sont relativement petits, typiquement pour des $n, m \leq 8$.

Cependant, $\mathbb{F}_2$ étant un corps fini, toutes les fonctions de $\mathbb{F}_2^n$ vers $\mathbb{F}_2$ s'avèrent être des fonctions polynomiales. Les représentations polynomiales seront donc souvent privilégiées pour décrire et étudier une primitive.

2.2 Forme normale algébrique

La plus naturelle des représentations polynomiales pour une fonction $f: \mathbb{F}_2^n \rightarrow \mathbb{F}_2$ est sa représentation multivariée décrite ci-dessous.

Proposition 2.1 (Forme normale algébrique). *Soit n un entier naturel et $f: \mathbb{F}_2^n \rightarrow \mathbb{F}_2$ une fonction booléenne. Alors il existe un unique polynôme $P \in \mathbb{F}_2[X_1, \dots, X_n]/(X_1^2 + X_1, \dots, X_n^2 + X_n)$ vérifiant $\forall x \in \mathbb{F}_2^n, f(x) = P(x)$. Ce polynôme est appelé la forme normale algébrique de f .*

Démonstration. Il s'agit essentiellement d'une version multivariée du théorème d'interpolation de Lagrange.

Existence. Pour tout $1 \leq i \leq n$ et $y \in \mathbb{F}_2$, notons $P_{i,y} = X_i + (1 + y)$ l'interpolant qui vaut 1 lorsque $X_i = y$ et 0 sinon. Étant donné $(y_1, \dots, y_n) \in \mathbb{F}_2^n$, le polynôme multivarié $Q_{y_1, \dots, y_n} = \prod_{i=1}^n P_{i,y_i}$ vaut donc 1 lorsque $X_i = y_i$ pour tout i et 0 sinon. Ainsi le polynôme $P = \sum_{x \in \mathbb{F}_2^n} f(x) Q_x$ est un polynôme d'interpolation de la fonction f . Son degré en chaque variable est effectivement inférieur ou égal à 1.

Unicité. D'une part, il existe 2^{2^n} fonctions de $\mathbb{F}_2^n$ vers $\mathbb{F}_2$ (2^n choix d'images à faire parmi 2 valeurs). C'est également le nombre de polynômes dans $\mathbb{F}_2[X_1, \dots, X_n]/(X_1^2 + X_1, \dots, X_n^2 + X_n)$. En effet, à chaque vecteur binaire $(u_1, \dots, u_n) \in \mathbb{F}_2^n$ correspond un monôme multivarié $\prod_{i=1}^n X_i^{u_i}$. Il y a donc 2^n monômes dont le degré est inférieur ou égal à 1 en toutes les variables et donc 2^{2^n} polynômes (correspondant au choix des coefficients parmi 0 ou 1). L'unicité est donc prouvée. $\square$

De manière générale, pour un vecteur binaire $(u_1, \dots, u_n) \in \mathbb{F}_2^n$, on notera X^u le monôme $X^u \stackrel{\text{def}}{=} \prod_{i=1}^n X_i^{u_i}$ de sorte que l'ANF d'une fonction f puisse être décrite sous la forme suivante :

$$\sum_{u \in \mathbb{F}_2^n} a_u X^u, \quad \text{où } a_u \in \mathbb{F}_2, \forall u \in \mathbb{F}_2^n.$$

Du fait de cette correspondance bijective, on identifiera sans ambiguïté une fonction et sa représentation polynomiale.

En particulier, le *degré algébrique* de f , noté $\deg(f)$, n'est rien d'autre que le degré de son ANF. C'est cette notion de degré qui a été mentionnée précédemment comme critère de confusion et qui se doit d'être le plus haut possible comme nous le reverrons en Section 3.1.

L'ANF se généralise naturellement au cas d'une fonction vectorielle $F: \mathbb{F}_2^n \rightarrow \mathbb{F}_2^m$. Dans ce cas, il s'agit de la collection formée par les m ANF de ses fonctions *coordonnées* et le degré algébrique $\deg(F)$ est le maximum de leurs degrés respectifs.

Définition 2.2 (Coordonnées et composantes). Soit $F: \mathbb{F}_2^n \rightarrow \mathbb{F}_2^m$. Les fonctions coordonnées de F sont les m uniques fonctions booléennes $F_1, \dots, F_m: \mathbb{F}_2^n \rightarrow \mathbb{F}_2$ vérifiant la propriété suivante :

$$\forall x_1, \dots, x_n \in \mathbb{F}_2, \quad F(x) = (F_1(x_1, \dots, x_n), \dots, F_m(x_1, \dots, x_n)).$$

Toute combinaison linéaire de fonctions coordonnées, c'est-à-dire toute fonction de la forme $\sum_{i=1}^m \alpha_i F_i$ où $\alpha_i \in \mathbb{F}_2$ pour tout i est appelée une *composante* de F . Étant donné $\alpha = (\alpha_1, \dots, \alpha_m) \in \mathbb{F}_2^m$, on notera également $\alpha \cdot F \stackrel{\text{def}}{=} \sum_{i=1}^m \alpha_i F_i$ la composante associée à α (où $\cdot$ est le produit scalaire standard sur $\mathbb{F}_2^m$).

Il existe donc une correspondance bijective entre tables des valeurs et représentations multivariées. Cette correspondance est explicitée dans la proposition suivante. Étant donné $u, v \in \mathbb{F}_2^n$, on rappelle que le support de u est l'ensemble $\text{Supp}(u) \stackrel{\text{def}}{=} \{i \in \llbracket 1, n \rrbracket \mid u_i = 1\}$. On notera également $u \preceq v$ si le support de u est un sous-ensemble du support de v ou de manière équivalente si x^u divise x^v .

Proposition 2.3 (Relations coefficients-valeurs). Soit $f: \mathbb{F}_2^n \rightarrow \mathbb{F}_2$ la fonction booléenne $f = \sum_{u \in \mathbb{F}_2^n} a_u X^u$. Alors :

$$\forall x \in \mathbb{F}_2^n, \quad f(x) = \sum_{u \preceq x} a_u \quad \text{et} \quad \forall u \in \mathbb{F}_2^n, \quad a_u = \sum_{v \preceq u} f(v).$$

On peut par exemple en déduire que le degré des bijections n'atteint *jamais* le nombre de variables.

Proposition 2.4 (Degré des bijections). Soit $F: \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ une fonction bijective. Alors $\deg(F) \leq n - 1$.

Exercice 11.

1. Soit $u, v \in \mathbb{F}_2^n$. On rappelle que $v^u = \prod_{i=1}^n v_i^{u_i} \in \mathbb{F}_2$. Montrer que $v^u = 1$ si et seulement si $u \preceq v$. En déduire la première formule de la Proposition 2.3.
2. En déduire la deuxième formule de la Proposition 2.3.
3. Soit F une bijection. Que vaut $a_{(1,1,\dots,1)}$? En déduire la Proposition 2.4.

La conversion table des valeurs $\leftrightarrow$ ANF est donc une transformation de type « transformée de Möbius » qui s'implémente très bien en pratique avec un algorithme en $O(n2^n)$. Pour plus d'informations, voir [Can16, Section 1.1.2] ou [Jou09, Page 287].

2.3 Représentation univariée

Le résultat d'interpolation présenté plus haut avec l'ANF est en fait plus général.

Proposition 2.5 (Interpolation multivariée). Soit s un entier naturel et $q = p^t$ une puissance d'un nombre premier. Soit $F: \mathbb{F}_q^s \rightarrow \mathbb{F}_q$. Alors il existe un unique polynôme $P \in \mathbb{F}_q[X_1, \dots, X_s]/(X_1^q - X_1, \dots, X_s^q - X_s)$ vérifiant $\forall x \in \mathbb{F}_q^s, F(x) = P(x)$.

Démonstration. La preuve est la même en remplaçant l'interpolant $P_{i,y}$ par sa forme plus générale (et plus connue) $P_{i,y} := \prod_{x \in \mathbb{F}_q \setminus \{y\}} \frac{X_i - y}{x - y}$. $\square$

Dans le cas particulier d'une fonction booléenne vectorielle où $n = m$, c'est-à-dire $F: \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$, le résultat précédent fournit une nouvelle représentation polynomiale à F , cette fois-ci, sous forme *univariée*. En effet, le $\mathbb{F}_2$ -espace vectoriel (canonique) $\mathbb{F}_2^n$ peut être identifié avec n'importe quel espace vectoriel de même dimension, et notamment avec le corps fini à 2^n éléments $\mathbb{F}_{2^n}$. Soit $\Phi, \Psi: \mathbb{F}_2^n \rightarrow \mathbb{F}_{2^n}$ des isomorphismes (d'espaces vectoriels). La fonction $\Psi \circ F \circ \Phi^{-1}$ est alors une fonction de $\mathbb{F}_{2^n}$ vers lui-même, qui, en vertu de la Proposition 2.5 (pour $n = 1, p = 2, t = n$), admet une représentation univariée $P = \sum_{i=0}^{2^n-1} a_i X^i$ avec $a_i \in \mathbb{F}_{2^n}$ pour tout i . Contrairement à l'ANF, sommes et produits ont ici lieu dans le corps $\mathbb{F}_{2^n}$.



Attention 2.6. La représentation univariée d'une fonction booléenne n'est ni cano-
nique, ni unique. Elle dépend intrinsèquement du choix des identifications Φ, Ψ qui
doivent être explicitées. Sans isomorphismes explicites, la fonction n'est pas uniquement
déterminée : à partir d'un polynôme $P \in \mathbb{F}_{2^n}[X]$, les fonctions booléennes $\Psi^{-1} \circ P \circ \Phi$
et $\Psi'^{-1} \circ P \circ \Phi'$ auront (pour un bon choix d'identifications) une même représentation
univariée mais *pas la même table des valeurs*. Il s'agit donc de fonctions distinctes. En
revanche, si Φ et Ψ sont fixés, la correspondance est dans ce cas bijective modulo $X^{2^n} + X$.

En pratique, les isomorphismes Φ, Ψ entre $\mathbb{F}_2^n$ et $\mathbb{F}_{2^n}$ sont dictés par la façon dont le
corps $\mathbb{F}_{2^n}$ est implémenté en machine. Le plus souvent, $\Phi = \Psi$ et ils correspondent à une
identification entre un mot binaire $(c_0, \dots, c_{n-1})$ et le polynôme associé $\sum_{i=0}^{n-1} c_i \alpha^i$ où α
est une racine d'un polynôme irréductible de degré n comme rappelé ci-dessous.

Proposition 2.7 (Corps finis à 2^n éléments). Soit n un entier naturel non nul. Soit P
un polynôme irréductible de degré n . Alors $\mathbb{F}_2[X]/(P)$ est un corps à 2^n éléments.

Exemple 2.8 (Boîte-S de l'AES). Comme vu précédemment, l'AES utilise comme sous-
routine une boîte $S: \mathbb{F}_2^8 \rightarrow \mathbb{F}_2^8$. Cette boîte-S est définie comme la composition $S = A \circ S'$
de deux opérations $A, S': \mathbb{F}_2^8 \rightarrow \mathbb{F}_2^8$ où :

- A est une bijection $\mathbb{F}_2$ -affine, et
- $S' = \Phi^{-1} \circ P \circ \Phi$ avec $P = X^{254}$, $\Phi: \mathbb{F}_2^8 \rightarrow \mathbb{F}_{256}, (c_0, \dots, c_7) \mapsto \sum_{i=0}^7 c_i X^i$ où les
opérations polynomiales sont effectuées modulo le polynôme irréductible $X^8 + X^4 +$
 $X^3 + X + 1$.

Par exemple, on vérifie facilement que cette boîte-S est bijective puisque A et Φ le sont et
que l'opération $X \mapsto X^{254}$ correspond (presque) à l'inversion dans $\mathbb{F}_{256}$ qui est par nature
une opération bijective. En effet $0^{254} = 0$ et pour tout $x \neq 0$, $x \times x^{254} = x^{255} = 1$ donc
 $x^{254} = x^{-1}$.

Il existe également des liens forts entre ANF et représentation univariée. Par exemple, le degré algébrique (à ne pas confondre avec le degré univarié) peut se lire directement depuis la forme univariée.

Proposition 2.9. Soit $F: \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ et $P = \sum_{i=0}^{2^n-1} a_i X^i$ une représentation univariée de F . Alors $\deg_{\text{alg}}(F) = \max \{\text{wt}(i) \mid 0 \leq i < 2^n, a_i \neq 0\}$. Autrement dit, le degré algébrique est le plus haut poids de Hamming des exposants présents dans une représentation univariée.

Exercice 12. Soit $F: \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$. Notons $M(P) = \max \{\text{wt}(i) \mid 0 \leq i < 2^n, a_i \neq 0\}$ le plus haut poids de Hamming des exposants d'un polynôme $P \in \mathbb{F}_2[X]$.

1. Soit $\Phi, \Psi: \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ des isomorphismes d'espaces vectoriels. En développant l'expression de $\Psi^{-1} \circ F \circ \Phi$, montrer que $\deg_{\text{alg}}(F) \leq M(\Psi^{-1} \circ F \circ \Phi)$.
2. Pour tout $0 \leq d \leq n$, on considère les fonctions φ_d définies par :

$$\varphi_d: \left\{ P \in \mathbb{F}_2[X] / (X^{2^n} + X), M(P) \leq d \right\} \longrightarrow \left\{ F: \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n, \deg_{\text{alg}}(F) \leq d \right\}$$

$$P \longmapsto (x \mapsto P(x)).$$

Montrer que les fonctions φ_d sont injectives et en déduire l'égalité $\deg_{\text{alg}}(F) = M(\Psi^{-1} \circ F \circ \Phi)$ par un argument de dénombrement.

3. Quel est le degré algébrique de la boîte-S de l'AES ?

2.4 Transformée de Walsh

Comme déjà mentionné, une fonction cryptographique se doit d'être « la plus éloignée possible » d'une fonction linéaire. Si le degré algébrique est un premier indicateur d'une telle distance, il existe des fonctions de haut degré dont les propriétés sont très proches de fonctions linéaires ou affines. Pensez par exemple à une fonction $x \mapsto x_1 x_2 \dots x^n + x_1$ qui se comporte presque comme $x \mapsto x_1$ puisque $x_1 x_2 \dots x^n$ est très souvent nul. Pour mesurer une telle distance, une représentation plus adaptée à l'opération d'addition est souvent utilisée : la transformée de Walsh.

Nous avons déjà montré avec l'ANF que la famille de fonctions $x \mapsto Q_y$ pour tout $y \in \mathbb{F}_2^n$ est une base¹ des fonctions booléennes puisque $f = \sum_{y \in \mathbb{F}_2^n} f(y) Q_y$, et même des fonctions de $\mathbb{F}_2^n$ à valeurs dans $\mathbb{C}$. La théorie de Fourier fournit une autre base, la base des caractères (additifs) de $\mathbb{F}_2^n$.

Définition 2.10 (Caractère). Soit $\mathbb{G}$ un groupe. Un caractère de $\mathbb{G}$ est un morphisme de $\mathbb{G}$ dans $\mathbb{C}^*$.

Proposition 2.11 (Caractères de $\mathbb{F}_2^n$). Les caractères de $\mathbb{F}_2^n$ sont les fonctions χ_u pour $u \in \mathbb{F}_2^n$ définies par : $\chi_u: x \mapsto (-1)^{u \cdot x}$, où $u \cdot x$ est le produit scalaire usuel.

1. Il s'agit en fait de la base des fonctions de Dirac puisque Q_y est défini comme valant 1 en y et 0 ailleurs.

Exercice 13 (Preuve de Proposition 2.11).

1. Montrer que les χ_u sont des caractères de $\mathbb{F}_2^n$.
2. Soit χ un caractère. Montrer que $\forall x \in \mathbb{F}_2^n, \chi(x) \in \{-1, 1\}$.
3. En décomposant x dans une base $(f_1, \dots, f_n)$ et en explicitant la valeur de chaque $\chi(f_i)$, déduire que $\chi = \chi_u$ pour un certain $u \in \mathbb{F}_2^n$.
4. Conclure.

Théorème 2.12 (Transformée de Fourier discrète). *La famille $(\chi_u)_{u \in \mathbb{F}_2^n}$ est une base de $\mathbb{C}^{\mathbb{F}_2^n}$. Autrement dit, pour toute fonction $g: \mathbb{F}_2^n \rightarrow \mathbb{C}$, il existe une unique suite $(\hat{g}(u))_{u \in \mathbb{F}_2^n}$ de coefficients vérifiant $g = \sum_{u \in \mathbb{F}_2^n} \hat{g}(u) \chi_u$. Ces coefficients sont appelés coefficients de Fourier de g et la fonction définie par $u \mapsto \hat{g}(u)$, naturellement notée $\hat{g}$, est appelée transformée de Fourier de g .*

Exercice 14 (Preuve de la transformée de Fourier).

1. Soit $u \in \mathbb{F}_2^n$. Montrer que :

$$\sum_{v \in \mathbb{F}_2^n} \chi_u(v) = \begin{cases} 2^n & \text{si } u = 0 \\ 0 & \text{sinon,} \end{cases}$$

2. Soient $u, w \in \mathbb{F}_2^n$. À l'aide de la question précédente, montrer la relation d'orthogonalité des caractères :

$$\sum_{v \in \mathbb{F}_2^n} \chi_u(v) \chi_w(v) = \begin{cases} 2^n & \text{si } u = w, \\ 0 & \text{sinon.} \end{cases}$$

3. En déduire que la famille $(\chi_u)_{u \in \mathbb{F}_2^n}$ est une famille libre.
4. Finir la preuve du Théorème 2.12.

Le résultat précédent est un résultat non-constructif puisqu'il exprime g fonction de $\hat{g}$ sans expliciter $\hat{g}$. Il existe une formule bien connue permettant au contraire d'exprimer $\hat{g}$ fonction de g , essentiellement de la même manière.

Proposition 2.13 (Transformée de Fourier inverse). *Soit $g: \mathbb{F}_2^n \rightarrow \mathbb{C}$. Alors la fonction $\hat{g}$ est donnée par : $\hat{g} = 2^{-n} \sum_{u \in \mathbb{F}_2^n} g(u) \chi_u$.*

Exercice 15. Prouver le résultat précédent.

Étant donnée une fonction booléenne $f: \mathbb{F}_2^n \rightarrow \mathbb{F}_2$, il est tout à fait possible d'étudier sa transformée de Fourier en identifiant $\mathbb{F}_2$ avec $\{0, 1\} \subset \mathbb{C}$. En pratique, il est cependant beaucoup plus agréable de l'identifier avec $\{-1, 1\} \in \mathbb{C}^*$. On utilisera donc quasi-exclusivement la transformée de Fourier de la fonction $x \mapsto (-1)^{f(x)}$.

Définition 2.14 (Transformée de Walsh). Soit $f: \mathbb{F}_2^n \rightarrow \mathbb{F}_2$. La transformée de Walsh de f , notée W_f , est (à un scalaire près) la transformée de Fourier de $(-1)^f$. Plus précisément, elle est définie par :

$$\forall v \in \mathbb{F}_2^n, \quad W_f(v) \stackrel{\text{def}}{=} \sum_{u \in \mathbb{F}_2^n} (-1)^{f(u)} \chi_u(v) = \sum_{u \in \mathbb{F}_2^n} (-1)^{f(u)+v \cdot u}.$$

Par définition, les coefficients de Walsh d'une fonction booléenne f capturent la distance (signée) entre f et les fonctions linéaires. En effet, de manière analogue à Proposition 2.11, les fonctions linéaires de $\mathbb{F}_2^n$ vers $\mathbb{F}_2$ sont exactement les fonctions $x \mapsto u \cdot x$ pour un certain $u \in \mathbb{F}_2^n$. La valeur $W_f(u)$ mesure donc le nombre de fois où f coïncide avec $x \mapsto u \cdot x$.

Exercice 16. Soit $f: \mathbb{F}_2^n \rightarrow \mathbb{F}_2$. Soit $u \in \mathbb{F}_2^n$. Montrer que

$$W_f(u) = 2 |\{x \in \mathbb{F}_2^n \mid f(x) = u \cdot x\}| - 2^n.$$

On notera en particulier que le coefficient en 0 donne le *biais* de la fonction f , c'est-à-dire la différence entre le nombre de fois où la fonction prend la valeur 0 et la valeur 1. Puisqu'il est attendu qu'une fonction tirée aléatoirement prenne autant de fois les deux valeurs, les fonctions *équilibrées*, c.-à-d. qui vérifient $W_f(0) = 0$ sont privilégiées en cryptographie.

Pour une fonction booléenne vectorielle $F: \mathbb{F}_2^n \rightarrow \mathbb{F}_2^m$, on peut étudier la transformée de Walsh de toutes ces composantes. Pour tout $\alpha \in \mathbb{F}_2^n, \beta \in \mathbb{F}_2^m$, on notera donc $W_F(\alpha, \beta)$ le coefficient de Walsh de $\beta \cdot F$ en α , ou encore :

$$W_F(\alpha, \beta) \stackrel{\text{def}}{=} W_{\beta \cdot F}(\alpha) = \sum_{x \in \mathbb{F}_2^n} (-1)^{\alpha \cdot x + \beta \cdot F(x)}.$$

Une fonction cryptographique doit donc avoir des coefficients de Walsh aussi petits que possible pour être « éloignée » d'un comportement linéaire. On définit d'ailleurs généralement l'indicateur suivant comme un indicateur à minimiser lors de la conception d'une fonction cryptographique.

Définition 2.15 (Linéarité). Soit $F: \mathbb{F}_2^n \rightarrow \mathbb{F}_2^m$. La linéarité de F est l'entier $\mathcal{L}(F)$ défini par :

$$\mathcal{L}(F) \stackrel{\text{def}}{=} \max_{\alpha \in \mathbb{F}_2^n, \beta \in \mathbb{F}_2^m \setminus \{0\}} |W_F(\alpha, \beta)|.$$

La cryptanalyse dite linéaire [TG92, Mat94a, Mat94b], présentée en Section 3.4 repose précisément sur l'exploitation de coefficients de Fourier anormalement élevés qui se traduit par une linéarité élevée.

Il est donc naturel de chercher les fonctions les plus éloignées des fonctions affines possibles. On peut montrer que pour $f: \mathbb{F}_2^n \rightarrow \mathbb{F}_2$, la linéarité de f vérifie $\mathcal{L}(f) \geq 2^{\frac{n}{2}}$. Les fonctions pour lesquelles il y a égalité sont appelées fonctions *courbes* et n'existent que pour n pair. Il existe de nombreux problèmes ouverts relatifs à ces questions comme déterminer

la linéarité minimale en dimension impaire (supérieure à 9) ou encore la linéarité minimale pour les fonctions équilibrées en dimension paire.

3 Quelques principes de cryptanalyse et de conception

Maintenant munis de représentations adéquates, nous pouvons aborder certaines techniques d'attaques et les contre-mesures mises en place pour tenter de les rendre infructueuses.

3.1 Attaques algébriques

Un premier vecteur d'attaques. Dans un contexte d'attaques à clairs connus contre un chiffrement par blocs, un adversaire récupère certains couples clair-chiffré $y = E_k(x)$ pour une fonction E_k inconnue (puisque E_k dépend de la clé secrète). L'attaque sans doute la plus naturelle, et dont les idées sont déjà présentes dans les travaux de Shannon [Sha49], consiste à modéliser algébriquement les relations entre données connues et données inconnues. Comme vu précédemment, une fois la clé fixée, chaque bit de sortie $z \in \mathbb{F}_2$ peut être exprimé polynomialement en fonction des entrées $x \in \mathbb{F}_2^n$ (grâce à l'ANF f de la fonction coordonnée en question) :

$$z = f(x) = \sum_{u \in \mathbb{F}_2^n} a_u(k) x^u, \quad (1)$$

où $a_u(k)$ est une quantité *inconnue* qui dépend des spécifications du chiffrement et surtout *dépendante de la clé*.

Lorsque la clé k est fixée (comme c'est le cas en pratique), connaître la valeur de tous les $a_u(k)$, $u \in \mathbb{F}_2^n$, c'est être en mesure de chiffrer sans même nécessairement connaître k , voire déchiffrer en inversant le système. Or dans un contexte à clair connu, le clair x est connu, donc les x^u aussi et le chiffré z l'est également. En d'autres termes, chaque couple clair-chiffré fournit, pour chaque bit de sortie, une équation linéaire en les $a_u(k)$. A priori, le nombre d'inconnues est de 2^n . En utilisant le pivot de Gauss, il faudrait donc environ 2^n données et environ $2^{n\omega}$ opérations binaires où ω est la constante d'algèbre linéaire ($2 \leq \omega \leq 3$), ce qui ne constitue pas une attaque. Il s'agit cependant d'une borne supérieure très grossière qui suppose que pour chaque $u \in \mathbb{F}_2^n$, la fonction $k \mapsto a_u(k)$ n'est pas identiquement nulle ce qui n'est que rarement, voire jamais² le cas. Ainsi, une borne d sur le degré de f en les variables x , aussi grossière soit-elle, fait baisser ces complexités à M en données et M^ω en temps où $M = \sum_{i=0}^d \binom{n}{i}$. Il est donc primordial de s'assurer du degré mais aussi et surtout de la *densité* des équations qui peuvent être construites par un adversaire (puisque une équation de haut degré peut tout à fait contenir peu de termes $a_u(k)$). C'est à ce titre que la confusion est nécessaire.

2. Nous avons par exemple vu plus haut que le degré d'une bijection est toujours borné par $n - 1$ et donc le coefficient $a_{(1, \dots, 1)}$ est toujours nul, indépendamment de k .

Attaques par linéarisation. Il n'existe cependant pas qu'une seule modélisation algébrique et toute nouvelle description est un potentiel vecteur d'attaque supplémentaire. Par exemple, la modélisation précédente fait intervenir M inconnues où M est le nombre de a_u non identiquement nul. Pourtant, fidèle au principe de Kerckhoffs [Ker83], les seules véritables inconnues sont les κ bits de clé. Dit autrement, chaque coefficient a_u admet lui aussi une ANF : $a_u(k) = \sum_{v \in \mathbb{F}_2^\kappa} \alpha_{u,v} k^v$ dans laquelle les $\alpha_{u,v}$ sont *a priori* connus de tous. Il est donc tout à fait possible de ré-exprimer l'équation 1 comme suit :

$$z = f(x) = \sum_{u \in \mathbb{F}_2^n} \sum_{v \in \mathbb{F}_2^\kappa} \alpha_{u,v} k^v x^u = \sum_{v \in \mathbb{F}_2^\kappa} \underbrace{\left(\sum_{u \in \mathbb{F}_2^n} \alpha_{u,v} x^u \right)}_{\beta_v} k^v = \sum_{v \in \mathbb{F}_2^\kappa} \beta_v k^v.$$

Il s'agit désormais d'une équation non-linéaire en les bits de clé puisque les β_v sont connus. Savoir résoudre un système constitué de telles équations, c'est être en mesure de récupérer la clé secrète k .

Pour cela, la méthode la plus naïve est la *linéarisation*. Cette méthode consiste à convertir un système non-linéaire en un système linéaire (donc facile à résoudre) avec (beaucoup) plus d'inconnues. Pour cela, il suffit de remplacer chaque $k^v, v \in \mathbb{F}_2^\kappa$ par une nouvelle variable y_v et de résoudre le système linéaire formé d'équations de la forme :

$$z = \sum_{v \in \mathbb{F}_2^\kappa} \beta_v y_v.$$

Parmi les solutions de ce système se trouve la solution du système initial. Le coût de la résolution est alors de N^ω en temps et N en données où N est désormais le nombre de monômes distincts en les variables k_i qui apparaissent dans les ANF des coefficients $a_u(k)$. Là encore, il est donc nécessaire que leur nombre soit aussi grand que possible, et en particulier que le degré en les variables de clé soit élevé. À l'inverse, si le degré en les variables de clé est borné par d , alors $N \leq \sum_{i=0}^d \binom{\kappa}{i}$.

Remarque 3.1. Pour monter une telle attaque, un adversaire doit donc être en mesure de calculer, voire de stocker la valeur de tous les $\alpha_{u,v}$ et/ou β_v . Si ces valeurs sont théoriquement publiques, cela nécessite en pratique de calculer tout ou partie de l'ANF de E_k . Si l'ANF de la fonction de tour F est usuellement simple à calculer, le dimensionnement du nombre de tours R est notamment guidé par la volonté de rendre le calcul de l'ANF de F^R (par exemple en composant R fois avec elle-même l'ANF de F) impossible à calculer et/ou à stocker.

Des attaques algébriques (beaucoup) plus élaborées. Les attaques algébriques ne se résument en pratique pas à ces simples modélisations et il existe de nombreuses autres techniques de *modélisation* et de *résolution* souvent conçues conjointement et adaptées à une primitive spécifique.

Concernant les modélisations, on notera que les équations utilisées plus haut sont linéaires en les bits de sortie (puisque seul z apparaît). Néanmoins, un adversaire peut

tout à fait identifier des relations entre clair et chiffré qui sont de degré plus élevé, comme énoncé ci-dessous.

Proposition 3.2. *Soit $F: \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$. Il existe au moins $\sum_{i=0}^d \binom{i}{2n} - 2^n$ relations de degré au plus d entre les variables d'entrées et de sorties d'une fonction F .*

Démonstration. Adaptée de [Can16]. Une relation de degré au plus d est la donnée de coefficients $c_{u,v}$ pour tout $u, v \in \mathbb{F}_2^n$ vérifiant l'égalité³ suivante :

$$\forall x \in \mathbb{F}_2^n, \quad \sum_{u,v} c_{u,v} x^u F(x)^v = 0.$$

Un tel vecteur $(c_{u,v})_{u,v \in \mathbb{F}_2^n}$ est donc un élément du noyau de la matrice M définie par

$$M = (x^u F(x)^v)_{(u,v) \in \mathbb{F}_2^n \times \mathbb{F}_2^n, x \in \mathbb{F}_2^n},$$

formée de $\sum_{i=0}^d \binom{2n}{i}$ colonnes (une pour chaque monôme $x^u y^v$) et de 2^n lignes (une pour chaque x). Puisque M admet 2^n lignes, son rang vérifie $\text{rk}(M) \leq 2^n$. On en déduit donc par théorème du rang que la dimension du noyau vérifie $\dim(\ker(M)) \geq \sum_{i=0}^d \binom{2n}{i} - 2^n$. $\square$

Ce type de relations entre entrées et sorties donnent lieu à de nouvelles modélisations. On peut par exemple identifier des relations entre entrées et sorties d'une fonction de tour F , et introduire des variables pour les valeurs d'états intermédiaires. Il faut alors estimer précisément les gains ou surcoûts liés à la diminution ou l'augmentation du nombre de variables et du degré des équations.

Concernant les méthodes de résolution autres que la linéarisation, les techniques utilisant les *bases de Gröbner* généralisent l'approche à base d'algèbre linéaire. Elles permettent parfois de mieux prendre en compte les structures algébriques sous-jacentes et sont utilisables dans des contextes où les données sont limitées. Les stratégies de *guess-and-solve* prennent le parti de deviner la valeur de certaines variables et de tenter de résoudre le système simplifié obtenu, avant de répéter l'opération si la supposition était fausse. Là encore, l'estimation du gain de résolution nécessite une analyse dédiée fine.

Exemple 3.3 (Le cas de l'AES). D'après la Proposition 2.9, la boîte-S de l'AES est de degré $\text{wt}(254) = 7$. Puisque le reste de la fonction de tour est constitué de fonctions linéaires (donc de degré 1), on en déduit que la fonction de tour est elle aussi de degré 7. On peut en fait vérifier que toutes les composantes de S sont de degré 7. Avec les notations de l'Exemple 2.8, $S = A \circ \Phi^{-1} \circ P \circ \Phi$ où $P(X) = X^{254}$. On observe en particulier que $x^2 P(x) = x^{256} = x$ pour tout $x \in \mathbb{F}_{256}$. Or $X \mapsto X^2$ est une application linéaire dans $\mathbb{F}_{256}$, on en déduit donc que l'équation $x^2 P(x) = x$ dans $\mathbb{F}_{256}$ fournit 8 relations quadratiques dans $\mathbb{F}_2$ entre entrée et sortie de P : une par coordonnée. La composition avec les bijections affines A, Φ^{-1} et Φ préserve l'existence de ces relations quadratiques et fournit 8 relations

3. Dans l'égalité en question $F(x)^v$ signifie $\prod_i F_i(x)^{v_i}$. Si les sorties de F sont notées y_i , cela correspond au monôme y^v en les variables de sortie.

quadratiques pour S .

Exercice 17. Soit $F: \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ et $A, B: \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ deux bijections affines. Montrer que $\deg(B \circ F \circ A) = \deg(F)$.

Exercice 18 (Composante de la boîte-S d'AES). On rappelle que la trace (absolue) de $\mathbb{F}_{2^n}$ est la fonction $\text{Tr}: \mathbb{F}_{2^n} \rightarrow \mathbb{F}_2$ définie par $\text{Tr}: X \mapsto \sum_{i=0}^{n-1} X^{2^i}$.

1. Montrer que la trace est une forme linéaire.
2. Montrer que toute forme linéaire de $\mathbb{F}_{2^n}$ vers $\mathbb{F}_2$ s'écrit de manière unique comme $X \mapsto \text{Tr}(\lambda X)$ pour un certain $\lambda \in \mathbb{F}_{2^n}$.
3. En déduire que les composantes non-nulles de $P: X \mapsto X^{2^{54}}$ sont toutes de degré 7.

Exercice 19 (À programmer). En s'inspirant de la preuve de la Proposition 3.2, construire la matrice dont le noyau contient toutes les relations quadratiques pour la boîte S de l'AES. Vérifier que sa dimension est 39.

3.2 Attaques statistiques

Les attaques algébriques décrites ci-dessus exploitent des relations entre données connues et inconnues qui sont *toujours vraies*. Une généralisation naturelle est de tenter de déterminer puis d'exploiter des relations statistiques qui ne sont vraies *qu'avec une certaine probabilité* ou de manière équivalente, que pour certains couples d'entrée-sortie.

Ces propriétés fournissent la majorité du temps des *distingueurs* contre le chiffrement par blocs. Plus exactement, en mettant en avant une propriété commune à la (quasi) totalité des fonctions $(E_k)_{k \in \mathbb{F}_2^k}$ et valable pour une large portion de couples entrées-sorties, une propriété statistique permet de distinguer la famille de bijections d'une famille de bijections tirée aléatoirement. Une telle propriété est indésirable puisqu'elle indique que le cadre théorique des PRP dans lequel les preuves de sécurité des modes d'opérations dans lesquels les chiffrements par blocs sont utilisés n'est effectivement pas vérifié. Si cela ne compromet⁴ pas directement la sécurité du chiffrement, dans le cas des chiffrements itérés au moins, un distingueur sur une version réduite à moins de tours peut très souvent se transformer en une attaque de recouvrement de clé.

Attaque sur le dernier tour. À partir d'un distingueur sur $R - 1$ tours d'un chiffrement à clé alternée, il est possible de construire une attaque de recouvrement de clé visant (en premier lieu) la dernière clé de tour. Notons $G_k = F_{k^{(r-1)}}^{(r-1)} \circ \dots \circ F_{k^{(1)}}^{(1)}$ les $R - 1$ premiers tours, de sorte que $E_k = F_{k^{(r)}}^{(r)} \circ G_k$. Étant données des couples clairs-chiffrés (x_i, z_i) pour E_k , un adversaire peut calculer, pour chaque valeur possible pour la dernière clé de tour $\hat{k}$, des couples clairs-chiffrés (x_i, y_i) pour G_k en posant $y_i = (F_{\hat{k}}^{(r)})^{-1}(z_i) = (F_{\hat{k}}^{(r)})^{-1} \circ E_k(x_i)$. L'adversaire peut alors mettre en oeuvre l'attaque distinguante. Si $\hat{k} = k^{(r)}$, alors les y_i sont de vrais chiffrés pour G_k : $y_i = (F_{k^{(r)}}^{(r)})^{-1}(z_i) = (F_{k^{(r)}}^{(r)})^{-1} \circ E_k(x_i) = G_k(x_i)$ et G_k

4. Si la sécurité n'est pas nécessairement compromise directement, la notion de sécurité IND-CPA est elle invalidée.

sera donc distinguée d'une bijection aléatoire. À l'inverse, si $\hat{k} \neq k^{(r)}$, sous l'hypothèse que $G_k \circ (F_{\hat{k}}^{(r)})^{-1}$ se comporte aléatoirement, alors le distingueur désignera $G_k \circ (F_{\hat{k}}^{(r)})^{-1}$ comme aléatoire. La clé $k^{(r)}$ est ainsi retrouvée. On pourra alors mener le même raisonnement sur les autres clés de tour et/ou mener une recherche exhaustive sur la partie restante de la clé maître.

Pour qu'un tel procédé fonctionne, il faut en réalité que le calcul des y_i (ou seulement des parties nécessaires au distingueur) ne dépende pas de la totalité des bits de la dernière clé de tour. Autrement dit, on doit pouvoir inverser partiellement le dernier tour à partir *seulement d'une partie de la clé*. Sinon la recherche exhaustive de $k^{(r)}$ coûte de l'ordre de 2^n , ce qui rend cette attaque caduque. Plus d'informations à ce sujet peuvent être trouvées dans le polycopié d'A. Canteaut [Can23, Section 2.4] et/ou dans le tutoriel suivant [Hey02].

Dans la suite, nous présentons deux des plus grandes familles d'attaques statistiques.

3.3 Cryptanalyse différentielle

3.3.1 Principe

La cryptanalyse différentielle a été introduite par Biham & Shamir [BS91a, BS91b] pour attaquer le standard historique DES. Cette technique exploite l'existence d'un distingueur (dit différentiel) d'une fonction $F: \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$, c'est-à-dire l'existence d'un couple de différences $(\Delta^{\text{in}}, \Delta^{\text{out}})$, appelé une *différentielle*, pour lequel l'équation ci-dessous admet beaucoup de solutions $x \in \mathbb{F}_2^n$:

$$F(x) + F(x + \Delta^{\text{in}}) = \Delta^{\text{out}}.$$

Pour une fonction aléatoire, il est attendu que le nombre de solutions d'une telle équation soit en moyenne de 1. L'existence d'un distingueur différentiel est donc révélatrice d'un comportement qui s'éloigne de l'aléatoire.



Attention 3.4. Un tel distingueur nécessite généralement un cadre d'attaque à clairs choisis. En effet, on souhaite obtenir deux clairs (et les chiffrés associés) qui diffèrent d'une différence fixée choisie par l'adversaire, ce qui n'arrive (presque) jamais aléatoirement.

Exercice 20. Montrer qu'une fonction $F: \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ affine est, du point de vue différentiel, le pire possible.

Pour distinguer différentiellement un chiffrement par blocs d'une famille tirée aléatoirement, il est nécessaire de trouver une paire $(\Delta^{\text{in}}, \Delta^{\text{out}})$ qui fonctionne pour toutes les clés $k \in \mathbb{F}_2^{\kappa}$ (ou presque). Le distingueur obtenu consiste alors à faire la requête de nombreuses paires de couples (x, y) et $(x + \Delta^{\text{in}}, y')$, et à compter le nombre de fois où $y + y' = \Delta^{\text{out}}$. Si ce nombre est grand, on estime qu'il s'agit du chiffrement par blocs, sinon qu'il s'agit du cas aléatoire. Le nombre de paires nécessaire est alors inversement proportionnel à

la probabilité qu'une paire $(x, x + \Delta^{\text{in}})$ soit solution de l'équation différentielle : si cette probabilité vaut $p \gg 2^{-1}$, le distingueur requiert de l'ordre de $1/p$ paires $(x, x + \Delta^{\text{in}})$.

Pour le cryptanalyste, l'objectif est donc de déterminer le nombre maximal de solutions des équations différentielles d'une fonction.

Définition 3.5 (Uniformité différentielle [Nyb94]). Soit $F: \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$. L'uniformité différentielle de F est la quantité δ_F définie par :

$$\delta_F \stackrel{\text{def}}{=} \max_{\Delta^{\text{in}} \neq \Delta^{\text{out}}} \delta_F(\Delta^{\text{in}}, \Delta^{\text{out}}) \quad \text{où} \quad \delta_F(\Delta^{\text{in}}, \Delta^{\text{out}}) \stackrel{\text{def}}{=} \left| \left\{ x \in \mathbb{F}_2^n \mid F(x) + F(x + \Delta^{\text{in}}) = \Delta^{\text{out}} \right\} \right|.$$

Pour le concepteur, l'uniformité différentielle est donc un indicateur qui doit être minimisé. Cela passe le plus souvent par le calcul de la *table de distribution différentielle* (DDT en anglais), c'est-à-dire par la construction de la matrice carrée de taille 2^n définie par $(\delta_F(\Delta^{\text{in}}, \Delta^{\text{out}}))_{\Delta^{\text{in}}, \Delta^{\text{out}} \in \mathbb{F}_2^n}$.

3.3.2 Différentielles et chemins différentiels

Si pour de petites valeurs de n , l'uniformité différentielle peut facilement être calculée exhaustivement, pour un chiffrement avec un état de $n \geq 64$ bits, la tâche est compliquée : il faut a priori parcourir toutes les valeurs de $\Delta^{\text{in}}, \Delta^{\text{out}}$ et x , soit une complexité naïve en temps 2^{3n} .

Exercice 21. La complexité naïve du calcul de l'uniformité différentielle peut en fait être ramenée à 2^{2n} en temps, quitte à utiliser de la mémoire. Donner une description d'un tel algorithme.

Le cryptanalyste n'a (actuellement ?) pas d'autre choix que d'estimer cette uniformité différentielle. Dans le cas des chiffrements itérés, l'idée la plus naturelle est de se reposer sur la décomposition en tours : plutôt que de chercher directement une différentielle $\Delta^{\text{in}} \xrightarrow{F} \Delta^{\text{out}}$, on peut tenter de construire un *chemin différentiel*, c'est-à-dire une suite de différences $\Delta = (\Delta^{(0)}, \dots, \Delta^{(R)})$, et chercher la probabilité $p(\Delta)$ que les différences intermédiaires lors du chiffrement de deux messages $X, X + \Delta^{(0)}$ prennent successivement les valeurs $\Delta^{(1)}, \dots, \Delta^{(R)}$. Autrement dit, $p(\Delta)$ est définie comme :

$$p(\Delta) \stackrel{\text{def}}{=} \mathbb{P}_{X_0} \left[X_1 + X'_1 = \Delta^{(1)}, \dots, X_R + X'_R = \Delta^{(R)} \mid X_0 + X'_0 = \Delta^{(0)} \right], \quad (2)$$

où l'on a noté X_i le i -ème état intermédiaire : $X_{i+1} = F^i \circ \dots \circ F^0(X_0)$, ou encore $X_{i+1} = F^{(i)}(X_i)$.

La probabilité d'un chemin différentiel Δ telle que $\Delta^{(0)} = \Delta^{\text{in}}$ et $\Delta^{(R)} = \Delta^{\text{out}}$ permet d'estimer la probabilité $\delta_F(\Delta^{\text{in}}, \Delta^{\text{out}})$ puisque la relation suivante est toujours vérifiée :

$$\frac{\delta_F(\Delta^{\text{in}}, \Delta^{\text{out}})}{2^n} = \sum_{\Delta \text{ t.q. } \Delta^{(0)} = \Delta^{\text{in}}, \Delta^{(R)} = \Delta^{\text{out}}} p(\Delta) \quad (3)$$

On obtient donc un minorant $p(\Delta) \leq \frac{\delta_F(\Delta^{\text{in}}, \Delta^{\text{out}})}{2^n}$ en ne considérant qu'un chemin (ou qu'une partie des chemins).

Le problème se ramène donc au calcul de la probabilité $p(\Delta)$. Reprenant l'Eq. (2), on observe que *si la différence $\Delta^{(i+1)}$ au tour $i+1$ dépend exclusivement de la différence $\Delta^{(i)}$* , alors les événements successifs sont indépendants et la probabilité $p(\Delta)$ peut alors être calculée comme :

$$\begin{aligned} p(\Delta) &= \prod_{i=0}^{R-1} \mathbb{P}_{X_i}[X_{i+1} + X'_{i+1} = \Delta^{(i+1)} \mid X_i + X'_i = \Delta^{(i)}] \\ &= \prod_{i=0}^{R-1} \mathbb{P}_{X_i}[F^{(i)}(X_i) + F^{(i)}(X_i + \Delta^{(i)}) = \Delta^{(i+1)}] \\ &= \prod_{i=0}^{R-1} \frac{\delta_{F^{(i)}}(\Delta^{(i)}, \Delta^{(i+1)})}{2^n}. \end{aligned}$$

Cette hypothèse, connue sous le nom d'*hypothèse de Markov* [LMM91], est par exemple vérifiée si les clés de tour des fonctions $F^{(i)}$ sont indépendantes et uniformément distribuées. Ce n'est en pratique jamais le cas puisque les clés de tour sont toutes dérivées d'une (petite) clé maître.⁵ En général l'approximation $p(\Delta) \simeq \prod_{i=0}^{R-1} \frac{\delta_{F^{(i)}}(\Delta^{(i)}, \Delta^{(i+1)})}{2^n}$ reste néanmoins une bonne approximation (qui se doit d'être vérifiée autant que possible).

3.3.3 Analyse d'une fonction de tour

D'après l'approximation $p(\Delta) \simeq \prod_{i=0}^{R-1} \frac{\delta_{F^{(i)}}(\Delta^{(i)}, \Delta^{(i+1)})}{2^n}$ vraie sous hypothèse, le calcul de la probabilité $p(\Delta)$ d'un chemin $\Delta = (\Delta^{(0)}, \dots, \Delta^{(R)})$ se réduit donc à l'étude des différentielles des fonctions de tour $F^{(i)}$. Celles-ci étant construites comme composition d'une fonction linéaire et d'une couche de boîtes-S, nous pouvons commencer par étudier ces deux couches indépendamment.

Proposition 3.6 (Différentielle d'une couche de boîte-S). *Soit n, m, s tels que $n = ms$. Soit $S: \mathbb{F}_2^m \rightarrow \mathbb{F}_2^m$ une boîte-S et $\mathcal{S}: \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ la couche de boîtes-S associée, c'est-à-dire définie par :*

$$\forall x = (x_1, \dots, x_s) \in (\mathbb{F}_2^m)^s, \quad \mathcal{S}(x) \stackrel{\text{def}}{=} (S(x_1), \dots, S(x_s)).$$

Soit $\Delta^{\text{in}} = (\Delta_1^{\text{in}}, \dots, \Delta_s^{\text{in}})$ et $\Delta^{\text{out}} = (\Delta_1^{\text{out}}, \dots, \Delta_s^{\text{out}}) \in \mathbb{F}_2^{ms}$. Alors :

$$\delta_{\mathcal{S}}(\Delta^{\text{in}}, \Delta^{\text{out}}) = \prod_{i=1}^s \delta_S(\Delta_i^{\text{in}}, \Delta_i^{\text{out}}).$$

En particulier, $\delta_{\mathcal{S}} = 2^{m(s-1)} \delta_S$.

Exercice 22. Prouver (en une ligne) l'énoncé ci-dessus.

5. Pour que les $R+1$ clés de tour soient indépendantes et uniformément distribuées, il faut que la clé maître soit au moins de taille $(R+1)n$.

Proposition 3.7 (Différentielles et application linéaire). Soit $A: \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ une application affine.

(i) Soit $\Delta^{\text{in}}, \Delta^{\text{out}} \in \mathbb{F}_2^n$ Alors :

$$\delta_A(\Delta^{\text{in}}, \Delta^{\text{out}}) = \begin{cases} 2^n & \text{si } A(\Delta^{\text{in}}) + A(0) = \Delta^{\text{out}}, \\ 0 & \text{sinon.} \end{cases}$$

En particulier $\delta_A = 2^n$.

(ii) Soit $F: \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$. Soit $A, B: \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ des applications affines bijectives dont les parties linéaires respectives sont L_A et L_B . Soit G la fonction définie par $G = B \circ F \circ A$. Alors :

$$\forall \Delta^{\text{in}}, \Delta^{\text{out}} \in \mathbb{F}_2^n, \quad \delta_G(\Delta^{\text{in}}, \Delta^{\text{out}}) = \delta_F(L_A(\Delta^{\text{in}}), L_B^{-1}(\Delta^{\text{out}})).$$

En particulier l'uniformité différentielle est inchangée par composition par des applications affines bijectives : $\delta_{B \circ F \circ A} = \delta_F$. On dit que l'uniformité différentielle est invariante par équivalence affine.

Exercice 23. Prouver chacun des énoncés ci-dessus.

Exemple 3.8 (Chemins différentiels sur deux tours de l'AES). On souhaite calculer la probabilité des chemins différentiels de deux tours de l'AES représentés à gauche en Fig. 7. La deuxième addition de clé est omise car quelle que soit F , la différence entre les sorties de deux messages x, x' à travers $\text{Add}_k \circ F$ ou F est la même. En effet $(F(x) + k) + (F(x') + k) = F(x) + F(x')$. De plus, on observe que la couche de boîtes-S et l'opération **ShiftRows** commutent entre elles et on obtient donc la représentation équivalente à droite en Fig. 7. On appellera couche de *super-boîtes* [DR06] la composition $\mathcal{S} \circ \text{Add}_k \circ M \circ \mathcal{S}$. Ce nom est dû au fait que cette fonction peut être vue comme la concaténation de 4 boîtes-S 16×16 (dépendantes d'une clé) : elle agit indépendamment sur les colonnes de l'état.

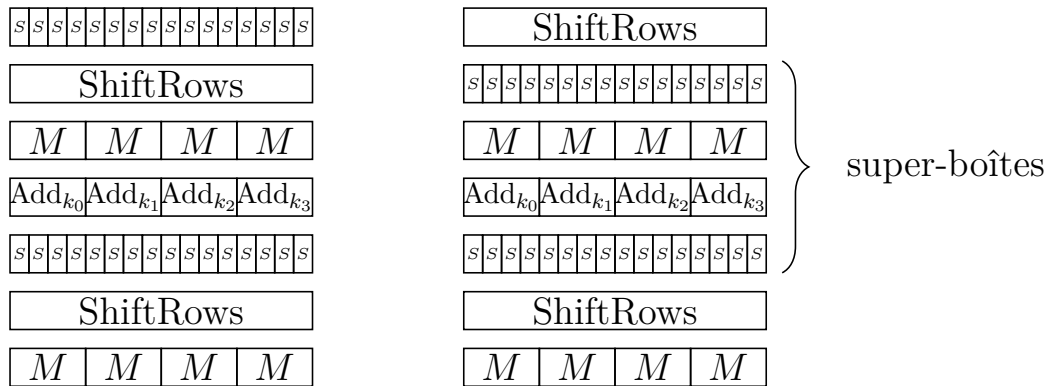


FIGURE 7 : Représentation alternative de deux tours de l'AES.

Enfin, on observe grâce à l'énoncé (ii) de la Proposition 3.7, que la première couche

ShiftRows et les deuxièmes couches **ShiftRows** et **MixColumns** peuvent complètement être omises dans notre étude puisque la composition (en entrée et/ou en sortie) par des applications linéaires bijectives réordonne la DDT d'une fonction mais ne change pas les valeurs présentes dans celle-ci.

Notons $\alpha, \beta \in \mathbb{F}_2^{128}$ les différences en entrée et sortie de la première couche de boîtes-S et $\gamma, \delta \in \mathbb{F}_2^{128}$ en entrée et sortie de la deuxième couche de boîtes-S. D'après le premier point de la Proposition 3.7, une transition différentielle est déterministe à travers une couche linéaire et la différence après la première couche est donc nécessairement égale à $\mathbf{MC}(\beta)$ (où l'on abrège **MixColumns** en **MC**). Puisque la différence est inchangée par l'ajout d'une clé de tour, les chemins valides (c'est-à-dire avec probabilité non-nulle) $(\alpha, \beta, \gamma, \delta)$ vérifient nécessairement $\gamma = \mathbf{MC}(\beta)$.

Enfin, en supposant que la clé de tour est uniformément distribuée, la couche de super-boîtes vérifie l'hypothèse de Markov et la probabilité $p(\alpha, \beta, \mathbf{MC}(\beta), \delta)$ d'un chemin sur 2 tours peut être calculée comme :

$$p(\alpha, \beta, \mathbf{MC}(\beta), \delta) = \mathbb{P}[\mathcal{S}(x + \alpha) + \mathcal{S}(x) = \beta] \times \mathbb{P}[\mathcal{S}(x + \mathbf{MC}(\beta)) + \mathcal{S}(x) = \delta].$$

En utilisant la Proposition 3.6, on en déduit donc que :

$$p(\alpha, \beta, \mathbf{MC}(\beta), \delta) = \prod_{i=1}^{16} \mathbb{P}[S(x_i + \alpha_i) + S(x_i) = \beta_i] \times \prod_{i=1}^{16} \mathbb{P}[S(x_i + \mathbf{MC}(\beta)_i) + S(x_i) = \delta_i].$$

On observe donc que $p(\alpha, \beta, \mathbf{MC}(\beta), \delta) \neq 0$ si et seulement si chacune des probabilités $\mathbb{P}[S(x_i + \alpha_i) + S(x_i) = \beta_i]$, $\mathbb{P}[S(x_i + \mathbf{MC}(\beta)_i) + S(x_i) = \delta_i]$ est non nulle. Cela impose que $[\alpha_i = \beta_i = 0]$ ou $[\alpha_i \neq 0 \text{ et } \beta_i \neq 0]$ pour tout i , ou autrement dit que α et β ont même support. En effet S étant bijective, une différence est non nulle en entrée (deux antécédents distincts) si et seulement si la différence en sortie est non nulle en sortie (deux images distinctes). De même $\mathbf{MC}(\beta)$ et δ doivent avoir même support. Si α, β, δ sont choisis de sorte à vérifier ces deux conditions, alors :

$$\begin{aligned} p &= \prod_{i \in \text{Supp}(\beta)} \mathbb{P}[S(x_i + \alpha_i) + S(x_i) = \beta_i] \prod_{i \in \text{Supp}(\mathbf{MC}(\beta))} \mathbb{P}[S(x_i + \mathbf{MC}(\beta)_i) + S(x_i) = \delta_i] \\ &\leq \left(\frac{\delta_S}{2^m} \right)^{\text{wt}(\beta)} \left(\frac{\delta_S}{2^m} \right)^{\text{wt}(\mathbf{MC}(\beta))} = \left(\frac{\delta_S}{2^m} \right)^{\text{wt}(\beta) + \text{wt}(\mathbf{MC}(\beta))}. \end{aligned}$$

3.3.4 Choix des composants du point de vue différentiel

L'inégalité obtenue $p \leq \left(\frac{\delta_S}{2^m} \right)^{\text{wt}(\beta) + \text{wt}(\mathbf{MC}(\beta))}$ est précieuse pour le concepteur. En effet, puisque minimiser la probabilité des chemins différentiels permet de minimiser la probabilité des différentielles (cf. Eq. (3)), deux options sont envisageables et utilisées en pratique. D'une part l'uniformité différentielle de la boîte-S δ_S peut être choisie aussi petite que possible. D'autre part, il est souhaitable que $\text{wt}(\beta) + \text{wt}(\mathbf{MC}(\beta))$ soit aussi grand que possible pour toute valeur de $\beta \neq 0$. Cela donne lieu à l'indicateur suivant qui guide le choix des couches linéaires.

Définition 3.9 (Branch number différentiel [DR01]). Soit $L: (\mathbb{F}_2^m)^s \rightarrow (\mathbb{F}_2^m)^s$ une application $\mathbb{F}_2$ -linéaire. Le branch number différentiel (relatif à $\mathbb{F}_2^m$) de L est la quantité $\mathcal{B}(L)$ définie par :

$$\mathcal{B}(L) = \min_{x \neq 0} (\text{wt}(x) + \text{wt}(Lx)),$$

où la fonction de poids compte le nombre de coordonnées (de m bits) non nulles.

Contrairement à l'uniformité différentielle, le branch number est un indicateur de diffusion qui doit être maximisé. C'est précisément la stratégie adoptée par les concepteurs de l'AES :

- (i) La boîte-S de l'AES a une uniformité différentielle de 4 (ce qui est petit comparé au maximum 2^8)
- (ii) La matrice M de MixColumns (qui est appliquée à chaque colonne) a un branch number (relatif aux octets) maximal de 5. On parle de matrice MDS [DR02] en référence aux codes linéaires à distance séparable maximale auxquels ces matrices sont étroitement liées. L'intéressé.e pourra consulter les références suivantes sur les codes linéaires, en français [Cou20, Dem97] ou en anglais [MS77].

Les bonnes propriétés locales de la boîte-S couplées aux bonnes propriétés de diffusion de la couche linéaire font de l'AES un chiffrement par blocs toujours considéré comme très résistant (à la cryptanalyse différentielle comme à tout autre technique), et ce, après plus de 25 ans d'analyse par la communauté internationale.

3.3.5 Quelques mots sur la résistance optimale aux attaques différentielles

Comme vu précédemment, l'uniformité différentielle est la quantité à minimiser pour résister aux attaques différentielles. Il est donc naturel de se demander quelles sont les fonctions résistantes de manière optimale à cette technique.

Premièrement, on observe que les solutions x de l'équation $F(x) + F(x + \Delta^{\text{in}}) = \Delta^{\text{out}}$ où $\Delta^{\text{in}} \neq 0$ vont par paires : en effet x est solution si et seulement si $x + \Delta^{\text{in}}$ est aussi solution (et $x + \Delta^{\text{in}} \neq x$ puisque $\Delta^{\text{in}} \neq 0$). Le nombre de solutions est donc toujours pair.

D'autre part, puisque F est une fonction, étant donné $\Delta^{\text{in}} \neq 0$, il existe toujours une valeur Δ^{out} telle que $\delta_F(\Delta^{\text{in}}, \Delta^{\text{out}}) > 0$, par exemple $\Delta^{\text{out}} = F(0) + F(\Delta^{\text{in}})$.

On en déduit donc que l'uniformité différentielle δ_F vérifie $2 \leq \delta_F$ et on définit donc naturellement la classe de fonctions suivante.

Définition 3.10 (Fonction APN [NK93]). Une fonction $F: \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ est dite presque-parfaitement non-linéaire (almost-perfect nonlinear ou APN en anglais) si elle vérifie $\delta_F = 2$ ou de manière équivalente si pour toutes différences $(\Delta^{\text{in}}, \Delta^{\text{out}})$, $\Delta^{\text{in}} \neq 0$, l'équation différentielle $F(x) + F(x + \Delta^{\text{in}}) = \Delta^{\text{out}}$ admet soit 0, soit 2 solutions.

Exemple 3.11 (La fonction cube). Soit n un entier et $F: \mathbb{F}_{2^n} \rightarrow \mathbb{F}_{2^n}$ définie (sous forme univariée) par $F: X \mapsto X^3$. Soit $\Delta^{\text{in}} \neq 0$. On observe que $F(X) + F(X + \Delta^{\text{in}}) =$

$X^3 + (X + \Delta^{\text{in}})^3 = X^3 + X^3 + \Delta^{\text{in}} X^2 + \Delta^{\text{in}^2} X + \Delta^{\text{in}^3} = \Delta^{\text{in}} X^2 + \Delta^{\text{in}^2} X + \Delta^{\text{in}^3}$. Il s'agit donc d'une équation de degré 2 qui admet donc au plus 2 solutions dans $\mathbb{F}_{2^n}$. La fonction F est donc APN et ce quelle que soit la dimension n .

En pratique, les constructions de fonctions APN connues sont peu nombreuses et les propriétés de ces fonctions restent très énigmatiques. À titre d'exemple, la quasi-totalité des fonctions APN connues sont *équivalentes*⁶ (pour l'équivalence CCZ [CCZ98]) à des fonctions de degré algébrique 2 (comme le cube ci-dessus). Les seules exceptions sont des fonctions APN monomiales $X \mapsto X^d$ pour quelques valeurs d appelées exposants de Gold, Niho, Kasami, Welch ou Dobbertin (et la fonction inverse lorsque n est impaire) [Nyb94, Kas71, Dob99b, Dob99a, Dob01]. Nous ne connaissons à ce jour qu'un seul exemple d'une fonction APN qui ne tombe pas dans l'une de ces deux catégories, à savoir une seule fonction qui n'est ni CCZ-équivalente à une fonction quadratique, ni CCZ-équivalente à une fonction monomiale [BL08, EP09].

Peut-être plus surprenant encore, en dimension n paire, nous ne connaissons qu'une seule fonction APN bijective [BDMW10]. Ce sont pourtant deux propriétés que l'on souhaiterait avoir pour construire une boîte-S optimale. La question est restée ouverte pendant près de 20 ans avant qu'un exemple en dimension 6 soit mis au jour. Cette bijection est appelée *permutation de Dillon* en hommage au chercheur de la NSA qui l'a découverte en 2009. Depuis, la question de l'existence d'autres exemples en dimension 6, comme en dimensions paires supérieures demeure ouverte.

Lorsque n est impair, la situation est très différente puisque $X \mapsto X^3$ fournit un exemple de fonction APN bijective pour toute dimension impaire.

3.4 Cryptanalyse linéaire



Attention 3.12 (En construction). Cette section est encore en construction. Nous renvoyons pour le moment aux articles fondateurs [TG92, Mat94a, Mat94b] et/ou au polycopié d'A. Canteaut [Can23, Section 2.5]. La thèse d'A. Florez-Gutierrez [FG22] (ancien étudiant du Master AAC) porte également en grande partie sur la cryptanalyse linéaire.

Références

- [AES01] Advanced Encryption Standard (AES). National Institute of Standards and Technology, NIST FIPS PUB 197, U.S. Department of Commerce, November 2001.
- [BDMW10] K. A. Browning, J.F. Dillon, M. T. McQuistan, and A. J. Wolfe. An APN permutation in dimension six. In *Post-proceedings of the 9-th International*

6. L'équivalence CCZ est une notion d'équivalence plus générale que l'équivalence affine mentionnée plus haut. C'est, à ce jour, la relation d'équivalence la plus générale qui préserve l'uniformité différentielle et donc la propriété APN.

- Conference on Finite Fields Their Appl.*, volume 518, pages 33–42. American Mathematical Society, 2010.
- [BL08] Marcus Brinkmann and Gregor Leander. On the classification of APN functions up to dimension five. *Designs, Codes and Cryptography*, 49(1):273–288, Dec 2008.
- [BS91a] Eli Biham and Adi Shamir. Differential cryptanalysis of DES-like cryptosystems. In Alfred J. Menezes and Scott A. Vanstone, editors, *CRYPTO’90*, volume 537 of *LNCS*, pages 2–21, Santa Barbara, CA, USA, August 11–15, 1991. Springer Berlin Heidelberg, Germany.
- [BS91b] Eli Biham and Adi Shamir. Differential cryptanalysis of DES-like cryptosystems. *Journal of Cryptology*, 4(1):3–72, January 1991.
- [BSS⁺13] Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks, and Louis Wingers. The SIMON and SPECK families of lightweight block ciphers. Cryptology ePrint Archive, Report 2013/404, 2013.
- [Can16] Anne Canteaut. Lecture notes on cryptographic boolean functions, March 2016. <https://www.rocq.inria.fr/secret/Anne.Canteaut/poly.pdf>.
- [Can23] Anne Canteaut. Lecture notes on error-correcting codes and their applications to symmetric cryptography, December 2023. <https://www.paris.inria.fr/secret/Anne.Canteaut/MPRI/poly-sans-stream.pdf>.
- [CCZ98] Claude Carlet, Pascale Charpin, and Victor Zinoviev. Codes, bent functions and permutations suitable for des-like cryptosystems. *Designs, Codes and Cryptography*, 15:125–156, 1998.
- [Cou20] Alain Couvreur. Introduction to coding theory. Lecture notes, 2020. 108 pages.
- [Dem97] Michel Demazure. *Cours d’algèbre*, volume 1 of *Nouvelle bibliothèque mathématique*. Cassini, Paris, 1997.
- [Dob99a] Hans Dobbertin. Almost perfect nonlinear power functions on $\text{GF}(2^n)$: The niho case. *Inf. Comput.*, 151(1-2):57–72, 1999.
- [Dob99b] Hans Dobbertin. Almost perfect nonlinear power functions on $\text{GF}(2^n)$: The welch case. *IEEE Transactions on Information Theory*, 45(4):1271–1275, 1999.
- [Dob01] Hans Dobbertin. Almost perfect nonlinear power functions on $\text{GF}(2^n)$: A new case for n divisible by 5. In Dieter Jungnickel and Harald Niederreiter, editors, *Finite Fields and Applications*, pages 113–121, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.
- [DR01] Joan Daemen and Vincent Rijmen. The wide trail design strategy. In Bahram Honary, editor, *8th IMA International Conference on Cryptography and Coding*, volume 2260 of *LNCS*, pages 222–238, Cirencester, UK, December 17–19, 2001. Springer Berlin Heidelberg, Germany.

- [DR02] Joan Daemen and Vincent Rijmen. *The Design of Rijndael : AES – The Advanced Encryption Standard*. Information Security and Cryptography. Springer, 2002.
- [DR06] Joan Daemen and Vincent Rijmen. Understanding two-round differentials in AES. In Roberto De Prisco and Moti Yung, editors, *SCN 06*, volume 4116 of *LNCS*, pages 78–94, Maiori, Italy, September 6–8, 2006. Springer Berlin Heidelberg, Germany.
- [EP09] Yves Edel and Alexander Pott. A new almost perfect nonlinear function which is not quadratic. *Adv. Math. Commun.*, 3(1):59–81, 2009.
- [FG22] Antonio Florez Gutierrez. *Improved Techniques in the Cryptanalysis of Symmetric Primitives*. PhD thesis, Sorbonne Université, September 2022. https://inria.hal.science/tel-03878739v2/file/FLOREZ_GUTIERREZ_Antonio_2022.pdf.
- [Hey02] Howard M. Heys. A tutorial on linear and differential cryptanalysis. Technical report, Memorial University of Newfoundland, 2002.
- [Hua25] Huawei Technologies Co., Ltd. Global digitalization and intelligence index 2025. Technical report, Huawei Technologies Co., Ltd., 2025.
- [Int25] International Telecommunication Union. Guidelines for the implementation of low-carbon data centres to climate change mitigation and adaptation. ITU-T Recommendation L.1308, International Telecommunication Union, December 2025.
- [Jou09] Antoine Joux. *Algorithmic Cryptanalysis*. CRC Press Cryptography and Network Security Series. CRC Press, Boca Raton, FL, 2009.
- [Kas71] Tadao Kasami. The weight enumerators for several classes of subcodes of the 2nd order binary reed-muller codes. *Inf. Control.*, 18(4):369–394, 1971.
- [Ker83] Auguste Kerckhoffs. La cryptographie militaire. *Journal des Sciences Militaires*, 9:5–38,161–191, 1883.
- [Lai88] Charles-Ange Laisant. Sur la numération factorielle, application aux permutations. *Bulletin de la Société mathématique de France*, 16:176–183, 1888.
- [LMM91] Xuejia Lai, James L. Massey, and Sean Murphy. Markov ciphers and differential cryptanalysis. In Donald W. Davies, editor, *EUROCRYPT’91*, volume 547 of *LNCS*, pages 17–38, Brighton, UK, April 8–11, 1991. Springer Berlin Heidelberg, Germany.
- [Mat94a] Mitsuru Matsui. The first experimental cryptanalysis of the data encryption standard. In Yvo Desmedt, editor, *CRYPTO’94*, volume 839 of *LNCS*, pages 1–11, Santa Barbara, CA, USA, August 21–25, 1994. Springer Berlin Heidelberg, Germany.
- [Mat94b] Mitsuru Matsui. Linear cryptanalysis method for DES cipher. In Tor Helleseth, editor, *EUROCRYPT’93*, volume 765 of *LNCS*, pages 386–397, Lofthus, Norway, May 23–27, 1994. Springer Berlin Heidelberg, Germany.

- [MS77] Florence Jessie MacWilliams and Neil James Alexander Sloane. *The Theory of Error-Correcting Codes*, volume 16 of *North-Holland Mathematical Library*. North-Holland, Amsterdam, 1977.
- [Nat77] National Bureau of Standards. Data encryption standard. Technical Report FIPS PUB 46, U.S. Department of Commerce, January 1977.
- [NK93] Kaisa Nyberg and Lars R. Knudsen. Provable security against differential cryptanalysis (rump session). In Ernest F. Brickell, editor, *CRYPTO'92*, volume 740 of *LNCS*, pages 566–574, Santa Barbara, CA, USA, August 16–20, 1993. Springer Berlin Heidelberg, Germany.
- [Nyb94] Kaisa Nyberg. Differentially uniform mappings for cryptography. In Tor Helleseth, editor, *EUROCRYPT'93*, volume 765 of *LNCS*, pages 55–64, Lofthus, Norway, May 23–27, 1994. Springer Berlin Heidelberg, Germany.
- [Sha49] Claude E. Shannon. Communication theory of secrecy systems. *Bell Systems Technical Journal*, 28(4):656–715, 1949.
- [TG92] Anne Tardy-Corffdir and Henri Gilbert. A known plaintext attack of FEAL-4 and FEAL-6. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 172–181, Santa Barbara, CA, USA, August 11–15, 1992. Springer Berlin Heidelberg, Germany.
- [Wik26a] Wikipédia. Code de lehmer. https://fr.wikipedia.org/wiki/Code_de_Lehmer, 2026. Wikipédia, l'encyclopédie libre, consulté le 27 septembre 2026.
- [Wik26b] Wikipédia. Complexité de Kolmogorov. https://fr.wikipedia.org/wiki/Complexit%C3%A9_de_Kolmogorov, 2026. Wikipédia, l'encyclopédie libre.
- [YZS⁺15] Gangqiang Yang, Bo Zhu, Valentin Suder, Mark D. Aagaard, and Guang Gong. The simeck family of lightweight block ciphers. In Tim Güneysu and Helena Handschuh, editors, *CHES 2015*, volume 9293 of *LNCS*, pages 307–329, Saint-Malo, France, September 13–16, 2015. Springer Berlin Heidelberg, Germany.