République Algérienne Démocratique et Populaire
Université Abderrahmane MIRA de Béjaia
Faculté des Sciences Exactes

Département de Recherche Opérationnelle

Mémoire Présenté pour L'obtention du Diplôme de Master
en Mathématiques Appliquées

Spécialité : Science des Données et Aide à la Décision

Exploration de la cryptanalyse différentielle et son application sur un réseau SPN

Présenté par :
BAALI Mohammed Amokrane
ALLAOUA Amel

Sous la direction de : DJABRI Rabah

Défendu le 29/06/2025, devant le jury composé de :

M. SOUFIT Massinissa	M.C. classe/ B	Président de jury	UAMB - Bejaia
M^{me} CHIBANE Zebida	doctorante	Examinatrice	UAMB - Bejaia
M^{me} HAMZA Lamia	M.C. classe/ A	Examinatrice	UAMB - Bejaia
M. ASLI Larbi	M.C. classe/ A	Examinateur	UAMB - Bejaia

Année Universitaire 2024 – 2025

Table des matières

Table des figures

Liste des tableaux

Liste des symboles

$\mathbb{Z}$	Ensemble des entiers relatifs
$\mathbb{Z}^*$	Entiers relatifs non nuls ou inversibles
$\mathbb{N}$	Ensemble des entiers naturels
$\mathbb{N}^*$	Entiers naturels non nuls
$\in$	Appartenance à un ensemble
$\subseteq$	Inclusion
$\neq$	Différent de
a|b	a divise b
|a|	Valeur absolue de a
$\cap$	Intersection
max	Maximum
min	Minimum
$\Rightarrow$	Implication logique
$\Leftrightarrow$	Équivalence logique
$\equiv$	Congruence modulo n
$\exists$	Il existe
mod	Opération modulo
$[a]_n$	Classe de résidus de a modulo n
$\mathbb{Z}_n$	Ensemble des classes de résidus modulo n
$\oplus$	Opération OU exclusif (XOR)

Liste des abréviations

PGCD	Plus grand commun diviseur
PPCM	Plus petit commun multiple
D(a)	Ensemble des diviseurs de a
DES	Data Encryption Standard
ASCII	American Standard Code for Information Interchange
E	Expansion (fonction d'expansion de 32 à 48 bits)
IP	Permutation initiale
S_i	La i-ème boîte de substitution DES
P	Permutation
KS	Algorithme de générateur des sous-clés du DES (key schedule)
PC_1	Permutation de choix 1
PC_2	Permutation de choix 2
leftshift_j	Rotation circulaire à gauche de j positions
SPN	Réseau de substitution-permutation (substitution-permutation network)
XOR	OU exclusif (exclusive OR)

Introduction Générale

Le seul système vraiment sécurisé est celui qui est éteint, enfermé dans du béton, et scellé dans une pièce blindée. – Gene Spafford

Depuis les premières tentatives d'écriture sécurisée, la cryptographie s'est imposée comme une réponse à la nécessité de transmettre des informations tout en les protégeant de regards indésirables. Elle a accompagné l'évolution des sociétés, des champs de bataille aux réseaux numériques, en cherchant à garantir que les messages restent confidentiels, intègres et accessibles uniquement aux bonnes personnes.

En parallèle, la cryptanalyse, discipline sœur mais antagoniste, s'est développée comme l'art d'exploiter les faiblesses de ces protections. Tout au long de l'histoire, des personnes comme Vigenère ou Babbage ont montré les deux côtés de la cryptographie. Certains inventaient des méthodes complexes pour cacher les messages, tandis que d'autres cherchaient à les casser en observant les petits détails. Ce jeu entre création et attaque a aidé à construire les règles de sécurité qu'on utilise aujourd'hui.

Dans ce contexte, la cryptanalyse différentielle marque une avancée majeure. Elle ne s'appuie pas seulement sur l'observation extérieure des messages chiffrés, mais pénètre au cœur des algorithmes, en exploitant les effets de petites variations contrôlées dans les données d'entrée. Là où les méthodes classiques pouvaient échouer face à la complexité apparente d'un système, l'approche différentielle révèle des motifs internes exploitables, menaçant directement la robustesse des algorithmes symétriques.

Dans un monde numérique où la protection des données sensibles revêt une importance capitale, les algorithmes de chiffrement modernes reposent sur des structures complexes, telles que les réseaux SPN (Substitution-Permutation Network) et les réseaux de Feistel. Pourtant, même avec une architecture considérée comme robuste, ces systèmes présentent encore certaines vulnérabilités. La cryptanalyse, et plus particulièrement la cryptanalyse différentielle, constitue un outil puissant permettant d'analyser et de mettre en évidence ces failles potentielles.

L'objectif de ce mémoire est d'explorer la cryptanalyse différentielle et de l'appliquer sur un réseau SPN, en l'implémentent en Python. Pour cela, nous avons construit un reseaux SPN

simplifié comprenant une S-boîte de taille 44, une permutation sur 12 bits, et un schéma de chiffrement utilisant trois sous-clés. L'étude a débuté par la génération de la table des différences de la S-boîte, qui permet d'identifier les caractéristique d'entrée-sortie les plus probables.

Cette mémoire est organisée en quatre chapitres disposés de la manière suivante :

— La première pose les bases mathématiques nécessaires à la suite, notamment les notions d'arithmétique dans $\mathbb{Z}$, de congruence, d'inversibilité dans $\mathbb{Z}_n$, ainsi que les algorithmes classiques comme celui d'Euclide.

— La deuxième partie introduit les concepts fondamentaux de la cryptologie : elle couvre les méthodes de cryptographie classique et moderne, ainsi que les principales approches de cryptanalyse, des plus simples aux plus récentes.

— La troisième partie est consacrée à la cryptanalyse différentielle, en présentant son principe, son application aux chiffrements par blocs et son efficacité dans l'analyse des structures de type SPN.

— Enfin, la dernière partie propose une mise en œuvre pratique de cette méthode, avec un cas d'étude détaillé, en s'appuyant sur des outils de développement adaptés et des exemples concrets.

CHAPITRE 1

Arithmétique et théorie des nombres

L'arithmétique et la théorie des nombres sont des branches fondamentales des mathématiques qui étudient les propriétés des entiers et les relations entre eux. Ces domaines jouent un rôle essentiel dans de nombreux aspects des mathématiques pures et appliquées, notamment en cryptographie, en algorithmique et en analyse numérique.

1.1 Divisibilité dans $\mathbb{Z}$

On note $\mathbb{Z}$ l'ensemble des nombres entiers relatifs, i.e. :

$$\mathbb{Z} = \{\dots, -3, -2, -1, 0, 1, 2, 3, \dots\}.$$

On note $\mathbb{Z}^*$ l'ensemble des nombres entiers relatifs non nuls, i.e. :

$$\mathbb{Z}^* = \{\dots, -3, -2, -1, 1, 2, 3, \dots\}.$$

Et on note $\mathbb{N} \subseteq \mathbb{Z}$ l'ensemble des nombres entiers naturels, i.e. :

$$\mathbb{N} = \{0, 1, 2, 3, 4, \dots\}.$$

Et on note $\mathbb{N}^* \subseteq \mathbb{Z}$ l'ensemble des nombres entiers naturels non nuls, i.e. :

$$\mathbb{N}^* = \{1, 2, 3, 4, \dots\}$$

.

Définition 1.1. Soit $a, b \in \mathbb{Z}$ avec $a \neq 0$. On dit que a divise b et on écrit $a|b$ s'il existe $k \in \mathbb{Z}$ tel que $b = ka$. Dans ce cas, on dit aussi que b est divisible par a, ou que a est un diviseur de b, ou que b est un multiple de a .

Lemme 1. Soit $a, b \in \mathbb{Z}$ avec $a \neq 0$.

(a) Si $a|b$ alors soit $b = 0$ soit $|b| \geq |a|$.

(b) Si $a|b$ et $a|c$, alors $a|(bx+cy)$ pour tout $x, y \in \mathbb{Z}$.

(c) Si $a|b$ et $c|d$, alors $ac|bd$.

(d) Si $a|b$ et $b|c$, alors $a|c$.

(e) Si $c \neq 0$ et $ac|bc$, alors $a|b$.

1.2 Division euclidienne

La division euclidienne est un outil essentiel pour determiner si un nombre est divisible par un autre, trouver des multiples ou décomposer un nombre en terme de diviseurs.[2]

Théorème 1. Soit a, $b \in \mathbb{Z}$ avec $a \neq 0$. Il existe $q, r \in \mathbb{Z}$ uniques tels que

$$b = q \cdot a + r \quad \text{et} \quad 0 \leq r < |a|.$$

Les nombres q et r sont appelés le quotient et le reste de la division de b par a, respectivement.

1.3 Plus grand commun diviseur

Le PGCD (plus grand commun diviseur) est un concept mathématique essentiel utilisé pour trouver le plus grand diviseur commun de deux nombres entiers .

Définition 1.2. On dit que d est un diviseur commun de a et b si $d|a$ et $d|b$.
On note $D(a,b)$ l'ensemble des diviseurs communs de a et b, $D(a)$ les diviseurs de a. Alors :

$$D(a,b) = D(a) \cap D(b).$$

Définition 1.3. Soient $a, b \in \mathbb{Z}$ qui ne sont pas les deux égaux à 0. Leur plus grand diviseur commun qu'on note par PGCD(a,b) est défini par :

$$\text{PGCD}\ (a,b) = max\{d \in \mathbb{N} : d|a \quad \text{et} \quad d|b\}.$$

Exemple 1. Pour trouver le *PGCD* de 24 et 16, on peut le déterminer en passant par ces étapes Nous avons :

$$\begin{aligned} D(24) &= \{-24,-12,-8,-6,-4,-2,-1,1,2,4,6,8,12,24\}, \\ \text{Alors} \quad D(16) &= \{-16,-8,-4,-2,-1,1,2,4,8,16\}. \\ D(24) \cap D(16) &= \{-8,-4,-2,-1,1,2,4,8\}. \\ \text{donc} \quad \text{PGCD}(24,16) &= 8. \end{aligned}$$

Mais dans le cas où on travaille avec des grands nombres on peut utiliser un algorithme plus rapide, il s'agit de l'algorithme d'Euclide.

1.4 Plus petit commun multiple

Si a et b sont deux entiers relatifs non nuls, le plus petit nombre entier qui est un multiple commun de ces deux nombre c'est le PPCM de a et b. On le note PPCM(a,b). Il existe plusieurs méthodes pour déterminer le PPCM de deux entiers a et b, dans chapitre on va s'interesser a une de ces methodes .

Définition 1.4. Un commun multiple de deux nombres entiers est un nombre entier qui est divisible par ces deux nombres.
Soit a,b et $m \in \mathbb{N}^*$, et m est un multiple commun de a et $b \Leftrightarrow a|m$ et $b|m$.

Définition 1.5. Soit $a,b \in \mathbb{Z}^*$, leurs plus petit commun multiple qu'on note par PPCM(a,b) est défini par :

$$\text{PPCM}(a,b) = min\{m \in \mathbb{N}^* : a|m \quad et \quad b|m\}.$$

Théorème 2. D'abord, dans cette méthode, on dois calculer le PGCD de a et b, aprés on utilise la propriété mathématique suivent :

$$|a \times b| = \text{PGCD}(a,b) \cdot \text{PPCM}(a,b).$$

Exemple 2. Pour trouver le PPCM$(91,65)$ on a

$$\begin{aligned} \text{PGCD}(91,65) &= 13, \\ 91 \cdot 65 &= 13 \cdot \text{PPCM}(91,65), \\ \text{d'où} \quad \text{PPCM}(91,65) &= \frac{91 \cdot 65}{13} = \frac{5915}{13}, \\ \text{ce qui donne} \quad \text{PPCM}(91,65) &= 455. \end{aligned}$$

Donc, le PPCM$(91,65)$ est égal à 455.

1.5 Algorithme d'Euclide

L'algorithme d'Euclide est une méthode mathématique utilisée pour trouver le plus grand commun diviseur PGCD de deux nombres.
L'idée fondamentale de l'algorithme d'Euclide repose sur le fait que le PGCD de deux nombres reste le même, même si on les divise l'un par l'autre plusieurs fois, cela permet de simplifier le calcul du PGCD en effectuant une série de divisions successives [17].

Algorithm 1 Algorithme d'Euclide pour le PGCD

1: **Entrée :** deux entiers a et b tels que $a > b > 0$
2: **Sortie :** PGCD(a,b)
3: **while** $b \neq 0$ **do**
4: Calculer le reste $r \leftarrow a \bmod b$
5: Affecter $a \leftarrow b$
6: Affecter $b \leftarrow r$
7: **end while**
8: **Retourner** a

Théorème 3. Soit $a,b,r_i,q_i \in \mathbb{N}^*$ pour $i = 1,\ldots,N$ tels que

$$\begin{aligned}
a &= q_1 \cdot b + r_1 && \text{avec } 0 \leq r_1 < b,\\
b &= q_2 \cdot r_1 + r_2 && \text{avec } 0 \leq r_2 < r_1,\\
r_1 &= q_3 \cdot r_2 + r_3 && \text{avec } 0 \leq r_3 < r_2,\\
&\vdots\\
r_{n-1} &= q_{n+1} \cdot r_n + r_{n+1} && \text{avec } 0 \leq r_{n+1} < r_n,\\
r_n &= q_{n+2} \cdot r_{n+1} + 0 && \text{donc } r_{n+2} = 0.
\end{aligned}$$

Alors, PGCD$(a,b) = r_{n+1}$.

Exemple 3. Pour déterminer le PGCD de 91 et 65 en utilisant l'algorithme d'Euclide, on suit les étapes suivantes :

$$\begin{aligned}
91 &= 1 \cdot 65 + 26\\
65 &= 2 \cdot 26 + 13\\
26 &= 2 \cdot 13 + 0.
\end{aligned}$$

Donc, PGCD$(91,65) = 13$.

1.6 Algorithme d'Euclide étendu

En mathématiques, l'algorithme d'Euclide étendu est une version améliorée de l'algorithme d'Euclide. Quand on a deux nombres entiers a et b, cet algorithme calcule leur plus grand commun diviseur PGCD et trouve aussi deux nombres, appelés **coefficients de Bézout** (notés u et v). Ces coefficients vérifient une équation importante, appelée **identité de Bézout** :

$$a \cdot u + b \cdot v = \text{PGCD}(a,b).$$

Algorithm 2 Algorithme d'Euclide étendu

Entrée : deux entiers a et b tels que $a > b > 0$
Sortie : PGCD(a,b) et deux entiers u, v tels que $ua + vb = \text{PGCD}(a,b)$
Initialiser : $r_0 \leftarrow a$, $r_1 \leftarrow b$
Initialiser : $u_0 \leftarrow 1$, $v_0 \leftarrow 0$, $u_1 \leftarrow 0$, $v_1 \leftarrow 1$
$i \leftarrow 1$
while $r_i \neq 0$ **do**
    Calculer le quotient $q_i \leftarrow r_{i-1} \div r_i$
    Calculer le reste $r_{i+1} \leftarrow r_{i-1} - q_i \cdot r_i$
    Calculer $u_{i+1} \leftarrow u_{i-1} - q_i \cdot u_i$
    Calculer $v_{i+1} \leftarrow v_{i-1} - q_i \cdot v_i$
    Incrémenter $i \leftarrow i + 1$
end while
Retourner r_{i-1} comme PGCD, u_{i-1} et v_{i-1} comme coefficients de Bézout

Exemple 4. On appliquera l'algorithme d'Euclide étendu pour $a = r_0 = 243$ et $b = r_1 = 198$. Pour cela, on effectuera à gauche les étapes de l'algorithme d'Euclide (calcul des restes r_i et des quotients q_i). En même temps, on écrira à droite les calculs pour u_i et v_i, tels que $r_i = u_i \cdot a + v_i \cdot b$.

$$
\begin{aligned}
r_{i-2} &= q_{i-1} \cdot r_{i-1} + r_i & r_i &= [u_i] \cdot a + [v_i] \cdot b \\
243 &= 1 \cdot 198 + 45 & 45 &= [1] \cdot 243 + [-1] \cdot 198 \\
198 &= 4 \cdot 45 + 18 & 18 &= 198 - 4 \cdot 45 \\
& & &= 198 + (-4) \cdot (243 - 1 \cdot 198) \\
& & &= [-4] \cdot 243 + [5] \cdot 198 \\
45 &= 2 \cdot 18 + 9 & 9 &= 45 - 2 \cdot 18 \\
& & &= 45 - 2 \cdot (198 - 4 \cdot 45) \\
& & &= 9 \cdot 45 - 2 \cdot 198 \\
& & &= 9 \cdot (243 - 198) - 2 \cdot 198 \\
& & &= [9] \cdot 243 + [-11] \cdot 198 \\
18 &= 2 \cdot 9 + 0
\end{aligned}
$$

Nous avons alors $\text{PGCD}(243, 198) = 9 = [9] \cdot 243 + [-11] \cdot 198$.

1.7 Théorème de Bézout

Définition 1.6. Soient $a, b \in \mathbb{Z}$. Il existe deux entiers $u, v \in \mathbb{Z}$ tels que

$$a \cdot u + b \cdot v = \text{PGCD}(a,b).$$

L'existence des entiers u et v est donnée par l'algorithme d'Euclide étendu .

Théorème 4. Soient $a, b \in \mathbb{Z}$. Les entiers a, b sont premiers entre eux si et seulement s'il existe deux entiers u et v tels que :

$$a \cdot u + b \cdot v = 1.$$

Théorème 5. Théorème de Bézout Soient $a, b \in \mathbb{Z}$. Les entiers a, b sont premiers entre eux si et seulement s'il existe deux entiers u et v tels que :

$$a \cdot u + b \cdot v = 1.$$

1.8 Décomposition en facteurs premiers

La factorisation en nombres premiers est une méthode fondamentale de la théorie des nombres qui consiste à décomposer un nombre en un produit de facteurs premiers. Cette technique est utilisée depuis des siècles et trouve des applications dans de nombreux domaines, les mathématiques appliquées, la recherche de diviseurs communs et les multiples communs [6].

Théorème 6. Pour tout $n > 1$, il exsiste une décomposition en facteurs premiers de n, c'est-à-dire :

$$n = p_1^{\alpha_1} \cdot p_2^{\alpha 2} \cdot \cdots \cdot p_r^{\alpha_r}$$

où $p_1, p_2, \ldots, p_r$ sont des nombres premiers distincts et $\alpha_1, \alpha_2, \ldots, \alpha_r$ sont des exposants entiers positifs.
Si $p_1 < p_2 < \cdots < p_r$ donc il exsiste une unique décomposition.
Pour la décompotion d'un nombre $n > 1$ en facteurs premiers, on suit les étapes suivantes :

1. Diviser n par son plus petit diviseur noté p_i, on note le resultat de cette opération n_i.
2. Si $n_i = 1$ l'opération est terminée.
3. Sinon, si $n_i \geq 2$ retourner à l'étape (1) et faire de même pour n_i.

Exemple 5. On veut décomposer n, tel que $n = 728$

1. Le plus petit facteur premier de 728 est 2 donc $n_1 = \frac{728}{2} = 364$.
2. Le plus petit facteur premier de 364 est encore 2 donc $n_2 = \frac{364}{2} = 182$.
3. Le plus petit facteur premier de 182 est toujours 2
   donc $n_3 = \frac{182}{2} = 91$.
4. Le plus petit facteur premier de 91 est 7 et $n_4 = \frac{91}{7} = 13$.
5. Enfin, 13 est premier, donc on s'arrête ici.

Alors :

$$728 = 2 \cdot 2 \cdot 7 \cdot 13 = 2^2 \cdot 7^1 \cdot 13^1.$$

Remarque 1. (Nombres premiers) On dit qu'un nombre naturel $p > 1$ est premier si ces diviseurs positifs sont 1 et p.

$$\text{p est premier} \Leftrightarrow D(p) = \{1, p\}.$$

1.9 Congruences

Une congruence est une relation d'équivalence entre deux entiers qui indique qu'ils donnent le même reste lorsqu'ils sont divisés par un même nombre appelé le modulo, et on note le symbole de la congruence $\equiv$.

Définition 1.7. Soit a, b des entiers et $n \in \mathbb{N}^*$. On dit que a est congru à b modulo n si n divise $(a-b)$.
On note $a \equiv b[n] \quad \Leftrightarrow \quad n|(a-b) \quad \Leftrightarrow \quad \exists k \in \mathbb{Z} \quad \text{avec} \quad a = kn+b$.
On peut aussi noter les congruences par cette formule $a \equiv b \pmod{n}$.

Remarque 2. Si $a \equiv 0[n] \quad alors \quad a = nk \quad \text{avec} \quad k \in \mathbb{Z}$.

Lemme 2. Soit a, $b \in \mathbb{Z}$ et $n \in \mathbb{N}^*$

1. $a \equiv a[n]$.
2. Si $a \equiv b[n]$, alors $b \equiv a[n]$.
3. Si $a \equiv b[n]$ et $b \equiv c[n]$, alors $a \equiv c[n]$.
4. Si $a \equiv b[n]$ et $c \equiv d[n]$, alors $a+c \equiv b+d[n]$.
5. Si $a \equiv b[n]$, alors $a \cdot c \equiv b \cdot c[n]$ et $a^k \equiv b^k[n]$.

Exemple 6. Voici quelques exemples sur les propriétés du lemme précédent :

1. $7+2 = 9 \equiv 1[8]$
2. $(7+2)^{15} \equiv 1^{15}[8] \equiv 1[8]$
3. $14 \cdot (7+2)^{15} \equiv 14 \cdot 1[8]$
4. $14 \cdot (7+2)^{15} \equiv 6[8]$.

1.10 Classes des résidus

La classe des résidus, parfois appelée classe de congruence, est une notion fondamentale en arithmétique modulaire. Elle regroupe des entiers qui ont le même reste lorsqu'ils sont divisés par un nombre fixé, appelé le modulo.
Cette notion est très utile en cryptographie, où elle permet de créer des structures mathématiques basées sur les nombres entiers et leurs restes.
Le symbole $\mathbb{Z}$ représente l'ensemble des entiers relatifs. Cet ensemble comprend tous les nombres entiers positifs, négatifs et zéro$\{\ldots, -3, -2, -1, 0, 1, 2, 3, \ldots\}$.
n est un entier positif donné. Il représente le modulo ou le diviseur par lequel les entiers sont divisés pour former des classes d'équivalence. Les classes d'équivalence sont basées sur le reste de la division des entiers par $n[6]$.

Définition 1.8. Soit $a, b \in \mathbb{Z}$ et $n \in \mathbb{N}^*$, lorsque nous avons $a \equiv b[n]$, nous disons que b est un résidu de a modulo n. La classe de résidu de a modulo n, notée $[a]_n$, est définie comme l'ensemble de tous les entiers x tels que x est un entier et x est congru à a modulo n. $[a]_n = \{x : x \in \mathbb{Z} \quad et \quad x \equiv a[n]\}$.

Propriété 1. Pour un entier positif n, il existe n classes de résidus distinctes qui sont $[0]_n, [1]_n, \ldots, [n-1]_n$
Alors $\mathbb{Z}_n = \{[0]_n, [1]_n, \ldots, [n-1]_n\}$.

Exemple 7. Si nous considérons l'arithmétique modulo 5, les classes résidus sont les ensembles suivants

1. $[0]_5 = \{0, 5, 10, 15, \ldots\}$.
2. $[1]_5 = \{1, 6, 11, 16, \ldots\}$.
3. $[2]_5 = \{2, 7, 12, 17, \ldots\}$.
4. $[3]_5 = \{3, 8, 13, 18, \ldots\}$.
5. $[4]_5 = \{4, 9, 14, 19, \ldots\}$.

1.11 Inversibilité dans $\mathbb{Z}_n$

Propriété 2. On dit que $a \in \mathbb{Z}_n$ est **inversible** s'il existe $b \in \mathbb{Z}_n$, appelé l'**inverse** de a et noté a^{-1}, tel que :

$$a \cdot b = 1.$$

Les inversibles de $\mathbb{Z}_n$ sont exactement les k, où k est un entier premier avec n.

Exemple 8. Inversibilité de 17 dans $\mathbb{Z}_{31}$

1. Trouver l'inverse de 17 modulo 31, c'est-à-dire un entier b tel que :

$$17 \cdot b \equiv 1[31].$$

2. Vérification de l'inversibilité : On calcule PGCD$(17, 31)$:

$$\begin{aligned} 31 &= 1 \cdot 17 + 14 \\ 17 &= 1 \cdot 14 + 3 \\ 14 &= 4 \cdot 3 + 2 \\ 3 &= 1 \cdot 2 + 1 \\ 2 &= 2 \cdot 1 + 0. \end{aligned}$$

   Le PGCD est 1, donc 17 est inversible modulo 31.

3. Algorithme d'Euclide étendu : On remonte l'algorithme pour exprimer 1 comme combinaison linéaire de 17 et 31

$$\begin{aligned}
1 &= 3 - 1 \cdot 2 \\
&= 3 - 1 \cdot (14 - 4 \cdot 3) \\
&= 5 \cdot 3 - 1 \cdot 14 \\
&= 5 \cdot (17 - 1 \cdot 14) - 1 \cdot 14 \\
&= 5 \cdot 17 - 6 \cdot 14 \\
&= 5 \cdot 17 - 6 \cdot (31 - 1 \cdot 17) \\
&= 11 \cdot 17 - 6 \cdot 31.
\end{aligned}$$

On obtient ainsi

$$1 = 11 \cdot 17 - 6 \cdot 31.$$

4. L'équation montre que
$$11 \cdot 17 \equiv 1[31].$$
Ainsi, l'inverse de 17 modulo 31 est 11.
5. On vérifie que
$$17 \cdot 11 = 187$$
$$187 \equiv 1[31].$$
La vérification confirme que 11 est bien l'inverse de 17 modulo 31.
6. **Résultat** : L'inverse de 17 dans $\mathbb{Z}_{31}$ est 11, c'est-à-dire
$$17^{-1} \equiv 11[31].$$

Ce chapitre a permis de poser les bases théoriques nécessaires à la compréhension des calculs sur les entiers, en introduisant des outils tels que la division euclidienne, les algorithmes de calcul du PGCD, les congruences et les inverses modulaires. Ces éléments fondamentaux de la théorie des nombres seront largement exploités dans l'étude des systèmes cryptographiques et des méthodes de cryptanalyse .

CHAPITRE 2

Notions et définitions

Ce chapitre vise à poser les bases conceptuelles indispensables à la compréhension des systèmes de chiffrement et de déchiffrement. Il présente les définitions clés de la cryptologie, ainsi que les différentes formes de cryptographie, depuis les méthodes classiques jusqu'aux techniques modernes. Nous aborderons également les notions fondamentales de cryptanalyse, qui permettent d'évaluer la sécurité des algorithmes cryptographiques.

2.1 Cryptologie

La cryptologie est un mot composé qui tiré son origine du Grec : crypto qui signifie secret et logie qui signifie science. En fait , c'est la science du secret et ne peut être vraiment considérée ainsi que depuis peu de temps. Elle englobe la cryptographie,et la cryptanalyse .
Le chemin si dessue présente la structure de la cryptologie [23].

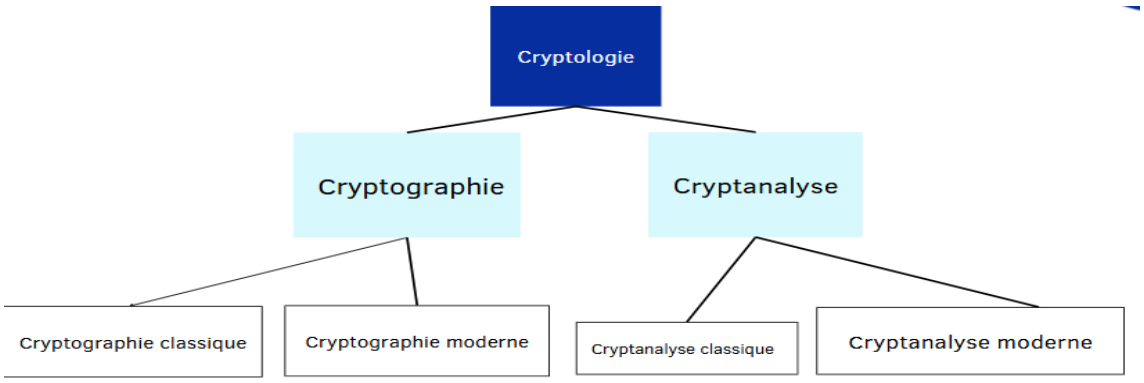

FIGURE 2.1 – Organigramme de la cryptologie

2.2 Cryptographie

Le terme cryptographie vient en effet de deux mots grecs : Kruptus qu'on peut traduire comme secret et Graphein pour écriture. La cryptographie est l'art de cacher l'information pour qu'elle soit incompréhensible, elle désigne l'ensemble des techniques qui permettent de chiffrer les messages, son objectif principal est de permettre à deux personnes alice et Bob de communiquer à travers un canal peu sécurisé de telle sorte qu'un opposant Eve ne puisse pas comprendre ce qui est échangé, on utilise une clé appelée clé de chiffrement pour le processus de chiffrement. Pour rendre l'information à nouveau compréhensible on utilise une clé appelée clé de déchiffrement [12].

Objectifs de la cryptographie La cryptographie est l'étude des techniques mathématiques qui sont utilisées pour accomplir plusieurs objectifs pour garantir la sécurité de communication, ces objectifs sont :

1. **La confidentialité :** Le message chiffré doit être lisible seulement par les personnes autorisées. Il ne doit pas pouvoir être lu par quelqu'un d'autre.
2. **L'intégrité :** La personne qui reçoit le message doit pouvoir vérifier qu'il n'a pas été changé pendant l'envoi. Une autre personne ne doit pas pouvoir remplacer un vrai message par un faux.
3. **L'authentification :** Permettre à la personne qui reçoit un message de vérifier qui l'a envoyé, pour s'assurer que personne n'a utilisé une fausse identité.

2.3 Mots clés

1. **Chiffrement (cryptage)** Est un procédé de cryptographie grâce auquel on souhaite rendre la compréhension d'un document impossible à toute personne qui n'a pas la clé de déchiffrement.
2. **Clé de chiffrement** Paramètre constitué d'une séquence de symboles et utilisé, avec un algorithme cryptographique, pour transformer, valider, authentifier, chiffrer ou déchiffrer des données.
3. **Déchiffrement** C'est la méthode ou l'algorithme utilisé pour transformer un texte chiffré en un texte en clair.
4. **Texte en clair** C'est le message à protéger
5. **Texte chiffré** C'est le résultat du chiffrement du texte en clair.
6. **Chiffrement par bloc** Technique de chiffrement où le texte clair est divisé en blocs de taille fixe, et chaque bloc est chiffré indépendamment à l'aide d'une clé et d'un algorithme[3].

2.4 Types de cryptographies

La cryptographie peut être classée en trois grands types selon les époques et les technologies utilisées cryptographie classique, cryptographie moderne et cryptographie quantique.

2.5 Cryptographie classique

La cryptographie classique a été créée avant l'invention des ordinateurs. Dans la cryptographie classique, la méthode et la clé pour chiffrer et déchiffrer les messages sont connues par la personne qui envoie le message et celle qui le reçoit.

2.5.1 Chiffrement par décalage

Le chiffrement par décalage, aussi appelé chiffre de César, est une méthode très simple pour coder des messages. Jules César l'utilisait pour protéger ses messages secrets. Le principe est facile : chaque lettre du message est remplacée par une autre lettre de l'alphabet, en la décalant d'un certain nombre de places[7].
La fonction de chiffrement $e_k : \mathbb{Z}_d \to \mathbb{Z}_d$ est donnée par

$$e_k(m) = m + k.$$

Avec k la clé.
La fonction de déchiffrement $d_k : \mathbb{Z}_d \to \mathbb{Z}_d$ est donnée par

$$d_k(c) = c - k$$

Exemple 9. Considérons le chiffrement de César défini par $P = C = K = \mathbb{Z}_{26}$, représentant les 26 lettres de l'alphabet (A = 1, B = 2, …, Y = 25, Z = 0).
Chiffrement des lettres A, F, X avec $k = 7$

- Pour A nous avons : $e_7(1) = 1 + 7 = 8$ = H
- Pour F nous avons : $e_7(6) = 6 + 7 = 13$ = M
- Pour X nous avons : $e_7(24) = 24 + 7 = 31 \mod 26 = 5$ = E

Déchiffrement du texte chiffré RLF Convertir les lettres en nombres : R = 18, L = 12, F = 6 :

$$d_7(18) = 18 - 7 = 11 = K$$
$$d_7(12) = 12 - 7 = 5 = E$$
$$d_7(6) = 6 - 7 = -1 \mod 26 = 25 = Y$$

Message déchiffré : KEY

2.5.2 Chiffrement affine

Le chiffrement affine utilise une formule mathématique simple pour transformer chaque lettre du message en une autre lettre. Cette méthode est une version améliorée du chiffrement de César.
La fonction de chiffrement $e_k : \mathbb{Z}_d \to \mathbb{Z}_d$ est donnée par

$$e_k(m) = k_1 \cdot m + k_2$$

La fonction de déchiffrement $d_k : \mathbb{Z}_d \to \mathbb{Z}_d$ est donnée par

$$d_k(c) = k_1^{-1} \cdot (c - k_2)$$

où k_1^{-1} est l'inverse de k_1 modulo d.

2.5.3 Chiffrement de Vigenère

Le chiffrement de Vigenère est une méthode de chiffrement par substitution poly-alphabétique, inventée par Blaise de Vigenère.
Il utilise une clé pour déterminer plusieurs décalages différents dans l'alphabet.
Voici comment cela fonctionne :
La fonction de chiffrement $e_k : \mathbb{Z}_d^n \to \mathbb{Z}_d^n$ est donnée par

$$e_k(m) = m + k$$

La fonction de déchiffrement $d_k : \mathbb{Z}_d^n \to \mathbb{Z}_d^n$ est donnée par

$$d_k(c) = c - k$$

2.5.4 Chiffrement de Hill

Le chiffrement de Hill est une méthode de chiffrement par blocs basée sur l'algèbre linéaire. Il utilise une matrice inversible k pour chiffrer un message m sous forme de vecteur. La sécurité repose sur la difficulté à inverser la matrice sans connaître la clé.
Fonction de chiffrement $e_k : \mathbb{Z}_d^n \to \mathbb{Z}_d^n$ est donnée par

$$e_k(m) = k \cdot m$$

où $m \in \mathbb{Z}_d^n$ est le message et $k \in \mathrm{GL}(n, \mathbb{Z}_d)$ est la matrice de chiffrement.
Fonction de déchiffrement $d_k : \mathbb{Z}_d^n \to \mathbb{Z}_d^n$ est donnée par

$$d_k(c) = k^{-1} \cdot c$$

où $c \in \mathbb{Z}_d^n$ est le texte chiffré et k^{-1} est l'inverse de la matrice k.

Exemple 10. Cryptage de Hill (A = 1, B = 2, ..., Z = 26)
Données

- Texte clair : "HILL"
- Alphabet : A = 1, B = 2, ..., Z = 26.
- Matrice de chiffrement k :

$$k = \begin{pmatrix} 3 & 3 \\ 2 & 5 \end{pmatrix}$$

- Blocs de 2 lettres : "HI" et "LL".

Conversion en nombres

- "H" = 8, "I" = 9 $\to$ Premier bloc : $\begin{pmatrix} 8 \\ 9 \end{pmatrix}$
- "L" = 12, "L" = 12 $\to$ Deuxième bloc : $\begin{pmatrix} 12 \\ 12 \end{pmatrix}$

1. Chiffrement
   Appliquons $e_k(m) = k \cdot m \quad$ dans $\mathbb{Z}_{26}$.
   1. Pour le premier bloc $\begin{pmatrix} 8 \\ 9 \end{pmatrix}$:

$$\begin{pmatrix} 3 & 3 \\ 2 & 5 \end{pmatrix} \cdot \begin{pmatrix} 8 \\ 9 \end{pmatrix} = \begin{pmatrix} 3 \cdot 8 + 3 \cdot 9 \\ 2 \cdot 8 + 5 \cdot 9 \end{pmatrix}$$
$$= \begin{pmatrix} 24 + 27 \\ 16 + 45 \end{pmatrix}$$
$$= \begin{pmatrix} 51 \\ 61 \end{pmatrix}$$
$$= \begin{pmatrix} 25 \\ 9 \end{pmatrix}$$

   Résultat : "YI".

2.5.5 Chiffrement de Hill affine

Le chiffrement de Hill affine est une extension du cryptage de Hill. Il combine une transformation linéaire (matrice) et une translation (vecteur) pour améliorer la sécurité.
Fonction de chiffrement $e_k : \mathbb{Z}_d^n \to \mathbb{Z}_d^n$ est donnée par

$$e_k(m) = k_1 \cdot m + k_2$$

où $k = (k_1, k_2) \in \mathrm{GL}(n, \mathbb{Z}_d) \times \mathbb{Z}_d^n$ est la clé, k_1 est la matrice de chiffrement, et k_2 est le vecteur de translation.
Fonction de déchiffrement $d_k : \mathbb{Z}_d^n \to \mathbb{Z}_d^n$ est donnée par

$$d_k(c) = k_1^{-1} \cdot (c - k_2)$$

où k_1^{-1} est l'inverse de la matrice k_1.

2.6 Cryptographie moderne

La cryptographie moderne est une discipline avancée de la sécurité informatique qui vise à protéger les informations dans un environnement numérique. Elle assure des services essentiels tels que la confidentialité, l'intégrité, l'authentification, et la non-répudiation, en utilisant des méthodes plus sûres et plus complexes que la cryptographie classique. Elle se divise en deux techniques principales : le chiffrement symétrique et le chiffrement asymétrique [5].

2.6.1 Cryptographie symétrique

La cryptographie symétrique, ou cryptographie à clé secrète, est une méthode de protection des messages qui utilise une seule clé pour chiffrer (coder) et déchiffrer (décoder) les informations. Cette clé doit rester secrète et être partagée uniquement entre l'émetteur et le destinataire. Si la clé est compromise, un tiers peut accéder aux messages. Pour garantir la sécurité, la clé doit être complexe et difficile à deviner. La photo ci-dessous est un schéma de cryptographie symétrique.

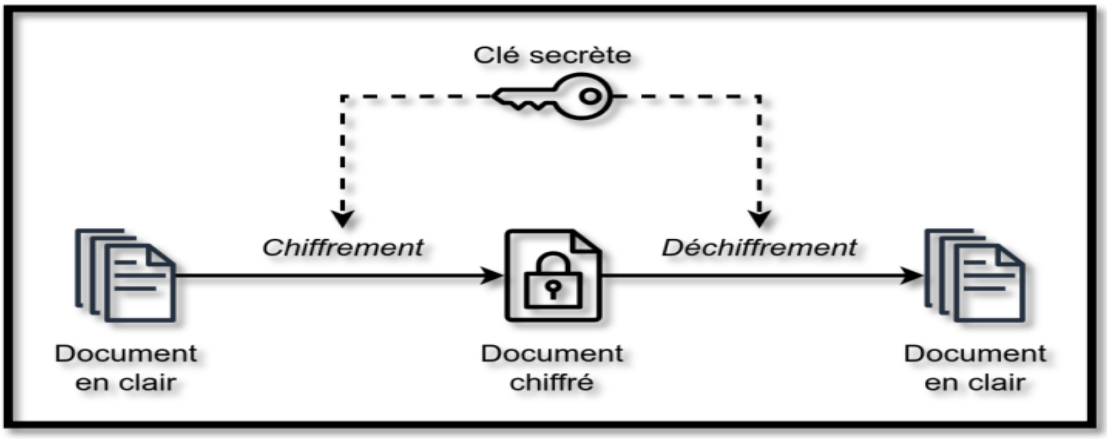

FIGURE 2.2 – Chiffrement symétrique[23]

Pour la sécurité du message, il existe plusieurs techniques de cryptage :

2.6.1.1 Réseau de Feistel

Le réseau de Feistel est une structure utilisée pour chiffrer les messages de manière sécurisée. Il fonctionne en divisant le message en deux blocs de taille égale, appelées Gauche (L) et Droite (R), puis en appliquant plusieurs étapes, appelées tours. À chaque tour, on utilise une fonction spéciale (F) et une clé temporaire (sous-clé) pour modifier les données. Ce qui rend cette méthode pratique, c'est qu'elle peut être utilisée à la fois pour coder et décoder un message il suffit d'inverser les étapes [1].

2.6.1.2 Chiffrement DES

Le Data Encryption Standard (DES) est un algorithme de chiffrement symétrique par blocs, fondé sur une structure en réseau de Feistel. Il opère sur des blocs de 64 bits, qu'il chiffre en utilisant une clé principale de 56 bits. Cette clé est dérivée d'une clé saisie de 64 bits, dont un bit par octet est réservé à la parité et donc ignoré dans le processus de chiffrement. L'algorithme effectue 16 tours successifs, combinant à chaque étape des opérations de substitution, à l'aide de S-boîtes, et de permutation, afin d'assurer la confusion et la diffusion des données. Cette

architecture rend possible le déchiffrement en appliquant les mêmes opérations dans l'ordre inverse des sous-clés[27].

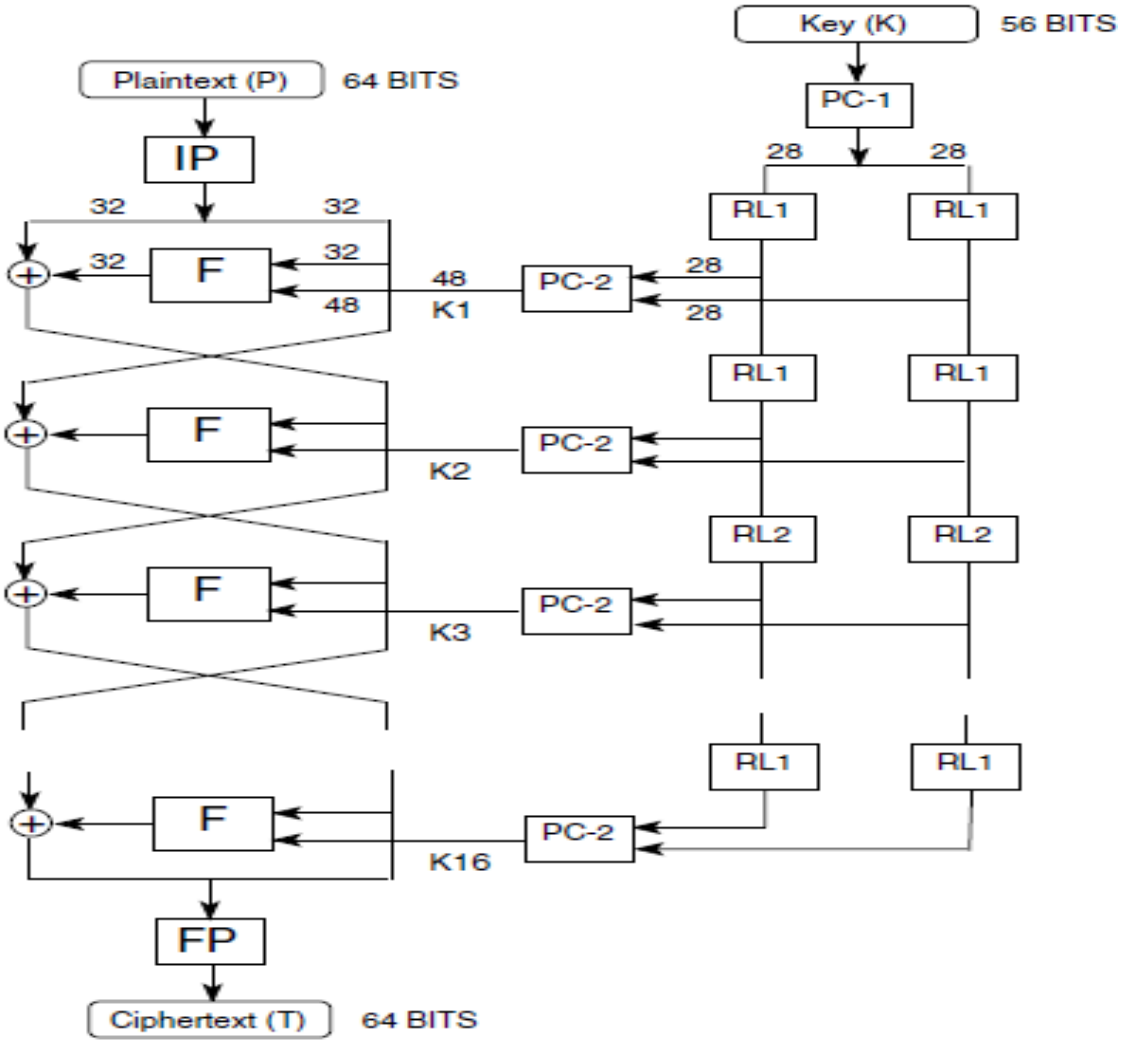

FIGURE 2.3 – Chiffrement DES

Algorithm 3 L'algorithme du Data Encryption Standard (DES)

Choisir une clé k de 64 bits
Soit f la fonction de ronde (ou fonction de chiffrement)
f est construite à partir des S-boîte S_i (fonctions de sélection) et de la permutation π
Soient $k_1, k_2, \ldots, k_{16}$ les sous-clés pour les 16 tours
Chaque k_i est un bloc de 48 bits
Les sous-clés sont données par :

$$k_i = \mathrm{KS}(i,k) \quad \text{pour } i = 1,2,\ldots,16$$

Soit x un bloc de texte clair de 64 bits
Appliquer la permutation initiale :

$$\mathrm{IP}(x) = (L_0, R_0) \quad \text{avec } |L_0| = |R_0| = 32$$

for $i = 1$ **à** 16 **do**
    $L_i = R_{i-1}$
    $R_i = L_{i-1} \oplus f_{k_i}(R_{i-1})$
end for
Le couple final obtenu est (R_{16}, L_{16})
Appliquer la permutation finale inverse :

$$c = \mathrm{IP}^{-1}(R_{16}, L_{16})$$

Déchiffrement
Appliquer IP sur c
for $i = 16$ **à** 1 **par pas de** -1 **do**
    $R_{i-1} = L_i$
    $L_{i-1} = R_i \oplus f_{k_i}(L_i)$
end for
Reconstituer le texte clair :

$$x = \mathrm{IP}^{-1}(L_0, R_0)$$

Algorithm 4 L'algorithme de génération des sous-clés (DES)

1: **Choisir** une clé de 56 bits (en réalité une clé de 64 bits avec 8 bits de parité)
2: Appliquer la permutation PC_1 à la clé k :

$$k \leftarrow \mathrm{PC}_1(k)$$

3: Diviser le résultat en deux moitiés :

$$k = (a_0\, b_0)$$

4: **for** $r = 1$ **à** 16 **do**
5: $\quad a_r = \mathrm{leftshift}_j(a_{r-1})$
6: $\quad b_r = \mathrm{leftshift}_j(b_{r-1})$
7: $\quad k_r = \mathrm{PC}_2(a_r, b_r)$
8: **end for**
9: Où $\mathrm{leftshift}_j$ désigne une rotation circulaire à gauche de j positions, avec :

$$j = \begin{cases} 1 & \text{si } r \in \{1,2,9,16\} \\ 2 & \text{sinon} \end{cases}$$

10: À l'issue des 16 itérations, on obtient :

$$\mathrm{KS}(k) = (k_1, k_2, \ldots, k_{16})$$

11: PC_1 est une permutation des positions de la clé k
12: PC_2 extrait 48 bits de (a_r, b_r)

• La fonction $\mathrm{f}_{\mathrm{k}_{i-1}}(\mathrm{R}) = \pi\left[S_1\left((E(R) \oplus k)_1\right), S_2\left(((E(R) \oplus k)_2\right), \ldots, S_8\left((E(R) \oplus k)_8\right)\right]$
Où $S_i : \mathbb{B}^6 \to \mathbb{B}^4$ est la i-ème boîte S (avec $\mathbb{B} = \{0,1\}$)

• Si l'entrée est $x = (x_1, \ldots, x_{64})$, alors $(\mathrm{IP}(x))_1 = x_{58}, \quad (\mathrm{IP}(x))_2 = x_{50}, \ \ etc.$
• Si la pré-sortie est $x = (x_1, \ldots, x_{64})$, alors $(\mathrm{IP}^{-1}(x))_1 = x_{40}, \quad (\mathrm{IP}^{-1}(x))_2 = x_8, \ \ etc.$
• Fonction d'expansion : $E : \mathbb{B}^{32} \to \mathbb{B}^{48}$ prend 32 bits en entrée et en génère 48 en sortie.
• Fonction $\mathrm{S_i}$ (S-boîte) : $S_i : \mathbb{B}^6 \to \mathbb{B}^4$, C'est la i$^{\mathrm{e}}$ boîte de substitution. Une S-boîte S_i est définie par une matrice M_i de dimension 4×16, dont les entrées sont données en base 10.

$$\begin{aligned} S_i(x_1,x_2,x_3,x_4,x_5,x_6) &= \mu \circ M_i\left((x_1,x_6)_2, (x_2,x_3,x_4,x_5)_2\right) \\ &= \mu(y_1,y_2,y_3,y_4) \\ &= (y_1,y_2,y_3,y_4) \end{aligned}$$

Exemple 11. • Texte clair x : `0123456789ABCDEF`

• Clé (key) : `133457799BBCDFF1`

TOUR 1

Étape 1 : Conversion en binaire

- Texte clair (64 bits) x
  `00000001 00100011 01000101 01100111 10001001 10101011 11001101 11101111`
- Clé (64 bits) :
  `00010011 00110100 01010111 01111001 10011011 10111100 11011111 11110001`

Étape 2 : Permutation initiale (IP)

La permutation initiale donne :

$$L_0 = \texttt{11001100 00000000 11001100 11111111}$$

$$R_0 = \texttt{11110000 10101010 11110000 10101010}$$

Étape 3 : Génération des sous clés

• Application de la permutation PC_1 (56 bits) :

`1111000 0110011 0010101 0101111 0101010 1011001 1001111 0001111`

• Division en deux moitiés (28 bits)

— Partie droite a_0 :

`1111000 0110011 0010101 0101111`

— Partie gauche b_0 :

`0101010 1011001 1001111 0001111`

• Rotations circulaires gauche (décalage de 1 bit)

$$a_1 = \texttt{1110000 1100110 0101010 1011111}$$
$$b_1 = \texttt{1010101 0110011 0011110 0011110}$$

$x_1 = (a_1, b_1)$

$= \texttt{1110000 1100110 0101010 1011111 1010101 0110011 0011110 0011110}$

• Application de la permutation PC_2

$$K_1 = \texttt{000110 110000 001011 101111 111111 000111 000001 110010}$$

Étape 3 : Algorithme de la fonction f

• Apllication d'expansion E sur R_0

$$E(R_0) = \texttt{011110100001010101010101011110100001010101010101}$$

• Application de XOR entre $E(R_0)$ et K_1

$$E(R_0) = \texttt{011110100001010101010101011110100001010101010101}$$
$$K_1 = \texttt{000110110000001011101111111111000111000001110010}$$

On effectue le XOR bit à bit :

$$\begin{array}{l} \texttt{011110 100001 010101 010101 011110 100001 010101 010101} \\ \oplus \\ \texttt{000110 110000 001011 101111 111111 000111 000001 110010} \\ \hline \texttt{011000 010001 011110 111010 100001 100110 010100 100111} \end{array}$$

M_i est la matrice associée à la boite S_i

$$\begin{aligned} S_1(x_1,x_2,x_3,x_4,x_5,x_6) &= S_1(011000) = \mu \circ M_1\big((0,0)_2,(1,1,0,0)_2\big) \\ &= \mu \circ M_1(0,12) = \mu(5) = (0101)_2 = \texttt{0101} \\ S_2(x_7,x_8,x_9,x_{10},x_{11},x_{12}) &= S_2 = 010001 = \mu \circ M_2\big((0,1)_2,(1,0,0,0)_2\big) \\ &= S_2 = \mu \circ M_2(1,8) = \mu(12) = (1100)_2 = \texttt{1100} \end{aligned}$$

De même, pour les autres blocs, on obtient directement :

$$\begin{aligned} \mathrm{S}_3(x_{13},x_{14},x_{15},x_{16},x_{17},x_{18}) &= \texttt{1000} \\ \mathrm{S}_4(x_{19},x_{20},x_{21},x_{22},x_{23},x_{24}) &= \texttt{0010} \\ \mathrm{S}_5(x_{25},x_{26},x_{27},x_{28},x_{29},x_{30}) &= \texttt{1011} \\ \mathrm{S}_6(x_{31},x_{32},x_{33},x_{34},x_{35},x_{36}) &= \texttt{0101} \\ \mathrm{S}_7(x_{37},x_{38},x_{39},x_{40},x_{41},x_{42}) &= \texttt{1001} \\ \mathrm{S}_8(x_{43},x_{44},x_{45},x_{46},x_{47},x_{48}) &= \texttt{0111} \end{aligned}$$

Alors $[S_1((E(R)\oplus k)_1), S_2(((E(R)\oplus k)_2), \ldots, S_8((E(R)\oplus k)_8)] = 0101\ 1100\ 1000\ 0010\ 1011\ 0101\ 1001\ 0$

- En appliquant la permutation π au résultat précédent, ce qui donne

$$f_{k_1}(R_0) = \texttt{0010 0011 0100 1010 1010 1001 1011 1011}$$

- Calcul de L_1 et R_1

$$\begin{array}{l} L_1 = R_0 = \texttt{1111 0000 1010 1010 1111 0000 1010 1010} \\ R_1 = L_0 \oplus f(R_0,K_1) \\ \quad = \texttt{1100 1100 0000 0000 1100 1100 1111 1111} \\ \quad \oplus\ \texttt{0010 0011 0100 1010 1010 1001 1011 1011} \\ \hline R_1 = \texttt{1110 1111 0100 1010 0110 0101 0100 0100} \end{array}$$

2.6.1.3 Réseaux de substitution-permutation (SPN)

Le réseau de substitution-permutation (SPN) est un système de chiffrement par blocs qui fonctionne en plusieurs tours successifs. Chaque tour se compose généralement de trois étapes : l'ajout de clé, la substitution, puis la permutation [22].

— Ajout de clé : consiste à combiner le bloc courant avec une sous-clé à l'aide d'un opérateur XOR.

— Substitution : applique des S-boîtess sur des sous-blocs afin de mélanger les bits localement.

— Permutation : réorganise les bits dans tout le bloc pour assurer une diffusion globale de l'information.

.

Chiffrement SPN-12 :

Avant de présenter l'algorithme, précisons les éléments utilisés :

— Texte clair : $x \in \mathbb{B}^{12}$, un mot de 12 bits divisé en 3 blocs de 4 bits :

$$x = (x_1, x_2, x_3)$$

— Clé : $k = (k_1, \ldots, k_{n+1})$, où chaque $k_i \in \mathbb{B}^{12}$, divisée également en 3 sous-blocs de 4 bits.

— Substitution : $y = \sigma(x)$, appliquée bloc par bloc (chaque bloc passe par une S-boîte).

— Permutation : $y = \pi(y)$, où π est une permutation fixe des positions de bits.

— Sortie : Le texte chiffré $c \in \mathbb{B}^{12}$

Algorithm 5 Algorithme de chiffrement SPN

1: **Entrée :** Texte clair $x \in \mathbb{B}^{12}$ (divisé en 3 blocs de 4 bits)
2: **Clé :** $k = (k_1, k_2, ..., k_r, k_{r+1})$, avec chaque $k_i \in \mathbb{B}^{12}$
3: **for** $i = 1$ **à** $r - 1$ **do**
   $x \leftarrow x \oplus k_i$
   $x \leftarrow f_\sigma(x)$
   $x \leftarrow \pi(x)$
4: **end for**
5: **for** $i = r$ **do**
   $x \leftarrow x \oplus k_r$
   $x \leftarrow f_\sigma(x)$
   $x \leftarrow x \oplus k_{r+1}$
   $c \leftarrow x$
6: **Sortie :** Le texte chiffré $c \in \mathbb{B}^{12}$

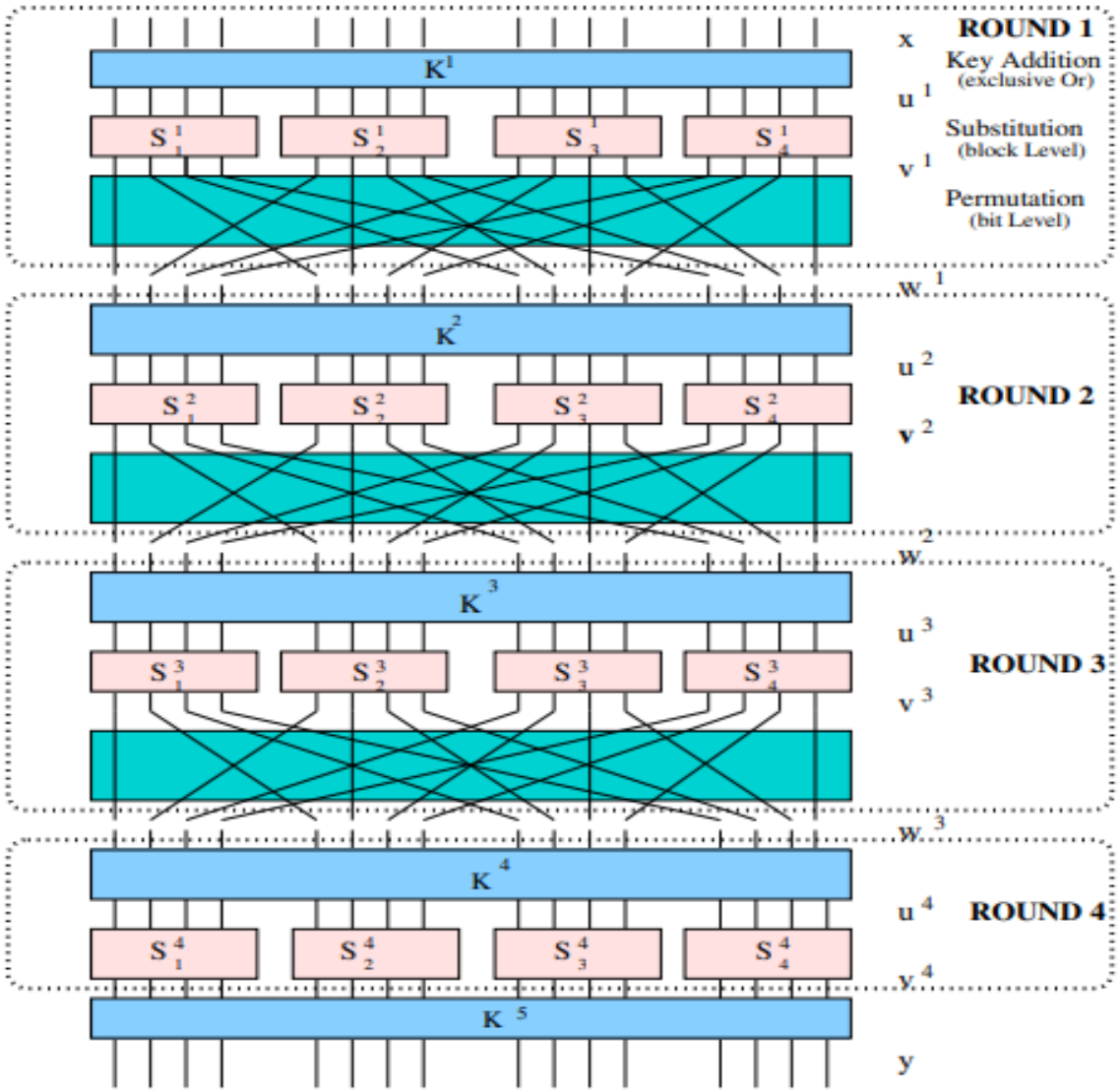

FIGURE 2.4 – SPN : Nombre de tours = 4, taille de bloc de la S-boîte = 4 bits, taille du bloc = 16 bits [21]

Exemple 12. Exemple détaillé de chiffrement SPN de 2 tours :

S-boîte (substitution de 4 bits en sortie) :

Chaque valeur d'entrée (de 0 à 15) est transformée selon la S-boîte (voir la table 13) :

Permutation sur 12 bits :

Le bit de position i dans le bloc est déplacé à la position $\pi(i)$ (voir la table 14) :

Texte clair

$$\text{Texte clair initial} = 7bA = 011110111010$$

Tour 1

Étape 1 : XOR avec la clé du tour 1

$$011110111010 \oplus k_1 = \texttt{110010101111}$$

Étape 2 : Substitution par blocs

$$\begin{aligned}
\text{Bloc 1 :} \quad & \texttt{1100} \rightarrow \texttt{1001} \quad (\sigma(12) = 9) \\
\text{Bloc 2 :} \quad & \texttt{1010} \rightarrow \texttt{0000} \quad (\sigma(10) = 0) \\
\text{Bloc 3 :} \quad & \texttt{1111} \rightarrow \texttt{1010} \quad (\sigma(15) = A)
\end{aligned}$$

Résultat de la substitution :

$$\texttt{100100001010}$$

Étape 3 : Permutation

Position i	Bit	Position après $\sigma(i)$
0	1	0
1	0	4
2	0	7
3	1	11
4	0	1
5	0	6
6	0	9
7	0	3
8	1	2
9	0	8
10	1	10
11	0	5

Résultat après permutation :

$$\texttt{100000010110}$$

Tour 2

Étape 1 : XOR avec la clé du tour 2

$$100000010110 \oplus k_2 = \texttt{101100111110}$$

Étape 2 : Substitution par blocs

$$\begin{aligned} \text{Bloc 1 :} \quad & \texttt{1011} \to \texttt{1111} \quad (\sigma(11) = F) \\ \text{Bloc 2 :} \quad & \texttt{0011} \to \texttt{0001} \quad (\sigma(3) = 1) \\ \text{Bloc 3 :} \quad & \texttt{1110} \to \texttt{0101} \quad (\sigma(14) = 5) \end{aligned}$$

Résultat après substitution :

$$\texttt{111100010101}$$

(Note : pas de permutation au dernier tour)

Finalisation

$$111100010101 \oplus k_3 = \texttt{010111110001}$$

Texte chiffré final

$$\text{Binaire final} = \texttt{010111110001} \quad \Rightarrow \quad \text{hexadécimal} = \texttt{5F1}$$

2.6.2 Cryptographie asymétrique

La cryptographie asymétrique, ou à clé publique, repose sur une paire de clés : une clé publique, que l'on peut partager librement, et une clé privée, qui reste secrète. Ce système permet de chiffrer un message avec l'une des clés et de le déchiffrer uniquement avec l'autre. La photo ci-dessous est un schéma de cryptographie asymétrique

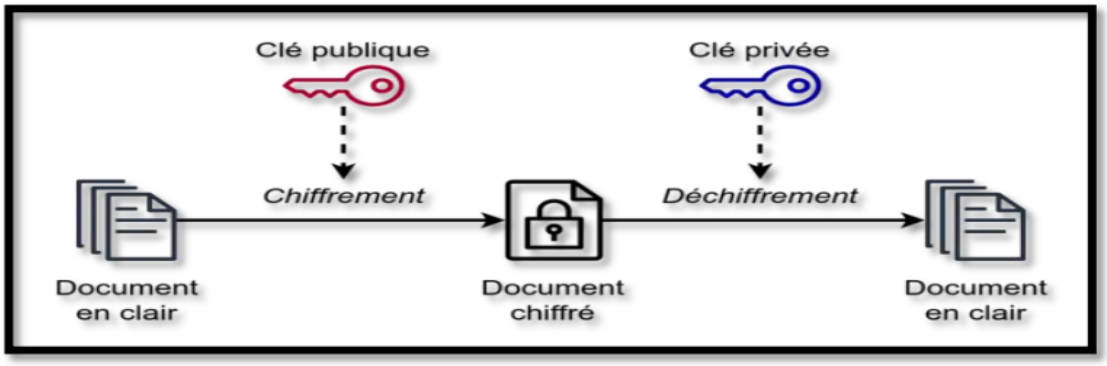

FIGURE 2.5 – Chiffrement asymétrique[23]

2.6.2.1 Chiffrement RSA

Le chiffrement RSA porte le nom de ses inventeurs : Ron Rivest, Adi Shamir et Leonard Adleman. C'est le premier algorithme de chiffrement asymétrique. Il a été créé en 1977 au Massachusetts Institute of Technology (MIT).

Le principe du RSA est simple :

— On utilise une clé publique pour chiffrer les données.

— On utilise une clé privée pour déchiffrer les données.

Maintenant, nous allons expliquer les opérations que l'on peut faire avec cet algorithme.

Génération des clés

Le RSA utilise deux nombres premiers. Ces nombres doivent être très grands. Ensuite, on calcule la clé privée et la clé publique avec l'algorithme suivant

Algorithm 6 Algorithme de génération des clés RSA

1: **Entrées :** deux nombres premiers p et q
2: **Sorties :** une clé publique (N,e) et une clé privée (N,d)
3: **if** $p = q$ **then**
4: Choisir deux nouveaux nombres premiers p et q
5: **end if**
6: Calculer le module : $N = p \times q$
7: Calculer la fonction d'Euler : $\varphi(N) = (p-1) \times (q-1)$
8: Choisir un entier e tel que $1 < e < \varphi(N)$ et $\text{PGCD}(e, \varphi(N)) = 1$
9: Calculer d, l'inverse de e modulo $\varphi(N)$, tel que $d < \varphi(N)$
10: **return** (N,e) comme clé publique et (N,d) comme clé privée.

Chiffrement d'un message

Supposons que deux personnes, A et B, possèdent chacune leur propre système RSA :

— A a son module N_A, sa clé publique e_A et sa clé privée d_A.

— B a son module N_B, sa clé publique e_B et sa clé privée d_B.

Si A veut envoyer un message m à B, voici comment faire :

Algorithm 7 Algorithme de chiffrement RSA

1: **Entrées :** un message clair m et la clé publique de B (N_B, e_B)
2: **Sortie :** un message chiffré c
3: Transformer le message m en un entier $< N_B - 1$
4: Calculer le message chiffré :

$$c \leftarrow m^{e_B} \mod N_B$$

5: Envoyer le message c à B

Déchiffrement du message

B a reçu un message chiffré c de la part de A. Pour le déchiffrer, B utilise sa clé privée d_B, comme décrit dans l'algorithme suivant :

Algorithm 8 Algorithme de déchiffrement RSA

Require: Un message chiffré c, la clé privée de B (N_B, d_B)

Ensure: Le message clair m

1: Calculer $m \leftarrow c^{d_B} \mod N_B$

2: **return** m

Exemple 13. Exemple de chiffrement RSA avec le message "BAALI"

Étape 1 : Génération des clés

- Choisir deux nombres premiers : $p = 17$ et $q = 19$.
- Calculer $N = p \times q = 17 \times 19 = 323$.
- Calculer $\phi(N) = (p-1) \times (q-1) = 16 \times 18 = 288$.
- Choisir la clé publique $e = 5$ (car PGCD$(5, 288) = 1$).
- Calculer la clé privée $d = 173$ (car $5 \times 173 = 865 \equiv 1 \mod 288$).

Résultat :

- Clé publique : $(e, N) = (5, 323)$.
- Clé privée : $(d, N) = (173, 323)$.

2.7 Cryptanalyse

La cryptanalyse est une technique qui permet de transformer un message chiffré en texte clair sans connaître la clé. Si on réussit, on peut obtenir soit le texte clair, soit la clé elle-même. Chaque tentative s'appelle une attaque, et si elle fonctionne, elle devient une méthode. On distingue deux types de cryptanalyse **la cryptanalyse classique** et **la cryptanalyse moderne** [26].

2.7.1 Cryptanalyse classique

Les techniques de cryptanalyse classiques sont des méthodes anciennes utilisées pour déchiffrer des messages cryptés avant l'apparition des systèmes de chiffrement modernes. Elles étaient souvent basées sur des calculs manuels et des analyses simples. Voici quelques méthodes :

2.7.1.1 Attaque par force brute (attaque exhaustive)

L'attaque par force brute consiste à tester toutes les clés possibles une par une jusqu'à trouver la bonne. C'est la seule méthode pour retrouver la clé dans des algorithmes modernes. Le nombre maximum de clés à tester dépend de la longueur de la clé (n). Ce nombre est calculé par la formule :

$$\text{Nombre de clés} = 2^n$$

Cependant, cette technique ne fonctionne bien que si la clé est courte. Pour les clés très longues ou les mots de passe complexes, le temps nécessaire devient trop important, ce qui rend l'attaque peu pratique.

Taille de clé (bits)	Nombre de clés alternatives	Temps requis pour 1 chiffrement/µs	Temps requis pour 10^6 chiffres/µs
32	$2^{32} = 4{,}3 \times 10^9$	2^{31} µs = 35,8 minutes	2,15 millisecondes
56	$2^{56} = 7{,}2 \times 10^{16}$	2^{55} µs = 1142 ans	10,01 heures
128	$2^{128} = 3{,}4 \times 10^{38}$	2^{127} µs = $5{,}4 \times 10^{24}$ ans	$5{,}4 \times 10^{18}$ ans
168	$2^{168} = 3{,}7 \times 10^{50}$	2^{167} µs = $5{,}9 \times 10^{36}$ ans	$5{,}9 \times 10^{30}$ ans
26 caractères (permutation)	$26! = 4 \times 10^{26}$	2×10^{26} µs = $6{,}4 \times 10^{12}$ ans	$6{,}4 \times 10^6$ ans

FIGURE 2.6 – Sécurité fournie selon la taille de la clé.

D'après les données du tableau, plus la taille de la clé augmente, plus le temps nécessaire pour la casser devient exponentiellement long. Par exemple, une clé de 128 bits nécessiterait des milliards d'années pour être cassée[4].

2.7.1.2 Attaque par analyse fréquentielle

L'analyse fréquentielle en cryptanalyse est une méthode qui étudie la fréquence des lettres dans un texte chiffré pour trouver des correspondances avec les lettres les plus utilisées dans la langue d'origine[4].

Les conditions de l'analyse de fréquence

Pour que cette attaque fonctionne, plusieurs conditions doivent être réunies :

1. L'algorithme de chiffrement doit utiliser des substitutions comme le code César (par décalage) ou Vigenère .
2. Le message chiffré doit être suffisamment long, pour des résultats précis.
3. La langue du message doit être connue pour pouvoir utiliser les statistiques de fréquence appropriées .

Méthodologie de l'analyse de fréquence :

1. On commence par calculer combien de fois chaque lettre apparaît dans le texte chiffré, en pourcentage ou en nombre.
2. Ensuite, on compare ces résultats avec les fréquences normales des lettres dans la langue utilisée.
3. Les lettres les plus fréquentes dans le message chiffré sont alors associées aux lettres les plus fréquentes dans la langue, ce qui aide à deviner les remplacements et à retrouver une partie de la clé de chiffrement.

Limitations
L'analyse fréquentielle est peu efficace contre les méthodes de chiffrement modernes et les chiffrements qui modifient ou masquent les fréquences des lettres.

1. Les méthodes modernes mélangent les données de manière très complexe. Par exemple, *DES* utilise des opérations mathématiques qui rendent le texte chiffré complètement aléatoire.
2. Les méthodes qui perturbent les fréquences

   Au lieu de remplacer toujours la lettre « A » par « X », ces méthodes changent la substitution à chaque fois.

   Par exemple, « A » peut devenir « X », « P » ou « 9 » de manière aléatoire.

   la lettre « E » (la plus fréquente en français) n'apparaît plus plus souvent que les autres, ce qui rend l'analyse fréquentielle inefficace.

Table des fréquences Voici les fréquence de caractères de certains langages latins :

Lettre	Allemand	Anglais	Espagnol	Français	Italien	Portugais
A	6.51	8.167	11.525	7.636	11.74	14.63
B	1.89	1.492	2.215	0.901	0.92	1.04
C	3.06	2.782	4.019	3.260	4.5	3.88
D	5.08	4.253	5.510	3.669	3.73	4.99
E	17.40	12.702	12.681	14.715	11.79	12.57
F	1.66	2.228	0.692	1.066	0.95	1.02
G	3.01	2.015	1.768	0.866	1.64	1.30
H	4.76	6.094	0.703	0.737	1.54	1.28
I	7.55	6.966	6.247	7.529	11.28	6.18
J	0.27	0.153	0.443	0.613	0.00	0.40
K	1.21	0.772	0.011	0.049	0.00	0.02
L	3.44	4.025	4.967	5.456	6.51	2.78
M	2.53	2.406	3.157	2.968	2.51	4.74
N	9.78	6.749	6.712	7.095	6.88	5.05
O	2.51	7.507	8.683	5.598	9.83	10.73
P	0.79	1.929	2.510	2.521	3.05	2.52
Q	0.02	0.095	0.877	1.362	0.51	1.20
R	7.00	5.987	6.871	6.693	6.37	6.53
S	7.27	6.327	7.977	7.948	4.98	7.81
T	6.15	9.056	4.632	7.244	5.62	4.74
U	4.35	2.758	2.927	6.311	3.01	4.63
V	0.67	0.978	1.138	1.838	2.10	1.67
W	1.89	2.360	0.017	0.074	0.00	0.01
X	0.03	0.150	0.215	0.427	0.00	0.21
Y	0.04	1.974	1.008	0.128	0.00	0.01
Z	1.13	0.074	0.517	0.326	0.49	0.47

FIGURE 2.7 – Fréquences des caractères pour quelques langues latines

Exemple 14. Nous voulons déchiffrer ce texte en utilisant l'analyse fréquentielle, en sachant que le texte est en français.
Texte chiffré : OD FUBSWRJUDSKLH HVW XQH GLVFLSOLQH TXL SURWHJH OHV LQIRUPDWLRQV HQ OHV UHQGDQW LQFRPSUHKHQVLEOHV.

1. D'abord, on va calculer les fréquences des lettres du texte chiffré.

Lettre	**Occurrence**	**Lettre**	**Occurrence**
D	5	K	1
F	4	L	7
U	6	H	7
B	2	E	5
S	5	V	4
W	2	X	1
R	6	Q	7
J	3	N	5
G	5	I	4
A	1	C	2
K	1	P	3
L	7	O	4
H	7	M	1

TABLE 2.1 – Occurrences des lettres

2. On observe dans le texte chiffré que :
   La lettre H est très fréquente $\rightarrow$ en français, la lettre la plus fréquente est E.
3. Testons un décalage de -3 lettres :

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

TABLE 2.2 – Correspondance des lettres

4. On retrouve le texte d'origine :
   LA CRYPTOGRAPHIE EST UNE DISCIPLINE QUI PROTEGE LES INFORMATIONS EN LES RENDANT INCOMPREHENSIBLES.

2.7.2 Cryptanalyse moderne

2.7.2.1 Cryptanalyse linéaire

La cryptanalyse linéaire est une technique introduite par MATSUI en 1993 pour attaquer les chiffrements par blocs. Elle consiste à exploiter des approximations linéaires entre les bits du texte clair, du texte chiffré et de la clé secrète. [20].

2.7.2.2 Objectifs de la cryptanalyse linéaire

L'objectif principal de la cryptanalyse linéaire est d'identifier des relations linéaires biaisées entre les bits du texte clair, du texte chiffré et de la clé. Ces relations permettent de révéler des informations partielles sur la clé secrète. Plus précisément, cette méthode permet de :

— Évaluer la résistance d'un algorithme face aux attaques analytiques.

— Exploiter les biais statistiques dans les approximations linéaires.

— Réduire l'espace de recherche en éliminant les clés incompatibles avec les biais observés.

2.7.2.3 Cryptanalyse différentielle

La cryptanalyse différentielle est une technique introduite par ELI BIHAM et ADI SHAMIR au début des années 1990 pour attaquer aussi les chiffrements par blocs. Elle repose sur l'analyse des différences entre des paires de textes clairs et les différences correspondantes entre les textes chiffrés[11] .

2.7.2.4 Objectifs de la cryptanalyse différentielle

L'objectif principal de la cryptanalyse différentielle est d'étudier comment des différences spécifiques dans les entrées (textes clairs) influencent les différences dans les sorties (textes chiffrés), afin de retrouver des informations sur la clé secrète. Plus précisément, cette méthode permet de :

— Déterminer des paires de textes clairs avec une différence fixée (XORée).

— Identifier des caractéristiques différentielles qui se produisent avec une probabilité plus élevée que prévue.

— Analyser les effets de ces différences à travers les différentes étapes du chiffrement.

Ce chapitre a permis d'établir un cadre clair des notions essentielles en cryptologie et cryptographie. En présentant les différentes méthodes de chiffrement ainsi que les approches classiques et modernes de cryptanalyse, il prépare le terrain pour l'étude approfondie des techniques spécifiques et des attaques, notamment celles basées sur l'analyse différentielle, développées dans les chapitres suivants.

CHAPITRE 3

Cryptanalyse différentielle

Dans ce chapitre, nous explorons une technique puissante d'attaque cryptographique : la cryptanalyse différentielle. Elle permet d'analyser les faiblesses structurelles des chiffrements en étudiant l'évolution des différences entre paires de textes clairs et leurs chiffrés. Cette méthode est appliquée à des chiffrements bien connus comme le *SPN* afin de mieux comprendre son efficacité et ses limites.

3.1 Fonctionnement de la cryptanalyse différentielle

La cryptanalyse différentielle est une méthode qui permet d'exploiter les propriétés statistiques d'un algorithme de chiffrement pour retrouver des informations sur la clé secrète. Elle repose sur l'analyse de la propagation des différences entre paires de textes clairs à travers les différentes étapes du chiffrement.

— Choix d'une différence d'entrée : on sélectionne une différence fixée $\Delta x = x \oplus x'$ entre deux textes clairs x et x', souvent exprimée en binaire ou en hexadécimal.

— Propagation de la différence : cette différence est suivie à travers les différents tours du chiffrement, notamment les opérations de substitution (S-boîte) et de permutation. L'objectif est de déterminer les effets de cette différence sur les données intermédiaires du chiffrement.

— Caractéristiques différentielles : on identifie des *caractéristiques différentielles*, c'est-à-dire des successions de différences intermédiaires qui ont une probabilité de survenue plus élevée que la moyenne. Par exemple, certaines différences en sortie de S-boîte sont statistiquement plus fréquentes pour une différence d'entrée donnée.

— Construction de la table différentielle : pour chaque S-boîte, on établit une table listant les probabilités $\Pr[\Delta x \to \Delta y]$, c'est-à-dire la probabilité qu'une différence d'entrée Δx donne une différence de sortie Δy.

— Exploitation statistique : en générant un grand nombre de paires de textes clairs ayant la même différence Δx, on chiffre chaque paire et on observe les différences Δc des textes chiffrés. On ne conserve que les paires conformes à la caractéristique différentielle choisie.

— Deviner une sous-clé : à l'aide des paires retenues, on effectue une attaque ciblée sur les derniers tours du chiffrement. On suppose plusieurs valeurs possibles pour une partie de la sous-clé, et on mesure laquelle génère le plus de résultats cohérents avec la caractéristique.

— Identification de la clé : la sous-clé qui maximise le nombre de succès est retenue comme candidate. Cette étape peut être répétée pour reconstruire la clé complète.

3.2 Cryptanalyse différentielle d'un réseau SPN

Le réseau SPN est une structure de chiffrement par blocs susceptible d'être attaquée par cryptanalyse différentielle. Nous présentons ci-dessous le fonctionnement de cette méthode d'attaque[11].

Algorithm 9 Attaque différentielle sur un chiffrement SPN

Construire une caractéristique différentielle $(\alpha, \ldots, \beta)$ de longueur $r-1$
Calculer sa probabilité p en utilisant la DDT

Choisir un échantillon de N paires(N=C/p et C dépend du nombre de bits dans la clé à déterminer)
    $[(x_1, x'_1), \ldots, (x_N, x'_N]$ tels que $x_j \oplus x'_j = \alpha$,$j = 1, \ldots, N$
Écrire la liste de sous-clés partielles de la clé final
    $k_1, \ldots, k_m$
Pour chaque sous-clé k_i calculer
    $\chi_i = \text{card}\{j : \sigma^{-1}(c_j \oplus k_i) \oplus \sigma^{-1}(c'_j \oplus k_i) = \beta\}$

Prendre la clé qui maximise χ_i, c'est la clé estimée .

Remarque 3. Cette attaque permet uniquement de retrouver la dernière sous-clé.

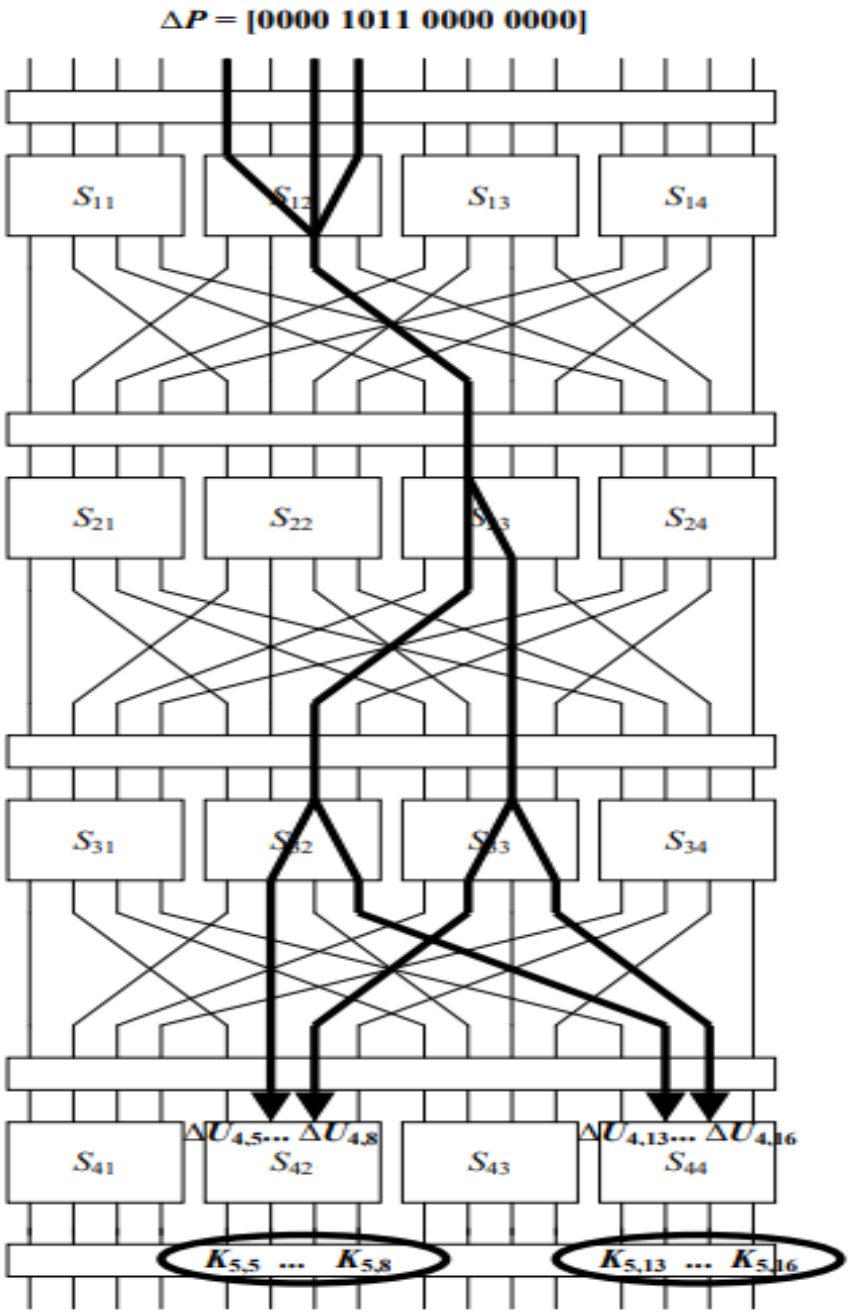

FIGURE 3.1 – Caractéristique différentielle [11]

Exemple 15. Attaque Différentielle sur un SPN à 1 tour et 3 bits :

1. La S-boîte est une fonction de substitution non-linéaire qui transforme une entrée de 3 bits en une sortie de 3 bits.(voir la table 9).
2. Analyse des paires différentielles (voir la table 11) .

$$A_7 = \{(x,x') \in \mathbb{B}^3 \times \mathbb{B}^3 : x \oplus x' = 7\}$$

Toutes les paires (x,x') tel que $x \oplus x' = 7$

Exemple de raisonnement SPN (A) :

— On choisit $x = 4 \rightarrow \texttt{100}$.

— On choisit une clé $k = 5 \rightarrow \texttt{101}$.

— $\alpha = 7 \rightarrow \texttt{111}$.

— $x' = x \oplus \alpha \rightarrow 4 \oplus 7 = 3 \rightarrow \texttt{011}$.

— Donc $x' = 3$, et $(x,x') = (4,3)$

L'attaque :

1. $(x,x') \rightarrow (y,y') \rightarrow (\sigma(y), \sigma(y'))$.
   $x \oplus x' = \alpha \rightarrow y \oplus y' = \alpha$
   car :
   $y = x \oplus k$ et $y' = x' \oplus k$ donc $y \oplus y' = x \oplus x' \oplus k \oplus k = x \oplus x' = \alpha$.
   $y \oplus y' = \alpha$
2. Nous connaissons (x,x') et $(\sigma(y) = c, \sigma(y') = c')$

$$\begin{aligned} \text{Nous avons} \sigma(y) &= \sigma(x \oplus k) \\ &= \sigma(4 \oplus 5) \\ &= \sigma(1) \\ &= 4 \\ \sigma(y') &= \sigma(x' \oplus k) \\ &= \sigma(3 \oplus 5) \\ &= \sigma(6) \\ &= 7 \\ \sigma(y) \oplus \sigma(y') &= 4 \oplus 7 \\ &= 3 \end{aligned}$$

on obtient $(\alpha, \beta) = (7,3)$.

3. On cherche les couples (x,x') tels que $x \oplus x' = 7$ et $\sigma(x) \oplus \sigma(x') = 3$.

$$\begin{aligned} \text{Nous avons} \quad A_{\alpha,\beta} &= A_{7,3} = \{(6,1),(1,6)\} \\ &= \{(y,y'),(y,y')\} = \{((x \oplus k, x' \oplus k),(x \oplus k, x' \oplus k))\} \end{aligned}$$

$$\text{alors} \quad \begin{cases} x \oplus k = 6 \\ x' \oplus k = 1 \end{cases} \quad \text{ou} \quad \begin{cases} x \oplus k = 1 \\ x' \oplus k = 6 \end{cases}$$

Ce qui donne

$$\text{Alors} \begin{cases} 4 \oplus k = 6 \to k = 2 \\ 3 \oplus k = 1 \to k = 2 \end{cases} \quad \text{où} \quad \begin{cases} 4 \oplus k = 1 \to k = 5 \\ 3 \oplus k = 6 \to k = 5 \end{cases}$$

On obtient

$$k \in \{2,5\}$$

4. On prend une nouvelle paires de (x,x') :
   — On choisit $x = 6 \to \texttt{110}$.
   — $\alpha = 2 \to \texttt{010}$.
   — $x' = x \oplus \alpha \to 6 \oplus 2 = 4 \to \texttt{100}$.
   — Donc $x' = 4$, et $(x,x') = (6,4)$
5. On calcule $\sigma(y)$ et $\sigma(y')$

$$\begin{aligned} \sigma(y) &= \sigma(x \oplus k) \\ &= \sigma(6 \oplus 5) \\ &= \sigma(3) \\ &= 1 = c \\ \sigma(y') &= \sigma(x' \oplus k) \\ &= \sigma(4 \oplus 5) \\ &= \sigma(1) \\ &= 4 = c' \\ \sigma(y) \oplus \sigma(y') &= 1 \oplus 4 \\ &= 5 \end{aligned}$$

$$\text{on obtient} \quad (\alpha,\beta) = (2,5)$$

6. On cherche les couples (x,x') tels que $x \oplus x' = 2$ et $\sigma(x) \oplus \sigma(x') = 5$.

$$\begin{aligned} A_{\alpha,\beta} = A_{2,5} &= \{(0,2),\ (1,3),\ (2,0),\ (3,1)\} \\ &= \{(y,y'),\ (y,y'),\ (y,y'),\ (y,y')\} \\ &= \{(x \oplus k,\ x' \oplus k),\ (x \oplus k,\ x' \oplus k),\ (x \oplus k,\ x' \oplus k),\ (x \oplus k,\ x' \oplus k)\} \end{aligned}$$

$$\text{ce qui donne} \begin{cases} 6 \oplus k = 0 \\ 4 \oplus k = 2 \end{cases} \quad \text{ou} \quad \begin{cases} 6 \oplus k = 1 \\ 4 \oplus k = 3k = 7 \end{cases}$$

$$\text{ou} \begin{cases} 6 \oplus k = 2 \\ 4 \oplus k = 0 \end{cases} \quad \text{ou} \quad \begin{cases} 6 \oplus k = 3 \\ 4 \oplus k = 1 \end{cases}$$

$$\text{Alors} \quad k \in \{6,7,4,5\}$$

$$\text{Comme} \quad k \in \{2,5\} \quad \text{et} \quad k \in \{6,7,4,5\} \quad \text{donc} \quad k = 5.$$

3.3 Contexte de la cryptanalyse différentielle

La cryptanalyse différentielle constitue l'une des premières méthodes d'analyse statistique puissantes visant à évaluer la sécurité des algorithmes de chiffrement symétriques. Cette technique a été introduite au début des années 1990 par Eli Biham et Adi Shamir, à travers une série de travaux devenus fondamentaux en cryptologie :

— Biham, E., & Shamir, A. (1990). *Differential Cryptanalysis of DES-like Cryptosystems*.

— Biham, E., & Shamir, A. (1991). *Differential Cryptanalysis of the Data Encryption Standard*.

Initialement développée pour analyser le DES, basé sur une structure de Feistel, la cryptanalyse différentielle s'applique bien au-delà de ce cadre. Elle est aujourd'hui un outil de référence pour l'évaluation des chiffrements par blocs, y compris ceux reposant sur une architecture de type (SPN) qui constitue précisément le cadre de ce projet.

Dans le contexte des réseaux SPN, cette attaque permet d'exploiter la manière dont des différences contrôlées à l'entrée peuvent traverser les couches de substitution et de permutation, révélant ainsi des modèles statistiques exploitables pour deviner une partie de la clé secrète.

Ce chapitre a présenté les principes fondamentaux de la cryptanalyse différentielle ainsi que son application sur différents cryptosystèmes comme le SPN . En mettant en évidence les relations entre les différences d'entrée et de sortie, cette approche constitue une base solide pour l'évaluation de la robustesse des algorithmes de chiffrement symétrique.

CHAPITRE 4

Résultats et discussions

Ce chapitre présente l'application d'une attaque différentielle sur un chiffrement SPN 12 bits. À partir de caractéristiques différentielles à forte probabilité, nous générons des paires de textes clairs spécifiques et analysons leur propagation à travers le réseau. L'objectif est de retrouver la clé finale en testant différentes hypothèses et en observant les différences de sortie. Les résultats expérimentaux permettent d'évaluer l'efficacité de l'attaque en fonction du nombre de paires utilisées.

4.1 Outils de programmation

L'implémentation de ce projet a été effectuée en Python, un langage de programmation de haut niveau, interprété, dynamique et orienté objet. Python est aujourd'hui l'un des langages les plus utilisés dans le domaine scientifique et technique, notamment en cryptographie, en traitement des données et en simulation d'algorithmes [25].

Ce choix s'explique par plusieurs raisons :

- Sa **syntaxe simple et lisible**, qui facilite la compréhension du code et la rapidité de développement ;
- Son **écosystème riche** en bibliothèques spécialisées, adapté aux besoins de l'analyse et du traitement de données ;
- Sa **portabilité** et sa **grande communauté**, garantissant un bon support technique et une documentation abondante.

FIGURE 4.1 – Logo Python

Pour développer ce projet, nous avons utilisé Jupyterlab, un environnement interactif particulièrement pratique pour écrire, tester et organiser du code. Il permet de créer des *notebooks*, où l'on peut combiner du texte, du code Python, des graphiques et même des visualisations interactives.

JupyterLab est très apprécié dans le monde de la recherche et de la science des données, car il offre une interface claire et simple d'utilisation, tout en proposant des fonctionnalités puissantes. Parmi celles-ci, on retrouve [25]

- l'exécution du code bloc par bloc (cellule par cellule) .
- la coloration syntaxique .
- la complétion automatique .
- l'affichage direct des résultats (graphiques, tableaux, etc.) .
- une bonne intégration avec la ligne de commande.

FIGURE 4.2 – Logo JupyterLab

Dans le cadre de ce projet, une seule bibliothèque standard de Python a été utilisée : `random`.

Elle permet de générer des nombres pseudo-aléatoires, ce qui est essentiel pour la création de valeurs d'entrée aléatoires (comme des paires de textes pour une attaque différentielle ou des clés de chiffrement). Cette bibliothèque a été suffisante pour simuler les comportements aléatoires nécessaires à l'implémentation.

4.2 Méthode de la cryptanalyse différentielle

La cryptanalyse différentielle est une technique d'analyse statistique puissante qui exploite la manière dont des différences spécifiques entre paires de textes clairs influencent les différences observées entre les textes chiffrés correspondants. En étudiant ces corrélations, l'attaquant peut identifier des biais dans la propagation des différences et en déduire des informations sur les clés utilisées.

Le principe fondamental consiste à introduire une différence contrôlée Δx entre deux textes clairs x et x', puis à suivre l'évolution de cette différence tout au long des transformations du chiffrement. Certaines différences à la sortie $\Delta c = c \oplus c'$ apparaissent avec une probabilité significativement plus élevée que d'autres, révélant des caractéristiques internes exploitables du chiffrement.

Les étapes typiques de l'attaque différentielle sont les suivantes :

1. Choix d'une différence d'entrée Δx, soigneusement sélectionnée en fonction des propriétés de la S-boîte.
2. Génération de nombreuses paires de textes clairs (x, x') vérifiant $\Delta x = x \oplus x'$.
3. Chiffrement des paires sous une même clé secrète pour obtenir (c, c').
4. Calcul des différences de sortie $\Delta c = c \oplus c'$.
5. Analyse statistique des Δc obtenues pour identifier les sorties les plus probables.
6. Formulation d'hypothèses sur la sous-clé de la dernière tour, en se basant sur les transitions les plus fréquentes.
7. Validation des hypothèses par la vérification de leur cohérence statistique sur l'ensemble des paires.

4.3 Application de la cryptanalyse différentielle sur un réseau SPN

Dans ce projet, nous avons appliqué la cryptanalyse différentielle à un algorithme de chiffrement symétrique structuré selon un réseau SPN . Ce type d'architecture est couramment utilisé dans les chiffrements modernes en raison de sa simplicité conceptuelle et de sa capacité à assurer une bonne diffusion et confusion, éléments essentiels pour garantir la sécurité cryptographique.

Le chiffrement SPN étudié repose sur les caractéristiques suivantes :

— Une taille de bloc de 12 bits, divisée en 3 sous-blocs de 4 bits chacun.

— Une structure composée de 2 rondes successives.

— Chaque tour inclut :

  — une substitution non linéaire via une S-boîte

  — une permutation fixe sur les 12bits pour assurer la diffusion

  — une opération XOR avec une sous-clé tour.

— L'algorithme utilise au total 3 sous-clés : une pour chaque ronde.

4.3.1 Mise en œuvre de l'attaque

L'attaque différentielle a été menée en plusieurs étapes :

1. **Étude de la S-boîte** : une table différentielle a été construite afin d'identifier les différences d'entrée donnant lieu à des différences de sortie avec une probabilité élevée. Cela permet de cibler les caractéristiques différentielles exploitables.
2. **Sélection d'une différence d'entrée** Δx : cette différence a été choisie pour maximiser la probabilité d'une propagation contrôlée des différences à travers les premières rondes du chiffrement.
3. **Génération de paires de textes clairs** : des paires de textes (x, x') ont été générées, vérifiant toutes la condition $\Delta x = x \oplus x'$.
4. **Chiffrement des paires** : chaque paire a été chiffrée avec les mêmes clé secrète à travers les 2 tour du SPN.
5. **Analyse des sorties** : les différences de sortie $c \oplus c'$ ont été examinées pour détecter celles qui apparaissent avec des fréquences anormales, révélant un biais exploitable.
6. **Deviner la sous-clé finale** : en se concentrant sur la dernière ronde, différentes valeurs de sous-clé ont été testées pour décrypter partiellement les sorties, et la valeur menant aux correspondances les plus fréquentes a été retenue comme hypothèse plausible.

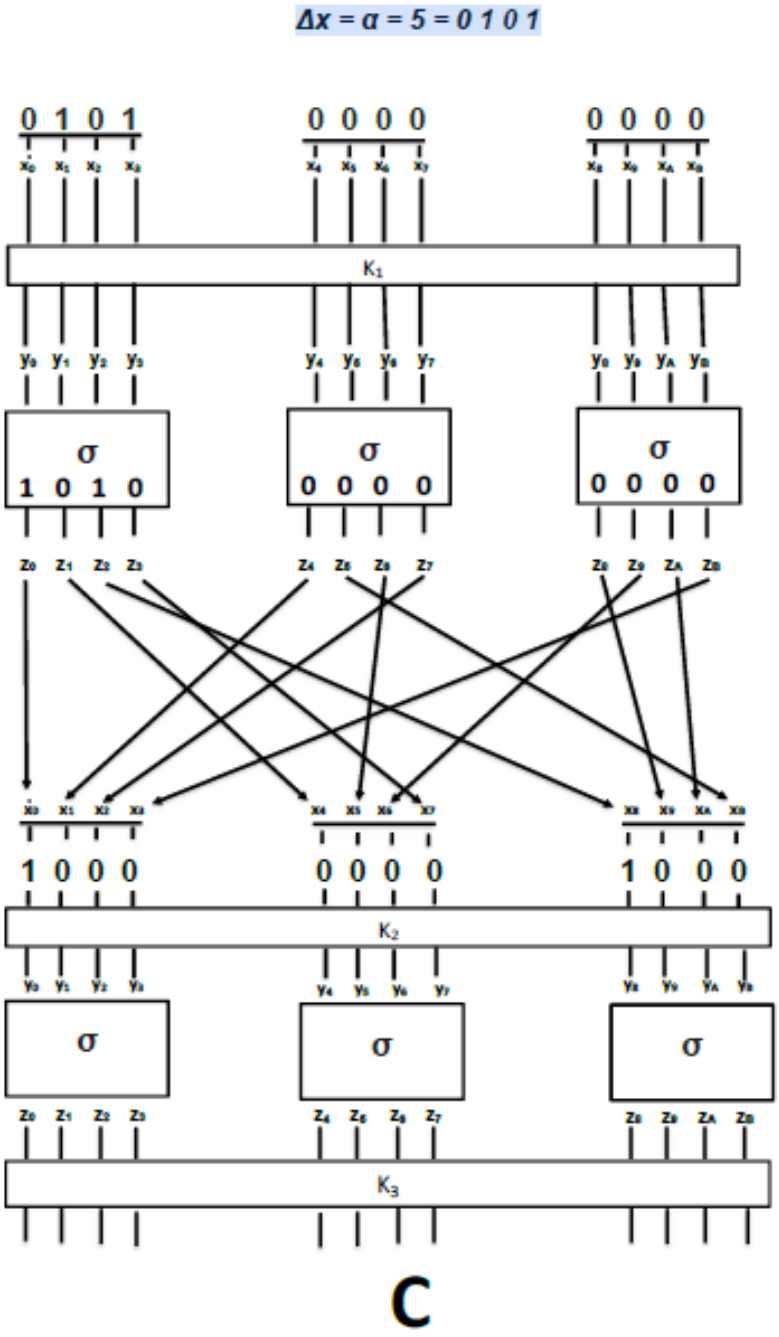

FIGURE 4.3 – Caractéristique différentielle

La différence d'entrée $\Delta x = 0101$ (en binaire) est injectée dans le premier tour du réseau SPN. Après le premier XOR avec la clé k_1, les blocs passent par les boîtes de substitution s_i, donnant une différence intermédiaire $\Delta u = 1010$.

Cette différence est ensuite diffusée par la permutation linéaire, produisant une nouvelle différence $\Delta x_1 = 100000001000$. Un second tour de substitution est appliqué avec s_i.

La probabilité associée à cette transition différentielle est d'environ $\frac{1}{2}$.

4.4 Implémentation et résultats

Dans cette section, nous proposons un cas d'étude qui illustre de manière concrète l'implémentation et le fonctionnement de notre algorithme.

L'objectif est de simuler le processus de chiffrement à l'aide d'un réseau SPN, puis d'appliquer une attaque différentielle dans le but d'extraire des informations sur les sous-clés utilisées au cours du chiffrement.

Définition de la S-boîte

Ce code Python construit automatiquement la table de vérité du XOR, les transitions pour chaque différence d'entrée α, et la table des différences (DDT)de la S-boîte (voir la table 13) :

```
# Definition de la S-bo\^ite
sbox = [0xD, 0xB, 0x3, 0x1, 0x6, 0x7, 0xE, 0x4,
       0x2, 0xC, 0x0, 0xF, 0x9, 0x8, 0x5, 0xA]
```

Construction de la table des différences (DDT)

Le code suivant implémente la construction de la DDT en parcourant toutes les paires $(x, x \oplus \Delta x)$ possibles et en comptant les occurrences de chaque différence de sortie β .

Ce code constitue un extrait de l'implémentation complète réalisée dans le cadre de notre étude sur la cryptanalyse différentielle.

```
# 3. Generation de la DDT
def generer_ddt(sbox):
    taille = len(sbox)
    ddt = [[0 for _ in range(taille)] for _ in range(taille)]

    for delta_x in range(taille):
        for x in range(taille):
            x_prime = x ^ delta_x
            delta_y = sbox[x] ^ sbox[x_prime]
            ddt[delta_x][delta_y] += 1

    return ddt
```

La DDT (voir la table 24) est une matrice de taille 16×16 où chaque ligne correspond à une différence d'entrée Δx, et chaque colonne à une différence de sortie Δy. Les valeurs dans la matrice indiquent le nombre d'occurrences où $\sigma(x) \oplus \sigma(x \oplus \Delta x) = \Delta y$, pour tous les x possibles.

Construction d'une caractéristique différentielle

Dans le cadre de l'analyse différentielle appliquée à notre chiffre SPN sur 12 bits, nous avons construit une caractéristique différentielle sur un seul tour, en utilisant une différence d'entrée bien définie.

La différence choisie est :

$$\Delta X_0 = \texttt{0x500} = 0101\ 0000\ 0000$$

Cette différence affecte uniquement le bloc de poids fort (bloc 2), tandis que les deux autres blocs restent inchangés (différence nulle). Cette configuration permet de concentrer l'analyse sur un seul bloc actif à l'entrée, ce qui facilite l'observation du comportement de la S-boîte et de la permutation lors de la propagation de la différence.

Lors de l'application de la différence d'entrée $\Delta X_0 = \texttt{0x500}$ à travers la S-boîte, la sortie obtenue est :

$$\Delta u_1 = \texttt{0xA00}$$

```
 # Definition de la permutation
PERM = [0x0, 0x4, 0x7, 0xB, 0x1, 0x6, 0x9, 0x3, 0x2, 0x8, 0xA, 0x5]
```

Ensuite, après l'application de la permutation, la nouvelle différence en sortie est donnée dans la table 14 :

$$\Delta x_1 = \texttt{0x808}$$

Cette valeur indique que deux blocs sont désormais actifs :

- Le bloc de poids fort (bloc 2) ;
- Le bloc de poids faible (bloc 0) ;

Tandis que le bloc du milieu (bloc 1) est resté inactif.

Probabilité de la caractéristique

Lors de l'application de la S-boîte à la différence $\Delta x_0 = \texttt{0x500}$, nous observons les comportements suivants :

- Les blocs inactifs (valeur d'entrée égale à 0) produisent des différences de sortie nulles avec une probabilité de 1.
- Le bloc actif (bloc 2), avec une différence d'entrée $\Delta = \texttt{0x5}$, génère une sortie $\Delta = \texttt{0xA}$ avec une probabilité de $\frac{8}{16} = 0{,}5$, selon la DDT de la S-boîte.
  - Bloc 0 : $\Delta x_0 = 0000 \Rightarrow \Delta y_0 = 0000$ (probabilité = 1)
  - Bloc 1 : $\Delta x_1 = 0000 \Rightarrow \Delta y_1 = 0000$ (probabilité = 1)
  - Bloc 2 : $\Delta x_2 = 0101 \Rightarrow \Delta y_2 = 1010$ (probabilité = 0.5)

En multipliant les probabilités associées à chaque bloc, on obtient la probabilité totale de cette caractéristique différentielle sur un tour :

$$p = 1 \times 1 \times \frac{1}{2} = \frac{1}{2}$$

Cette valeur exprime la probabilité que la différence d'entrée ΔX_0 soit transformée en une différence $\Delta X_1 = \texttt{0x808}$ après un tour complet (substitution + permutation).

```
🔎 Construction  de la caractéristique différentielle:
ΔX0 = 500 (010100000000)
🔸 S-box differences :
  Bloc 0: Δ_in = 0000 → Δ_out = 0000 (proba = 1)
  Bloc 1: Δ_in = 0000 → Δ_out = 0000 (proba = 1)
  Bloc 2: Δ_in = 0101 → Δ_out = 1010 (proba = 0.50)
ΔU1 = A00 (101000000000)
ΔX1 = 808 (100000001000)

✅ Probabilité totale : 0.500000 (≈ 1/2)
```

FIGURE 4.4 – La caractéristique différentielle

La figure illustre l'exécution de la première ronde du chiffrement SPN avec une différence d'entrée fixée. Elle montre la propagation de cette différence à travers la S-boîte, puis après permutation, ainsi que les valeurs binaires et hexadécimales associées. Cette représentation visuelle permet de vérifier que la caractéristique différentielle attendue est bien respectée et que les calculs de probabilité sont cohérents avec la table de différences.

Génération et chiffrement de paires différentielles

L'objectif de cette étape est de générer automatiquement des paires de textes clairs (x, x') telles que $x' = x \oplus \alpha$, avec une différence d'entrée fixée à $\alpha = 0x500$. Ces paires sont ensuite chiffrées à l'aide d'un réseau SPN sur 12 bits, afin d'analyser les différences obtenues en sortie (c, c') et de repérer les paires qui suivent la caractéristique différentielle attendue.

Dans le cadre de cette attaque, et selon les principes de la cryptanalyse différentielle, le nombre de paires nécessaires est estimé par la formule :

$$N = C \cdot \frac{1}{p}$$

où p représente la probabilité de la caractéristique différentielle (ici $p = \frac{1}{2}$), et C est une constante empirique . En prenant $C = 8$, on obtient donc :

$$N = 8 \cdot \frac{1}{\frac{1}{2}} = 16$$

Ainsi, 16 paires de textes clairs sont générées (x, x')

Le chiffrement de chaque paire (x, x') est effectué à l'aide d'un réseau SPN utilisant les clés suivantes, fixées manuellement

$$k_1 = [\mathtt{B}, 1, 5], \quad k_2 = [3, 2, 8], \quad k_3 = [\mathtt{A}, \mathtt{E}, 4]$$

Les couples chiffrés (c, c') sont ensuite comparés pour vérifier si la différence de sortie est bien conforme à celle prédite par la caractéristique différentielle (par exemple $\beta = \mathtt{0x808}$), ce qui permet d'identifier les paires valides à conserver pour l'attaque.

Les résultats ci-dessous présentent les paires (x, x') générées ainsi que leurs textes chiffrés (c, c'), obtenus après application du chiffrement SPN sur 12 bits.

N	x	x'	c	c'
0	E81	B81	E25	02B
1	C41	941	17B	195
2	0E9	5E9	F42	44D
3	273	773	07E	09E
4	795	295	A9D	A7D
5	EDD	BDD	D17	214
6	DFF	8FF	6EF	1F8
7	67F	37F	008	E0F
8	681	381	03B	E35
9	005	505	362	C6D
10	42B	12B	839	736
11	D09	809	3B2	CAD
12	B65	E65	A2C	923
13	03C	53C	662	16D
14	84F	D4F	1A8	6BF
15	BEF	EEF	046	E49

FIGURE 4.5 – Paires générées

Chaque paire (x, x') est ensuite chiffrée indépendamment à l'aide de l'algorithme SPN. Celui-ci applique successivement plusieurs transformations sur 12 bits répartis en 3 blocs de 4 bits :

- **XOR avec une sous-clé** : chaque bloc est combiné à une sous-clé par une opération $\oplus$ bit à bit .
- **Substitution (S-boîte)** : chaque bloc de 4 bits subit une substitution non-linéaire selon une table S-boîte fixe .
- **Permutation** : les bits de l'état intermédiaire sont réorganisés selon une permutation prédéfinie.

Ces opérations sont répétées pour les deux premiers tours. Le troisième (et dernier) tour se termine par une substitution suivie d'un dernier XOR avec la clé finale, sans permutation finale. On obtient ainsi deux textes chiffrés c et c', respectivement pour x et x'.

Attaque différentielle

Pour chaque clé partielle candidate , on construit une clé de la forme $(a_i, 0, b_i)$, en supposant que seuls les blocs de poids fort et de poids faible sont actifs. L'objectif est alors de vérifier si cette clé respecte la caractéristique différentielle précédemment établie.

Pour cela, on applique un processus de test à chaque paire de sorties chiffrées (c, c') :

- On simule l'annulation de la dernière sous-clé en calculant $c \oplus k$ et $c' \oplus k$.
- On applique l'inverse de la S-boîte sur les résultats obtenus.
- On calcule la différence suivante :

$$\Delta u = \sigma^{-1}(c \oplus k) \oplus \sigma^{-1}(c' \oplus k)$$

```
    for c, c_prime in pairs_y:
              uy = inv_sbox_on_three_nibbles(c ^ k)
              uyp = inv_sbox_on_three_nibbles(c_prime ^ k)
              diff = uy ^ uyp

              if diff == beta:
                  count += 1

          counters[(ai, bi)] = count
```

- Si cette différence Δu est égale à la différence attendue $\beta = \texttt{0x808}$, un compteur associé à la clé partielle est incrémenté.

Ce test est réalisé pour l'ensemble des $2^8 = 256$ clés partielles possibles et pour toutes les paires de données chiffrées.

Étant donné que la probabilité de propagation de la différence est égale à $1/2$, la bonne clé partielle est censée produire le résultat attendu dans environ 50 % des cas. Elle se distingue ainsi par un nombre de succès ("hits") significativement plus élevé que les autres, permettant son identification comme composante probable de la clé secrète.

Afin d'évaluer l'efficacité de l'attaque différentielle en fonction du nombre de paires disponibles, des tests ont été réalisés avec différentes tailles d'échantillons : 2, 4, 6, 8, 10, etc. L'objectif était d'observer à partir de quelle quantité minimale de données la clé correcte commence à émerger de façon significative parmi les candidats. Les résultats ont montré que, pour de faibles volumes de paires (inférieurs à 10), il est difficile de distinguer clairement la bonne clé partielle. En effet, plusieurs combinaisons (a,b) obtiennent un nombre de paires conformes (ou "compteur") très proche, voire identique, ce qui rend la prise de décision incertaine à ce stade.

Ce n'est qu'à partir de 11 paires que la clé partielle correcte $(A,0,4)$ commence à se détacher nettement du reste. Elle atteint alors un compteur de 6, ce qui représente plus de la moitié des paires utilisées. Cette observation montre que, même avec un petit échantillon, l'attaque peut déjà donner des indices pertinents sur la clé, à condition de dépasser un certain seuil. Ce comportement est cohérent avec la probabilité théorique de la caractéristique différentielle utilisée (environ $1/2$), qui ne devient statistiquement visible qu'après un nombre suffisant d'occurrences.

À l'issue de l'attaque différentielle menée à partir de 16 paires de textes clairs (x,x') avec une différence d'entrée $\alpha = 0x500$, les sorties chiffrées (c,c') ont été analysées afin de tester toutes les clés partielles de la forme $(a,0,b)$. Pour chaque clé partielle, on a compté combien

de paires vérifiaient la différence de sortie cible $\beta = 0x808$ après application de l'inverse de la S-boîte.

La figure ci-dessus présente les 16 meilleures clés candidates, triées par ordre décroissant de leurs compteurs, c'est-à-dire de paires conformes à la différence attendue. La clé partielle $(A,0,4)$ se démarque nettement avec un score de 7, ce qui représente presque 50% des paires utilisées — un résultat en parfaite cohérence avec la probabilité théorique de la caractéristique différentielle, estimée à $1/2$.

```
🧠 Top 10 clés (a_i, b_i) avec le plus grand nombre de hits :
Clé partielle: a=A, b=4 → (a,0,b) = (A,0,4) | compteur = 7
Clé partielle: a=A, b=F → (a,0,b) = (A,0,F) | compteur = 5
Clé partielle: a=4, b=4 → (a,0,b) = (4,0,4) | compteur = 3
Clé partielle: a=A, b=1 → (a,0,b) = (A,0,1) | compteur = 3
Clé partielle: a=A, b=A → (a,0,b) = (A,0,A) | compteur = 3
Clé partielle: a=F, b=4 → (a,0,b) = (F,0,4) | compteur = 3
Clé partielle: a=0, b=4 → (a,0,b) = (0,0,4) | compteur = 2
Clé partielle: a=1, b=4 → (a,0,b) = (1,0,4) | compteur = 2
Clé partielle: a=1, b=F → (a,0,b) = (1,0,F) | compteur = 2
Clé partielle: a=5, b=4 → (a,0,b) = (5,0,4) | compteur = 2
Clé partielle: a=B, b=4 → (a,0,b) = (B,0,4) | compteur = 2
Clé partielle: a=D, b=1 → (a,0,b) = (D,0,1) | compteur = 2
Clé partielle: a=D, b=4 → (a,0,b) = (D,0,4) | compteur = 2
Clé partielle: a=D, b=A → (a,0,b) = (D,0,A) | compteur = 2
Clé partielle: a=D, b=F → (a,0,b) = (D,0,F) | compteur = 2
Clé partielle: a=0, b=3 → (a,0,b) = (0,0,3) | compteur = 1
```

FIGURE 4.6 – Résultats des clés partielles

Ces résultats confirment que l'attaque différentielle permet bien d'identifier la sous-clé finale en exploitant la propagation différentielle dans le dernier tour du chiffrement SPN. Les autres clés (comme (A,F) ou $(4,4)$) montrent quelques occurrences, mais avec des scores nettement inférieurs, ce qui renforce la fiabilité du candidat principal.

En augmentant le nombre de paires (x,x') à 1000, la clé partielle correcte $(A,0,4)$ ressort nettement avec un compteur de 499, soit près de la moitié des paires utilisées. Ce résultat confirme l'adéquation de cette clé avec la caractéristique différentielle choisie, dont la probabilité était de $1/2$.

```
🧠 Top 10 clés (a_i, b_i) avec le plus grand nombre de hits :
Clé partielle: a=A, b=4 → (a,0,b) = (A,0,4) | compteur = 499
Clé partielle: a=1, b=4 → (a,0,b) = (1,0,4) | compteur = 306
Clé partielle: a=A, b=F → (a,0,b) = (A,0,F) | compteur = 294
Clé partielle: a=4, b=4 → (a,0,b) = (4,0,4) | compteur = 255
Clé partielle: a=F, b=4 → (a,0,b) = (F,0,4) | compteur = 253
Clé partielle: a=A, b=A → (a,0,b) = (A,0,A) | compteur = 244
Clé partielle: a=A, b=1 → (a,0,b) = (A,0,1) | compteur = 238
Clé partielle: a=A, b=B → (a,0,b) = (A,0,B) | compteur = 201
Clé partielle: a=5, b=4 → (a,0,b) = (5,0,4) | compteur = 192
Clé partielle: a=1, b=F → (a,0,b) = (1,0,F) | compteur = 187
Clé partielle: a=1, b=A → (a,0,b) = (1,0,A) | compteur = 152
Clé partielle: a=F, b=F → (a,0,b) = (F,0,F) | compteur = 148
Clé partielle: a=4, b=F → (a,0,b) = (4,0,F) | compteur = 146
Clé partielle: a=1, b=1 → (a,0,b) = (1,0,1) | compteur = 141
Clé partielle: a=A, b=5 → (a,0,b) = (A,0,5) | compteur = 140
Clé partielle: a=A, b=E → (a,0,b) = (A,0,E) | compteur = 134
```

FIGURE 4.7 – Résultats pour 1000 paires

Après avoir choisi une nouvelle différence en entrée $\alpha = 0x800$, on obtient, après une analyse du chiffrement, une différence en sortie $\beta = 0x898$. En consultant la table DDT , on observe que cette transition possède une probabilité de 0,375. L'attaque alors été menée sur un ensemble de 21 paires de textes clairs (x, x').
Contrairement à la configuration précédente où seule une clé partielle $(a, 0, b)$ était visée, l'objectif ici est de retrouver la sous-clé complète $k_3 = (a, b, c)$ en testant l'ensemble des 4096 clés possibles sur 12 bits.

```
🔎 Construction automatique de la caractéristique différentielle sur 1 tour(s) :
ΔX0 = 800 (100000000000)
🔸 S-box differences :
  Bloc 0: Δ_in = 0000 → Δ_out = 0000 (proba = 1)
  Bloc 1: Δ_in = 0000 → Δ_out = 0000 (proba = 1)
  Bloc 2: Δ_in = 1000 → Δ_out = 1111 (proba = 0.38)
ΔU1 = F00 (111100000000)
ΔX1 = 898 (100010011000)

✅ Probabilité totale : 0.375000
```

FIGURE 4.8 – Nouvelle caractéristique

Le tableau ci-dessus présente les 20 meilleures clés candidates, triées par ordre décroissant du nombre d'occurrences, c'est-à-dire le nombre de paires (y, y') pour lesquelles la différence obtenue après l'inverse S-boîte correspondait à la différence attendue $\beta = 0x898$.

🎯 Les 20 meilleures clés candidates :

Clé (a,b,c)	Occurrences
(A,E,4)	9
(A,E,A)	6
(A,E,F)	6
(5,E,4)	5
(A,2,4)	5
(A,2,F)	5
(A,E,1)	5
(A,F,4)	5
(0,E,4)	4
(1,E,4)	4
(4,E,4)	4
(A,0,4)	4
(A,1,4)	4
(A,F,A)	4
(B,E,4)	4
(F,E,4)	4
(1,E,5)	3
(1,E,A)	3
(1,E,B)	3

FIGURE 4.9 – Clé partielle complète

On observe que la clé $(A,E,4)$ se distingue nettement avec 9 occurrences sur 21, ce qui la place en tête des candidats. Les autres clés (par exemple (A,E,A), (A,E,F) ou $(5,E,4)$) présentent des scores plus faibles, renforçant la probabilité que la clé $(A,E,4)$ soit la bonne.

Ce chapitre a présenté l'étude d'un chiffrement SPN sur 12 bits et sa vulnérabilité face à une attaque différentielle. Après avoir implémenté le chiffrement et identifié des caractéristiques différentielles probables, nous avons généré des paires différentielles et analysé les sorties pour retrouver la sous-clé finale.

Les résultats obtenus montrent que, même avec un petit nombre de paires, l'attaque permet d'identifier efficacement la bonne clé, en accord avec la probabilité théorique de la caractéristique choisie. Cette expérience confirme l'intérêt de l'analyse différentielle pour évaluer la sécurité des algorithmes de chiffrement.

Conclusion

Ce mémoire a été conçu en réponse à une réalité simple mais persistante : malgré les avancées considérables réalisées en cryptanalyse depuis plusieurs décennies, les ressources pédagogiques accessibles sur les méthodes modernes et en particulier sur la cryptanalyse différentielle restent rares, fragmentées, voire obsolètes. Ce constat, partagé par plusieurs chercheurs et enseignants du domaine, a motivé notre volonté d'explorer plus en profondeur cette branche essentielle de la cryptologie.

Nous avons ainsi parcouru les bases arithmétiques nécessaires, puis rappelé les fondements de la cryptographie et de la cryptanalyse avant de nous focaliser sur la cryptanalyse différentielle, tant dans sa logique théorique que dans ses applications concrètes. Cette technique, devenue incontournable face aux algorithmes de chiffrement modernes, constitue aujourd'hui l'un des outils d'analyse les plus puissants contre les chiffrements par blocs.

L'objectif principal de ce travail n'a pas été seulement de présenter des attaques, mais de mieux comprendre la structure même des algorithmes ciblés, leur conception, leurs failles, et surtout les principes communs à la majorité des approches modernes en cryptanalyse. En ce sens, la cryptanalyse différentielle ne se limite pas à une attaque ponctuelle : elle incarne un cadre d'analyse plus général, inspirant de nombreuses autres méthodes. À l'heure où les systèmes cryptographiques deviennent de plus en plus complexes, et les attaques plus subtiles, la compréhension des mécanismes internes de la cryptographie reste indispensable.

La cryptanalyse, loin d'être un simple outil de cassure, est aujourd'hui un instrument de progrès, qui contribue à la conception de systèmes plus sûrs et plus robustes. C'est dans cet esprit que s'inscrit notre démarche apprendre à attaquer, pour mieux protéger.

Dans la dernière partie de ce mémoire, nous avons construit un réseau SPN afin d'appliquer concrètement la cryptanalyse différentielle. Nous avons défini une S-boîte 4x4, une permutation sur 12 bits, et choisi trois clés. À partir de la table des différences de la S-boîte, nous avons sélectionné une caractéristique différentielle avec une probabilité suffisamment élevée. Ce choix nous a permis de déterminer le nombre de paires différentielles (x, x') à générer (avec les (c, c')

chiffrés corresponds).

Dans un premier temps, nous avons choisie une caractéristique qui nous a permis de déterminer deux blocs de la troisième clé. Dans un second temps, en modifiant la caractéristique, nous avons pu aller plus loin et retrouver les trois blocs, c'est-à-dire la troisième clé complète.Ce cas pratique nous a permis de mieux comprendre toutes les étapes de l'attaque.

Annexes 1 : Permutation initiale

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

TABLE 1 – Permutation initiale IP

Annexes 2 : Permutation inverse

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

TABLE 2 – Permutation IP^{-1}

Annexes 3 : Permutationde choix 1

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

TABLE 3 – PC_1

Annexes 4 : Permutation de choix 2

14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

TABLE 4 – PC_2

Annexes 5 : Table de leftshift

Tour	Nombre de positions de leftshift
1	1
2	1
3	2
4	2
5	2
6	2
7	2
8	2
9	1
10	2
11	2
12	2
13	2
14	2
15	2
16	1

TABLE 5 – Table de leftshift

Annexes 6 : Expansion

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

TABLE 6 – Expansion E

Annexes 7 : les 8 S-boîte

S1																
L/C	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
S2																
0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
2	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
3	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
S3																
0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
2	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
3	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
S4																
0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
2	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14
S5																
0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
2	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
3	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3
S6																
0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
2	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
3	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13
S7																
0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
2	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
3	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12
S8																
0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
2	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
3	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

TABLE 7 – Les 8 S-boîte

Annexes 8 : Permutation π

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

TABLE 8 – Permutation π

Annexes 9 : S-boîte 3 bits

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

TABLE 9 – S-Boîte

Annexes 10 : Table XOR pour 3 bits

$\oplus$	0 (000)	1 (001)	2 (010)	3 (011)	4 (100)	5 (101)	6 (110)	7 (111)
0 (000)	0	1	2	3	4	5	6	7
1 (001)	1	0	3	2	5	4	7	6
2 (010)	2	3	0	1	6	7	4	5
3 (011)	3	2	1	0	7	6	5	4
4 (100)	4	5	6	7	0	1	2	3
5 (101)	5	4	7	6	1	0	3	2
6 (110)	6	7	4	5	2	3	0	1
7 (111)	7	6	5	4	3	2	1	0

TABLE 10 – Tableau XOR (en décimal)

Annexes 11 : Table des différences pour $\alpha = 7$

x	$x \oplus 7$	$\sigma(x \oplus 7)$	$\sigma(x)$	$\sigma(x) \oplus \sigma(x \oplus 7)$
0	7	2	5	7
1	6	7	4	3
2	5	6	0	6
3	4	3	1	2
4	3	1	3	2
5	2	0	6	6
6	1	4	7	3
7	0	5	2	7

TABLE 11 – $\alpha = 7$

Annexes 12 : Table des différences pour 3 bits

α \ β	0	1	2	3	4	5	6	7
0	8	0	0	0	0	0	0	0
1	0	4	0	0	0	4	0	0
2	0	0	0	0	4	4	×	×
3	×	4	×	×	4	×	×	×
4	×	×	2	2	×	×	2	2
5	×	×	2	2	×	×	2	2
6	×	×	2	2	×	×	2	2
7	×	×	2	2	×	×	2	2

TABLE 12 – Table des différences

Annexes 13 : S-boîte utilisée dans le chiffrement SPN

i	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$\sigma(i)$	D	B	3	1	6	7	E	4	2	C	0	F	9	8	5	A

TABLE 13 – S-boîte utilisée dans le chiffrement SPN

Annexes 14 : Table de permutation utilisée dans le chiffrement SPN

i	0	1	2	3	4	5	6	7	8	9	A	B
$\pi(i)$	0	4	7	B	1	6	9	3	2	8	A	5

TABLE 14 – Table de permutation utilisée dans le chiffrement SPN

Annexes 15 : Table XOR

$\oplus$	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1	1	0	3	2	5	4	7	6	9	8	B	A	D	C	F	E
2	2	3	0	1	6	7	4	5	A	B	8	9	E	F	C	D
3	3	2	1	0	7	6	5	4	B	A	9	8	F	E	D	C
4	4	5	6	7	0	1	2	3	C	D	E	F	8	9	A	B
5	5	4	7	6	1	0	3	2	D	C	F	E	9	8	B	A
6	6	7	4	5	2	3	0	1	E	F	C	D	A	B	8	9
7	7	6	5	4	3	2	1	0	F	E	D	C	B	A	9	8
8	8	9	A	B	C	D	E	F	0	1	2	3	4	5	6	7
9	9	8	B	A	D	C	F	E	1	0	3	2	5	4	7	6
A	A	B	8	9	E	F	C	D	2	3	0	1	6	7	4	5
B	B	A	9	8	F	E	D	C	3	2	1	0	7	6	5	4
C	C	D	E	F	8	9	A	B	4	5	6	7	0	1	2	3
D	D	C	F	E	9	8	B	A	5	4	7	6	1	0	3	2
E	E	F	C	D	A	B	8	9	6	7	4	5	2	3	0	1
F	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0

TABLE 15 – Table XOR

Annexes 16 : Analyse des transitions pour plusieurs différences d'entrée α

$\alpha = 0$					$\alpha = 1$				
x	$x \oplus 0$	$\sigma(x)$	$\sigma(x \oplus 0)$	$\sigma(x) \oplus \sigma(x \oplus 0)$	x	$x \oplus 1$	$\sigma(x)$	$\sigma(x \oplus 1)$	$\sigma(x) \oplus \sigma(x \oplus 1)$
0	0	D	D	0	0	1	D	B	6
1	1	B	B	0	1	0	B	D	6
2	2	3	3	0	2	3	3	1	2
3	3	1	1	0	3	2	1	3	2
4	4	6	6	0	4	5	6	7	1
5	5	7	7	0	5	4	7	6	1
6	6	E	E	0	6	7	E	4	A
7	7	4	4	0	7	6	4	E	A
8	8	2	2	0	8	9	2	C	E
9	9	C	C	0	9	8	C	2	E
A	A	0	0	0	A	B	0	F	F
B	B	F	F	0	B	A	F	0	F
C	C	9	9	0	C	D	9	8	1
D	D	8	8	0	D	C	8	9	1
E	E	5	5	0	E	F	5	A	F
F	F	A	A	0	F	E	A	5	F

TABLE 16 – Tables des différences — $\alpha = 0$ et $\alpha = 1$

$\alpha = 2$					$\alpha = 3$				
x	$x \oplus 2$	$\sigma(x)$	$\sigma(x \oplus 2)$	$\sigma(x) \oplus \sigma(x \oplus 2)$	x	$x \oplus \alpha$	$\sigma(x)$	$\sigma(x \oplus \alpha)$	$\sigma(x) \oplus \sigma(x \oplus 2)$
0	2	D	3	E	0	3	D	1	C
1	3	B	1	A	1	2	B	3	8
2	0	3	D	E	2	1	3	B	8
3	1	1	B	A	3	0	1	D	C
4	6	6	E	8	4	7	6	4	2
5	7	7	4	3	5	6	7	E	9
6	4	E	6	8	6	5	E	7	9
7	5	4	7	3	7	4	4	6	2
8	A	2	0	2	8	B	2	F	D
9	B	C	F	3	9	A	C	0	C
A	8	0	2	2	A	9	0	C	C
B	9	F	C	3	B	8	F	2	D
C	E	9	5	C	C	F	9	A	3
D	F	8	A	2	D	E	8	5	D
E	C	5	9	C	E	D	5	8	D
F	D	A	8	2	F	C	A	9	3

TABLE 17 – Tables des différences — $\alpha = 2$ et $\alpha = 3$

$\alpha = 4$					$\alpha = 5$				
x	$x \oplus 4$	$\sigma(x)$	$\sigma(x \oplus 4)$	$\sigma(x) \oplus \sigma(x \oplus 4)$	x	$x \oplus 5$	$\sigma(x)$	$\sigma(x \oplus 5)$	$\sigma(x) \oplus \sigma(x \oplus 5)$
0	4	D	6	B	0	5	D	7	A
1	5	B	7	C	1	4	B	6	D
2	6	3	E	D	2	7	3	4	7
3	7	1	4	5	3	6	1	E	F
4	0	6	D	B	4	1	6	B	D
5	1	7	B	C	5	0	7	D	A
6	2	E	3	D	6	3	E	1	F
7	3	4	1	5	7	2	4	3	7
8	C	2	9	B	8	D	2	8	A
9	D	C	8	4	9	C	C	9	5
A	E	0	5	5	A	F	0	A	A
B	F	F	A	5	B	E	F	5	A
C	8	9	2	B	C	9	9	C	5
D	9	8	C	4	D	8	8	2	A
E	A	5	0	5	E	B	5	F	A
F	B	A	F	5	F	A	A	0	A

TABLE 18 – Tables des différences — $\alpha = 4$ et $\alpha = 5$

$\alpha = 6$					$\alpha = 7$				
x	$x \oplus 6$	$\sigma(x)$	$\sigma(x \oplus 6)$	$\sigma(x) \oplus \sigma(x \oplus 6)$	x	$x \oplus 7$	$\sigma(x)$	$\sigma(x \oplus 7)$	$\sigma(x) \oplus \sigma(x \oplus 7)$
0	6	D	E	3	0	7	D	4	9
1	7	B	4	F	1	6	B	E	5
2	4	3	6	5	2	5	3	7	4
3	5	1	7	6	3	4	1	6	7
4	2	6	3	5	4	3	6	1	7
5	3	7	1	6	5	2	7	3	4
6	0	E	D	3	6	1	E	B	5
7	1	4	B	F	7	0	4	D	9
8	E	2	5	7	8	F	2	A	8
9	F	C	A	6	9	E	C	5	9
A	C	0	9	9	A	D	0	8	8
B	D	F	8	7	B	C	F	9	6
C	A	9	0	9	C	B	9	F	6
D	B	8	F	7	D	A	8	0	8
E	8	5	2	7	E	9	5	C	9
F	9	A	C	6	F	8	A	2	8

TABLE 19 – Tables des différences — $\alpha = 6$ et $\alpha = 7$

$\alpha = 8$					$\alpha = 9$				
x	$x \oplus 8$	$\sigma(x)$	$\sigma(x \oplus 8)$	$\sigma(x) \oplus \sigma(x \oplus 8)$	x	$x \oplus 9$	$\sigma(x)$	$\sigma(x \oplus 9)$	$\sigma(x) \oplus \sigma(x \oplus 9)$
0	8	D	2	F	0	9	D	C	1
1	9	B	C	7	1	8	B	2	9
2	A	3	0	3	2	B	3	F	C
3	B	1	F	E	3	A	1	0	1
4	C	6	9	F	4	D	6	8	E
5	D	7	8	F	5	C	7	9	E
6	E	E	5	B	6	F	E	A	4
7	F	4	A	E	7	E	4	5	1
8	0	2	D	F	8	1	2	B	9
9	1	C	B	7	9	0	C	D	1
A	2	0	3	3	A	3	0	1	1
B	3	F	1	E	B	2	F	3	C
C	4	9	6	F	C	5	9	7	E
D	5	8	7	F	D	4	8	6	E
E	6	5	E	B	E	7	5	4	1
F	7	A	4	E	F	6	A	E	4

TABLE 20 – Tables des différences — $\alpha = 8$ et $\alpha = 9$

$\alpha = A$					$\alpha = B$				
x	$x \oplus A$	$\sigma(x)$	$\sigma(x \oplus A)$	$\sigma(x) \oplus \sigma(x \oplus A)$	x	$x \oplus B$	$\sigma(x)$	$\sigma(x \oplus B)$	$\sigma(x) \oplus \sigma(x \oplus B)$
0	A	D	0	D	0	B	D	F	2
1	B	B	F	4	1	A	B	0	B
2	8	3	2	1	2	9	3	C	F
3	9	1	C	D	3	8	1	2	3
4	E	6	5	3	4	F	6	A	C
5	F	7	A	D	5	E	7	5	2
6	C	E	9	7	6	D	E	8	6
7	D	4	8	C	7	C	4	9	D
8	2	2	3	1	8	3	2	1	3
9	3	C	1	D	9	2	C	3	F
A	0	0	D	D	A	1	0	B	B
B	1	F	B	4	B	0	F	D	2
C	6	9	E	7	C	7	9	4	D
D	7	8	4	C	D	6	8	E	6
E	4	5	6	3	E	5	5	7	2
F	5	A	7	D	F	4	A	6	C

TABLE 21 – Tables des différences — $\alpha = A$ et $\alpha = B$

$\alpha = C$					$\alpha = D$				
x	$x \oplus C$	$\sigma(x)$	$\sigma(x \oplus C)$	$\sigma(x) \oplus \sigma(x \oplus C)$	x	$x \oplus D$	$\sigma(x)$	$\sigma(x \oplus D)$	$\sigma(x) \oplus \sigma(x \oplus D)$
0	C	D	9	4	0	D	D	8	5
1	D	B	8	3	1	C	B	9	2
2	E	3	5	6	2	F	3	A	9
3	F	1	A	B	3	E	1	5	4
4	8	6	2	4	4	9	6	C	A
5	9	7	C	B	5	8	7	2	5
6	A	E	0	E	6	B	E	F	1
7	B	4	F	B	7	A	4	0	4
8	4	2	6	4	8	5	2	7	5
9	5	C	7	B	9	4	C	6	A
A	6	0	E	E	A	7	0	4	4
B	7	F	4	B	B	6	F	E	1
C	0	9	D	4	C	1	9	B	2
D	1	8	B	3	D	0	8	D	5
E	2	5	3	6	E	3	5	1	4
F	3	A	1	B	F	2	A	3	9

TABLE 22 – Tables des différences — $\alpha = C$ et $\alpha = D$

$\alpha = E$					$\alpha = F$				
x	$x \oplus E$	$\sigma(x)$	$\sigma(x \oplus E)$	$\sigma(x) \oplus \sigma(x \oplus E)$	x	$x \oplus F$	$\sigma(x)$	$\sigma(x \oplus F)$	$\sigma(x) \oplus \sigma(x \oplus F)$
0	E	D	5	8	0	F	D	A	7
1	F	B	A	1	1	E	B	5	E
2	C	3	9	A	2	D	3	8	B
3	D	1	8	9	3	C	1	9	8
4	A	6	0	6	4	B	6	F	9
5	B	7	F	8	5	A	7	0	7
6	8	E	2	C	6	9	E	C	2
7	9	4	C	8	7	8	4	2	6
8	6	2	E	C	8	7	2	4	6
9	7	C	4	8	9	6	C	E	2
A	4	0	6	6	A	5	0	7	7
B	5	F	7	8	B	4	F	6	9
C	2	9	3	A	C	3	9	1	8
D	3	8	1	9	D	2	8	3	B
E	0	5	D	8	E	1	5	B	E
F	1	A	B	1	F	0	A	D	7

TABLE 23 – Tables des différences — $\alpha = E$ et $\alpha = F$

Annexes 17 : Table des Différences

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	4	2	0	0	0	2	0	0	0	2	0	0	0	2	4
2	0	0	4	4	0	0	0	0	2	0	2	0	2	0	2	0
3	0	0	2	2	0	0	0	0	2	2	0	0	4	4	0	0
4	0	0	0	0	2	6	0	0	0	0	0	4	2	2	0	0
5	0	0	0	0	0	2	0	2	0	0	8	0	0	2	0	2
6	0	0	0	2	0	2	4	4	0	2	0	0	0	0	0	2
7	0	0	0	0	2	2	2	2	4	4	0	0	0	0	0	0
8	0	0	0	2	0	0	0	2	0	0	0	2	0	0	4	6
9	0	6	0	0	2	0	0	0	0	2	0	0	2	0	4	0
A	0	2	0	2	2	0	0	2	0	0	0	0	2	6	0	0
B	0	0	4	2	0	0	2	0	0	0	0	2	2	2	0	2
C	0	0	0	2	4	0	2	0	0	0	0	6	0	0	2	0
D	0	2	2	0	4	4	0	0	0	2	2	0	0	0	0	0
E	0	2	0	0	0	0	2	0	6	2	2	0	2	0	0	0
F	0	0	2	0	0	0	2	4	2	2	0	2	0	0	2	0

TABLE 24 – Table des différences de la S-boîte

Bibliographie

[1] Cryptographie moderne à clé secrète. https://yd-fsm.weebly.com/uploads/4/6/7/1/46716243/cours_mr2_ch3.pdf, n.d. Consulté le 15 avril 2025.

[2] S. Al-Fakir. *Algèbre et théorie des nombres, cryptographie, primalité*. Ellipses, 32, rue Bargue 75740 Paris cedex 15, 2003. page 9,12,21,34.

[3] T. Bekkouche. *Développement et implémentation des techniques de cryptage des données basées sur les transformées discrètes*. PhD thesis, 2018.

[4] R. Yakhlaf & M. Bougaada. Cryptanalyse d'un algorithme de chiffrement par des techniques heuristiques, 2024. University Center of Abdalhafid Boussouf - Mila.

[5] B. Debbagh & N. Bounegeb. *Eude et comparaison de principaux systèmes crypto Fournis par le package de Bouncy Castle Plate forme Java SDK*. PhD thesis, 2016.

[6] D Daniel. *Théorie des nombres*. Dunod, Paris, 1998. page 51,87.

[7] D. El Khier. Etude et comparaison des principaux systèmes de cryptage. *diplôme de magister en informatique, université M'sila*, 2006.

[8] A. Eli Biham. Differential cryptanalysis of des-like cryptosystems. *Journal of Cryptology*, 1990.

[9] A. Eli Biham. *Differential Cryptanalysis of the Data Encryption Standard*. Springer, 1991.

[10] F. Grosshans. La cryptographie quantique : L'incertitude quantique au service de la confidentialité. *Photoniques (Orsay)*, (8) :28–32, 2002.

[11] H. Heys. A tutorial on linear and differential cryptanalysis. *Cryptologia*, 26(3) :189–221, 2002.

[12] N. Sad houari. *"Conception Et Réalisation D'un Système Collaboratif Pour Les Experts Métier À Base D'agents Et Des Algorithmes De Cryptage"*. PhD thesis, "Université Ahmed Ben Bella - Oran 1", 2017.

[13] J.Jeanérémy. *Cryptanalyse de primitives symétriques basées sur le chiffrement AES*. PhD thesis, Citeseer, 2013.

[14] R. Madagasikara. Sécurisation des images à l'aide de la neuro-cryptographie et du tatouage par ondelette.

[15] P. Garg & SH. Varshney & M.Bhardwaj. Cryptanalysis of simplified data encryption standard using genetic algorithm. *American Journal of Networks and Communications*, 4(3) :32–36, 2015.

[16] N. Merabet. *Développement et mise en oeuvre de méthodes de cryptographie et de tatouage pour la protection de données numériques*. PhD thesis, Thèse de doctorat ,Université de Batna 2, 2018.

[17] J. De Koninck & A. Mercier. *1001 Problèmes en théorie des nombres*. Ellipses, 32, rue Bargue 75740 Paris cedex 15, 2004. page 19,27,43,.

[18] J. Masereel & E. Volte & V. Nachef. Évolution de la cryptographie à travers les âges : Cours, exercices corrigés, algorithmes en scratch et python. 2024.

[19] R. Oppliger. *Cryptography 101 : From Theory to Practice*. Artech House, 2021.

[20] L. Rezkallah. *De la cryptographie classique à la cryptographie moderne théorie et application*. PhD thesis, 2007.

[21] G. Sakthivel. Differential cryptanalysis of substitution permutation networks and rijndael-like ciphers. Master's project report, Rochester Institute of Technology, Department of Computer Science, 2005.

[22] G. Sakthivel. Differential cryptanalysis of substitution permutation networks and rijndael-like ciphers. 2006.

[23] B. Schneier. *"Cryptographie appliquée : Algorithme, protocoles et codes sources en C"*. PhD thesis, Université ferhat abbas setif-1, 2001.

[24] H. Kruppa & S. Shahy. Differential and linear cryptanalysis in evaluating aes candidate algorithms. Technical report, Technical report, National Institute of Standards and Technology, 1998.

[25] W.McKinney. *Python for data analysis : Data wrangling with pandas, numpy, and jupyter*. " O'Reilly Media, Inc.", 2022.

[26] R. Yende. Support de cours de sécurité informatique et crypto. 2018.

[27] Y. Kernouf & C. Zerrouki. *Simulation de quelques attaques sur le cryptosystème RSA*. PhD thesis, Université Mouloud Mammeri, 2020.

Résumé

Dans ce mémoire, nous avons étudié la cryptanalyse différentielle et son application sur un réseau SPN. Après avoir posé les bases mathématiques et introduit les notions clés de la cryptologie, nous avons implémenté un réseau SPN simplifié en Python. Grâce à une analyse différentielle rigoureuse, nous avons réussi à retrouver partiellement, puis complètement, une sous-clé du chiffrement. Ce travail met en lumière la puissance de cette méthode pour identifier des failles dans les algorithmes symétriques modernes.
Mots clés : Réseau SPN, cryptologie, cryptanalyse différentielle, caractéristiques, table des différences,S-boîte, clé.

Abstract

In this thesis, we explored differential cryptanalysis and its application to a Substitution-Permutation Network (SPN). After establishing the mathematical foundations and introducing key concepts in cryptology, we implemented a simplified SPN in Python. Through careful differential analysis, we were able to partially, and then fully, recover one of the subkeys. This work highlights the effectiveness of differential cryptanalysis in identifying weaknesses in modern symmetric encryption algorithms.
Keywords : SPN network, Cryptology, differential cryptanalysis, characteristics, difference table, S-box, key.