Problème 1

a) La table de vérité a été construit çi dessous
b) Les 16 fonctions affine ont été mis dans le tableau aussi
c) De même la distance Hamming a éte calculer

La distance de Hamming d(f, g) est la somme des différences dans les valeurs de f(x) et g(x) pour toutes les entrées possibles x.

Construisons la table de vérité pour g(x) :

x_1	x_2	x_3	$f(x) = x_1 \cdot x_2 \oplus x_3$	g(x) = 0	g(x) = 1	$g(x) = x_3$	$g(x) = x_3 \oplus 1$
0	0	0	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	0	1
0	1	1	1	0	1	1	0
1	0	0	0	0	1	0	1
1	0	1	1	0	1	1	0
1	1	0	1	0	1	0	1
1	1	1	0	0	1	1	0
La distance de Hamming d(f, g)				4	4	2	6

Suite de la table

x_1	x_2	x_3	$f(x) = x_1 \cdot x_2 \oplus x_3$	$g(x) = x_2$	$g(x) = x_2 \oplus 1$	$g(x) = x_2 \oplus x_3$	$g(x) = x_2 \oplus x_3 \oplus 1$
0	0	0	0	0		0	1
0	0	1	1	0		1	0
0	1	0	0	1		1	0
0	1	1	1	1		0	1

1	0	0	0	0	1 1 0 0 1 1 0 0	0	1
1	0	1	1	0		1	0
1	1	0	1	1		1	0
1	1	1	0	1		0	1
La distance de Hamming d(f, g)				4	4	2	6

Suite de la table

x1	x2	x3	$f(x) = x_1 \cdot x_2 \oplus x_3$	$g(x) = x_1$	$g(x) = x_1 \oplus 1$	$g(x) = x_1 \oplus x_3$	$g(x) = x_1 \oplus x_3 \oplus 1$
0	0	0	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	0	1
0	1	1	1	0	1	1	0
1	0	0	0	1	0	1	0
1	0	1	1	1	0	0	1
1	1	0	1	1	0	1	0
1	1	1	0	1	0	0	1
La distance de Hamming d(f, g)				4	4	2	6

Suite de la table

x_1	x_2	x_3	$f(x) = x_1 \cdot x_2 \oplus x_3$	$g(x) = x_1 \oplus x_2$	$g(x) = x_1 \oplus x_2 \oplus 1$	$g(x) = x_1 \oplus x_2 \oplus x_3$	$g(x) = x_1 \oplus x_2 \oplus x_3 \oplus 1$
0	0	0	0	0	1	0	1

0	0	1	1	0	1	1	0
0	1	0	0	1	0	1	0
0	1	1	1	1	0	0	1
1	0	0	0	1	0	1	0
1	0	1	1	1	0	0	1
1	1	0	1	0	1	0	1
1	1	1	0	0	1	1	0
La distance de Hamming d(f, g)				4	4	6	2

La non-linéarité NL (f) de f est la distance minimum de Hamming pour toutes

$$NL(f) = \min_{g \text{ affine}} \{d_H(f, g)\}$$

d) NL (f) = 2

PROBLEME 2 : Utilisation de la transformée de Walsh–Hadamard

Pour se faire j'ai exécuté walsh_hadamard.py qui est le programme fournis par l'énoncé que je vais inclure dans Exercice 3 et j'ai obtenue les résultant suivant

a)

python walsh_hadamard.py pour le lancer

Fonction : f(x1, x2, x3) = x1·x2 ⊕ x3

Coefficients de Walsh-Hadamard:

Wf([0, 0, 0]) = 0

Wf([0, 0, 1]) = 4

Wf([0, 1, 0]) = 0

Wf([0, 1, 1]) = 4

Wf([1, 0, 0]) = 0

Wf([1, 0, 1]) = 4

Wf([1, 1, 0]) = 0

Wf([1, 1, 1]) = -4

b)

W_max = 4

c)Non-linéarité NL(f) = 2.0 et oui ça correspond

d)La fonction a une bonne non-linéarité (2.0/3)

Problème 3 Non-linéarité de la S-box AES

a) Ce programme a pour objectif d'évaluer la résistance cryptographique de la S-Box AES face aux attaques de cryptanalyse linéaire.

Il mesure cette résistance en calculant la non-linéarité, un indicateur quantitatif de l'écart entre le comportement de la S-Box et celui d'une fonction linéaire.

Pour ce faire, le programme s'appuie sur la transformée de Walsh–Hadamard, un outil mathématique qui permet de décomposer une fonction booléenne en ses composantes linéaires et d'en mesurer l'amplitude.

Ainsi, au-delà d'une simple vérification du bon fonctionnement de la S-Box, le programme offre une analyse scientifique et mesurable de sa robustesse cryptographique.

Il constitue donc un outil d'évaluation permettant de confirmer que la conception de la S-Box AES répond aux plus hautes exigences de sécurité face aux attaques linéaires.

b) Je vais maintenant modifier le code comme suite : Calcule la non-linéarité théorique maximale 8 bits, la non-linéarité de chaque bit de sortie de 0 à 7 de la S-box et afficher les résultats avec des statistiques et une évaluation

Problème 4 : Signification cryptographique

a) Expliquez pourquoi une forte non-linéarité est essentielle dans les fonctions cryptographiques.

-Car c'est possible pour les mathématiciens de faire le calcul des systèmes linéaire
-Sécurité : Les fonctions avec forte linéarité sont plus difficiles à prédire ce qui augmente la sécurité contre les attaques grâce à la cryptographie linéaire
-Complexité et robustesse : la non linéarité est liée à la complexité computationnel et à la robustesse des fonctions, ce qui est crucial pour la sécurité

b) Discutez comment la non-linéarité de la S-box AES contribue à sa sécurité. — Expliquez l'importance d'une forte non-linéarité dans les S-box. — Comparez la non-linéarité obtenue avec celle théoriquement maximale pour une fonction sur 8 bits.

La non-linéarité de la Sbox AES contribue à sa sécurité en introduisant des transformations non linéaires dans le processus de chiffrement. Cela permet de garantir que chaque octet de la sortie subit une transformation non linéaire, ce qui contribue à la sécurité globale de l'algorithme AES.

La non-linéarité dans les S-box AES est cruciale pour plusieurs raisons :

Protection contre les attaques cryptographiques : Une forte non-linéarité rend les S-box moins susceptibles d'être piratées par des algorithmes d'attaque basés sur des modèles linéaires.
Confusion des données : Les S-box sont conçues pour créer une confusion dans le chiffrement, rendant les données difficiles à décrypte

Non-linéarité moyenne de la S-box AES : 112.00

Non-linéarité théorique maximale : 120

La non-linéarité obtenue est inférieur à théoriquement maximale

c) Quels défis computationnels auriez-vous rencontrés si, au lieu d'utiliser la transformée de Walsh-Hadamard, le programme calculait la non-linéarité de la S-Box AES en utilisant la distance de Hamming minimale par rapport aux fonctions affines de 8 bits ?

   En utilisant la distance de Hamming pour mesurer la non-linéarité de la S-box AES, on rencontrerait deux grands défis.
   D'abord, il faudrait comparer la S-box à toutes les fonctions affines possibles sur 8 bits, ce qui représente énormément de calculs et rend la tâche très longue.
   Ensuite, cette méthode demanderait beaucoup de mémoire et de puissance de calcul, car chaque fonction affine devrait être évaluée et stockée.
   C'est pour éviter cette lourdeur que le programme utilise plutôt la transformée de Walsh–Hadamard, qui permet

d’obtenir le même résultat de manière beaucoup plus rapide et efficace.