Exercices de Cryptographie Moderne

PARTIE 1 : EXERCICES À LA MAIN

Exercice 1 : Chiffrement XOR de base

Objectif : Comprendre l'opération XOR et son utilisation en cryptographie

a) Alice veut envoyer le message M = 10110101 à Bob en utilisant la clé K = 11001010
   - Calculez le message crypté C = M $\oplus$ K
   - Vérifiez que Bob peut retrouver M en calculant M = C $\oplus$ K

Exercice 2 : Algorithme Blum Blum Shub (BBS)

Objectif : Générer des bits pseudo-aléatoires avec BBS

Données : - p = 7, q = 11 (nombres premiers, tous deux congrus à 3 modulo 4) - M = p × q = ? - x_0 = 5 (graine, premier avec M)

a) Calculez M

b) Vérifiez que p ≡ 3 (mod 4) et q ≡ 3 (mod 4)

c) Vérifiez que x_0 = 5 est bien premier avec M (utilisez pgcd)

d) Calculez les 5 premiers termes de la suite : x_1, x_2, x_3, x_4, x_5 avec la formule x_{n+1} = x_n^2 mod M

e) Extrayez le bit de poids faible (LSB) de chaque terme pour obtenir une séquence de 5 bits pseudo-aléatoires

---

Exercice 3 : Cryptographie symétrique vs asymétrique

Objectif : Comprendre les différences conceptuelles

a) **Multiplication facile, factorisation difficile** :

   - Calculez 17 × 23 = ?
   - Maintenant factorisez 1147 (trouvez p et q tels que p × q = 1147)
   - Quel exercice était le plus facile ?

Exercice 4 : Entropie et trappes

Objectif : Comprendre les concepts de sécurité

a) **Entropie** : Une clé de 8 bits peut prendre combien de valeurs différentes ? Et une clé de 128 bits ?

PARTIE 2 : EXERCICES PYTHON

Exercice 5 : Générateur Blum Blum Shub

Objectif : Implémenter le générateur BBS

```python
import math

def blum_blum_shub(seed, p, q, iterations):
    """
    Génère des bits pseudo-aléatoires avec BBS
    seed: graine (x0)
    p, q: nombres premiers congrus à 3 mod 4
    iterations: nombre de bits à générer
    retourne: liste de bits
    """
    # TODO: Vérifier que p % 4 == 3 et q % 4 == 3

    # TODO: Calculer M = p * q

    # TODO: Vérifier que pgcd(seed, M) == 1

    # TODO: Générer la suite et extraire les bits LSB
    pass

# Tests
bits = blum_blum_shub(seed=3, p=11, q=23, iterations=10)
print(f"Bits générés : {bits}")

# Question : Les bits semblent-ils aléatoires ?
```

Extensions :

1. Modifiez la fonction pour extraire la parité au lieu du LSB

2. Testez avec de très grands nombres premiers (utilisez `p=2027, q=2039`)

3. Comparez la sortie de BBS avec `random.randint(0,1)`