# HAI709I - TD 3

Rocco Mora

29 Septembre, 2025

## Exercice 1 :

Calculez les 10 premiers bits de sortie (ou plus si nécessaire) des LFSR suivants, calculez leur période et indiquez s'ils sont de longueur maximale :

1. Le LFSR de degré 4 avec des coefficients de rétroaction $c_0 = c_1 = 0, c_2 = c_3 = 1$ et un état initial $(s_3, \ldots, s_0) = (1, 0, 0, 0)$.
2. Le LFSR de degré 4 avec des coefficients de rétroaction $c_0 = c_1 = 1, c_2 = c_3 = 0$ et un état initial $(s_3, \ldots, s_0) = (1, 0, 0, 0)$.
3. Le LFSR de degré 5 avec des coefficients de rétroaction $c_0 = c_2 = c_4 = 1$, $c_1 = c_3 = 0$ et un état initial $(s_4, \ldots, s_0) = (1, 1, 0, 1, 1)$.

## Exercice 2 :

1. Considérons un générateur de combinaisons non linéaires construit à partir d'un LFSR de degré $n$, mais dont la sortie est donnée par la fonction non linéaire $g(s_0, \ldots, s_{n-1}) = s_0 \wedge s_1$. Calculez la probabilité que le bit de sortie soit 0 ou 1 et déduisez que $g$ ne donne pas un bon générateur pseudo-aléatoire.
2. Considérons maintenant la fonction non linéaire $g(s_0, \ldots, s_{n-1}) = (s_0 \wedge s_1) \oplus s_2$. Quelle est la probabilité que le bit de sortie soit 0 ou 1 ? Montrez que ce choix de $g$ est également peu sûr en fournissant une attaque de distinction. Vous pouvez suivre le guide ci-dessous.
   - Étape 1 : Supposons que $s_1^{(0)} = 0$. Dérivez $s_2^{(0)}$ en fonction de $y_0$.
   - Étape 2 : Montrez par induction que tous les $s_j^{(0)}$, $j = 1, \ldots, n-1$, peuvent être obtenus à partir de $y_i$, $i = 1, \ldots, n-1$.
   - Étape 3 : Montrez qu'avec ces informations, un attaquant PPT peut deviner avec une probabilité relativement élevée les $n$ bits suivants de la sortie.

## Exercice 3 :

Attaquons-nous RC4 ! Nous montrons en particulier que le deuxième octet de sortie de RC4 est biaisé vers 0. Soit $S_t$ le tableau $S$ après $t$ itérations de `Next` = `GetBits`. Considérons $S_0$ (le tableau initial obtenu après `Init`) comme une permutation uniforme de $\{0, \ldots, 255\}$.

- Étape 1 : Calculez la probabilité que $S_0[2] = 0$ et $X \stackrel{\text{def}}{=} S_0[1] \neq 2$.
- Étape 2 : en supposant que ce soit le cas, effectuez une première itération de `GetBits`. En particulier, calculez $S_1[X]$.
- Étape 3 : effectuez une deuxième itération de `GetBits`. Quelle est la sortie $y$ ?
- Étape 4 : quelle est la probabilité que le deuxième bit de sortie $y$ soit 0 lorsque $S_0[2] \neq 0$ à la place ? Dérivez la probabilité totale que le deuxième bit de sortie soit 0 et expliquez pourquoi il s'agit d'une attaque statistique.

**Remarque :** il existe d'autres attaques plus graves contre RC4, par exemple lorsqu'un $IV$ est ajouté au début de la clé.

## Exercice sur ordinateur:

Nous allons tester avec Python que la probabilité que le bit de sortie d'un bon chiffrement par flux est approximativement égal à 1/2.

1. Étape 1 : Installez le paquet `pycryptodome`.
2. Étape 2 : complétez le script suivant

```
from Crypto.Cipher import ChaCha20
from Crypto.Random import get_random_bytes

def count_bits(data: bytes):
"""Renvoie le nombre de 0 et de 1 dans la séquence d'octets."""
        # Réécrire les octets sous forme de bits
        bits =
        # calculer le nombre de 0 et de 1 avec la mèthode "count"
        zeros =
        ones =
        return zeros, ones

def main():
        # Étape 1 : Générer une clé aléatoire et un nonce de
        # 256 et 64 bits respectivement,
        # à l'aide de la fonction get_random_bytes
        # (combien d'octets correspondent à 256 et 64 bits ?)

        key =
        nonce =

        # Étape 2 : Initialiser le chiffrement ChaCha20 à l'aide
        # des variables key et nonce
        cipher =

        # Étape 3 : Initialiser le texte en clair.
        # Ensuite, chiffrer à l'aide du chiffrement défini précédemment.
        plaintext = b"\x00" * 10_000    # 10 KB of zeros
        keystream =

        # Étape 4 : Compter les 0s et les 1s dans le flux de clés
        # à l'aide de la fonction "count_bits"
        zeros, ones =

        # Étape 5 : Calculer les probabilités d'obtenir 0 et 1
        # et afficher les résultats


#exécuter
main()
```