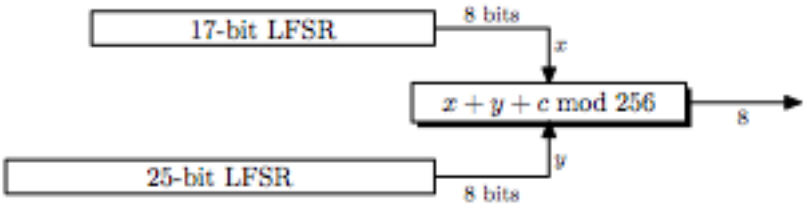
Attaque sur CSS

Dans ce TP vous pouvez réutiliser les fonctions `LFSR_Step` et `LFSR` issues du TP2 (récupérez le fichier `tp5_code.py`).

Le Content Scrambling System (CSS) est utilisé pour sécuriser le contenu des disques DVDs. CSS souffre de plusieurs faiblesses. Nous discutons ici du stream cipher CSS utilisé pour chiffrer les contenus des films. CSS a été conçu dans les années 1980 quand le chiffrement exportable était restreint à 40 bits. En conséquence, CSS utilise des clés secrètes de 40 bits.

Le PRG de CSS (Algorithm 1)[1] utilise deux LFSRs.



Algorithm 1 CSS

Entrée : seed $s \in \{0,1\}^{40}$

Écrire $s = s_1 \| s_2$ où $s_1 \in \{0,1\}^{16}$ et $s_2 \in \{0,1\}^{24}$
Charger $s_1 \| 1$ dans le LFSR de 17 bits
Charger $s_2 \| 1$ dans le LFSR de 25 bits
$c \leftarrow 0$ ▷ carry bit
for $i = 1$ à N **do**
  exécuter les deux LFSRs pendant 8 cycles et obtenir $x, y \in \{0,1\}^8$
  traiter x et y comme des entiers dans $0 \ldots 255$
  Renvoyer $z = x + y + c \bmod 256$
  si $x + y > 255$, alors $c \leftarrow 1$, sinon $c \leftarrow 0$ ▷ carry bit
end for

Le PRG retourne un octet à chaque itération. L'ajout du bit à 1 pour s_1 et s_2 garantit que les LFSRs ne seront pas initialisés à 0. Les polynômes de rétroaction des LFSR de 17 et 25 bits sont respectivement :

$$\begin{cases} P_{17} = X^{17} + X^3 + 1 \\ P_{25} = X^{25} + X^{22} + X^{21} + X^{13} + 1 \end{cases}$$

Ces deux polynômes sont primitifs.

Question 1. *Programmez le PRG de CSS. Pour représenter simplement la sortie, utilisez une liste d'entiers python entre 0 et 255.*

Question 2. *Soit x_1, x_2, x_3 les 3 premiers octets de sortie du LFSR de 17 bits. Montrez que l'état initial s_2 du second LFSR peut être obtenu en fonction de (z_1, z_2, z_3) (les trois premiers octets de sortie du PRG) et (x_1, x_2, x_3).*

[1] Il s'agit en réalité d'une petite variation ; dans le véritable CSS le chargement des clés et l'exécution sont un peu différents.

Question 3. *On suppose que l'on connaît les 5 premiers octets $z_1, z_2, \ldots, z_5$. Montez une attaque du type guess-and-determine de complexité 2^{16}.*

Question 4. *Programmez l'attaque contre ce générateur.*

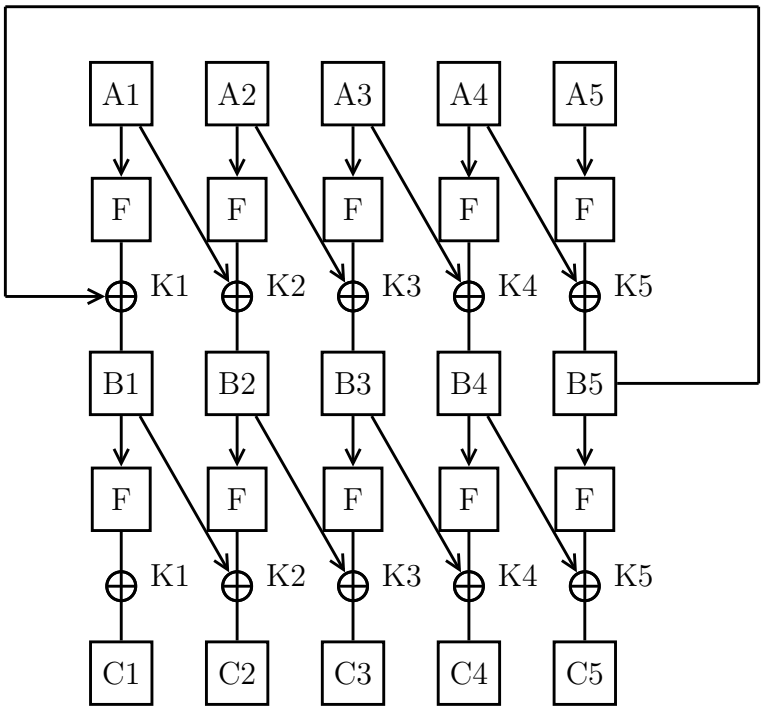


Figure 1: "Mangling function" dans CSS.

Pour certaines applications de CSS, la suite chiffrante n'est pas immédiatement XORée au clair. Soient $K_1, \ldots, K_5$ 5 octets de la suite chiffrante. On définit une boîte S notée F, dont on suppose qu'elle est connue et publique. En partant de cinq octets de clair $A_1, \ldots, A_5$, les 5 octets de chiffré $C_1, \ldots, C_5$ sont obtenus à l'aide d'une "mangling function" représentée sur la Figure 1. Sur cette figure, les $\oplus$ indiquent des XOR où interviennent les octets de suite chiffrante et des valeurs intermédiaires ; les B_i sont également des valeurs intermédiaires.

Question 5. *Montrer qu'à partir de 5 octets de clair et de chiffré connu, on peut retrouver 5 octets de suite chiffrante (de manière beaucoup plus efficace que la force brute).*