TP4 : Attaques contre le chiffrement à flot

7 février 2025

1 Attaques contre le chiffrement de Geffe

Le chiffrement de Geffe est un chiffrement à flot par registres combinés, proposé par Geffe en 1973. Il est composé de trois LFSR de longueurs distinctes combinés par la fonction booléenne

$$f(x_1, x_2, x_3) = x_1x_2 + x_2x_3 + x_3.$$

Dans le cadre de cet exercice, on suppose que les LFSR internes sont les suivants :

— `L1` de longueur 13, de polynôme de rétroaction $P_1 \stackrel{\text{def}}{=} 1 + X^3 + X^4 + X^{13}$.

— `L2` de longueur 11, de polynôme de rétroaction $P_2 \stackrel{\text{def}}{=} 1 + X^2 + X^{11}$.

— `L3` de longueur 9, de polynôme de rétroaction $P_3 \stackrel{\text{def}}{=} 1 + X^4 + X^9$.

1.1 Implémentation du générateur de Geffe

Q1) D'après la définition d'un LFSR combiné, quelle est la valeur de $z(t)$ en fonction de $s_1(t), s_2(t)$ et $s_3(t)$?

Q2) Programmer une fonction `geffe(S1,S2,S3,N)` qui prend en entrée les trois états initiaux et un entier N, et retourne les N premiers bits de la suite.

Pour tester si votre générateur fonctionne, vous pouvez vérifier que les 20 premiers bits de la suite générée par ce générateur, initialisé avec les états

$$S_1 = [1,0,1,0,1,0,1,0,1,0,1,0,1], \quad S_2 = [1,0,1,0,1,0,1,0,1,0,1], \quad S_3 = [1,0,1,0,1,0,1,0,1]$$

est

$$s = 1,0,1,0,1,0,1,0,1,1,1,0,1,0,1,0,1,1,1,0,\ldots$$

1.2 Attaque par corrélation

L'objectif de cette partie est de décrire et implémenter l'*attaque par corrélation*, telle que proposée par Siegenthaler [1, 2], dans le cas particulier du générateur de Geffe.

Q3) Quelle est la complexité de la recherche exhaustive pour déterminer l'état initial du générateur de Geffe ?

Faisons l'hypothèse qu'à chaque instant t, les bits $s_1(t), s_2(t), s_3(t)$ produits par les LFSR internes sont des variables aléatoires indépendantes et uniformément distribuées dans $\mathbb{F}_2$. Ainsi, $z(t) = f(s_1(t), s_2(t), s_3(t))$ est aussi une variable aléatoire, à valeurs dans $\mathbb{F}_2$.

Q4) Montrer que

$$\mathbb{P}[z(t) = s_1(t)] = \mathbb{P}[z(t) = s_3(t)] = \frac{3}{4}.$$

Q5) Soit x une variable aléatoire uniformément distribuée sur $\mathbb{F}_2$, et indépendante de $z(t)$. Quelle est la probabilité $p = \Pr[z(t) = x]$?

La question **Q4** démontre que la sortie du LFSR combiné de Geffe est fortement corrélée à la valeur des deux LFSR internes `L1` et `L3`.

Q6) Exploiter cette dépendance pour déduire une attaque sur le générateur de Geffe. Quelle est sa complexité ?

Q7) Implémenter cette attaque contre le générateur de Geffe.

1.3 Attaque *guess and determine*

Q8) Montrer que si un attaquant connaît le contenu du deuxième LFSR du générateur de Geffe, il peut retrouver le contenu des registres `L1` et `L3` avec une attaque de complexité de l'ordre de $\ell_1^3 + \ell_3^3$, où ℓ_1 et ℓ_3 sont les longueurs du premier et troisième LFSR respectivement. En déduire une attaque de complexité de l'ordre de $2^{\ell_2}(\ell_1^3 + \ell_3^3)$ opérations élémentaires, où ℓ_2 est la longueur du deuxième registre.

Q9) Implémenter l'attaque guess and determine sur le générateur de Geffe.

2 Attaque contre CSS

Le Content Scrambling System (CSS) est un chiffrement à flot utilisé pour sécuriser le contenu des disques DVDs en chiffrant le contenu des films. CSS a été conçu dans les années 1980, époque où la clé secrète des chiffrements exportables était limitée à 40 bits. En conséquence, CSS utilise une clé de 40 bits seulement. Ce chiffrement est aujourd'hui connu d'avoir plusieurs faiblesses et le but de cette partie est d'implémenter des attaques pratiques contre ce système.

Le chiffrement CSS est construit à partir de deux LFSRs s_1 et s_2 de tailles respectives de 17 et 25 bits. Le fonctionnement de ce chiffrement est détaillé dans l'algorithme 1 et peut être illustré par la figure 1.

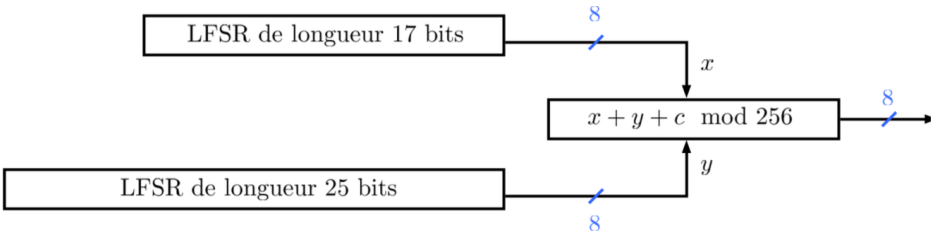

FIGURE 1 – Le Content Scrambling System (CSS)

Algorithme 1 : CSS

Données : Une valeur initiale $s \in \{0,1\}^{40}$

Écrire $s = s_1 \| s_2$ où $s_1 \in \{0,1\}^{16}$ et $s_2 \in \{0,1\}^{24}$;

Charger $1 \| s_1$ dans le LFSR de 17 bits;

Charger $1 \| s_2$ dans le LFSR de 25 bits;

$c \leftarrow 0$; `/* carry bit */`

tant que *forever* **faire**

Exécuter les deux LFSRs pendant 8 cycles et obtenir $x, y \in \{0,1\}^8$;

Traiter x et y comme des entiers dans $\{0 \dots 255\}$;

return $z = x + y + c \bmod 256$;

Si $x + y > 255$, alors $c \leftarrow 1$, sinon $c \leftarrow 0$; `/* carry bit */`

CSS renvoie un octet à chaque itération. Le premier bit sorti de chaque LFSR formera le bit de poids faible du mot x et y respectivement. Les coefficients de retroaction pour les deux LFSRs sont fixes.

Le LFSR de 17 bits a comme seuls coefficients non-nuls les coefficients $\{14, 0\}$ et celui de 25 bits les coefficients $\{12, 4, 3, 0\}$. L'ajout du bit à 1 pour s_1 et s_2 dans la position à poids fort ($\texttt{s}_1[\texttt{16}]$ et $\texttt{s}_2[\texttt{24}]$), garantit que les LFSRs ne seront pas initialisés à 0. Les autres bits (à part le bit de poids fort de l'un et le bit de poids fort de l'autre) des deux LFSRs sont initialisés avec les 40 (16 + 24) bits de la clé secrète.

Q10) Programmer l'opération de chiffrement et de déchiffrement d'un texte avec CSS. Chiffrer le message $m = \texttt{0xfffffffffff}$ avec CSS initialisé à $s = \texttt{0x0} \in \{0,1\}^{40}$. Vérifier que le chiffré correspondant est $c = \texttt{0xffffb66c39}$.

Q11) Soit x_1, x_2, x_3 les 3 premiers octets de sortie du LFSR de 17 bits. Montrer que l'état initial s_2 du second LFSR peut être obtenu en fonction de (z_1, z_2, z_3) et (x_1, x_2, x_3).

Q12) On suppose que l'on connaît les 6 premiers octets $z_1, z_2, \ldots, z_6$. Décrire une attaque de complexité 2^{16} qui permet de récupérer l'initialisation du générateur, en exploitant le point 4 ci-dessus.

Q13) Programmer l'attaque contre ce générateur. Pour cela, initialiser le générateur avec une valeur aléatoire $s \in \{0,1\}^{40}$ et générer par la suite 6 octets $z_1, z_2, \ldots, z_6$. Vérifier que votre attaque permet de bien retrouver l'état initial.

Q14) On suppose maintenant que l'on ne connaît que les 5 premiers octets $z_1, z_2, \ldots, z_5$. Monter une attaque de complexité 2^{17}.

Q15) Programmer l'attaque contre ce générateur.

Références

[1] Thomas Siegenthaler. Correlation-immunity of nonlinear combining functions for cryptographic applications. *IEEE Trans. Inf. Theory*, 30(5) :776–780, 1984.

[2] Thomas Siegenthaler. Decrypting a class of stream ciphers using ciphertext only. *IEEE Trans. Computers*, 34(1) :81–85, 1985.