TD3 - Chiffrement par Flot et Cryptanalyse

Responsable : M. Bombar

1 Algorithme de Berlekamp-Massey

1.1 Remarque introductive

Cet exercice était déjà présent dans le TD2 qui était bien évidemment très long. Cependant, l'algorithme de Berlekamp-Massey est un algorithme important pour comprendre la sécurité des chiffrements par flot, donc vous le remets.

1.2 L'algorithme

Le but de cet exercice est de voir qu'il est « facile » en pratique de déterminer la complexité linéaire et le polynôme minimal d'une suite chiffrante, en ayant accès à quelques bits de la suite. C'est la raison pour laquelle on a introduit les LFSR combinés, ou encore les LFSR filtrés, dans le cours d'aujourd'hui.

Soit $\mathbf{s} \overset{\text{def}}{=} (s_n)_{n\in\mathbb{N}} \in \mathbb{F}_q^{\mathbb{N}}$ une suite ultimement périodique. On note $\Lambda(\mathbf{s}) \geq 1$ sa complexité linéaire.

(Q1) Montrez que $2\Lambda(\mathbf{s})$ termes consécutifs de la suite caractérisent entièrement $\mathbf{s}$. Plus précisément, montrez que si on connaît $\Lambda(s)$, ainsi que $2\Lambda(s)$ termes *consécutifs* de $\mathbf{s}$, alors il est possible de calculer son polynôme de rétroaction minimal P.

Indication : Écrire les coefficients de P comme les inconnues d'un système linéaire.

On suppose que l'on connaît une borne supérieure $\Lambda(s) \leq \ell$ sur la complexité linéaire de $\mathbf{s}$. Déterminez un algorithme qui prend en entrée $\mathbf{s}$ et retourne $\Lambda(\mathbf{s})$ ainsi que le polynôme de rétroaction minimal en temps $O(\ell^4)$. Quelle est sa complexité en mémoire ?

La Question (Q1) nous donne déjà un algorithme *polynomial* pour caractériser la suite $\mathbf{s}$ en n'observant qu'un nombre restreint de coefficients. En particulier, même si $\mathbf{s}$ a d'excellentes propriétés statistiques, elle n'est absolument pas *imprédictible*. Cependant, la complexité $O(\ell^4)$ peut parfois être assez grosse pour monter des attaques en pratique, surtout lorsque les LFRS sont combinés à d'autres opérations, ou lorsque la borne supérieure

ℓ n'est pas connue. Cependant, on va voir qu'il est possible de faire beaucoup mieux étant donnée la *structure* des systèmes linéaires mis en jeu.

Pour cela, on va décrire, et implémenter, un algorithme dû à James Massey [Mas69], qui a montré comment une procédure proposée deux ans plus tôt par Elwyn Berlekamp [Ber68] afin de décoder une certaine famille de codes correcteurs d'erreurs appelés codes BCH, pouvait être utilisée pour déterminer le LFSR minimal qui génère une suite donnée. Plus précisément, étant donnée une suite $\mathbf{s} = (\mathbf{s}_0, \ldots, \mathbf{s}_{n-1})$ de longueur n, ou plus précisément la suite prolongée par périodicité, l'algorithme de Berlekamp-Massey va faire n itérations. À l'itération t, l'algorithme de Berlekamp-Massey détermine un LFSR de longueur minimale qui produit les t premiers symboles de $\mathbf{s}$.

Algorithm 1: Algorithme de Berlekamp-Massey

Input: $\mathbf{s}^{(n)} = (s_0, \ldots, s_{n-1})$ une suite de n éléments de $\mathbb{F}_q$.
Output: $\Lambda(\mathbf{s}^{(n)})$ la complexité linéaire de $\mathbf{s}^{(n)}$ et P, le polynôme de rétroaction de degré minimal qui engendre $\mathbf{s}^{(n)}$.

```
/* Initialisation */
P(X) ← 1
Q(X) ← 1
Λ ← 0
m ← −1
d′ ← 1
/* Algorithme */
for t ∈ {0, ..., n − 1} do
    d ← s_t + Σ_{i=1}^{Λ} p_i s_{t−i}
    if d ≠ 0 then
        T(X) ← P(X)                          ▷ Polynôme temporaire
        P(X) ← P(X) − d(d′)^{−1} Q(X) X^{t−m}
        if 2Λ ≤ t then
            Λ ← t + 1 − Λ
            m ← t
            Q(X) ← T(X)
            d′ ← d
return Λ et P(X)
```

(Q2) Décrire les étapes de l'algorithme de Berlekamp-Massey appliqué à la suite binaire de longueur 7 et définie par

$$\mathbf{s}^{(7)} \stackrel{\text{def}}{=} (0,1,1,1,1,0,0).$$

(Q3) Implémentez l'algorithme de Berlekamp-Massey avez Sage.

(Q4) Quelle est sa complexité ?

(Q5) Montrez que l'algorithme est correct, c'est-à-dire qu'à la fin de l'algorithme on a bien $\Lambda = \Lambda(s^{(n)})$ et que P est bien le polynôme de rétroaction minimal de $\mathbf{s}^{(n)}$.

2 Cryptanalyse du chiffrement de Geffe

2.1 Le générateur de Geffe

Le chiffrement de Geffe est un chiffrement par flots par registres combinés, proposé par Geffe en 1973 [Gef73]. Il combine trois LFSR binaires par la fonction booléenne

$$f(x_1,x_2,x_3) \stackrel{\text{def}}{=} x_3 + x_2x_3 + x_1x_2.$$

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

- LFSR1 de longueur 13, de polynôme de rétroaction $P_1 \stackrel{\text{def}}{=} 1+X+X^3+X^4+X^{13}$.
- LFSR2 de longueur 11, de polynôme de rétroaction $P_2 \stackrel{\text{def}}{=} 1+X^2+X^{11}$.
- LFSR3 de longueur 9, de polynôme de rétroaction $P_3 \stackrel{\text{def}}{=} 1+X^4+X^9$.

À l'initialisation, les trois LFSR sont initialisés avec leurs états initiaux respectifs S_1, S_2 et S_3. On note respectivement $s_{(1)}(t), s_{(2)}(t), s_{(3)}(t)$ les bits de sortie respectifs des trois LFSR au temps t, et $z(t)$ le bit de sortie du LFSR combiné.

(Q6) 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 $s_3(t)$?

(Q7) Programmez en Sage une fonction `geffe(S1,S2,S3,N)` qui prend en entrée les trois états initiaux, et un eniter 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

$$S_1 \stackrel{\text{def}}{=} [1,0,1,0,1,0,1,0,1,0,1,0,1] \qquad S_2 \stackrel{\text{def}}{=} [1,0,1,0,1,0,1,0,1,0,1]$$

et

$$S_3 \stackrel{\text{def}}{=} [1,0,1,0,1,0,1,0,1]$$

sont

$$\mathbf{Z} = 1,0,1,0,1,0,1,0,1,1,1,0,1,0,0,0,0,1,0,1,\cdots$$

(Q8) Quelle est la complexité linéaire de chacun des 3 LFSR internes ?

(Q9) Selon la théorie, quelle doit être la complexité linéaire du générateur de Geffe ?

(Q10) Vérifiez que Berlekamp-Massey vous retourne bien les bonnes valeurs.

2.2 Attaque par Corrélation

(Q11) Quelle est la complexité de la recherche exhaustive pour déterminer l'état initial du générateur de Geffe ?

L'objectif de cette partie est de décrire l'*attaque par corrélation*, telle que proposée par Siegenthaler en 1985 [Sie84 ; Sie85], dans le cas particulier du générateur de Geffe.

Faisons l'hypothèse qu'à chaque instant t, les bits $s_1(t), s_2(t)$ et $s_3(t)$ 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 $\mathbb{F}_2$.

(Q12) Montrez que

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

et

$$\mathbb{P}\Big[z(t) = s_2(t)\Big] = \frac{1}{2}$$

(Q13) Soit x une variable aléatoire uniformément distribuée sur $\mathbb{F}_2$, et indépendante de $z(t)$. Quelle est la probabilité $p \stackrel{\text{def}}{=} \mathbb{P}\Big[z(t) = x\Big]$?

La question (Q12) démontre que la sortie du LFSR combiné de Geffe est fortement corrélée [1] à la valeur des deux LFSR internes LFSR1 et LFSR3. Le but d'une attaque par

1. Remarquez que cette corrélation ne dépend pas de la structure de ces LFSR, qui n'interviendra que dans le calcul de la complexité de l'attaque.

corrélation est d'exploiter cette propriété pour en déduire les états initiaux de chacun des registres.

Attaque par corrélation

On suppose que par une attaque à clairs connus on a réussit à déterminer ℓ bits consécutifs de la suite chiffrante, de $z(t_0)$ à $z(t_0 + \ell - 1)$.

Détermination de l'état interne du LFSR1 On va alors faire une recherche exhaustive sur l'état du registre LFSR1 à l'instant t_0, et pour chaque état candidat $\widetilde{S_1}$, on calcule la suite $(s_1(t))$: D'après la question (Q12), lorsque l'état interne est correct, on s'attend à ce que $(s_1(t))$ coïncide avec $\approx \frac{3}{4}$ des valeurs obtenues si ℓ est suffisamment grand. Dans le cas contraire, d'après la question (Q13), on s'attend à ce que seuls $\approx p\ell$ éléments de la suite coïncident.

Détermination de l'état interne du LFSR3 On fait la même chose avec le LFSR3, et on devine un candidat $\widetilde{S_3}$ pour l'état interne du LFSR3 au temps t_0.

Détermination de l'état interne du LFSR2 Enfin, pour le registre LFSR2, on procède à une recherche exhaustive sur tous les états possibles.

(Q14) Quelle est la complexité de l'attaque pour retrouver les états S_1 et S_3 ?

(Q15) En déduire la complexité de cette attaque par corrélation. Est-elle meilleure que la recherche exhaustive ?

2.3 Challenge

Le fichier suiteGeffe.sage contient une suite de bits produits par le générateur de Geffe. Téléchargez et enregistrez ce fichier (par exemple dans le même répertoire que ce TD), et chargez le avec Sage :

```
sage: load("suiteGeffe.sage")
```

À l'aide d'une attaque par corrélation, retrouvez l'état initial des trois LFSR qui ont engendré cette suite. Calculez le temps pris par chaque étape, ainsi que le temps global de votre attaque.

Vous pouvez par exemple utiliser la fonction `cputime()` qui renvoie le nombre de secondes CPU depuis que Sage a été lancé. Utilisé avec un argument `t`, alors `cputime(t)` renvoie le temps CPU depuis "`t`".

Références

[Ber68] Elwyn R. Berlekamp. "Factoring Polynomials over finite fields". In : *Algebraic Coding Theory*. Sous la dir. d'E. R. Berlekamp. McGraw-Hill, 1968. Chap. 6.

[Gef73] Philip R Geffe. "How to Protect Data with Ciphers that are Really Hard to Break". In : *Electronics* 46.1 (1973), p. 99-101.

[Mas69] James L. Massey. "Shift-Register Synthesis and BCH Decoding". In : *IEEE Trans. Inform. Theory* 15.1 (1969), p. 122-127.

[Sie84] Thomas Siegenthaler. "Correlation-immunity of Nonlinear Combining Functions for Cryptographic Applications". In : *IEEE Trans. Inf. Theory* 30.5 (1984), p. 776-780.

[Sie85] Thomas Siegenthaler. "Decrypting a Class of Stream Ciphers Using Ciphertext Only". In : *IEEE Trans. Computers* 34.1 (1985), p. 81-85.