TD3 - Chiffrement par Flot et Cryptanalyse

Responsable : M. Bombar

1 Cryptanalyse du chiffrement de Geffe

1.1 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.

(Q1) Exprimez $z(t)$ en fonction de $s_1(t), s_2(t)$ et $s_3(t)$.

(Q2) Programmez en Sage 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, 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

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

(Q3) Quelle est la complexité linéaire de chacun des 3 LFSR internes ?

(Q4) Selon la théorie, quelle doit être la complexité linéaire du générateur de Geffe ?

(Q5) Vérifiez que Berlekamp-Massey vous retourne bien les bonnes valeurs.

1.2 Attaque par Corrélation

(Q6) 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. On modélise à chaque instant t les bits $s_1(t), s_2(t)$ et $s_3(t)$ produits par les LFSR internes par 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 $\mathbb{F}_2$.

(Q7) Montrez que

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

et

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

(Q8) 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 (Q7) démontre que la sortie du LFSR combiné de Geffe est fortement corrélée [1] à la valeur produite par les deux LFSR internes LFSR1 et LFSR3. Le but d'une attaque par corrélation est d'exploiter cette propriété pour en déduire les états initiaux de chacun des registres.

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.

Attaque par corrélation

On suppose que par une attaque à clairs connus on a réussi 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 (Q7), 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 (Q8), 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.

(Q9) Quelle est la complexité de l'attaque pour retrouver les états S_1 et S_3 ?

(Q10) En déduire la complexité de cette attaque par corrélation. Est-elle meilleure que la recherche exhaustive ?

1.3 Mise en Application

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`".

2 Challenge (*)

Après avoir rendu votre rapport d'audit de l'entreprise précédente (voir au TD2), le responsable de l'équipe de sécurité dit qu'il a appris sa leçon. En particulier, il affirme qu'il utilise maintenant un puissant LFSR, combinant quatre LFSR internes :

— LFSR1 de longueur 5 et de polynôme de rétroaction $P_1(X) = X^5 + X^2 + 1$.

— LFSR2 de longueur 9 et de polynôme de rétroaction $P_2(X) = X^9 + X^4 + 1$.

— LFSR3 de longueur 7 et de polynôme de rétroaction $P_3(X) = X^7 + X + 1$.

— LFSR4 de longueur 11 et de polynôme de rétroaction $P_4(X) = X^{11} + X^2 + 1$.

Ces quatre LFSR internes sont combinés par la fonction booléenne

$$f(x_1, x_2, x_3, x_4) = x_1x_2 + x_1 + x_2x_4 + x_5.$$

Comme ils vous l'expliquent, cette fonction n'a pas été choisie au hasard, et ils ont même démontré qu'elle n'était corrélée à aucune des 4 variables afin d'échapper à l'attaque décrite sur le générateur de Geffe.

À l'occasion d'un pentest, vous arrivez à récupérer la base de données `password.db` contenue dans le dossier `SecretDirectory`, mais celle-ci a été chiffrée grâce à ce LFSR. Dans votre rapport, vous allez rendre compte de votre intrusion, mais vous n'avez pas encore dit votre dernier mot : vous pensez toujours que ce LFSR combiné est vulnérable à une attaque par corrélation. Il vous faudra seulement être plus malins que l'attaque de Geffe.

(Q11) Retrouvez le flag qui a été caché dans `password.db`. Toute réponse partielle sérieuse sera prise en compte (mais évidemment, pas comme un challenge complet). Vous expliquerez pour cela votre démarche dans un fichier texte.
Si vous parvenez à retrouver le flag, mettez le moi dans un fichier `NOM_flag3.txt`.

2.1 Annexe : Quelques commandes SQLite

Le fichier `password.db` contient une base de donnée SQLite3. Voici quelques commandes faciles dont vous pourriez avoir besoin :

— Pour vous connecter à la base de données, exécutez la commande suivante dans un terminal :

```
sqlite3 passwords.db
```

— Une fois connectés, vous pouvez afficher les tables présentes via la commande `.tables` comme décrit ci-dessous :

```
SQLite version 3.46.1 2024-08-13 09:16:08
Enter ".help" for usage hints.
sqlite> .tables
secret_data
```

On voit par exemple que cette base de données contient une table appelée `secret_data`.

— De manière générale, les commandes à la base de données commencent généralement par un `.`

— Vous pouvez exécuter `.help` pour afficher une liste assez exhaustives de commandes disponibles.

— Vous pouvez aussi exécuter une commande SQL, qui doit nécessairement se terminer par un `;`. Par exemple

```
SELECT * FROM secret_data;
```

À vous de jouer !

Références

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

[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.