TP3 : Attaques algébriques contre les LFSR filtrés

(Sujet dû à Anne Canteaut)

24 janvier 2025

1 Attaques algébriques contre les LFSR filtrés

Nous nous concentrons ici sur les attaques algébriques contre les LFSR filtrés produisant une suite chiffrante $s = (s_t)_{t\geq 0}$ définie par :

$$s_t = f(u_{t+\gamma_1}, u_{t+\gamma_2}, \ldots, u_{t+\gamma_n}), \forall t \geq 0$$

où $u = (u_t)_{t\geq 0}$ est la suite engendrée par un LFSR de longueur L avec un polynôme de rétroaction P. Le nombre de variables d'entrée de f, n, est inférieur ou égal à la longueur du LFSR et $(\gamma_i)_{1\leq i\leq n}$ est une suite décroissante d'entiers positifs qui définit les positions des bits de l'état interne pris en entrée de la fonction de filtrage.

Par exemple, le LFSR filtré représenté dans la figure 1 génère la suite s définie par :

$$s_t = u_{t+4}u_{t+3} + u_{t+1}u_t + u_{t+4}.$$

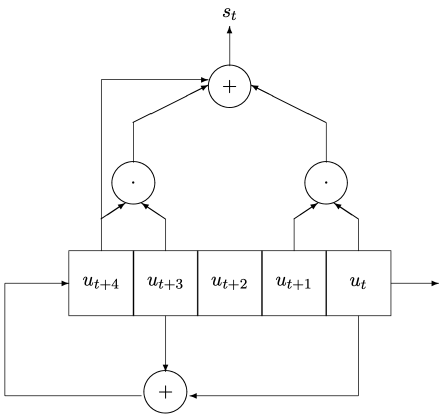

FIGURE 1 – Générateur pseudo-aléatoire par LFSR filtré

1.1 Attaque algébrique de base

Une des raisons pour lesquelles la fonction de filtrage doit avoir un degré élevé est qu'un degré faible rendrait le générateur vulnérable aux attaques algébriques de base. Ces attaques consistent à écrire chaque bit de la suite chiffrante s comme une fonction multivariée de degré $d = \deg(f)$ des L bits de l'état interne du LFSR, $u_0, \ldots, u_{L-1}$.

La connaissance de N bits de la suite chiffrante s conduit alors à un système non linéaire de N équations de degré $d = \deg(f)$ avec L inconnues. Ce système peut être résolu par des méthodes comme les bases de Gröbner ou par linéarisation, une technique plus simple mais moins efficace.

Exemple Dans le cas de la figure 1, un système quadratique en les 5 variables $u_0, \ldots, u_4$ est obtenu. Pour le linéariser, on introduit des variables comme suit :

$$x_0 = u_0, \ldots, x_4 = u_1, x_5 = u_0u_1,\ x_6 = u_0u_2, \ldots, x_{14} = u_3u_4.$$

Chaque équation de degré 2 peut alors être écrite comme une équation linéaire en $x_0, \ldots, x_{14}$, par exemple :

$$s_2 = x_5 + x_8 + x_{14} + x_{10} + x_{12} + x_1 + x_4.$$

Ensuite, la connaissance de 15 bits de la suite chiffrante conduit à un système linéaire de 15 équations avec 15 inconnues, qui peut être résolu par une élimination de Gauss. La complexité globale de l'algorithme est alors approximativement :

$$\left[\sum_{i=1}^{d} \binom{L}{i}\right]^3$$

où d est le degré de la fonction de filtrage et L est la longueur du LFSR.

1.2 Attaques algébriques évoluées

En 2003, Courtois et Meier [1] ont proposé une amélioration de cette attaque qui peut être applicable même si la fonction de filtrage a un degré élevé. Cette attaque s'applique s'il existe des relations de faible degré entre la sortie de f et ses entrées [2]. Plus précisément, l'attaquant recherche des fonctions booléennes g et h de petit degré telles que :

- $g(x_1, \ldots, x_n)f(x_1, \ldots, x_n) = 0, \forall (x_1, \ldots, x_n)$;
- $h(x_1, \ldots, x_n)[1 + f(x_1, \ldots, x_n)] = 0, \forall (x_1, \ldots, x_n)$.

Si de telles fonctions g et h de degré d existent, il est possible d'engendrer un système d'équations de degré d à partir de la suite chiffrante comme suit :

- Si $s_t = 1$, alors $g(u_t, \ldots, u_{t+L-1}) = 0$ où $(u_t, \ldots, u_{t+L-1})$ est l'état interne du LFSR à l'instant t ;
- Si $s_t = 0$, alors $h(u_t, \ldots, u_{t+L-1}) = 0$.

En écrivant l'état interne du LFSR à l'instant t comme une fonction linéaire des bits de l'état initial, on obtient comme précédemment un système d'équations de degré d en L variables.

1.3 Immunité algébrique de la fonction de filtrage

Pour éviter ces attaques, toutes les fonctions g et h satisfaisant les propriétés décrites précédemment doivent avoir un degré élevé. L'ensemble des annihilateurs de f, noté $\mathcal{AN}(f)$, est défini comme l'ensemble des fonctions g en n variables telles que $g(x_1, \ldots, x_n)f(x_1, \ldots, x_n) = 0$. L'*immunité algébrique* de f, notée $AI(f)$, est alors définie par :

$$AI(f) = \min\{\deg(g) : g \in \mathcal{AN}(f) \cup \mathcal{AN}(1 + f)\}.$$

L'objectif est de calculer cette valeur pour une fonction f donnée. On peut prouver qu'il existe toujours une fonction g dans $\mathcal{AN}(f)$ ou une fonction h dans $\mathcal{AN}(1 + f)$ de degré inférieur ou égal à $\lfloor \frac{n+1}{2} \rfloor$. Il s'agit donc de déterminer si des fonctions de degré strictement inférieur à $\lfloor \frac{n+1}{2} \rfloor$ existent dans ces deux ensembles.

Pour rechercher toutes les fonctions g de degré inférieur ou égal à un certain [1] d dans $\mathcal{AN}(f)$, nous utilisons le fait que toute fonction de ce type est une combinaison linéaire de monômes de degré inférieur ou égal à d, et qu'elle doit s'annuler en tous les points $(x_1, \ldots, x_n)$ tels que $f(x_1, \ldots, x_n) = 1$. En effet, pour tout $(x_1, \ldots, x_n)$ tel que $f(x_1, \ldots, x_n) = 1$, $g(x_1, \ldots, x_n) = 0$ si et seulement si $gf = 0$ en tous ces points.

À cet effet, nous considérons une matrice dont les lignes correspondent aux valeurs des différents monômes de degré inférieur ou égal à d en tous les points où $f = 1$. Toute combinaison linéaire des monômes qui s'annule en tous ces points correspond alors à une combinaison linéaire des lignes de cette matrice qui est égale à zéro. Une telle combinaison linéaire peut être trouvée en effectuant une élimination de Gauss.

1. En pratique $d = \lfloor \frac{n+1}{2} \rfloor - 1$ sera un choix naturel.

1.4 Exemple

Prenons un exemple simple où f est une fonction de 4 variables de degré 3 définie par :

$$f(x_1, x_2, x_3, x_4) = x_1 + x_2x_3 + x_1x_3x_4.$$

Nous voulons déterminer les fonctions de degré 2 dans $\mathcal{AN}(f)$. Puisque f est équilibrée, elle vaut 1 pour 8 des 16 mots d'entrée, à savoir :

$$S(f) = \{(1,0,0,0), (1,1,0,0), (1,0,1,0), (1,1,1,0), (1,0,0,1), (1,1,0,1), (0,1,1,1), (1,1,1,1)\}.$$

Nous construisons maintenant la matrice formée par les lignes correspondant aux 8 valeurs prises sur $S(f)$ par les monômes de degré inférieur ou égal à 2. Cette matrice est définie dans la Figure 2.

	$x_1x_2x_3x_4$							
monôme	1000	1100	1010	0110	1001	1101	0111	1111
1	1	1	1	1	1	1	1	1
x_1	1	1	1	0	1	1	0	1
x_2	0	1	0	1	0	1	1	1
x_3	0	0	1	1	0	0	1	1
x_4	0	0	0	0	1	1	1	1
x_1x_2	0	1	0	0	0	1	0	1
x_1x_3	0	0	1	0	0	0	0	1
x_1x_4	0	0	0	0	1	1	0	1
x_2x_3	0	0	0	1	0	0	1	1
x_2x_4	0	0	0	0	0	1	1	1
x_3x_4	0	0	0	0	0	0	1	1

FIGURE 2 – Matrice représentant les valeurs des monômes de degré inférieur ou égal à 2 sur $S(f)$.

Une élimination de Gauss sur cette matrice conduit à 3 lignes nulles correspondant aux fonctions suivantes :

$$1 + x_1 + x_2 + x_1x_2$$
$$1 + x_1 + x_3 + x_1x_3$$
$$1 + x_1 + x_4 + x_1x_4 + x_2x_3 + x_3x_4.$$

Nous en déduisons qu'il existe $2^3 - 1 = 7$ fonctions non nulles g de degré inférieur ou égal à 2 dans $\mathcal{AN}(f)$: elles correspondent aux combinaisons linéaires (non triviales) des 3 fonctions précédentes.

De manière similaire, l'ensemble des annihilateurs de $1+f$ est obtenu en appliquant le même algorithme à la matrice dont les colonnes correspondent aux points où $(1+f)$ vaut 1, c'est-à-dire où $f(x_1, \ldots, x_4) = 0$. Dans ce cas, nous obtenons 7 fonctions de degré 2 dans $\mathcal{AN}(1 + f)$, qui sont les combinaisons linéaires (non triviales) des fonctions :

$$x_1 + x_1x_3$$
$$x_1 + x_1x_2 + x_1x_4$$
$$x_1x_2 + x_2x_3.$$

2 Implémentation

Le programme doit prendre en arguments le nom d'un fichier contenant la forme algébrique normale de la fonction ainsi que le degré maximal d. Une valeur typique pour d est donnée par $d = \left\lfloor \frac{n+1}{2} \right\rfloor - 1$, où n est le nombre de variables de la fonction.

La fonction sera représentée dans le fichier comme suit :

- La première ligne contient le nombre de variables ;
- Les lignes suivantes donnent la forme algébrique normale de la fonction.

Par exemple :

```
4
x1 + x2x3 + x1x3x4
```

En sortie, le programme doit afficher les dimensions des idéaux $\mathcal{AN}(f)$ et $\mathcal{AN}(1+f)$, ainsi qu'une base des fonctions dans $\mathcal{AN}(f)$ (et dans $\mathcal{AN}(1+f)$) avec leurs degrés. Par exemple :

```
1 + x1 + x2 + x1x2                 degré = 2
1 + x1 + x3 + x1x3                 degré = 2
1 + x1 + x4 + x1x4 + x2x3 + x3x4   degré = 2

Dimension de AN(f) = 3

*********************************************

x1 + x1x3                          degré = 2
x1 + x1x2 + x1x4                   degré = 2
x1x2 + x2x3                        degré = 2

Dimension de AN(f+1) = 3
```

Un nombre raisonnable de variables pour f est $n \sim 10 - 15$.

2.1 Lecture de l'ANF

Cette fonction doit prendre en arguments un pointeur de type `unsigned int *` (qui correspond à l'adresse de n) ainsi que le nom du fichier contenant la forme algébrique normale (ANF). Elle retourne un tableau de 2^n éléments de type `unsigned short`, où les monômes sont représentés par des entiers comme suit : le monôme $x_{i_1}x_{i_2}\dots x_{i_d}$ est identifié par l'entier $2^{i_1-1}+2^{i_2-1}+\cdots+2^{i_d-1}$, car les variables sont numérotées ici de 1 à n.

Par exemple, la fonction à 4 variables dans l'exemple est représentée par un tableau de 16 éléments dont les éléments non nuls sont ceux d'indice $1 = 2^0$ (pour x_1), $6 = 2^2+2^1$ (pour x_2x_3) et $13 = 2^3+2^2+2^0$ (pour $x_1x_3x_4$).

2.2 Représentation de la fonction par son vecteur de valeurs

Écrire une fonction qui prend en entrée le tableau retourné par la fonction précédente et produit un tableau de type `unsigned long` correspondant au vecteur de valeurs de f. Cette fonction doit également calculer le poids de Hamming de f.

2.3 Tableau décrivant tous les monômes de degré au plus d

Puisque la matrice utilisée pour calculer l'ensemble des annihilateurs de degré au plus d est dérivée des valeurs en certains points des monômes de degré au plus d, une fonction qui définit ces monômes doit être écrite. Cette fonction doit construire un tableau dont les éléments sont les entiers correspondant à tous les monômes de n variables et de degré inférieur ou égal à d.

Par exemple, si $n = 4$ et $d = 2$, la taille de ce tableau est donnée par :

$$\sum_{i=0}^{2} \binom{4}{i} = 1 + 4 + 6 = 11,$$

et ses éléments sont 0 (pour le monôme constant 1), 1 (pour x_1), 2 (pour x_2), 4 (pour x_3), 8 (pour x_4), 3 (pour x_1x_2), 5 (pour x_1x_3), 9 (pour x_1x_4), 6 (pour x_2x_3), 10 (pour x_2x_4), 12 (pour x_3x_4).

Une fonction intermédiaire peut être écrite pour calculer les coefficients binomiaux $\binom{n}{i}$ pour $0 \le i \le d$, ainsi que leur somme, qui correspond au nombre de monômes à considérer.

2.4 Construction de la matrice

Écrire une fonction qui retourne, en tant que `unsigned long**`, la matrice dont les lignes consistent en les valeurs des monômes précédemment définis sur le support de f. Cette fonction doit prendre en arguments :

- le vecteur de valeurs de f,
- le poids de f,
- le nombre de variables n,
- le nombre de monômes,
- et le tableau définissant les monômes tel que retourné par la fonction précédente.

2.5 Élimination de Gauss

Écrire une fonction qui prend en arguments :

- la matrice décrite précédemment,
- le nombre de lignes (i.e., le nombre de monômes),
- le nombre de colonnes (i.e., le poids de la fonction),
- le nombre de variables,
- et le tableau décrivant les monômes.

Cette fonction doit retourner le nombre de lignes nulles dans la matrice après l'élimination de Gauss. Chaque fois qu'une ligne nulle est trouvée, la fonction doit afficher la forme algébrique normale de la fonction correspondante ainsi que son degré. Cette information peut être obtenue en enregistrant chaque fois qu'une ligne est remplacée par une autre.

L'algorithme utilisé pour effectuer une élimination de Gauss sur une matrice M de k lignes et m colonnes est décrit comme suit. La matrice T est utilisée pour enregistrer toutes les combinaisons linéaires effectuées par l'algorithme. T est une matrice carrée dont le nombre de lignes (et de colonnes) est égal au nombre de lignes de M. Elle est initialisée par la matrice identité. Dans la description suivante, X_i désigne la ligne i d'une matrice X.

Algorithm 1 Élimination de Gauss

```
1: examined_rows ← 0
2: for i de 0 à k − 1 do
3:    Chercher le premier indice j ≥ examined_rows tel que M_{i,j} ≠ 0.
4:    if M_{i,j} = 0 pour tout j then
5:       // La ligne i est une ligne nulle
6:       Incrémenter le nombre de lignes nulles et afficher la fonction correspondante et son degré.
7:    else
8:       if i ≠ j then
9:          Échanger les colonnes i et j dans M.
10:      end if
11:      for tous ℓ tels que i + 1 ≤ ℓ ≤ k et M_{ℓ,i} ≠ 0 do
12:         M_ℓ ← M_ℓ ⊕ M_i
13:         T_ℓ ← T_ℓ ⊕ T_i
14:      end for
15:      Incrémenter examined_rows.
16:   end if
17: end for
```

Références

[1] Nicolas T. Courtois and Willi Meier. Algebraic attacks on stream ciphers with linear feedback. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 345–359. Springer, 2003.

[2] Willi Meier, Enes Pasalic, and Claude Carlet. Algebraic attacks and decomposition of boolean functions. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 474–491. Springer, 2004.