Cryptanalyse par corrélation rapide

Sujet proposé par Nicolas Sendrier

Nicolas.Sendrier@inria.fr *Difficulté : moyen (**).*

1 Sujet

L'objet de ce projet est de mettre en oeuvre une attaque sur certains systèmes de chiffrement à flot.

1.1 Grandes lignes

On considère un générateur pseudo-aléatoire dont la sortie est constitué de n registres à décalage assemblés par une fonction booléenne (cf. FIG. 2). On souhaite pouvoir reconstituer l'initialisation de ce générateur à partir d'un petit échantillon de sa sortie.

En utilisant les défauts de la fonction booléenne, on peut établir une corrélation entre la sortie du générateur et celle d'un des registres. Ceci permet de ramener la cryptanalyse du générateur à un problème de décodage (dans le sens de la théorie des codes correcteurs d'erreurs).

Ce problème de décodage peut être résolu à l'aide de l'algorithme de Gallager [2], à condition d'avoir à sa disposition un nombre suffisant d'équations de parité de petit poids. Ces équations sont obtenues à l'aide du polynôme de rétroaction du registre lors d'une phase de pré-calcul

1.2 Le travail

Le travail demandé comportera au minimum les trois parties suivantes :

- réalisation d'un générateur pseudo-aléatoire par assemblage de registres à décalage à rétroaction linéaire (Linear Feedback Shift Register, LFSR), §2.1,
- recherche de trinômes multiples du polynôme de rétroaction d'un LFSR, §2.4,
- décodage d'une suite binaire bruitée à l'aide de l'algorithme de Gallager, §2.5

Les polynômes de rétroaction de chaque registre seront considérés comme connus, de même que la fonction de combinaison. Seules les initialisations des registres sont inconnues.

- Un premier programme devra sauver dans un fichier une suite binaire de longueur N produite par le générateur pseudo-aléatoire. Cette longueur dépend des paramètres du système, et est, en particulier, inversement proportionnelle à la corrélation.

- Un second programme déterminera l'initialisation de l'un des registres à partir de ce fichier.
- En option, on pourra écrire un programme capable de déterminer la totalité de l'initialisation.
- Des exemples de générateurs seront fournis, mais on pourra également en produire d'autres aléatoirement pour tester le programme.
- Une étude, basée sur le résultat de simulations, et ayant pour but de valider les formules données dans le texte ci-après, voire de faire toute autre constatation, sera appréciée.
- Enfin, la principale difficulté de ce projet réside dans la programmation, et pour pouvoir résoudre des problèmes de « taille réelle », il faudra porter une attention particulière à la représentation des données. À titre indicatif, les fonctions de combinaison ont usuellement entre 3 et 7 variables, les registres ont une mémoire comprise entre 20 et 60 bits, et le nombre de bits N requis pour que la cryptanalyse réussisse peut varier, en fonction de la corrélation, de quelques centaines à plusieurs dizaines milliers, voire un million dans les cas extrèmes.

2 Problématique cryptographique

2.1 Description des systèmes de chiffrement étudiés

Un système de chiffrement à clef secrète à flot (par opposition aux chiffrements par blocs) consiste à additionner bit à bit au texte clair une suite aléatoire de même longueur, appelée *suite chiffrante*. Ce système assure une sécurité parfaite sous la condition que la suite chiffrante soit une suite complètement aléatoire de la même taille que le message à chiffrer. Cependant, comme il n'est en général pas envisageable de partager une clef secrète qui soit aussi longue que le message à chiffrer, on utilise dans la pratique une *suite pseudo-aléatoire* générée de façon déterministe à partir d'un secret commun court qui, lui, peut être échangé plus facilement.

Une méthode classique pour générer une suite binaire pseudo-aléatoire est d'utiliser un registre à décalage à rétroaction linéaire (LFSR pour Linear Feedback Shift Register). Un LFSR de longueur L est composé d'un registre à décalage contenant une suite de L bits $(s_i, \dots, s_{i+L-1})$, et d'une fonction de rétroaction linéaire.

A chaque top d'horloge, le bit de poids faible s_i constitue la sortie du registre, et les autres bits sont décalés vers la droite. Le nouveau bit s_{i+L} placé dans la cellule de poids fort du registre est donné par une fonction linéaire des bits $(s_i, \dots, s_{i+L-1})$

$$s_{i+L} = c_1 s_{i+L-1} + c_2 s_{i+L-2} + \dots + c_L s_i \qquad (1)$$

où les coefficients de rétroaction $(c_i)_{1 \leq i \leq L}$ sont des éléments de $\mathbf{F}_2$, le corps à 2 éléments. Les bits $(s_0, \dots, s_{L-1})$, qui déterminent entièrement la suite, constituent *l'état initial du registre*. Nous avons

$$\sum_{i \geq 0} s_i X^i = \frac{Q(X)}{P(X)},$$

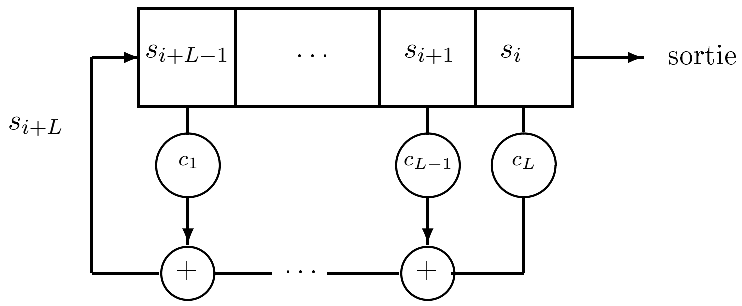

FIG. 1 – Fonctionnement d'un registre à décalage à rétroaction linéaire

où $P(X) = 1 + c_1X + \cdots + c_LX^L$ est un polynôme de $\mathbf{F}_2[X]$ de degré L, appelé *polynôme de rétroaction du registre*, et $Q(X)$ est un polynôme de $\mathbf{F}_2[X]$ de degré au plus $L-1$ dont la valeur dépend de l'initialisation. Un résultat classique est que la période de la suite binaire engendrée par un LFSR est maximale (et vaut $2^L - 1$) quand le polynôme de rétroaction est *primitif*. On se place donc dans ce cas dans la plupart des applications.

Toutefois la longueur du registre reste en pratique trop faible pour se mettre à l'abri d'une attaque à clair connu : il suffit de connaître L bits consécutifs d'un couple clair-chiffré pour retrouver l'initialisation du registre. Afin de surmonter cet obstacle, on utilise donc souvent n LFSRs en parallèle, dont les sorties sont combinées par une fonction booléenne $f : \mathbf{F}_2^n \to \mathbf{F}_2$, dite *fonction de combinaison* (ou d'assemblage).

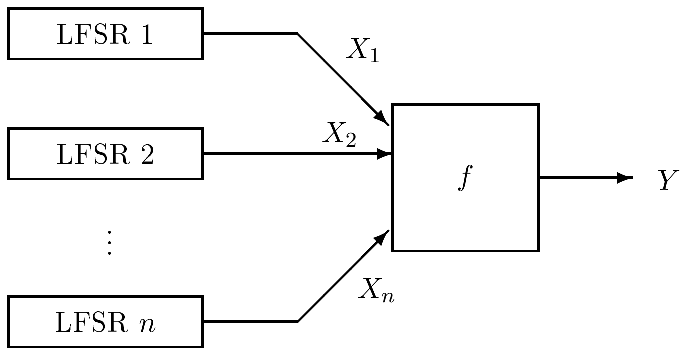

FIG. 2 – Combinaison de plusieurs LFSRs

Pour représenter la fonction booléenne f on peut utiliser soit sa table de vérité, soit sa *forme algébrique normale* : toute fonction booléenne à n variables s'écrit de manière unique sous forme d'un polynôme à coefficients dans $\mathbf{F}_2$:

$$f(x_1, \cdots, x_n) = \sum_{u \in \mathbf{F}_2^n} a_u \left(\prod_{i=1}^{n} x_i^{u_i} \right)$$

Par exemple, la fonction de Geffe [3] vaut $f(x_1, x_2, x_3) = x_1 + x_1x_2 + x_2x_3$.

Un tel système pourra être utilisé comme générateur pseudo-aléatoire dans les applications cryptographiques. La clef secrète correspond aux états initiaux des différents registres. *Tous les autres paramètres du générateur sont supposés connus*, notamment les polynômes de rétroaction des différents LFSRs et la fonction de combinaison.

2.2 Principe des attaques par corrélation

Dans toute la suite, on se place dans le contexte d'une attaque *à clair connu*. On supposera donc que l'adversaire connaît N bits de la suite chiffrante (sortie du générateur pseudo-aléatoire).

Dans ce contexte, les attaques par corrélation, introduites par Siegenthaler en 1985 [5], sont des algorithmes de type "diviser pour mieux régner", dont le but est de déterminer l'initialisation de chacun des registres indépendamment des autres. Une attaque par corrélation permet donc de retrouver l'initialisation des registres en $\sum_{i=1}^{n}(2^{L_i} - 1)$ essais, alors qu'une recherche exhaustive de la clef en nécessite $\prod_{i=1}^{n}(2^{L_i} - 1)$ (L_i désignant ici la longueur du i-ème registre).

Une attaque par corrélation exploite l'existence d'une éventuelle corrélation entre la sortie de la fonction de combinaison f et l'une de ses entrées. Elle repose sur le résultat suivant :

Théorème 1 *Soit f une fonction booléenne de combinaison à n variables. Pour $1 \leq i \leq n$, on note*

$$p_i = Pr[f(X_1, \cdots, X_n) \neq X_i] \tag{2}$$

où les X_i sont n variables aléatoires indépendantes, uniformément distribuées dans $\mathbf{F}_2$.

Soit $(s_n)_{n \geq 0}$ la suite produite par le générateur formé par la combinaison de plusieurs LFSRs par la fonction f. Supposons que N bits de la suite chiffrante sont connus. Alors la corrélation α_i entre la suite s et la sortie $s^{(i)}$ du i-ème LFSR, définie par

$$\alpha_i = \sum_{n=0}^{N-1} (1 - 2(s_n + s_n^{(i)}))$$

est une variable aléatoire gaussienne de moyenne m_i et de variance σ_i^2 avec

$$m_i = N(1 - 2p_i) \qquad et \qquad \sigma_i^2 = 4Np_i(1 - p_i) \ .$$

De même la corrélation α_0 entre la suite s et une suite aléatoire $s^{(0)}$ indépendante de $s^{(1)}, s^{(2)}, \cdots, s^{(n)}$ est une variable aléatoire gaussienne de moyenne m_0 et de variance σ_2^2 avec

$$m_0 = 0 \qquad et \qquad \sigma_0^2 = N$$

L'attaque par corrélation proposée par Siegenthaler consiste à essayer toutes les initialisations possibles pour le i-ème LFSR et à calculer pour chaque initialisation la corrélation entre la suite générée par ce LFSR et la suite s. Si l'initialisation du LFSR est incorrecte, la valeur trouvée correspond à la corrélation entre la suite s et une suite aléatoire indépendante des entrées de la fonction de combinaison. Elle est donc en moyenne proche de 0. Par contre, si l'initialisation du LFSR est exacte, la valeur de corrélation est en moyenne égale à $N(2p_i - 1)$. On peut donc distinguer par ce biais l'initialisation correcte de toutes les autres dès que $p_i \neq 0.5$.

Par exemple pour la fonction de Geffe ($f(x_1, x_2, x_3) = x_1 + x_1x_2 + x_2x_3$), on vérifie facilement que $p_1 = p_3 = 1/4$ et $p_2 = 1/2$, ce qui signifie que les registres 1 et 3 peuvent être facilement attaqués.

Pour minimiser les probabilités de fausse alarme P_f et de non-détection P_n, on teste la corrélation par rapport à un certain seuil T. Par exemple, pour

$$P_n = 1.3 \cdot 10^{-3} \qquad \text{et} \qquad P_f = \frac{1}{2^{L_i}}$$

où L_i est la longueur du i-ème LFSR, il est nécessaire de connaître N bits de la suite chiffrante où

$$N \simeq \left(\frac{\sqrt{\ln(2^{L_i - 1})} + 3\sqrt{2p_i(1-p_i)}}{\sqrt{2}(1/2 - p_i)} \right)^2 \tag{3}$$

Dans ce cas, le seuil est défini par

$$T = N(1 - 2p_i) - 6\sqrt{Np_i(1-p_i)} \tag{4}$$

Toutefois, cette attaque nécessite de passer en revue les 2^{L_i} valeurs possibles pour l'initialisation du registre. Elle devient donc hors de portée dès que les registres sont relativement longs. Il est possible de surmonter cet obstacle en utilisant une technique proposée par Meier et Staffelbach en 1988 [4] et appelée *attaque par corrélation rapide.*

2.3 Principe de l'attaque par corrélation rapide

L'attaque présentée ici est une variante de l'attaque originale de Meier et Staffelbach, qui tient compte des améliorations apportées par des travaux récents [1].

Le principe essentiel de l'attaque par corrélation rapide est d'assimiler la recherche de l'initialisation d'un registre à un problème de correction d'erreurs. On imagine que la sortie du générateur pseudo-aléatoire résulte de la transmission de la suite produite par un seul registre à travers un canal bruité. Les erreurs qui surviennent au cours de cette transmission proviennent en fait des autres registres du système. La probabilité d'erreur est donc d'autant plus élevée que les deux suites sont faiblement corrélées. Comme la suite générée par un seul registre est fortement redondante (il s'agit d'une suite engendrée par une relation de récurrence linéaire), on peut la reconstituer à l'aide d'un algorithme de décodage qui corrige les erreurs de transmission.

Pour simplifier les notations, on suppose que l'on veut retrouver l'initialisation du LFSR numéro i. On note L sa longueur, $P(X)$ son polynôme de rétroaction, σ la suite qu'il engendre et p la probabilité p_i définie par l'équation (2).

Il apparaît clairement que la suite formée par les N premiers bits de la suite chiffrante, $(s_0, \ldots, s_{N-1})$ peut être assimilée au résultat de la transmission de $(\sigma_0, \ldots, \sigma_{N-1})$ à travers un canal binaire symétrique de probabilité de transition (probabilité d'erreur) $Pr[s_n \neq \sigma_n] = p$. On supposera que $p < \frac{1}{2}$ (si $p > \frac{1}{2}$, on considérera la suite $(s_0 + 1, \ldots, s_{N-1} + 1)$).

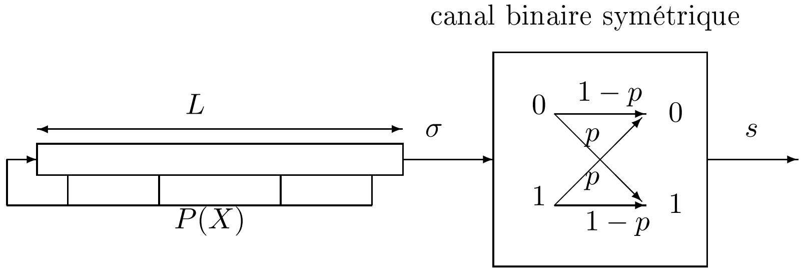

FIG. 3 – Modèle d'une attaque par corrélation rapide

La suite σ, par définition, vérifie l'équation de récurrence linéaire définie par le polynôme de rétroaction P. Un tel mot de N bits, $(\sigma_0, \ldots, \sigma_{N-1})$, appartient à un code correcteur d'erreurs linéaire, de longueur N et de dimension L, pour lequel nous disposons d'un algorithme de décodage. Cet algorithme est du à Gallager [2] et exploite l'existence d'équations de parité creuses pour ce code.

L'attaque se divise en 2 parties bien distinctes :

- **une phase de pré-calcul**, qui ne dépend que de la fonction de combinaison et du polynôme de rétroaction P. Cette phase consiste à déterminer des équations de parité de poids 3 pour la suite σ ;
- **une phase de décodage**, qui consiste à décoder la suite $(s_n)_{n<N}$ afin de retrouver $(\sigma_n)_{n<N}$.

2.4 Phase de pré-calcul

Le code linéaire C de longueur N que nous considérons est l'ensemble des mots binaires $(\sigma_0, \ldots, \sigma_{N-1})$ de longueur N produits par le LFSR de polynôme de retroaction $P(X)$. Par définition du LFSR, nous avons pour tout mot de code

$$(\sigma_0 + \sigma_1 X + \cdots + \sigma_{N-1} X^{N-1}) P(X) = Q(X) \mod X^N,$$

avec $\deg Q(X) < L$. Donc, pour tout multiple de $P(X)$ de la forme $1 + X^i + X^j$ avec $0 < i < j < N$ nous avons

$$(\sigma_0 + \sigma_1 X + \cdots + \sigma_{N-1} X^{N-1})(1 + X^i + X^j) = Q'(X) \mod X^N,$$

avec $\deg Q'(X) < j$, que l'on peut écrire sous la forme d'une équation linéaire, appelée ici *équation de parité*, vérifiée pour tout n, $j \leq n < N$,

$$\sigma_n + \sigma_{n-i} + \sigma_{n-j} = 0 \tag{5}$$

Dans la phase de pré-calcul, on recherche toutes les équations linéaires faisant intervenir exactement 3 bits de la suite $(\sigma_n)_{n<N}$. Pour calculer les multiples de $P(X)$ ayant la forme voulue, on utilisera l'**Algorithme 1**.

Dans ce qui suit, pour tout polynôme $q(X)$ de $\mathbf{F}_2[X]$, la valeur $q(2)$ sera calculée dans $\mathbf{Z}$, autrement dit $q(2)$ sera l'entier dont l'écriture en base 2 est donnée par les coefficients du polynôme $q(X)$.

Algorithme 1 – *Calculer tous les restes* $q_i(X) = X^i \bmod P(X)$ *pour* $1 \leq i < N$ *et les stocker dans un tableau* T *défini par*

$$\forall 0 \leq a < 2^L,\ T[a] = \{i, q_i(2) = a\}\ .$$

– *Pour tout* $i \in \{1, \ldots, N\}$,
calculer $A = 1 + q_i(2)$
Pour $j \in T[A]$, $1+X^i+X^j$ *est un multiple de* $P(X)$ *ayant la forme souhaitée.*

Proposition 2 *Le nombre* m *de multiples de poids 3 ainsi obtenus est de l'ordre de*

$$m \simeq \frac{N^2}{2^{L+1}} \tag{6}$$

2.5 Phase de décodage

On utilise maintenant toutes les équations de parité de poids 3 obtenues pour décoder le mot $(s_0, \ldots, s_{N-1})$. Rappelons que ce mot est égal à un mot de code $(\sigma_0, \ldots, \sigma_{N-1})$ auquel est ajouté modulo 2, position par position une erreur $(e_0, \ldots, e_{N-1})$. Chaque e_i vaudra 1 avec une probabilité $p < 1/2$. L'algorithme de décodage employé est un algorithme itératif qui repose sur le principe suivant : pour chaque position n, on calcule un rapport de vraisemblance, qui est mis à jour à chaque itération à l'aide des équations de parité.

Le *rapport (logarithmique) de vraisemblance* d'une variable aléatoire binaire u est défini par

$$L(u) = \log \frac{Pr[u = 0]}{Pr[u = 1]}\ .$$

Le signe de $L(u)$ donne la valeur la plus probable de u. Si $L(u)$ est positif, u a plus de chance de valoir 0. L'algorithme se déroule de la manière suivante :

Algorithme 2 *Soit* $(s_0, \ldots, s_{n-1})$ *une suite binaire. Pour tout* n, $0 \leq n < N$, *posons*

$$R_n = (-1)^{s_n} \log \frac{1-p}{p}.$$

Répéter jusqu'à ce que la suite s *ne varie plus :*

1. *Pour* n *variant de 0 à* $N - 1$
   (a) $R'_n \leftarrow R_n$,
   (b) *Pour toutes les équations de parité de la forme* $\sigma_n + \sigma_i + \sigma_j = 0$

$$R'_n \leftarrow R'_n + signe(R_i R_j) \min(|R_i|, |R_j|),$$

(c) Si $R_n R'_n < 0$ *alors* $s_n \leftarrow (1 - s_n)$

2. $R \leftarrow R'$.

Retourner s.

Chaque R_n doit être ici compris comme le rapport de vraisemblance $L(\sigma_n)$, et nous donne donc, si l'algorithme converge, la valeur la plus probable de la n-ième position du mot de code. La convergence de l'algorithme signifie évidemment qu'au bout d'un certain nombre d'itérations, toutes les équations de parité sont satisfaites par les s_i. À la fin de l'algorithme, les L premiers termes de la suite retournée correspondent à l'initialisation du LFSR.

Pour que l'algorithme de décodage converge, il faut cependant que l'on dispose de suffisamment d'équations de parité. On constate empiriquement que le nombre m de polynôme de la forme $1 + X^i + X^j$ nécessaires pour assurer la convergence doit vérifier

$$m \geq \frac{2}{C(p)} \tag{7}$$

où $C(p)$ est la capacité du canal binaire symétrique de probabilité d'erreur p, c'est-à-dire $C(p) = 1+p\log_2(p)+(1-p)\log_2(1-p)$. Chaque trinôme $1+X^i+X^j$, $0 < i < j < N$, multiple de $P(X)$ fournit jusqu'à 3 équations de parité faisant intervenir σ_n :

$$\begin{aligned}
\sigma_n &= \sigma_{n-i} + \sigma_{n-j} \text{ si } n - j \geq 0 \\
\sigma_n &= \sigma_{n+i} + \sigma_{n-j+i} \text{ si } n + i < N \text{ et } n - j + i \geq 0 \\
\sigma_n &= \sigma_{n+j} + \sigma_{n+j-i} \text{ si } n + j < N
\end{aligned}$$

Références

[1] Canteaut (A.) et Trabbia (M.). – Improved fast correlation attacks using parity-check equations of weight 4 and 5. *In : Advances in Cryptology - EUROCRYPT 2000*. pp. 573–588. – Springer-Verlag.

[2] Gallager (R.G.). – Low-density parity-check codes. *IRE Trans. Inform. Theory*, vol. IT-8, 1962, pp. 21–28.

[3] Geffe (P.R.). – How to protect data with ciphers that are really hard to break. *Electronics*, 1973, pp. 99–101.

[4] Meier (W.) et Staffelbach (O.). – Fast correlation attack on certain stream ciphers. *J. Cryptology*, 1989, pp. 159–176.

[5] Siegenthaler (T.). – Decrypting a class of stream ciphers using ciphertext only. *IEEE Transactions on Computers*, vol. C-34, n 1, janvier 1985, pp. 81–84.