IN603 - DM: Cryptographie Symétrique

Soursou Adrien

Avril 2020

1 Générateur de type Geffe pour le chiffrement à flot

1. Chemin du binaire: ./geffe
Pour afficher l'aide, utiliser la commande : ./geffe –help
Voir le README.txt pour plus de détails.

2. Trouver une corrélation revient à trouver un déséquilibre entre $x_0x_1x_2$ et la sortie du générateur. Le principe consiste à calculer la probabilité que $f_i = x_j$. Les 2 cas à traiter sont $P(x_j = 0)$ si $f_i = 0$ et $P(x_j = 1)$ si $f_i = 1$. Il faut une corrélation d'au moins 75 % pour effectuer une attaque. Comme chaque $f_i = \{0, 1\}$, les nombres de possibilités est de 2^8.

x_0	x_1	x_2	$F(x_0x_1x_2)$
0	0	0	$f_0 = 1$
1	0	0	$f_1 = 0$
0	1	0	$f_2 = 0$
1	1	0	$f_3 = 0$
0	0	1	$f_4 = 1$
1	0	1	$f_5 = 1$
0	1	1	$f_6 = 1$
1	1	1	$f_7 = 0$

Prenons l'exemple $f = (1, 0, 0, 0, 1, 1, 1, 0)$. Dans ce cas :

$$P(x_0 = F(x)) = \frac{1}{4} \qquad P(x_1 = F(x)) = \frac{1}{4} \qquad P(x_2 = F(x)) = \frac{3}{4}$$

La corrélation est de 75% entre x_2 et $F(x)$.
A partir de la corrélation, on sait que 75% des bits de L_2 sont égaux à ceux de la suite s.
Étant donné que $\forall i, s_i \in \{0, 1\}$, on sait aussi que 75% des bits de L_1 et L_0 sont différents de ceux de la suite s, donc :

$$P(x_0 \neq F(x)) = 1 - P(x_0 = F(x)) = \frac{3}{4} \qquad P(x_1 \neq F(x)) = 1 - P(x_1 = F(x)) = \frac{3}{4}$$

3. Le principe va consister à tester toutes les initialisations possibles de chaque registres, soit 2^{16} possibilités. Chaque registre est testé indépendamment. Pour chaque initialisation, on a besoin de vérifier que la sortie produite par le LFSR corresponde à la corrélation que l'on a trouvé

précédemment. Pour se faire, on peut utiliser l'inverse de la distance de Hamming entre la suite s et la suite $s2$ générée par le LFSR, afin de calculer le nombre de bits égaux entre les 2 suites. Si $P(x_i = F(x)) < \frac{1}{2}$, on peut inverser les bits de s pour considérer la probabilité comme étant $P(x_i \neq F(x))$.

L'algorithme est le suivant :

1. Pour chaque valeur possible du LFSR, c'est-à-dire de 0 à $2^{16} - 1$, calculer l'inverse de la distance de Hamming entre la suite s et la suite générée par le LFSR $s2$
2. Considérer l'initialisation du registre pour laquelle l'inverse de la distance de Hamming est la plus grande
3. Répéter l'opération pour les registres où $P(x_i = F(x)) \neq \frac{1}{2}$, donc les 3 registres dans notre exemple
4. Si $P(x_i = F(x)) = \frac{1}{2}$, il faut tester toutes les possibilités, nous n'avons pas à faire cette étape car x_0, x_1 et x_2 ont tous une corrélation non nulle

Il faudra évidemment prendre en considération le cas où la probabilité $P(x_i = F(x)) < \frac{1}{2}$. Dans ce cas, il faut calculer la distance de Hamming pour avoir le nombre d'occurrence où s est différent de $s2$.

4. Plus la corrélation est grande, moins il faudra de bits pour trouver la clé. Il faut un nombre de bits suffisamment grand pour deviner la clé avec précision. Si on veut être sûr de la clé, il faut utiliser un nombre de bits supérieur ou égale la taille d'un cycle, compris entre 0 et $2^{16} - 1$. On peut néanmoins considérer qu'à partir de 256 bits, la clé a statistiquement peu de chance de ne pas être la bonne. La complexité en temps est de $O(3 \cdot 2^{16})$ car il y a 3 registres de taille 16 et de $O(n)$ en mémoire, avec n la taille de la suite. La complexité pour rechercher la clé sans corrélations est en $O(2^{3 \cdot 16})$ car la clé peut avoir des valeurs entre 0 et $2^{48} - 1$. On peut donc en conclure que l'attaque par corrélation est strictement plus rapide, bien qu'elle manque de précision lorsque la taille de la suite est petite.

5. Chemin du binaire: ./geffe-cracker
Pour afficher l'aide, utiliser la commande : ./geffe-cracker –help
Voir le README.txt pour plus de détails.

6. Prenons la fonction $f = (0, 1, 1, 0, 1, 0, 0, 1)$
Les probabilités sont :

$$P(x_0 = F(x)) = \frac{1}{2} \qquad P(x_1 = F(x)) = \frac{1}{2} \qquad P(x_2 = F(x)) = \frac{1}{2}$$

Les sorties sont équilibrées. Cela implique que l'attaque par corrélation n'est plus possible.

2 Un chiffrement par bloc faible

1. $x_1^L = ((x_0^L \oplus x_0^R) <<< 7) \oplus k_0$

   x_1^L = 0x00d7818e

   $x_1^R = ((x_1^L \oplus x_0^R) <<< 7) \oplus k_1$

   x_1^R = 0x72af039a

2. $k_0 = x_1^L \oplus ((x_0^L \oplus x_0^R) <<< 7)$

   $k_1 = x_1^R \oplus ((x_1^L \oplus x_0^R) <<< 7)$

   Donc, il faut au minimum une paire de texte clair/chiffré.

   Chemin du binaire: ./feistel
Pour afficher l'aide, utiliser la commande : ./feistel –help
Voir le README.txt pour plus de détails.

3. Pour 12 tours, il faut effectuer 12 fois l'opération.

   L'équation de récurrence est la suivante :

   $k_0 = x_i^L \oplus ((x_{i-1}^L \oplus x_{i-1}^R) <<< 7)$

   $k_1 = x_i^R \oplus ((x_i^L \oplus x_{i-1}^R) <<< 7)$

   Il nous faut alors une paire de texte clair/chiffré, c'est-à-dire x_0 et x_{12}