

# Projet Crypto

BOUTON Nicolas, HADJAB Lynda, DEDARALLY Taariq

March 2020

## 1 Générateur de type Geffe pour le chiffrement à flot

### 1.1 Question 1

Voir le code geffe.c / geffe.h

On a :  $s_i = f(x_0x_1x_2)$

Donc pour chaque valeur possible de  $x_0x_1x_2$ , c-à-d  $2^3$ , on a

$$\left\{ \begin{array}{l} f(x_0x_1x_2) = f_i \\ f(000) = f_0 \\ f(100) = f_1 \\ f(010) = f_2 \\ f(110) = f_3 \\ f(001) = f_4 \\ f(101) = f_5 \\ f(011) = f_6 \\ f(111) = f_7 \end{array} \right.$$

Donc suivant les valeurs de  $f_i$  on peut retrouver  $x_0$ ,  $x_1$  et  $x_2$

- Le calcul de la corrélation entre la sortie du générateur  $s_i$  et la sortie de chaque LFSR.
- $f(0, 0, 0) = 0|1$
- $f(0, 0, 1) = 0|1$
- $f(0, 1, 0) = 0|1$
- $f(0, 1, 1) = 0|1$
- $f(1, 0, 0) = 0|1$
- $f(1, 0, 1) = 0|1$
- $f(1, 1, 0) = 0|1$

- $f(1, 1, 1) = 0|1$
- la sortie du  $s_i$  dépend de la fonction  $f$  comment elle fait ces calcul avec les 3 bits en entré ,si on veut générer une suite chiffrante de taille n alors la valeur de la suite chiffrante est dans l'intervalle  $[0.....0,1.....1]$ ;
- si on dit que la fonction f fait un xor de tous les bits, alors si tous les bits sont à 1 alors on aura  $s_i = 0$ , si on a 2 bits à 1 alors  $s_i = 0$ , sinon si un seul bit qui est à 1 alors on aura  $s_i = 1$ , donc la valeur de  $s_i$  dépend du fonctionnement de la fonction  $f$

## 1.2 Question 2

- voir test.c pour le code et res.txt pour le résultat

## 1.3 Question 3

- Définition de L'attaque diviser pour régner :
  - les attaques par corrélation, introduites par Siegenthaler en 1985 , sont des algorithmes de type “diviser pour mieux régner”, dont le but est de déterminer l'initialisation de chacun des registres (LFSR ) indépendamment des autres.
  - Une attaque par corrélation permet donc de retrouver l'initialisation des registres en  $= 1^n(2^{L_i} 1)$  essais.
  - 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.
- Exécution de l'attaque :
  - Pour l'attaque par coloration on a decider d'utiliser l'attaque par coloration rapide qui consiste en deux étape ,etape de phase de pré\_calcui et phase décodage
    - \* Phase de pré\_calcul :
      - Cette phase consiste à déterminer des équations de parité de poids 3 pour la suite  $\rho$  ;
      - $\rho$  est la sortie de chaque LFSR (tour apres tout cette valeur change)
    - \* Phase de décodage :
      - Cette phase consiste à décoder la suite  $(s_n)_n < N$  afin de retrouver  $(\rho_n)_n < N$ .
      - $s_n$  est la suite chiffrante apres le passage par la fonction f (la fonction qui permet de combiner les sorties de chaque LfSR de manière sécurisé )
      - L'entier N est le nombre de bits de mots (la clé)
      - $\rho_n$  est la suite chiffrante du taille N.

- Pour  $f = 10001110$  :
  - $f(000) = 1$
  - $f(001) = 1$
  - $f(010) = 0$
  - $f(011) = 1$
  - $f(100) = 0$
  - $f(101) = 1$
  - $f(110) = 0$
  - $f(111) = 0$
  - Corrélation  $x_0 = 25\%$
  - Corrélation  $x_1 = 25\%$
  - Corrélation  $x_2 = 75\%$
- Maintenant on sait que la sortie du 3eme LFSR correspond à 75% à la suite chiffrante
- Donc Pour toute les initialisation du 3eme LFSR on va regarder la corrélation entre la sortie de ce LFSR et la suite chiffrante, si la corrélation n'est pas de 75% on passe à une autre initialisation
- Maintenant on a trouvé l'initialisation du 3eme LFSR
- Déterminons le 1er et le 2eme LFSR
- Pour le 1er, on sait que la corrélation de la sortie du LFSR et de la suite chiffrante est de 25%, donc si on inverse la sortie du LFSR la corrélation sera de 75%. Donc comme pour le 3eme LFSR on va tester toute les initialisations possibles du LFSR et on va regarder la corrélation avec l'inverse de la sortie du LFSR et la suite chiffrante (ex: LFSR sortie : 1001, suite : 0101, corrélation entre 0110 et 0101) et si la corrélation est différente de 75% on passe à une autre initialisation
- Maintenant on connaît l'initialisation du 1er et du 3eme LFSR, on peut obtenir facilement le 2eme et testant toute les initialisations.

#### 1.4 Question 4

- nombre de bits : 1000 bits
- taille en mémoire : un int pour la suite chiffrante et 1 int pour chaque LFSR
- complexité en temps de l'attaque :  $2^{16} + 2^{16} + 2^{16} \approx 2^{17}$
- complexité en temps de la recherche exhaustive :  $2^{16} * 2^{16} * 2^{16} = 2^{48}$
- Rapport de temps :  $2^{48}/2^{17} = 2^{31}$
- donc l'attaque par corrélation est  $2^{31}$  plus rapide

## 1.5 Question 5 : Implémentations de L'attaque en C

Voir attaque\_geffe.c / attaque\_geffe.h

## 1.6 Question 6

Il faut une fonction  $f$  tel que la corrélation avec tous les bits de sorties soit de 50%.

Donc par exemple la fonction :

- $F = (f_0, f_1, f_2, f_3, f_4, f_5, f_6, f_7) = (0, 0, 0, 0, 0, 0, 0, 0)$
- $F = (f_0, f_1, f_2, f_3, f_4, f_5, f_6, f_7) = (0, 0, 1, 0, 0, 1, 0, 0)$

## 2 Exercice 2

### 2.1 Question 1

- Les fonction de chiffrement pour un seul tour sont :
  - $x_{r+1}^L = k_0 \oplus (x_r^R \oplus x_r^L) <<< 7$
  - $x_{r+1}^R = (x_r^L \oplus x_r^R) <<< 7 + K_0 \oplus x_r^R <<< 7 \oplus K_1$
  - Donc on a :
    - $x_{r+1}^L = k_0 \oplus (x_r^R \oplus x_r^L) <<< 7$
    - $x_{r+1}^R = (x_{r+1}^L \oplus x_r^R) <<< 7 \oplus K_1$
- Exemple :  $(x_0^L, x_0^R) = (0x45019824, 0x51023321)$
- Trouvons  $x_1^L, x_1^R$ , tel que  $k_0 = 0x01020304$  et  $k_1 = 0x98765432$
- $0x45019824$  en binaire c'est 01000101000000011001100000100100
- $0x51023321$  en binaire c'est 01010001000000100011001100100001
- En binaire on a  $K_0 = 00000001000000100000001100000100$  et  $K_1 = 10011000011101100101010000110010$
- $x_1^L = k_0 \oplus ((x_0^R \oplus x_0^L) <<< 7)$
- $x_1^R = k_1 \oplus ((x_0^R \oplus x_1^L) <<< 7)$
- Calcul de  $x_1^L$ 
  - calcul de  $\text{res} = x_0^R \oplus x_0^L$  on trouve 00010100000000111010101100000101
  - calcul de  $\text{res1} = \text{res} <<< 7$  on trouve  $\text{res1} = 00000001110101011000001010001010$
  - calcul de  $x_1^L = \text{res1} \oplus K_0$  on trouve  $x_1^L = 00000000110101111000000110001110$
  - la valeur en hexadécimal c'est  $x_1^L = 0x00D7818E$

- Calcul de  $x_1^R$ 
  - calcul de  $x_0^L \oplus x_0^R = res$  (on l'avait déjà calculé)
  - calcul de  $res <<< 7$  on trouve  $res1 = 00000001110101011000001010001010$
  - calcul de  $res1 \oplus K_0$  on trouve  $res2 = 0000000011010111000000110001110$
  - calcul de  $res3 = res2 \oplus x_0^R$  on trouve  $res3 = 010100011101010110110010101111$
  - calcul de  $res4 = res3 <<< 7$  on trouve  $res4 = 1110101011011001010101110101000$
  - calcul du  $x_1^R = res4 \oplus K_1$  on trouve  $x_1^R = 0111001010101110000001110011010$
  - La valeur en hexadécimal est  $x_1^R = 0x72AF039A$
- le couple  $(x_1^L, x_1^R) = (0x00D7818E, 0x72AF039A)$

## 2.2 Question 2

- Système d'équation :  $k_0$  et  $k_1$  sont des inconnues
  - $x_{r+1}^L = k_0 \oplus (x_r^R \oplus x_r^L) <<< 7$
  - $x_{r+1}^R = (x_r^L \oplus x_r^R) <<< 7 + K_0 \oplus x_r^R <<< 7 \oplus K_1$
  - Donc on a :
    - $x_{r+1}^L = k_0 \oplus (x_r^R \oplus x_r^L) <<< 7$
    - $x_{r+1}^R = (x_{r+1}^L \oplus x_r^R) <<< 7 \oplus K_1$
- Résoudre le système : Trouvons  $K_0$  et  $K_1$ 
  - on a  $A \oplus B = C$  implique  $A = C \oplus B$
  - On a  $A \oplus B = C$  implique  $B = A \oplus C$
  - $K_0 = ((x_r^R \oplus x_r^L) <<< 7) \oplus x_{r+1}^L$
  - $K_1 = x_{r+1}^R \oplus ((x_r^R \oplus x_{r+1}^L) <<< 7)$
  - Donc on a :
    - $K_0 = x_1^L \oplus ((x_0^R \oplus x_0^L) <<< 7)$
    - $K_1 = x_1^R \oplus ((x_0^R \oplus x_1^L) <<< 7)$
- Implémentions pour avoir les clé  $(K_0, K_1)$ , Voir le code src/block.c

## 2.3 Question 3

Regarder src/block.c

- $K0 = L1 \oplus ((R0 \oplus L0) <<< 7)$
- $K1 = R1 \oplus ((R0 \oplus L1) <<< 7)$
- ....

- $K0 = L12 \oplus ((R11 \oplus L11) <<< 7)$
- $K1 = R12 \oplus ((R11 \oplus L12) <<< 7)$

Donc si on prend un texte(clair/chiffre) de 12 tours et un 2ème décaler d'un tour, c'est à dire  $L1 = L0'$  et  $R1 = R0'$  alors on aura  $L12 = L11'$  et  $R12 = R11'$ . Donc on peut calculer  $K0$  et  $K1$  à partir de  $(L0, R0)$  et  $(L0', R0')$ , et on compare avec le  $K0$  et  $K1$  qu'on obtiens avec  $(L12, R12)$  et  $(L12', R12')$ . Normalement on trouve les mêmes clé car on a supposer que le deuxième texte était décaler d'un tour par rapport au premier.

## 2.4 Question 4

- Regarder le code

## 2.5 Question 5

- Non il ne le rendra pas plus solide car ca marche avec tous les nombres de tours
- Prenons n tours, on a :
- $K0 = L1 \oplus ((R0 \oplus L0) <<< 7)$
- $K1 = R1 \oplus ((R0 \oplus L1) <<< 7)$
- ....
- $K0 = LN \oplus ((RN-1 \oplus LN-1) <<< 7)$
- $K1 = RN \oplus ((RN-1 \oplus LN) <<< 7)$

Donc on a fait la même méthode on considère que pour le 2ème texte (clair/chiffré) le clair correspond au chiffré d'un tour du premier clair.

Donc on peut calculer  $K0$  et  $K1$  à partir de  $(L0, R0)$  et  $(L'_0, R'_0)$ , et on compare avec le  $K0$  et  $K1$  qu'on obtiens avec  $(LN, RN)$  et  $(L'_N, R'_N)$ . Normalement on trouve les mêmes clé car on a supposer que le deuxième texte était décaler d'un tour par rapport au premier.

## 2.6 Question 6

Il faut faire en sorte que l'on puisse pas calculer les clé avec un tour.