Projet Crypto

nicolas.bouton

March 2020

1 Générateur de type Geffe pour le chiffrement à flot

1. voir le code geffe.c, geffe.h
2. On a :
$$s_i = f(x_0x_1x_2)$$
Donc pour chaque valuer possible de $x_0x_1x_2$, c-à-d 2^3, on a
$$\begin{cases} f(x_0x_1x_2) = f_i \\ f(000) = f_0 \\ f(100) = f_1 \\ f(010) = f_2 \\ f(110) = f_3 \\ f(001) = f_4 \\ f(101) = f_5 \\ f(011) = f_6 \\ f(111) = f_7 \end{cases}$$
Donc suivant les valeurs de f_i on peut retrouver x_0, x_1 et x_2
    - le calcul de la correlation 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 a 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

1.1 Question 2

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

1.1.1 Fonction int_to_char

- Elle prend en paramètre une chaine de caracter dans laquelle écrire, le nombre a transformer en chaine, et la taille en bit du nombre
- Pour chaque itération de i, ca va prendre le bit a la position i (avec le &) et le décaler à la position 1 (avec le >>), ensuite on ajoute la valeur du caractère 0 pour avoir le caractère 0 ou 1.

1.1.2 Fonction my_pow

- Juste une fonction récursive qui calcule a^e.

1.2 Fonction fonction_f

- Calcule le bit de sorti s_i en fonction de f et $x_o x 1 x_2$
- On récupère d'abord individuellement x_0, x_1 et x_2
- Ensuite on fait les test
- Et on retourne le bon f_i en faisant en & pour prendre le bit et un >> pour le décaler à la position 1

1.2.1 Fonction main

- Tout d'abord les valeurs de f vont de $[00000000 - 11111111]$, donc on peut stoker f sur 8 bits avec un int par exemples (plus simples pour les calculs), pariel pour $x_0x_1x_2$ (sortie de chaque LFSR) qui prend une valeur dans l'intervalle $[000 - 111]$, donc ca prend 3 bits et on a aussi décidé de le stocker dans un int
- On init deux chaine pour pouvoir ensuite écrire dans un fichier les valeurs f et $x_0x_1x_2$
- Les variables nb_a, nb_b, nb_c permette de stocker combiens de fois la valeurs x_0, x_1 et x_2 sont égale à s_i respectivement.
- store et ret sont deux variables qui permette de sauvegarder le contenu de s_i, ret permet de sauvegarder une valeur de s_i et strore sauvegarde les 8 valeurs pour un f_i

- Ensuite comme f peut prendre 256 valeur on itère de 0 à 256, pareil pour n qui représente $x_0x_1x_2$ on itère de 0 à 8.
- Ensuite on applique des fonction pour écrire dans le fichier
- Dans la boucle de n, on calcule les bits de sortie et on les stocks dans ret, on fait un shift de store pour pouvoir le concatener avec le nouveau bit de sortie
- Maintenant dans store on a nos 8 bits de sortis s_i
- On va calcluez la corrélation
- Pour chaque valeur de n donc de $x_0x_1x_2$, on récupère individuellement les 3 valeurs en faisant un &, pour x_0 et x_1 on fait un shift pour le décaler à la position 1, et on prend le bit de sortie correspondant avec un & et on le décale avec un shift à la position 1, et si ca correspond on ajoute un au compteur.
- Et ensuite c'est trivial

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 ρ ;
      - ρ 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)
- · ρ_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 inatilisation 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%, doonc si on inverse la sortie du LFSR la corrélation sera de 75%. Donc comme pour le 3eme LFSR on va tester toute les initatialisations 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, correlation entre 0110 et 0101) et si la corrélation est différente de 75% on passe à une autre initialisation

- Maintenant on connait 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 : la taille du plus grand LFSR
- taille en mémoire : un int pour la suite chiffrante et 1 int pour chasue LFSR
- complexité en temps de l'attaque : $2^{16} + 2^{16} + 2^{16} \approx 2^{17}$
- complexité en temps exhaustive : $2^{16} * 2^{16} * 2^{16} = 2^{48}$
- $2^{48}/2^{17} = 2^{31}$
- donc l'attaque par corrélation est 2^{31} plus rapide

1.5 Question 5 : Implémentation de L'attaque en C

voir code

2 Exercice 2