Cryptanalyse — 4TCY902U
Responsable : G. Castagnos

TP 4 — Premières attaques sur un chiffrement à flot

On va appliquer plusieurs attaques cherchant à retrouver l'état interne d'un chiffrement à flot. Ces attaques sont génériques. Pour les tester, nous allons les appliquer sur une construction de type LFSR filtré définie comme suit. L'état interne est une chaîne S de ℓ bits. On note dans la suite $S^{(t)} = (S_0^{(t)}, \dots, S_{\ell-1}^{(t)})$, l'état interne au temps t. À l'initialisation, la clef secrète sk de ℓ bits est simplement chargée dans l'état interne : $S^{(0)} = sk$. On n'emploie pas de mécanisme d'IV.

Pour $t \geqslant 0$, la production de suite chiffrante $(z_t)_{t\in\mathbf{N}}$ et la mise à jour de l'état interne se font comme suit.

- Boucle :
  - Sortir le bit $z_t = S_0^{(t)} \oplus S_1^{(t)} \times S_2^{(t)}$ (calcul dans $\mathbf{F}_2$)
  - Mise à jour de l'état : $S^{(t+1)} = (S_1^{(t)}, S_2^{(t)}, \dots, S_{\ell-1}^{(t)}, S_0^{(t)} \oplus S_1^{(t)})$ (calcul dans $\mathbf{F}_2$)

Le chiffrement par flot est alors de type additif : le message clair est additionné bit à bit avec la suite chiffrante pour donner le chiffré : pour $t \geqslant 0$, $c_t = m_t \oplus z_t$.

$\boxed{1}$ Créer une fonction effectuant un tour de boucle : elle prend en entrée un état interne et elle retourne l'état interne mis à jour et un bit z de suite chiffrante. Créer ensuite une fonction prenant en entrée une clef secrète sk, un entier N et retournant les N premiers bits de suite chiffrante, $z_0, z_1, \dots, z_{N-1}$. On pourra considérer que les entrées et sorties des fonctions sont des éléments de `GF(2)`. Attention à la copie de liste !

Pour contrôler le bon fonctionnement de vos fonctions :

- Pour $\ell = 5$, avec la clef $0,1,0,1,0$ la suite chiffrante est $0,1,0,1,1,0,0,0,1,1\dots$
- Pour $\ell = 10$, avec la clef $1,0,1,0,1,0,1,0,1,0$ la suite chiffrante est $1,0,1,0,1,0,1,0,1,1\dots$

$\boxed{2}$ Dans la suite, on se place dans le cadre d'attaques à clair connu. Oscar récupère $n \geqslant \ell$ bits de message clair à partir du temps t_0 : $m_{t_0}, m_{t_0+1}, \dots, m_{t_0+n-1}$ ainsi que les chiffrés correspondants $c_{t_0}, c_{t_0+1}, \dots, c_{t_0+n-1}$. Donner une attaque lui permettant de décrypter les chiffrés $(c_i)_{i\geqslant t_0+n}$.

$\boxed{3}$ On pose $\ell = 5$, $t_0 = 100$ et $n := \ell + 2 = 7$. Donnez vous une clef secrète et générez l'état interne $S^{(t_0)}$ au temps t_0 et une liste Z de n bits de suite chiffrante produits à partir du temps t_0, c'est à dire $Z = z_{t_0}, z_{t_0+1}, \dots, z_{t_0+n-1}$.

On va programmer plusieurs attaques cherchant à retrouver l'état interne $S^{(t_0)}$.

$\boxed{4}$ Programmer **la recherche exhaustive** de l'état interne $S^{(t_0)}$: générer tous les états internes de $\{0,1\}^\ell$ et comparer la suite produite avec Z. Mesurer le temps nécessaire par `t = cputime() ; instructions ; print cputime(t)` pour $\ell = 5, 10, 15, 20, 21\dots$ En fonction de ℓ, quel est le coût calculatoire et la quantité de mémoire utilisée ?

$\boxed{5}$ Programmer **l'attaque par dictionnaire** : On passe par une phase de précalculs qui consiste, pour chaque état possible S à l'instant t_0, à calculer les n bits suivants de suite chiffrante, U. On considère ensuite l'entier u dont la représentation en base 2 est U. On stocke S dans un tableau à l'entrée u (on peut aussi utiliser les dictionnaires de Python). La phase active consiste à partir de la suite Z à regarder dans le tableau l'état correspondant. Mesurer le temps de cette attaque en faisant varier ℓ. En fonction de ℓ, quel est le coût calculatoire et la quantité de mémoire utilisée ?

$\boxed{6}$ On s'intéresse maintenant à **l'attaque par compromis temps mémoire**. On se donne maintenant une suite chiffrante Z de $N = \left\lceil \sqrt{2^\ell} \right\rceil + n - 1$ bits produits à partir du temps t_0. Pour la phase de précalculs, on procède comme précédemment mais avec « seulement » N états initiaux aléatoires. De même, pour la phase active, on regarde les $\left\lceil \sqrt{2^\ell} \right\rceil$ fenêtres de n bits consécutifs de Z. Si on trouve un indice correspondant dans le tableau précalculé, on aura retrouvé un état interne, en un temps $t_0 + i$ pour un certain $0 \leqslant i < \left\lceil \sqrt{2^\ell} \right\rceil$.

Programmer et vérifier le succès de cette attaque. Mesurer le temps pris en faisant varier ℓ. En fonction de ℓ, quel est le coût calculatoire et la quantité de mémoire utilisée ?

$\boxed{7}$ On veut minorer la probabilité de réussite de cette dernière attaque. Pour cela montrer la **variante du paradoxe des anniversaires** suivante :

Soit L un ensemble à m éléments. On construit un ensemble L_1 par tirages uniformes indépendants sans remise de k_1 éléments de L. On construit également un ensemble L_2 par tirages uniformes indépendants avec remise de k_2 éléments de L. Soit p la probabilité que l'on ait une collision, c'est à dire que $L_1 \cap L_2$ soit non vide. On a

$$p \geqslant 1 - e^{-\frac{k_1 k_2}{m}}.$$