Master CSI 2 2013-2014

Cryptanalyse — M1MA9W06
Responsable : G. Castagnos

TP 2 — Premières attaques sur un chiffrement à flot

On va appliquer plusieurs attaques cherchant à retrouver l'état interne d'un chiffrement à flot. Le chiffrement à flot que l'on va attaquer est une construction de type LFSR filtré. On le définit comme suit. L'état interne est une chaîne S de ℓ bits. On note dans la suite $S^{(t)} = (S_0^{(t)}, \ldots, S_{l-1}^{(t)})$, l'état interne au temps t. À l'initialisation, la clef secrète sk de ℓ bits est simplement chargée dans l'état interne : $S^{(0)} = sk$.

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 $z_t = S_0^{(t)} + S_1^{(t)} S_2^{(t)}$ (calcul dans $\mathbf{F}_2$)
  - Mise à jour de l'état : $S^{(t+1)} = (S_1^{(t)}, S_2^{(t)}, \ldots, S_{l-1}^{(t)}, S_0^{(t)} + S_1^{(t)})$ (calcul dans $\mathbf{F}_2$)

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, $z_0, z_1, \ldots, z_{N-1}$. 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 $[0,1,0,1,1,0,0,0,1,1,\ldots$
- Pour $\ell = 10$, avec la clef $1,0,1,0,1,0,1,0,1,0$ la suite chiffrante est $1,0,1,0,1,0,1,0,1,1,\ldots$

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

On va appliquer plusieurs attaques cherchant à retrouver l'état interne $S^{(t_0)}$ à partir de la suite chiffrante Z.

3 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 $\ell = 5, 10, 15, 20, 21, \ldots$. Quel est le nombre d'itérations et la quantité de mémoire utilisée ?

4 Programmer **l'attaque par dictionnaire**. La phase de précalculs consiste, pour chaque état possible S à l'instant t_0, à calculer les n premiers bits de suite chiffrante, U, et à considérer 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. Quelle est la mémoire nécessaire ?

5 On s'intéresse maintenant à **l'attaque par compromis temps mémoire**. On se donne maintenant une suite chiffrante Z de $\mathrm{N} = \lceil \sqrt{2^\ell} \rceil + n - 1$ éléments de suite chiffrante 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 $\lceil \sqrt{2^\ell} \rceil$ fenêtres de n bits consécutifs de Z. Programmer et vérifier le succès de cette attaque. Mesurer le temps pris en faisant varier ℓ. Quelle est la mémoire nécessaire ?