Master CSI 2 2013-2014

Cryptanalyse — M1MA9W06
Responsable : G. Castagnos

Devoir surveillé — 5 novembre 2012

Durée 1h30
accès aux fonctions programmées en TP, aux énoncés des TP et à la fiche d'initiation à Sage autorisés, autres documents non autorisés
Les deux exercices sont indépendants.

1 Implantation d'un LFSR avec des mots machines (exercice plutôt pratique)

(a) Donner le code d'une fonction qui simule une étape d'un LFSR : elle doit prendre en entrée un polynôme de rétroaction P de degré ℓ, un registre au temps t (de longueur ℓ) et ressortir le registre au temps $t+1$ et le bit de sortie.

(b) On considère le LFSR de longueur 20 et de polynôme de rétroaction $X^{20}+X^{10}+X^9+X^7+X^6+X^5+X^4+X+1$ initialisé par $[1,1,\ldots,1,1]$. Effectuer 10000 tours à vide (en jetant le bit de sortie) de ce LFSR, puis donner les 12 bits de sortie suivants (donner le code utilisé).

```
PR.<X> = PolynomialRing(GF(2))
P = X^20 + X^10 + X^9 + X^7 + X^6 + X^5+ X^4+ X+ 1

def LFSR_step(P, state):
   L = len(state)
   out = state[0]
   state = state[1:] + [sum(P[j+1]*state[L-1-j] for j in range(L))]
   return out, state

init = [GF(2)(1) for i in range(20)]
state = copy(init)
timing = cputime()
for i in range(10000):
   out,state = LFSR_step(P, state)
z = []
for i in range(12):
   out,state = LFSR_step(P, state)
   z += [out]
print "Time taken", cputime(timing),"s"
print z
```

retourne 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0.

On considère, dans la suite, le registre d'un LFSR de longueur ℓ à un instant t : $S^{(t)} = (S_0^{(t)}, \ldots, S_{\ell-1}^{(t)})$ et on suppose que $\ell = ds$ avec $d \geqslant 2$. De plus, on considère le registre comme d mots de s bits, c'est à dire que

$$S^{(t)} = (u_0, \ldots, u_{d-1})$$

avec $u_0 = (S_0^{(t)}, \ldots, S_{s-1}^{(t)})$, $u_1 = (S_s^{(t)}, \ldots, S_{2s-1}^{(t)}), \ldots, u_{d-1} = (S_{(d-1)s}^{(t)}, \ldots, S_{ds-1}^{(t)})$.
À l'instant $t+s$, on note de même $S^{(t+s)} = (v_0, \ldots, v_{d-1})$.

(c) Montrer comment obtenir $v_0, \ldots, v_{d-2}$ à partir de $u_0, \ldots, u_{d-1}$.

Comme on fait s itérations, le registre est décalé de s pas, donc on a $v_0 = u_1$, $v_1 = u_2, \ldots, v_{d-2} = u_{d-1}$.

(d) Pour rappel, la matrice de rétroaction d'un LFSR de longueur ℓ et de polynôme de rétroaction $P(X) = 1 + c_1 X + c_2 X^2 + \cdots + c_\ell X^\ell$ est la matrice M de taille $\ell \times \ell$ à coefficients dans $\mathbf{F}_2$:

$$M = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ \vdots & & \ddots & & \vdots \\ & & & & 0 \\ 0 & \cdots & & 0 & 1 \\ c_\ell & c_{\ell-1} & \cdots & c_2 & c_1 \end{pmatrix}$$

Pour $i \in \{0, \ldots, d-1\}$, on construit des matrices N_i de taille $s \times s$, comme sous matrice de la matrice M^s, à partir de ses s dernières lignes, de telle sorte que l'on ait M^s de la forme :

$$M^s = \begin{pmatrix} 0 & \cdots & 0 & 1 & 0 & \cdots & 0 \\ \vdots & & & \ddots & & & \vdots \\ & & & & & & 0 \\ 0 & \cdots & & & & 0 & 1 \\ N_0 & & N_1 & \cdots & & & N_{d-1} \end{pmatrix}$$

Donner le code Sage d'une fonction prenant en entrée le polynôme de rétroaction P et l'entier s et retournant la liste des matrices $N_0, \ldots, N_{d-1}$ définies ci-dessus.

```
def matrixN(P, s):
   L = P.degree()
   M = Matrix(GF(2),L,L)
   for i in range(L-1):
      M[i,i+1] = 1
   tmp = P.list()[1:]
   tmp.reverse()
   M[L-1] = vector(tmp)
   Ms = M^s
   d = L//s
   N = [0 for i in range(d)]
   for i in range(d):
      N[i] = Ms[-s:,i*s:(i+1)*s]
   return N
```

(e) Montrer que $v_{d-1} = \mathrm{N}_0 u_0 + \mathrm{N}_1 u_1 + \cdots + \mathrm{N}_{d-1} u_{d-1}$ (où l'on considère $v_{d-1}, u_0, u_1, \ldots, u_{d-1}$ comme des vecteurs colonnes).

On a en considérant les vecteurs en colonnes, $(v_0 || v_1 || \ldots || v_{d-1}) = \mathrm{M}^s(u_0 || u_1 || \ldots || u_{d-1})$ car M^s correspond à s itérations du LFSR. En particulier, v_{d-1} s'obtient en appliquant les s dernières lignes de M^s à $(u_0 || u_1 || \ldots || u_{d-1})$, ce qui donne l'égalité demandée.

(f) En déduire le code Sage d'une fonction effectuant d'un coup s étapes d'un LFSR en utilisant uniquement des opérations sur des mots de s bits : elle prend en entrée la liste des matrices N, un registre au temps t sous la forme d'une liste de d vecteurs de taille s et retourne le registre au temps $t + s$, toujours sous la forme d'une liste de d vecteurs de taille s, et s bits de sortie.

```
def LFSR_step_word(N, state):
   d = len(state)
   s = len(state[0])
   out = state[0]
   state = state[1:] + [sum(N[j] * state[j] for j in range(d))]
   return out, state
```

(g) On reprend l'exemple de la question (b). Comparer le temps mis pour ce calcul avec la fonction de la question (a) et celui avec la fonction de la question (f) en prenant $s = 4$ (Donner le code utilisé).

```
s = 4
d = 20//s
N = matrixN(P, s)
init = [vector(GF(2)(1) for i in range(s)) for j in range(d)]
state = copy(init)
zw = []
timing = cputime()
for i in range(10000//s):
   out,state = LFSR_step_word(N, state)
for i in range(12//s):
   out,state = LFSR_step_word(N, state)
   zw = zw + list(out)
print "Time taken", cputime(timing),"s"
```

La première fonction prend $1,095606s$ et celle ci $0,196597s$. On est donc beaucoup plus efficace, on le serait encore plus en utilisant des mots de la taille de l'architecture de l'ordinateur avec une implantation plus bas niveau.

2 Produit de suites (exercice théorique)

Dans tout l'exercice, on considère deux suites récurrentes z et z' de polynômes de rétroaction f et f' primitifs sur $\mathbf{F}_2$, de degré ℓ et ℓ' avec $\ell \neq \ell'$.

(a) Quelles sont les périodes et les complexités linéaires de z et z' ? Quelle est la complexité linéaire de la suite produit zz' définie par $(zz')_i = z_i z'_i$?

Comme les polynômes de rétroaction sont primitifs, les périodes sont respectivement $2^\ell - 1$ et $2^{\ell'} - 1$ et les complexités linéaires ℓ et ℓ'. Pour la suite produit, comme de plus $\ell \neq \ell'$, d'après le théorème du cours sur la complexité linéaire des LFSR combinés, la complexité linéaire est $\ell\ell'$.

Dans la suite, on s'intéresse au cas particulier $\ell = 2$ et $\ell' = 3$. On note α une racine de f dans $\mathbf{F}_{2^2}$, et α' une racine de f' dans $\mathbf{F}_{2^3}$. Les polynômes f et f' sont toujours supposés primitifs. On a : $\mathbf{F}_{2^2} = \mathbf{F}_2(\alpha)$ et $\mathbf{F}_{2^3} = \mathbf{F}_2(\alpha')$. On admet que $\mathbf{F}_{2^6} = \mathbf{F}_2(\alpha, \alpha')$.

C'est classique : $\mathbf{F}_2(\alpha, \alpha')$ est une extension de $\mathbf{F}_2(\alpha)$ de degré 3. Donc sur $\mathbf{F}_2$, c'est un extension de degré $d \leq 2 \times 3 = 6$. D'autre part, comme $\mathbf{F}_2(\alpha)$ et $\mathbf{F}_2(\alpha')$ sont contenus dans $\mathbf{F}_2(\alpha, \alpha')$, on a $2 \mid d$ et $3 \mid d$ donc $6 \mid d$ d'où $d = 6$.

(b) Quels sont les ordres de α dans $(\mathbf{F}_{2^2})^\times$ et de α' dans $(\mathbf{F}_{2^3})^\times$? Utiliser ces ordres pour montrer que α et α' appartiennent à $\mathbf{F}_2(\alpha\alpha')$. En déduire que $\mathbf{F}_2(\alpha\alpha') = \mathbf{F}_{2^6}$.

Comme les polynômes f et f' sont primitifs, on a α d'ordre $2^2 - 1 = 3$ et α' d'ordre $2^3 - 1 = 7$. Donc $(\alpha\alpha')^7 = \alpha^7 = \alpha$. Donc $\alpha \in \mathbf{F}_2(\alpha\alpha')$. De même avec α' : $(\alpha\alpha')^3 = \alpha'^3$. Puis comme 3 est premier avec 7, on met à la puissance l'inverse de 3 modulo 7 (c'est 5) pour retrouver α'.
On a donc $\mathbf{F}_2(\alpha) \subset \mathbf{F}_2(\alpha\alpha')$ et $\mathbf{F}_2(\alpha') \subset \mathbf{F}_2(\alpha\alpha')$. Donc si d est tel que $\mathbf{F}_2(\alpha\alpha') = \mathbf{F}_{2^d}$, on a $2 \mid d$ et $3 \mid d$, donc comme 2 et 3 sont premiers entre eux, $6 \mid d$. D'autre part, $\mathbf{F}_2(\alpha\alpha') \subset \mathbf{F}_2(\alpha, \alpha')$, donc $d \mid 6$, et $d = 6$.

(c) On rappelle que si $\gamma \in \mathbf{F}_{2^m}$, alors la trace de γ est $\mathrm{trace}(\gamma) = \gamma + \gamma^2 + \gamma^{2^2} + \cdots + \gamma^{2^{m-1}}$. D'après un exercice vu en TD, il existe $\beta \in \mathbf{F}_{2^2}$ et $\beta' \in \mathbf{F}_{2^3}$ tels que pour tout $i \in \mathbf{N}$, $z_i = \mathrm{trace}(\beta\alpha^{-i})$ et $z'_i = \mathrm{trace}(\beta'\alpha'^{-i})$. Montrer que pour tout $i \in \mathbf{N}$, $(zz')_i = \mathrm{trace}(\beta\beta'(\alpha\alpha')^{-i})$.

On calcule le produit de z_i par z'_i, en se servant du fait que si $\gamma \in \mathbf{F}_{2^2}, \gamma = \gamma^4 = \gamma^{16}$ et si $\gamma \in \mathbf{F}_{2^3}, \gamma = \gamma^8$, pour transformer les puissances de chaque terme :

$$z_i z'_i = (\beta\alpha^{-i} + \beta^2\alpha^{-2i})(\beta'\alpha'^{-i} + \beta'^2\alpha'^{-2i} + \beta'^4\alpha'^{-4i}) =$$

$$\beta\beta'\alpha^{-i}\alpha'^{-i} + \beta\beta'^2\alpha^{-i}\alpha'^{-2i} + \beta\beta'^4\alpha^{-i}\alpha'^{-4i} + \beta^2\beta'\alpha^{-2i}\alpha'^{-i} + \beta^2\beta'^2\alpha^{-2i}\alpha'^{-2i} + \beta^2\beta'^4\alpha^{-2i}\alpha'^{-4i} =$$
$$\beta\beta'\alpha^{-i}\alpha'^{-i} + \beta^{16}\beta'^{16}\alpha^{-16i}\alpha'^{-16i} + \beta^4\beta'^4\alpha^{-4i}\alpha'^{-4i} + \beta^8\beta'^8\alpha^{-8i}\alpha'^{-8i} + \beta^2\beta'^2\alpha^{-2i}\alpha'^{-2i} +$$
$$+\beta^{32}\beta'^{32}\alpha^{-32i}\alpha'^{-32i} = \mathrm{trace}(\beta\beta'(\alpha\alpha')^{-i}).$$

(d) En déduire la complexité linéaire de la suite zz' et sa période.

La complexité linéaire est le degré du polynôme minimal de $\alpha\alpha'$ soit $6 = 2 \times 3$, car $\mathbf{F}_2(\alpha\alpha') = \mathbf{F}_{2^6}$. La période est l'ordre de $\alpha\alpha'$ dans $(\mathbf{F}_{2^6})^\times$, et on vérifie facilement que cet ordre est 21.