MAT4216 — Introduction à la cryptologie

Examen du 9 janvier 2009, de 13h30 à 16h30, durée 3h.

Rédigez les deux parties sur des feuilles séparées.
Les paragraphes sont indépendants.
La note tiendra compte de la qualité de la rédaction.
Documents et calculatrices interdits.

Première partie — cours de Laurent Fousse

1. Chiffrements historiques

En tant que cryptanalyste, on vous transmet un cryptogramme intercepté. Il est écrit dans l'alphabet anglais à 26 lettres. Vous vous demandez s'il a été chiffré par une méthode de chiffrement polyalphabétique.

1.1. Rappelez ce qu'est un chiffrement polyalphabétique.

1.2. En supposant que le message clair est écrit en anglais, quels tests feriez vous pour confirmer cette hypothèse ?

1.3. Décrivez précisément l'attaque contre les chiffrements polyalphabétiques.

1.4. Cette méthode de chiffrement est-elle utilisable de façon sûre à l'heure actuelle ?

2. Synchronisation

2.1. Montrer que le chiffrement par flot suivant est auto-synchronisant au bout de t caractères transmis suite à une erreur :

$$\begin{aligned} \sigma_i &= (c_{i-t}, c_{i-t+1}, \ldots, c_{i-1}) \\ z_i &= g(\sigma_i, k) \\ c_i &= h(z_i, m_i) \end{aligned}$$

où l'état initial (public) est $\sigma_0 = (c_{-t}, c_{-t+1}, \ldots, c_{-1})$, k est la clef, g est la fonction générant le flux chiffrant z_i, et h est la fonction de sortie.

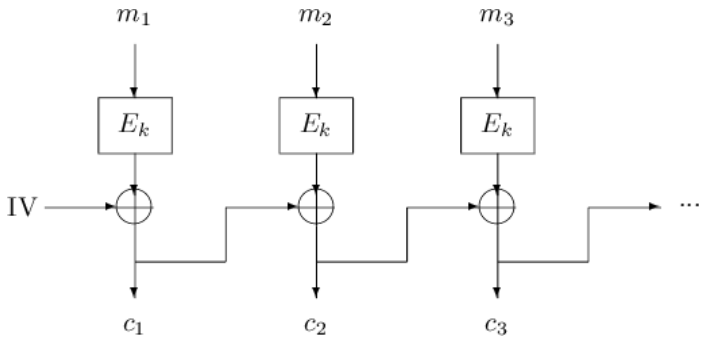
2.2. On considère ce chiffrement par flot :

$$\begin{aligned} \sigma_i &= (m_{i-t}, m_{i-t+1}, \ldots, m_{i-1}) \\ z_i &= g(\sigma_i, k) \\ c_i &= h(z_i, m_i) \end{aligned}$$

Est-il auto-synchronisant ?

3. MODES OPÉRATOIRES

On considère le mode opératoire suivant pour un chiffrement par bloc E_k :



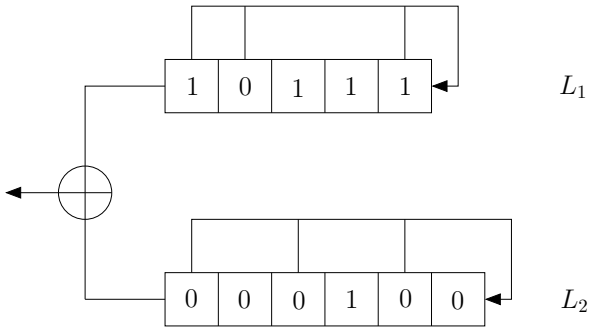
3.1. Écrire l'équation de chiffrement permettant de calculer c_i.

3.2. Écrire de même l'équation de déchiffrement permettant de retrouver m_i.

3.3. Si le bloc c_i est corrompu lors de la transmission, est-ce que le déchiffrement reprendra correctement ? Si oui, après combien de bloc mal déchiffrés ?

4. LFSR

On considère deux LFSRs (*Linear Feedback Shift Register*) L_1 et L_2 et on combine leurs sorties avec un ou exclusif (`XOR`) :



4.1. Calculer les 6 premiers bits de sortie pour L_1, pour L_2, et pour la sortie combinée.

4.2. Trouver la période de la sortie combinée, et proposer un LFSR unique la générant.

Seconde partie — cours de Michael Eisermann

5. Arithmétique modulaire

Pour $m \in \mathbb{Z}$ on note $\mathbb{Z}/_m$ l'ensemble des classes modulo m, et $\pi_m \colon \mathbb{Z} \to \mathbb{Z}/_m$ l'application quotient. On note $(\mathbb{Z}/_m)^\times$ le groupe des éléments inversibles dans $(\mathbb{Z}/_m, \cdot)$.

5.1. Énumérer les éléments du groupe $(\mathbb{Z}/_{10})^\times$ et écrire leur table de multiplication. Pour tout élément $x \in (\mathbb{Z}/_{10})^\times$ déterminer x^{10}.

5.2. Pour $p, q \in \mathbb{Z}$ expliciter l'homomorphisme naturel $\phi \colon \mathbb{Z}/_{pq} \to \mathbb{Z}/_p \times \mathbb{Z}/_q$: sous quelle condition est-il un isomorphisme ? Expliciter l'isomorphisme inverse ψ. (On ne demande pas de preuve, mais vous voulez peut-être vérifier votre réponse.)

5.3. Soit $p \in \mathbb{N}$ premier et soit $e \in \mathbb{Z}$, $e \geq 1$. Quel est l'ordre du groupe $(\mathbb{Z}/_{p^e})^\times$? Comment déterminer l'ordre du groupe $(\mathbb{Z}/_n)^\times$ pour $n \in \mathbb{N}$ quelconque ?

5.4. Énoncer le théorème de Lagrange sur l'ordre des sous-groupes H d'un groupe G. Que dire de l'ordre des éléments du groupe G ? (On ne demande pas de preuve.)

5.5. Décomposer $\mathbb{Z}/_{100}$ d'après le théorème chinois. Que peut-on en déduire pour le groupe $(\mathbb{Z}/_{100})^\times$? Quel est son cardinal ? Pour tout élément $x \in (\mathbb{Z}/_{100})^\times$ déterminer x^{100}. Justifiez votre réponse, brièvement mais suffisamment.

6. L'algorithme d'Euclide-Bézout

6.1. Énoncer l'algorithme d'Euclide-Bézout vu en cours : cet algorithme calcule itérativement le pgcd positif d de deux entiers $a_0, b_0 \in \mathbb{Z}$ ainsi que deux coefficients de Bézout $u, v \in \mathbb{Z}$ tel que $d = ua_0 + vb_0$ (voir la spécification ci-dessous).

Algorithme 1 Euclide–Bézout

Entrée: deux entiers $a_0, b_0 \in \mathbb{Z}$

Sortie: trois entiers d, u, v tels que
- d soit le pgcd positif de a_0 et b_0, et
- l'identité de Bézout $d = a_0 u + b_0 v$ soit satisfaite.

Rappeler sa preuve :

6.2. Pourquoi l'algorithme se termine-t-il ?
(Expliciter une mesure de complexité dans $\mathbb{N}$ qui décroît.)

6.3. Pourquoi l'algorithme renvoie-t-il un résultat correct ?
(Expliciter un invariant qui prouve votre affirmation.)

6.4. Quelle est la complexité de cet algorithme en nombre d'itérations ?
(Votre mesure de complexité, à quelle vitesse décroît-elle ?)

7. Analyse d'un algorithme

L'algorithme 2 prétend de calculer rapidement le pgcd de deux entiers $a_0, b_0 \in \mathbb{Z}$ sans jamais effectuer de division euclidienne.

Algorithme 2 pgcd binaire

Entrée: deux entiers $a_0, b_0 \in \mathbb{Z}$

Sortie: le pgcd positif de a_0 et b_0

$a \leftarrow |a_0|$, $b \leftarrow |b_0|$, $s \leftarrow 1$
tant que a et b sont pairs **faire** $a \leftarrow a/2$, $b \leftarrow b/2$, $s \leftarrow s \cdot 2$ **fin tant que**
tant que a est pair **faire** $a \leftarrow a/2$ **fin tant que**
tant que $b \neq 0$ **faire**
  tant que b est pair **faire** $b \leftarrow b/2$ **fin tant que**
  si $a > b$ **alors** échanger $a \leftrightarrow b$ **fin si**
  $b \leftarrow (b-a)/2$
fin tant que
retourner $a \cdot s$

On pourra considérer les développements binaires et leur longueur, c'est-à-dire le nombre de bits : pour $a \in \mathbb{N}$ on pose $\mathrm{len}(a) := \min\{n \in \mathbb{N} \mid a < 2^n\}$. Ainsi on a $\mathrm{len}\,0 = 0$, $\mathrm{len}\,1 = 1$, $\mathrm{len}\,2 = \mathrm{len}\,3 = 2$, $\mathrm{len}\,4 = \cdots = \mathrm{len}\,7 = 3$, $\mathrm{len}\,8 = \cdots = \mathrm{len}\,15 = 4$, etc.

7.1. Pour quelles données $a_0, b_0 \in \mathbb{Z}$ l'algorithme 2 se termine-t-il ?
Expliciter une mesure de complexité $\chi(a,b) \in \mathbb{N}$ qui décroît.

7.2. Pour quelles données $a_0, b_0 \in \mathbb{Z}$ l'algorithme 2 renvoie-t-il le résultat correct ?
Expliciter un invariant $\iota(a,b) \in \mathbb{N}$ qui prouve votre affirmation.

7.3. Rectifier l'algorithme 2, d'une manière simple, afin d'inclure les cas où il ne se termine pas ou bien ne renvoie pas le résultat correct.

7.4. Quelle est la complexité de l'algorithme 2 en nombre d'itérations ?
Votre mesure de complexité, à quelle vitesse décroît-elle ?

7.5. Voyez-vous d'éventuels avantages de l'algorithme 2 « pgcd binaire » vis-à-vis de l'algorithme d'Euclide présenté en cours ?