1 Primalité, ordre, racine carrée.

Exercice 1

Écrire un test de non primalité basé sur l'identité $a^{p-1} = 1 \pmod p$ pour p premier et a premier avec p. Déterminer les entiers non premiers plus petits que 1000 pour lesquels le test renvoie vrai. Majorer la complexité de cette recherche.

Exercice 2

Programmer le test de Miller-Rabin. Vérifier qu'il détecte les nombres non premiers non détectés dans l'exercice 1.

Exercice 3

Soit p un nombre premier tel que la factorisation de $p-1$ soit connue. Déterminer un élément primitif de $\mathbb{Z}/p\mathbb{Z}$. Indication : prendre a au hasard, vérifier que $a^{(p-1)/d} \neq 1 \pmod p$ pour tous les diviseurs premiers d de $p-1$.

Exercice 4

Trouver des nombres premiers p de la forme $1 + k2^{25}$. Déterminer une racine primitive 2^{25}-ième de l'unité. A quelle condition peut-on calculer le produit de deux polynômes à coefficients dans $\mathbb{Z}/p\mathbb{Z}$ par FFT ?
Le faire sur un exemple en utilisant l'instruction `fft` avec 3 arguments.

Exercice 5

Déterminer les racines carrées de 1235 modulo 12347.

2 RSA

Rappel du principe de codage RSA :

- Chaque personne souhaitant coder ou signer un message dispose d'une clef privée, un entier s connu de lui seul, et d'une clef publique, une paire d'entiers (c, n).
- n est le produit de 2 nombres premiers p et q, et s et c sont inverses modulo $\varphi(n)$, où $\varphi(n) = (p-1)(q-1)$ (le nombre d'entiers de l'intervalle $[1, n[$ premiers avec n),

$$(a^c \pmod n)^s \pmod n = (a^c \pmod n)^s \pmod n = a \pmod n \tag{1}$$

- Pour coder un message à destination d'une personne dont la clef publique est (c, n), on commence par le transformer en une suite d'entiers, On peut par exemple remplacer chaque caractère par un entier compris entre 0 et 255, son code ASCII.
- Puis on envoie la liste des nombres $b = a^c \pmod n$. En principe, seule la personne destinataire connait s et peut donc retrouver a à partir de b en calculant $b^s \pmod n$.
- On peut aussi authentifier qu'on est l'auteur d'un message en le codant avec sa clef privée, tout le monde pouvant le décoder avec la clef publique.

Exercice 1 : Générer une paire de clefs

Génèrez deux nombres premiers p et $q > 256$ au hasard, en utilisant par exemple les fonctions `nextprime` et `rand`. Calculez $n = p \times q$ puis $\varphi(n) = (p-1)(q-1)$ puis choisissez une clef secrète s inversible modulo $\varphi(n)$ et calculez son inverse c. Vérifiez sur quelques entiers la propriété (1), on utilisera la fonction `powmod(a,c,n)` pour calculer $a^c \pmod n$.

Exercice 2 : Codage et décodage d'un message

On transforme une chaine de caractère en une liste d'entiers et réciproquement avec `asc` et `char`. Pour l'appliquer à une liste `l`, on peut utiliser `map(l,powmod,c,n)`. En utilisant la paire de clefs de l'exercice 1, codez un message puis décodez ce message pour vérifier. Décodez le message authentifié situé à l'URL
`http://www-fourier.ujf-grenoble.fr/~parisse/mat249/rsa1`.

Exercice 3 : Attaque simple

L'utilisation de 256 valeurs possibles pour a se prête à une attaque très simple : la personne souhaitant décoder un message codé avec une clef publique sans en connaitre la clef secrète calcule simplement la liste des $a^c \pmod n$ pour les 256 valeurs possibles de a et compare au message. Décodez de cette manière le message situé à l'URL
`http://www-fourier.ujf-grenoble.fr/~parisse/mat249/rsa2`

Exercice 4 : Groupement de lettres

Pour parer à cette attaque, on va augmenter le nombre de valeurs possibles de a pour que le calcul de la liste de toutes les puissances des a possibles soit trop long. Pour cela, on groupe par paquets de x caractères et on associe à un groupe de caractères l'entier correspondant en base 256. Par exemple, si on prend des groupes de $x = 3$ caractères, "ABC" devient `65*256^2+66*256+67` car le code ASCII de A, B, C est respectivement 65, 66, 67. Donner une condition reliant n et x pour que le décodage redonne le message original. Choisissez une paire de clefs vérifiant cette condition pour $x = 8$. Ecrire un programme de codage et de décodage avec groupement (on commencera par compléter le message original par des espaces pour qu'il soit un multiple de 8 caractères, l'instruction `size` permet de connaitre la taille d'une chaine de caractères, on pourra utiliser la fonction d'écriture en base de la feuille d'exercices 2).

Exercice 5 : Sécurité du codage 1

Montrer que la connaissance de $\varphi(n)$ et de n permet de calculer p et q par résolution d'une équation de degré 2. La sécurité du codage repose donc sur la difficulté de factoriser n. Tester sur des entiers de taille croissante le temps nécessaire au logiciel pour factoriser p et q. Une valeur de n de taille 128 bits, 512 bits, 1024 bits parait-elle suffisante ?

Exercice 6 : Sécurité du codage 2

Le choix de c et de s est aussi important. Pour le comprendre, prenons $p = 11$ et $q = 13$. Représentez pour différentes valeurs de c les points $(a, a^c \pmod n)$, plus le dessin obtenu est aléatoire, plus il sera difficile à une personne mal intentionnée de déchiffrer un message sans connaitre la clef. On pourra utiliser les instructions `seq` pour générer une suite de terme général exprimé en fonction d'une variable formelle, et `scatterplot(l)` qui représente le nuage de points donné par une liste `l` de couples de coordonnées. Observez en particulier les cas où c n'est pas premier avec $\varphi(n)$ et $c = 3$. Conclusions ?

Textes du jury sur RSA

`http://agreg.org/Textes/583.pdf`
`http://agreg.org/Textes/public2018-C2.pdf`