TP 3 : Cryptographie, tests de primalité

Exercice 1.

On représente les 26 lettres (majuscules) de l'alphabet en utilisant les entiers de 1 à 26 (on utilise 0 pour l'espace). On utilise les entiers de 27 à 52 pour les lettres minuscules. On dispose ainsi d'un « alphabet modifié » à 53 caractères. Pour des raisons techniques dans les fonctions ci-dessous, il est commode d'ajouter un caractère non significatif au début. On obtient ainsi un alphabet à 54 symboles.

On veut, en fait, étant donné un (grand) entier n, pouvoir faire correspondre de manière unique un élément de $\mathbf{Z}/n\mathbf{Z}$ à un mot donné. Cela n'est possible que si le mot en question a au plus k lettres : $w_0 w_1 \cdots w_{k-1}$ où k est le plus grand entier tel que $54^k \leqslant n$. L'entier correspondant à ce mot est alors $\sum_{i=0}^{k-1} w_i 54^i$. Si l'on veut envoyer des mots plus longs, on découpe le message en une suite de mots ayant tous au plus k lettres.

En utilisant ce principe, on peut écrire une fonction `encode` prenant en entrée un mot m (ou une phrase), écrit entre guillements simples, et un entier k et renvoyant une liste de nombres ayant tous au plus k chiffres en base 54.

Réciproquement on peut, selon le même principe, écrire une procédure `decode` prenant pour entrées un entier N (en base 10) et renvoyant en sortie la signification de N (en toutes lettres).

1. Copier et tester les fonctions `encode` et `decode` à partir de l'adresse

   `www.math.u-bordeaux.fr/~fjouve001/DonneesRSA.rtf`

   (Remplace l'extension `.rtf` par `.txt` si cela ne fonctionne pas.) Ce fichier contient aussi l'alphabet à 54 caractères décrit ci-dessus. Il est également à copier-coller dans votre feuille de calcul.
2. Écrire une procédure `RSA(p,q)` prenant en entrée deux nombres premiers $p \neq q$ et renvoyant un couple clé publique/clé privée RSA : $(n = pq, e)$, $(\varphi(n), d)$. (Ici φ désigne la fonction indicatrice d'Euler.) Tester cette procédure pour générer des couples clé publique/clé privée à partir de nombres premiers p et q à 6 chiffres.
3. Écrire une procédure `encrypte(m,cle_pub)` prenant en entrée une phrase en toutes lettres m et une clé publique RSA (n, e) et renvoyant le message chiffré associé à m par le cryptosystème RSA relatif à (n, e). On pourra chercher à optimiser la longueur de chaque bloc de manière à ce que la liste de nombres obtenus ait le moins d'éléments possible. On pourra utiliser la syntaxe python `n, e = cle_pub` pour récupérer les deux parties de la clé au début de la fonction.

   Utiliser `encrypte` et une clé publique RSA créée à partir de nombres premiers à 6 chiffres pour chiffrer la phrase : `'les vacances approchent sourions un peu'`.
4. Écrire une procédure `decrypte(L, cle_priv)` prenant en entrée une liste L de nombres et une clé privée RSA $(\varphi(n), d)$ et renvoyant en sortie le message clair (en toutes lettres) correspondant.

   Tester `decrypte` sur la liste obtenue en sortie à la question précédente (avec bien entendu une clé privée correspondant à la clé publique de la question précédente), et retrouver la phrase en toutes lettres de départ.
5. Écrire, en utilisant `factor`, une procédure `casse(cle_pub)` pour casser une clé RSA, c'est à dire passer d'une clé publique (n, e) à la clé privée (n, d) correspondante.
6. Tester les procédures précédentes sur les clés publiques RSA (n_i, e_i), $i = 1, 2, 3$, et les listes de messages chiffrés trouvés à l'url ci-dessus.

Exercice 2. (*Cryptosystème de Rabin* ; exercice du TP2, ne pas refaire les questions déjà traitées)

On présente un système de cryptographie proche de RSA mais basé cette fois ci sur le fait qu'il est d'une part « facile »de calculer des racines carrées modulo p, pour p premier, mais d'autre part « difficile »de calculer des racines carrées modulo $n = pq$, où p et q sont des nombres premiers distincts « grands », si l'on ne connait que le produit n et pas ses facteurs p et q.

On fixe p et q deux nombres premiers congrus à 3 modulo 4.

1. Justifier que si ℓ est un nombre premier impair quelconque et si y n'est pas multiple de ℓ alors $y^{(\ell-1)/2} \equiv \pm 1 (\mathrm{mod} \ell)$.
2. Soient m et x des entiers vérifiant $m^2 \equiv x (\mathrm{mod} p)$. Quelle est la classe de congruence de $x^{(p+1)/4}$ modulo p ?
3. Alice veut envoyer à Bob un message clair $m \in \mathbf{Z}/n\mathbf{Z}$ via le système de Rabin : n est la clé publique de Bob (et le couple (p, q) est sa clé privée, connue de lui seul). Alice envoie alors le message chiffré $x \equiv m^2 (\mathrm{mod} n)$ à Bob. En utilisant la question précédente, expliquer comment Bob peut facilement déduire un ensemble $\mathcal{M}$ de 4 valeurs possibles pour le message initial m.
4. Écrire une procédure `ValPossibles(x,p,q)` prenant en entrée le triplet (x, p, q) où x est un carré modulo pq et renvoyant un ensemble $\mathcal{M}$ de 4 valeurs possibles pour un m vérifiant $m^2 \equiv x (\mathrm{mod}\, pq)$. (Pour le théorème des restes chinois, on pourra utiliser la fonction `crt`.)
5. Bob doit enfin arriver à déterminer lequel des 4 éléments de $\mathcal{M}$ est le « vrai »message initial. Pour l'aider, Alice inclut des données redondantes dans son message clair m : par exemple si elle veut envoyer 123 (écrit, disons, en base 54) elle enverra le message à 6 chiffres (toujours en base 54) $m = 123123$. On peut montrer que la probabilité qu'au moins 2 des 4 éléments de l'ensemble $\mathcal{M}$ obtenu par Bob présente ce type de redondance est très faible.
   Décoder la liste de mots ci-dessous (disponible ici :
   `www.math.u-bordeaux.fr/~fjouve001/DonneesEx2.txt`)
   chiffrée en utilisant d'une part la même correspondance lettre/chiffre (en base 54) que dans l'exercice précédent, et d'autre part le système de Rabin à clé publique
   $$n = 60002900033$$
   incluant le type de redondance donné en exemple ci-dessus :
   $$56502641148,\ 43664647296,\ 35860583742,\ 11494938440,$$
   $$53734232928,\ 44689084848,\ 42170931037,\ 28932214703,\ 10533212706\,.$$
6. Quelle taille doit on choisir pour p et q si l'on veut envoyer des mots de 5 lettres (avant ajout de redondance) ?

Exercice 3. (*Test de primalité de Miller-Rabin* ; exercice du TP2, ne pas refaire les questions déjà traitées)
Soit p un nombre premier impair.

1. Justifier que si $a^2 \equiv 1 (\mathrm{mod}\, p)$ alors $a \equiv 1 (\mathrm{mod}\, p)$ ou $a \equiv -1 (\mathrm{mod}\, p)$.
2. On écrit $p - 1 = 2^s t$ où s, t sont des entiers et t est impair. Soit a un entier qui n'est pas multiple de p et soit $A = a^t$. Justifier que A est inversible modulo p.
   On note α l'ordre de A en tant qu'élément de $(\mathbf{Z}/p\mathbf{Z})^\times$.
3. Montrer que α divise 2^s. Dans la suite on notera $\alpha = 2^j$.
4. Montrer que si $j = 0$ alors $a^t \equiv 1 (\mathrm{mod}\, p)$.
5. Montrer que si $j \geqslant 1$ alors $a^{t2^{j-1}} \equiv -1 (\mathrm{mod}\, p)$.
6. On considère le pseudo code suivant qui décrit le test de Miller-Rabin
   Entrée : un nombre impair n et un nombre entier a premier à n.
   (1) Calculer s et t tels que $n - 1 = 2^s \times t$, avec t impair.
   (2) Si $a^t \equiv 1 (\mathrm{mod}\, n)$, la procédure s'arrête et l'on renvoie « vrai ».
   (3) Pour i de 0 à $s - 1$:
   si $a^{t2^i} \equiv -1 (\mathrm{mod}\, n)$ alors la procédure s'arrête et l'on renvoie « vrai ».
   (4) On renvoie « faux ».

   Que conclure si l'algorithme renvoie « vrai » ? Et s'il renvoie « faux » ?
7. Implanter le pseudo code ci-dessus en une fonction `MillerRabin(n, a)` et le tester sur quelques valeurs de a et n.

Exercice 4. (*Un calcul de transformée de Fourier discrète*)
Soit $K = \mathbb{Z}/41\mathbb{Z}$.

1. Montrer que $\omega = 14$ est une racine primitive 8-ème de 1 dans K.
2. On note $\eta = \omega^2$ et $f = x^7 + 2x^6 + 3x^4 + 2x + 6 \in K[x]$. En utilisant une seule étape de récursion et l'évaluation en des puissances de η, calculer $\alpha = \mathrm{TFD}_\omega(f)$.
3. Calculer de même $\beta = \mathrm{TFD}_\omega(g)$ où $g = x^7 + 12x^5 + 35x^3 + 1 \in K[X]$. Calculer le produit coordonnée à coordonnée $\alpha \cdot \beta$.
4. Déduire $f *_8 g$ par transformation de Fourier inverse (on pourra commencer par calculer l'inverse de ω dans K). Comparer avec la valeur du produit fg.