SÉANCE 6. LE PROTOCOLE RSA

Les messages que l'on veut transmettre sont constitués des vingt-six lettres de l'alphabet (majuscules, disons), des quatre symboles de ponctuation `{ , . , ? , !}` et des dix chiffres de 0 à 9 ; on utilise donc quarante signes au total. Nous avons choisi de ne pas recourir aux lettres minuscules et aux autres signes de ponctuations afin de limiter le nombre de signes employés ; si vous souhaitez le faire, il n'y a pas de difficulté à adapter ce qui suit.

Exercice 1 (Codage et décodage) — On choisit de faire correspondre à chacun des 40 signes utilisés un nombre spécifique entre 0 et 39 ; par exemple, on peut opter pour le choix suivant :

_	A	B	…	Z	.	?	!	0	1	…	9
0	1	2	…	26	27	28	29	30	31	…	39

Un message M de q signes devient donc une suite $x_0, x_1, \dots, x_{q-1}$ de q éléments de $\{0,\dots,39\}$. Cette suite est ensuite convertie en un nombre entier $n(M)$ via l'écriture en base 40 :

$$n(M) = \sum_{k=0}^{q-1} x_k 40^k.$$

Par exemple, si M est le message « LYON 1 », alors

$$n(M) = 12 + 25\cdot 40 + 15\cdot 40^2 + 14\cdot 40^3 + 31\cdot 40^5 = 3\,175\,321\,012.$$

1. Sur Python, définir une fonction `Codage(`M`)` qui calcule le nombre entier $n(M)$ associé au message M.
2. Définir ensuite une fonction `Decodage(`n`)`, qui transforme le nombre entier n en un message $M(n)$, de telle façon que l'on ait `Decodage(Codage(`M`))` $= M$ pour tout message M.
3. Soit $N \geqslant 1$ un nombre entier. Si l'on ne veut utiliser que des entiers dans $\{0,\dots,N-1\}$, il est nécessaire de remplacer un entier trop grand par la liste de ses chiffres en base N. Définir une fonction `Coder(`M`,`N`)` qui associe au message M la liste des chiffres en base N de l'entier $n(M)$, puis une fonction `Decoder(`l`,`N`)` qui reconstitue le message à partir de la liste l.

Exercice 2 (Le protocole RSA) — Soit p et q deux nombres premiers distincts. Posons $N = pq$, ce qui entraîne $\varphi(N) = (p-1)(q-1)$. Choisissons enfin un nombre entier e premier à $\varphi(N)$.

1. Justifier l'existence d'un entier $d > 0$ tel que $ed \equiv 1 \pmod{\varphi(N)}$.
2. Définir une fonction `Inverse(`e`,`n`)`, pour deux entiers e et n premiers entre eux, renvoyant un entier *positif*, inverse modulo de e modulo n.

   Remarquer que, si $ex_0 + ny_0 = 1$, *alors* $e(x_0 + kn) + n(y_0 - ke) = 1$ *pour tout nombre entier* k.
3. Considérons les applications

$$C\colon \begin{array}{ccc} \mathbf{Z}/N\mathbf{Z} & \to & \mathbf{Z}/N\mathbf{Z} \\ x & \mapsto & x^e \end{array} \quad \text{et} \quad D\colon \begin{array}{ccc} \mathbf{Z}/N\mathbf{Z} & \to & \mathbf{Z}/N\mathbf{Z} \\ x & \mapsto & x^d \end{array}.$$

Démontrer que l'on a

$$D \circ C(x) = x$$

pour tout $x \in \mathbf{Z}/N\mathbf{Z}$.

Utiliser le théorème des restes chinois.

4. Définir une fonction `Puissance(l,k,N)`, associant à une liste $l = [u_0, u_1, \ldots, u_s]$ d'entiers entre 0 et $N-1$ la liste formée des restes modulo N de $u_0^k, u_1^k, \ldots, u_s^k$.

   Utiliser l'exponentiation rapide vue lors de la séance 4.

5. Définir des fonctions `Chiffrer(M)` et `Dechiffrer(l)` réalisant le protocole RSA. La première s'applique à un message M et renvoie une liste d'entiers dans $\{0, \ldots, N-1\}$; la seconde s'applique à une liste l d'entiers dans $\{0, \ldots, N-1\}$ et renvoie un texte.

6. La factorisation $N = pq$ permet de calculer $\varphi(N)$. Réciproquement, démontrer que l'on peut aisément déterminer p et q à partir de N et $\varphi(N)$.

   Observer que l'on a $\varphi(N) = pq - (p+q) + 1$...

   Il est donc aussi difficile de factoriser N que de calculer $\varphi(N)$.

Le couple (N, e) est la *clef publique* : elle permet à tout le monde de chiffrer un message. L'entier d n'est connu que des personnes habilitées à décrypter les messages chiffrés ; cela suppose évidemment de ne pas divulguer $\varphi(N)$ ou l'un des facteurs premiers de N.

7. Illustration numérique : vérifier que les entiers

   $$p = 440334654777631 \quad \text{et} \quad q = 145295143558111$$

   sont des nombres premiers, puis que

   $$e = 1193$$

   est premier à $(p-1)(q-1)$.

   (i) Utiliser ces données pour chiffrer le message de votre choix.

   (ii) Vérifier que vous pouvez le déchiffrer.

   (iii) Que se passe-t-il si vous modifiez un chiffre dans le message chiffré, puis que vous le déchiffrez ?

8. Avec Python, parvenez-vous à retrouver la factorisation de N ? Si oui, en combien de temps ?

Remarque — L'utilisation du protocole RSA repose sur la construction de (très) grands nombres premiers (ayant au moins une centaine de chiffres décimaux). Nous verrons lors d'une prochaine séance comment en produire.

Exercice 3 (Une attaque massive) — Voici une clef publique :

$$(N, e) = (61\,838\,105\,010\,447\,916\,231, 1000003).$$

Elle a été utilisée pour chiffrer le code d'une carte bancaire (quatre chiffres) :

$$[39\,871\,735\,737\,866\,571\,309].$$

Pouvez-vous déterminer le code de cette carte bancaire en factorisant N ? Et en comparant systématiquement ce résultat au chiffrement d'un nombre de quatre chiffres ?