INFORMATIQUE PSI P2B

Année 2024/25

TP n° 3

Cryptographie

Table des matières

Vincent ROHART

http://vrohart.e-monsite.com/

1 Le chiffrement de César et l'analyse fréquentielle

Si vous êtes à l'aise avec `Python` *vous pouvez passer cette section.*

1. Avant toute chose, programmer une fonction de signature `toutenmaj(M)` où M est une chaîne de caractères, qui renvoie la même chaîne de caractères où chaque minuscules (de `'a'` à `'z'`) est transformée en sa majuscule correspondante. Les autres caractères (espace, apostrophe, etc.) restant inchangés.

   Exemple. L'appel de `toutenmaj("Bonjour l'ami !")` devra renvoyer `"BONJOUR L'AMI !"`.

2. Programmer une fonction de signature `cesar(M,k)` où
   — `M` est une chaîne de caractères ne comportant pas de minuscules,
   — `k` est un entier compris entre 0 et 25,
   qui renvoie la chaîne de caractères obtenue après un décalage de k (par exemple, si $k = 2$, le message `'ABCD'` sera transformé en `'CDEF'`). Attention les caractères qui ne sont pas des lettres ne devront pas être transformés.

3. Programmer une fonction de signature `freq(M)` où `M` est une chaîne de caractères ne comportant pas de minuscules, qui renvoie un dictionnaire dont les clés sont les lettres de l'alphabet (de `'A'` à `'Z'`) et dont les valeurs sont des fréquence (en pourcentage) d'apparition de ces lettres.

   Exemple. L'appel de `toutenmaj("BONJOUR L'AMI !")` devra renvoyer

   ```
   {"A":9.09, "B":9.09, "I":9.09, "J":9.09, "L":9.09, "M":9.09, "N":9.09,
                        "O":18.18, "R":9.09, "U":9.09}
   ```

4. Faire une analyse fréquentielle sur le texte du fichier `message-crypte-cesar.tex` en ligne et le déchiffrer. Bonus : trouver le nom de l'œuvre dont est extrait ce texte.

2 Le chiffrement de Vigenère

1. Proposer une fonction de signature `vigenere(M,K)` où `M` est une chaîne de caractères ne comportant que des lettres majuscules, sans espace, et où `M` est une chaîne de caractère (la clé), qui renvoie le message `M` chiffré par le code de Vigenère grâce à la clé `K`.
2. Écrire alors une fonction `dech_vig(N,K)` qui décrypte le message `N` en connaissant la clé `K`.
3. Proposer un ensemble de fonctions mettant en place le principe de Babbage-Kasiski.
4. Déchiffrer le texte du fichier `message-crypte-vigenere.tex` en ligne et le déchiffrer. Bonus : trouver le nom de l'œuvre dont est extrait ce texte.

3 Le chiffrement RSA

1. Créer une fonction de signature `inv_mod(a,n)` qui prend en argument deux entiers naturels a et n (où $n \neq 0$) qui renvoie, si $\text{pgcd}(a, n) = 1$, l'inverse de a modulo n, c'est-à-dire l'unique entier b de $[\![1, n-1]\!]$ tel que $ab \equiv 1\ [n]$. Si $\text{pgcd}(a, n) \neq 1$, la fonction renverra un message d'erreur.
2. Écrire une fonction `ascii(M)` qui transforme le texte `M` en utilisant le code ASCII.

   Exemple. L'appel `ascii("Bonjour l'ami !")` devra renvoyer `"66 111 110 106 111 117 114 32 108 39 97 109 105 32 33"`
3. Écrire une fonction de signature `chiff_RSA(M,n,e)` qui chiffre un message `M` (chaîne de caractères) selon la méthode RSA avec la clé publique `(n,e)`, où `n` est le produit de deux nombres premiers de votre choix (pas trop grands). Se reporter au cours pour savoir qui est `e`.
4. Chiffrer le message « Gare ! Les cornichons sont parmi nous ! ».
5. Proposer une fonction de déchiffrement connaissant l'exposant `d` (censé rester secret).

4 Fonction à sens unique sur $\mathbb{F}_p$

Soit p un nombre premier.

1. L'exponentiation rapide consiste, pour calculer x^n, à remarquer que

$$x^n = \begin{cases} \left(x^{\frac{n}{2}}\right)^2 & \text{si } n \in 2\mathbb{N} \\ x\left(x^{\frac{n-1}{2}}\right)^2 & \text{sinon.} \end{cases}$$

   Programmer de façon récursive une fonction `exporap(x,n,p)` qui renvoie x^n si $x \in \mathbb{F}_p$ et $n \in \mathbb{N}$.
2. Écrire une fonction `primitif(p)` qui renvoie un élément primitif de $\mathbb{F}_p$.
3. Écrire une fonction `log_p(y,p)` qui renvoie l'unique $n \in \mathbb{F}_p$ tel que $x^n = y$, où x est un élément primitif de $\mathbb{F}_p$, puis comparer le temps d'exécution de `exporap` et `log_p` quand p est un nombre premier assez grand.

5 Cryptographie sur une courbe elliptique $(*)$

1. Soit p un nombre premier. Créer une fonction en `Python` permettant de savoir si les entier a et b définissent une courbe elliptique d'équation $y^2 = x^3 + ax + b$.
2. Le module `ecc.py` permet de définir la somme de deux points d'une courbe elliptique. Lire son contenu et le tester en calculant la somme de deux points de votre choix.
3. Créer une fonction à sens unique sur une courbe elliptique et mettre en place un procédé de Diffie-Hellman.