Corrigé

# Cryptographie à clef secrète

## Chiffrement de Vigenère

**Question 1.** On définit les fonctions :

```
def add(x, y):
    i, j = alph.index(x), alph.index(y)
    return alph[(i + j) % 26]

def sub(x, y):
    i, j = alph.index(x), alph.index(y)
    return alph[(i - j) % 26]
```

puis :

```
def chiffre(clef, message):
    s = ''
    for i in range(len(message)):
        s = s + add(message[i], clef[i % len(clef)])
    return s

def dechiffre(clef, message):
    s = ''
    for i in range(len(message)):
        s = s + sub(message[i], clef[i % len(clef)])
    return s
```

Le premier message une fois déchiffré se révèle être la *Chanson d'automne* de Paul Verlaine.

## Attaque du chiffrement de Vigenère

**Question 2.** Il s'agit de parcourir les lignes du fichier `hugo.txt` en ne prenant en compte que les caractères présents dans la chaîne `alph`. Dans le script qui suit, l'entier $n$ désigne le nombre de caractères de la sorte présents dans le fichier.

```
s = open('hugo.txt', 'r')
francais = [0] * 26
n = 0
for l in s:
    for c in l:
        if c in alph:
            n += 1
            francais[alph.index(c)] += 1
for i in range(26):
    francais[i] /= n
s.close()
```

**Question 3.** On procède de la même façon avec une chaîne de caractères, si ce n'est qu'ici on peut supposer que tous les caractères sont présents dans `alph`.

```
def frequence(texte):
    f = [0] * 26
    for c in texte:
        f[alph.index(c)] += 1
    for i in range(26):
        f[i] /= len(texte)
    return f
```

**Question 4.** On observe que $\sum_i (x_i - \overline{x})(y_i - \overline{y}) = \sum_i x_i y_i - n\overline{x}\,\overline{y}$, $\sum_i (x_i - \overline{x})^2 = \sum_i x_i^2 - n\overline{x}^2$ et $\sum_i (y_i - \overline{y})^2 = \sum_i y_i^2 - n\overline{y}^2$, ce qui permet un calcul en un seul parcours des deux listes X et Y :

```
def correlation(X, Y):
    n = len(X)
    sx = sy = sx2 = sy2 = sxy = 0
    for x, y in zip(X, Y):
        sx += x
        sy += y
        sx2 += x * x
        sy2 += y * y
        sxy += x * y
    mx, my = sx / n, sy / n
    return (sxy - n * mx * my) / sqrt(sx2 - n * mx * mx) / sqrt(sy2 - n * my * my)
```

**Question 5.** Pour chacune des lettres $x$ de la chaîne `alph` on dechiffre le message $M_k$ à l'aide de la clef $x$ puis on calcule le coefficient de corrélation du message obtenu avec la langue française.

```
def clefL(message, l):
    clef = ''
    for k in range(l):
        m = -1
        for x in alph:
            c = correlation(francais, frequence(dechiffre(x, message[k::l])))
            if c > m:
                m, d = c, x
        clef += d
    return clef
```

Cette fonction permet de trouver la clef du second message : `NERVAL`, et le message une fois déchiffré se révèle être le poème *El Desdichado* de Gérard de Nerval.

**Question 6.** Pour chaque valeur de $\ell$ on calcule à l'aide de la fonction précédente la clef la plus vraisemblable puis on calcule le coefficient de corrélation avec la langue française du message décrypté avec cette clef. On garde la clef associée au coefficient de corrélation maximal.

```
def clef(message):
    cmax = 0
    for l in range(4, 21):
        clef = clefL(message, l)
        decode = dechiffre(clef, message)
        c = correlation(francais, frequence(decode))
        if c > cmax:
            cmax, clemax = c, clef
    return clemax
```

Le troisième message est chiffré à l'aide de la clef : `APOLLINAIRE` et se révèle être le poème *Le Pont Mirabeau*, de Guillaume Apollinaire.