Pablo Rauzy <pr@up8.edu>
https://pablorauzy.fr/teaching/crypto

Introduction à la cryptologie

TP 1 : Cryptographie historique

Ce TP a été conçu par Julien Lavauzelle. L'original est accessible sur `https://www.lvzl.fr/teaching.html`.

Dans ce TP, les objectifs sont d'implanter :
- le chiffrement de César une généralisation possible, le chiffrement affine ;
- des attaques statistiques sur ces chiffrements ;
- le chiffrement de Vigenère ;
- une attaque statistique sur ce chiffrement.

Exercice 0.

Recommandations.

1. Ce TP est à faire en Python.
2. Faites les questions dans l'ordre et pensez à tester votre code.
3. N'hésitez jamais à rajouter de la sortie de debug pour comprendre tout ce qui se passe.
4. Téléchargez les fichiers à décrypter à l'adresse suivante :

   `https://pablorauzy.fr/teaching/crypto/02-tp-crypto-historique.zip`

Exercice 1.

Chiffrement affine

1. Dans cet exercice, on souhaite **implanter et attaquer un chiffrement affine simple**. On considère un ensemble de symboles (l'alphabet) formé des 26 lettres courantes en majuscule `A-Z`, et de trois caractères spéciaux : l'espace « », l'apostrophe « ' » et le point « . ».

   On souhaite encoder ces 29 caractères avec les entiers allant de 0 à 28, en commençant par le `A` et en terminant par le point.

   → Écrire des fonctions `encode` et `decode`, qui permettent respectivement d'encoder un caractère de l'alphabet (à 29 élément) en l'entier correspondant, et de décoder un entier en un caractère, selon la règle d'encodage énoncée ci-dessus.

2. Le chiffrement de Cesar vu en cours s'applique aussi bien sur l'alphabet classique à 26 éléments, que sur notre alphabet à 29 éléments. Par exemple, avec la clé `K` (caractère numéro 10), le texte clair « `HELLO WORLD.` » devient le chiffré « `ROVVYHDY'VNJ` ».

   → Écrire des fonctions `chiffre_cesar` et `dechiffre_cesar`, qui permettent respectivement de chiffrer un texte clair et de déchiffrer un texte chiffré avec une clé représentée sous la forme d'un entier compris entre 0 et 28.

3. Dans notre alphabet à 29 éléments, la distribution des caractères est différente de celle présentée en cours, car on a ajouté trois éléments dont le caractère d'espacement qui est très courant. Avec cet alphabet à 29 caractères, les caractères les plus fréquents sont les suivants :

caractère	espace	E	A	S	I
fréquence	0.165	0.136	0.076	0.069	0.065

   → Écrire une fonction `plus_frequent` qui retourne le caractère le plus fréquent d'un texte.

4. → Retrouver le texte clair associé au texte chiffré par un chiffrement de Cesar, qui est donné dans le fichier `enc_cesar.txt`.

5. On s'intéresse maintenant à une autre catégorie de chiffrement, appelée « chiffrement affine ». L'idée est de généraliser le chiffrement de Cesar à d'autres types de substitutions de lettres. Pour Cesar, la substitution était donnée par :

   $$x \mapsto x + \Delta \mod 29$$

   où x est la lettre à chiffrer, et Δ est la clé de décalage.

Pour un chiffrement affine, la permutation est de la forme :

$$x \mapsto ax + b \mod 29$$

où le couple (a, b) forme la clé secrète, avec comme seule contrainte que a soit être différent de 0. Pour déchiffrer, on doit alors appliquer la permutation inverse, à savoir :

$$y \mapsto (y - b) \cdot a^{-1} \mod 29$$

Remarque. Pour obtenir l'inverse modulaire « $a^{-1} \mod 29$ », en `python` il suffit d'exécuter `pow(a, -1, 29)`.

→ Écrire des fonctions `chiffre_affine` et `dechiffre_affine`, qui permettent respectivement de chiffrer un texte clair et de déchiffrer un texte chiffré avec une clé `cle` représentée sous la forme d'un couple d'entiers.

6. → Vérifier que le chiffré de `INFORMATIQUE` par la clé $(a = 13, b = 12)$ est `AHTUBXM'ARLG`, puis que le déchiffrement se déroule correctement.
7. Pour retrouver la clé Δ du chiffrement de Cesar, il a suffi de reconnaître le chiffré d'une seule lettre (en l'occurence, la plus fréquente). Pour retrouver la clé du chiffrement affine, qui est formée de deux entiers (a, b), il nous faudra la connaissance du chiffrement de deux lettres.

   Supposons que l'on connaisse y le chiffré d'un caractère x, et y' le chiffré de x'. Alors on a le système de deux équations à deux inconnues (a, b) :

$$\begin{cases} y = ax + b \mod 29 \\ y' = ax' + b \mod 29 \end{cases}$$

   que l'on sait résoudre pour obtenir

$$a = (y - y') \cdot (x - x')^{-1} \mod 29\,, \qquad \text{puis} \qquad b = y - ax \mod 29\,.$$

   → Écrire une fonction `deux_plus_frequents` qui retourne les deux caractères les plus fréquents d'un texte.
8. → Retrouver le texte clair associé au texte chiffré par un chiffrement affine, qui est donné dans le fichier `enc_affine.txt` du dossier `affine`.

Exercice 2.

Attaque sur le chiffrement de Vigenère.

1. On souhaite attaquer le chiffrement de Vigenère tel que vu en cours, c'est-à-dire avec un alphabet de 26 lettres.

   → Implanter les fonctions de chiffrement et de déchiffrement de Vigenère, puis tester ces fonctions sur un exemple.
2. L'indice de coincidence d'un texte est donné par la formule :

$$\texttt{IC} = \frac{1}{N(N-1)} \sum_{i=1}^{26} M_i(M_i - 1)$$

   où N est le nombre de caractères dans le texte, et M_i est le nombre de fois que le i-ème caractère apparaît dans le texte.

   → Implanter une fonction `IC` qui calcule l'indice de coincidence d'un texte.
3. On rappelle que, sur l'alphabet que l'on a considéré, l'indice de coincidence du français est de 0.078 tandis que celui d'un texte aléatoire est de 0.038. L'algorithme suivant doit donc permettre de deviner la longueur de la clé :

   Data : un chiffré C obtenu par une clé de longueur L (a priori inconnue)
   Result : la longueur L
   `test` $\leftarrow 0$
   $L \leftarrow 0$
   while `test` < 0.07 **do**
   $L \leftarrow L + 1$
   Former le morceau de texte $T \leftarrow$ `C[0] C[L] C[2L] C[3L]` $\cdots\cdots$
   `test` $\leftarrow \mathrm{IC}(T)$
   end
   return L

   → Implanter une fonction `calcule_longueur_cle` qui prend en entrée un texte chiffré, et qui retourne la longueur de clé probable utilisée pour obtenir ce chiffré.
4. → Implanter (ou réutiliser de l'exercice précédent) une fonction `calcule_decalage` qui calcule le décalage probable d'un morceau de texte chiffré avec le chiffrement de Cesar.
5. → Implanter une fonction `attaque_vigenere` qui décrypte un chiffré de Vigenère.
6. → Décrypter les chiffrés contenus dans les fichiers `file<i>_enc.txt` du dossier `vigenere`.