

# Attaques cryptographiques

## CR432 - Cybersécurité avec Python

Les étudiants doivent remettre un code illustrant la solution des deux parties. Le code en question devrait être bien commenté. Les étudiants doivent aussi remettre une vidéo à vive voix pour démontrer les réponses émises dans leurs codes. (Code et commentaires : 70 points, Explication via démo : 30 points)

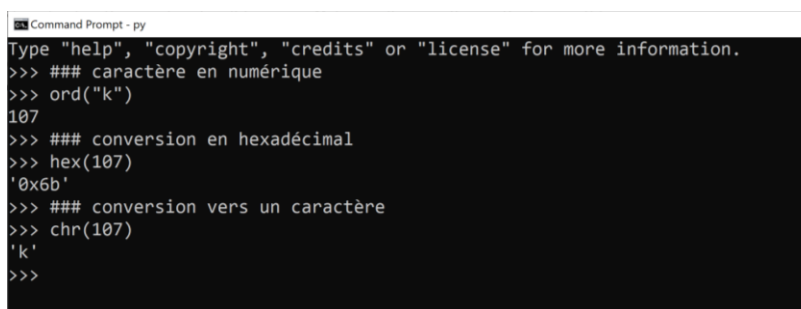
### Partie 1

Supposant que vous êtes un analyste crypto, vous avez intercepté le message crypté suivant:

Message = "gpkyfe zj r kilcp nfeuviwlc crexlrsv"

Vous suspectez que l'entité envoyant le message, a envoyé un code Caesar (César). Votre but est de trouver le message associé en clair (décrypté). Veuillez considérer ces suppositions :

1. Comme supposition de départ, on aura juste à utiliser des caractères ASCII en minuscules [a-z]. Les autres caractères sont inchangés lors du chiffrement.
2. Chaque caractère est codé sur un octet. À titre de rappel, un octet comprend 8 bits et une certaine représentation est utilisée pour les caractères.
3. Dans ce devoir, on utilisera la notation hexadécimale : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E et F, l'idée est d'avoir 16 chiffres sur 4 bits ( $2^4=16$ ).
4. Le code python suivant illustre des représentations des caractères en hexadécimales (e.g., « k ») est illustré dans la Figure 1.



```
Command Prompt - py
Type "help", "copyright", "credits" or "license" for more information.
>>> ### caractère en numérique
>>> ord("k")
107
>>> ### conversion en hexadécimal
>>> hex(107)
'0x6b'
>>> ### conversion vers un caractère
>>> chr(107)
'k'
>>>
```

Figure 1 : Conversions

5. Pour comprendre le chiffage du code Caesar, il s'agit d'utiliser un chiffre comme une clé. En considérant la figure suivante, la clé est égale à « 2 », la lettre « a » va être chiffré en « c », la lettre « b » va être chiffré en « d », etc. (voir Figure 2)

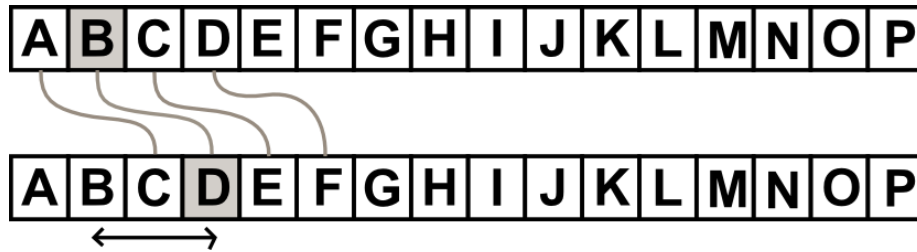


Figure 2 : illustration du code Caesar

#### Question 1 (5 points)

Veillez écrire une fonction « encrypt » qui prend comme entrées un texte (clair) composé d'une suite de caractères ASCII [a-z] et d'une clé et qui retourne le texte chiffré via le code de Cæsar.

#### Question 2 (5 points)

Veillez écrire une fonction « decrypt » qui prend comme entrées un texte chiffré et d'une clé et qui retourne le texte en clair via le code de Cæsar.

#### Question 3 (5 points)

Quelle est la relation entre le chiffrement et le déchiffrement ?

#### Question 4 (5 points)

En utilisant le module « *argparse* », veuillez créer un paramétrage pour votre script de tel sorte qu'il pourra prendre une chaîne de caractères en entrée ou un fichier contenant du texte et retourner en sortie un message ou un fichier chiffré.

#### Question 5 (5 points)

Veillez ajouter à votre script une fonction « brute\_force », qui prendra comme entrée un message chiffré et qui retournera un dictionnaire ayant comme clés les valeurs de clé de chiffrement ainsi que comme valeurs les valeurs des textes en clair (e.g.,  $\{k_0 : \text{texte\_clair}_0, k_1 : \text{texte\_clair}_1, \dots, k_n : \text{texte\_clair}_n\}$ )

#### Question 6 (5 points)

En testant la fonction « brute\_force » avec le message intercepté, veuillez déduire la valeur de la clé ainsi que le message en clair.

### Partie 2

En étant analyste, vous intercepté un fichier chiffré, ou vous suspectez que le texte est chiffré avec des valeurs de substitution sur un octets (1 byte). A titre de rappel, veuillez vous référer au code illustré dans la Figure 3.

```
Python 3.11.1 (tags/v3.11.1:a7a450f, Dec 6 2022, 19:58:39) [MSC v.1934 64 bit (AMD64)] on win
Type "help", "copyright", "credits" or "license()" for more information.
>>> import binascii
>>> ord('.')
46
>>> hex(46)
'0x2e'
>>> bytes('.', 'utf-8')
b'.'
```

Sachant que les lettres utilisées dans ce problème sont les lettres de l'alphabet en majuscule et minuscule, chiffres ainsi que les diverses ponctuations, simple espace et « \n ». Tous les symboles sont encodés en « utf-8 ». Le chiffrement en substitution consiste à utiliser une clé sous forme de dictionnaire, où chaque symboles (e.g., lettres) correspond une valeur de substitution d'un octet (8 bits). Voici un exemple, pour « a » qui est chiffré en « , » : { 'a' : b'2c', ... }. Les valeurs sont tirées au hasard pour chaque lettre et sont censés chiffrer un seul symbole. Il est impossible d'avoir une même clé pour deux ou plusieurs symboles différents.

Même si une attaque de type brute force est une option, on préférera faire une analyse cryptographique. De ce fait, on va calculer la fréquence des caractères dans un texte en clair - représentatif dans notre cas de la langue anglaise - ainsi que sur un texte chiffré. En supposant que la distribution des fréquences est semblable dans les deux textes, on pourra faire correspondre le symbole chiffré avec la plus haute fréquence avec le symbole en clair qui a la plus haute fréquence.

### Question 7 (5 points)

Veillez écrire une fonction « `frq_txt` » qui prend comme paramètre d'entrée un texte (représenté sous forme de chaîne de caractères) (voir le texte dans le fichier « `english_sample.txt` ») et retournera une liste de dictionnaires ou de tuples (votre choix), où chaque symbole est associé avec le nombre d'occurrence associé. La liste doit être ordonnée à partir du caractère le plus fréquent, jusqu'aux caractères les moins fréquents.

### Question 8 (5 points)

Veillez écrire une fonction « `frq_cpr` » qui prend comme paramètre d'entrée une séquence d'octets (voir le fichier « `cipher_sample.bin` ») et retournera une liste de dictionnaires ou de tuples (votre choix), ou chaque symbole est associé avec le nombre d'occurrence associé. La liste doit être ordonnée à partir du caractère le plus fréquent, jusqu'aux caractères les moins fréquents.

### Question 9 (5 points)

En faisant une correspondance entre les deux listes ordonnées, veuillez faire le parallèle entre le texte en clair et celui qui est chiffré. (Pas de code Python, il faudrait une explication)

**Question 10 (5 points)**

Vu que le chiffrement se fait via une clé sous forme de dictionnaire (e.g., `key={'a' : b'2d', ...}`), le déchiffrement se fait avec une clé inversée (e.g., `key={b'2d' : 'a', ...}`), en fonction de l'observation faite dans la question précédente, veuillez écrire une fonction « `build_dec_key` » qui va recouvrir la clé de déchiffrement partielle.

**Question 11 (5 points)**

Veuillez implémenter une fonction « `guess_txt` » ayant comme entrées le texte chiffré ainsi qu'une clé de déchiffrement partielle des caractères les plus fréquents faite sur la base des fréquences du texte de base en anglais. Si un caractère ne peut pas être déchiffré, vous pourrez faire la correspondance par « `$` ». La longueur de la clé partielle avec laquelle le test se fera est 17.

**Question 12 (5 points, suite de la question 11)**

Veuillez tester la fonction « `guess_txt` » sur le fichier « `cipher_sample.bin` »

**Question 13 (5 points)**

Veuillez déduire des mots ou des phrases à partir de l'analyse des fréquences du fichier « `english_sample.txt` », cela vous permettra de recouvrir la clé de déchiffrement

**Question 14 (5 points)**

Veuillez refaire le test en utilisant la clé de déchiffrement sur le fichier « `test.bin` » et en déduire la maximum de mots en clair