SAE Cryptographie

Binôme :

- Daniel Moreira 22
- Cassandra Maupou 23

Premier indice donné

Décryptage du premier message

Voici, ci-dessous, le premier indice qui nous a été donné :

> BDQE PG OTQYUZ EQ OMOTQ GZ FDQEAD
>
> MOODAOTQ M GZ MDNDQ FAGF DQOAGHQDF P'AD
>
> ZQ ZQSXUSQ BME XM VQGZQ BAGOQ RQGUXXG
>
> SDMZP QEF EAZ EQODQF YMXSDQ EM FMUXXQ YQZGQ
>
> DAZPQE QF OAXADQQE EAZF XQE NMUQE CG'UX BADFQ
>
> MZUEQQE QF EGODQQE, XQGDE EMHQGDE EAZF RADFQE.
>
> YMUE MFFQZFUAZ M ZQ BME XQE ODACGQD,
>
> YQYQ EU XM RMUY FUDMUXXQ FQE QZFDMUXXQE,
>
> QZ MGOGZ OME FG ZQ PAUE EGOOAYNQD

Toutes les phrases chiffrées ci-dessous utilisent le chiffrement de César. En décalant toutes les lettres de 12, nous pouvons obtenir le message en clair :

> PRES DU CHEMIN SE CACHE UN TRESOR
>
> ACCROCHE A UN ARBRE TOUT RECOUVERT D'OR
>
> NE NEGLIGE PAS LA JEUNE POUCE FEUILLU
>
> GRAND EST SON SECRET MALGRE SA TAILLE MENUE
>
> RONDES ET COLOREES SONT LES BAIES QU'IL PORTE
>
> ANISEES ET SUCREES, LEURS SAVEURS SONT FORTES.
>
> MAIS ATTENTION A NE PAS LES CROQUER,
>
> MEME SI LA FAIM TIRAILLE TES ENTRAILLES,
>
> EN AUCUN CAS TU NE DOIS SUCCOMBER

Pour retrouver le message en clair nous avons écrit une fonction permettant de déchiffrer le chiffre de César :

```
def decrypt_cesar(ciphertext, shift):
    decrypted_text = ""
    for char in ciphertext:
        if char.isalpha():
            shifted = ord(char) - shift
            if shifted < ord('A'):
                shifted += 26
            decrypted_char = chr(shifted)
            decrypted_text += decrypted_char
        else:
            decrypted_text += char
    return decrypted_text
```

Dans cette fonction nous mettons en paramètres : le texte que l'on souhaite déchiffrer et le chiffre de César, nous instancions également une variable locale nommée `decrypted_text` qui contiendra le message en clair. La fonction va ensuite vérifier pour chaque caractère présent dans le texte s'il s'agit, ou non, d'une lettre. S'il s'agit bien d'une lettre, nous allons stocker la différence de son code Unicode et du chiffre de César, que nous avions passé en paramètres, dans une variable locale nommée `shifted`. Nous regardons ensuite si la valeur stockée dans la variable locale est inférieure à la valeur Unicode de la lettre A pour éviter de gérer des caractères ne faisant pas parti de l'alphabet. Si oui, on rajoute 26 à la valeur stockée. Enfin, nous récupérons le caractère correspondant à la valeur Unicode stockée dans `shifted` et nous l'ajoutons dans `decrypted_text`. A la fin du programme, nous retournons le message en clair.

Décryptage du second message

Après avoir décrypté le premier message, nous avons trouvé une clé qui pourrait nous permettre de décrypter le second message qui nous a été confié. Pour trouver la clé, nous avons simplement pris la première lettre de chacune des phrases du premier message, ce qui nous a permis de trouver : PANGRAMME.

Grâce à cette clé, nous pouvons décrypter le second message, en utilisant le chiffre de Vigenère.

Voici le second chiffré qui nous a été donné :

> AE IOW ZQBLXR WASIXQ WJR YKJ KGYUJAGY UU OXSLN TXRCUQYM
>
> IY IRCTQ HPNF RR RQBIIIGOFN XQ WTCEKK DQ OIH MHXDUDQW BAYNVUDQYM
>
> NR MRRPQD SU CXVMUQV HOHLWLQ CYT LRY GRQYMTRRY RPBMVXTVUES
>
> QF EXNFO UEHAMAEM RV MQEWPGR IRCTQ HTREOVRQ XE HUOYKIFGXXOA

A l'aide d'une fonction, nous trouvons le message en clair ci-dessous :

> LE VIF ZEPHIR JUBILE SUR LES KUMQUATS DU CLOWN GRACIEUX
>
> IL CACHE DANS LA REPETITION LE SECRET DE CES MURMURES MALHEUREUX
>
> NE GARDEZ DU PREMIER SOUFFLE QUE LES PREMIERES APPARITIONS
>
> ET AINSI DEVOILEZ LE MESSAGE CACHE DERRIERE LA SUBSTITUTION

Pour retrouver le message en clair, nous avons utilisé la fonction suivante :

```
def decrypt_vigenere(message,cle):
    # effectue le decalage en fonction de la cle sur les caracteres de message
    n = 0
    chiffre=''
    for c in message:
        if c.isalpha():
            k = ord(cle[n%len(cle)])-65
            chiffre += decrypt_cesar(c,k)
            n+=1
        else:
            chiffre += c
    return chiffre
```

Cette fonction prend en paramètre : le message que nous souhaitons déchiffrer et une clé qui va nous permettre de déchiffrer notre message. Nous instancions ensuite deux variables locales. La première, `n`, va nous servir à compter le nombre de lettres qui ont été décryptées. La seconde, `chiffre`, va nous permettre de stocker le message en clair. Cette fonction va parcourir le message qu'on lui passera en paramètres puis vérifiera que chaque caractère consulté fait parti de l'alphabet. On instancie on autre variable locale se nommant `k` qui va nous permettre de connaître la substition que nous allons devoir utiliser. Pour cela, nous faisons en sorte de récupérer l'Unicode du caractère de la clé qui correspond à notre position dans le message puis on lui soustrait 65 (qui correspond aussi au code ASCII de la lettre A). Nous réutilisons la première fonction que nous avons écrite pour trouver un des caractères du message en clair. On ajoute 1 à notre première variable locale puis nous répétons le processus jusqu'à ce que tout le message soit déchiffré.

Décryptage du troisième message

Le chiffré qui nous a été fourni cette fois-ci était :

> ETLWK, WKOD LWFX PLPSF! BF VKIF L ZKOTSRT FDC: FBRXFEFCH

Pour ce message, nous avons supposé que la méthode de chiffrement utilisée était celle de la substitution. Nous avons remarqué que la première phrase du second message que nous avions déchiffré était en fait la clé de la substitution : Nous avons écrit une fonction qui permet de construire de façon automatique un dictionnaire qui nous donne la substitution :

```
def create_dico(message):
    dico={}
    alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
    liste_lettre = liste_mot_uni(message)
    for i in range(len(alphabet)):
        dico[liste_lettre[i]] = alphabet[i]
    return dico
```

Cette fonction est très simple, pour chaque caractère du message (qui deviendra une clé du dictionnaire), nous lui attribuons une lettre de l'alphabet en guise de valeur. Après l'exécution de cette fonction, nous

obtenons le dictionnaire suivant :

> {'L': 'A', 'E': 'B', 'V': 'C', 'I': 'D', 'F': 'E', 'Z': 'F', 'P': 'G', 'H': 'H', 'Y': 'I', 'R': 'J', 'J': 'K', 'U': 'L', 'B': 'M', 'S': 'N', 'K': 'O', 'M': 'P', 'Q': 'Q', 'A': 'R', 'T': 'S', 'D': 'T', 'C': 'U', 'O': 'V', 'W': 'W', 'N': 'X', 'G': 'Y', 'X': 'Z'}

Grâce à notre dictionnaire qui contient la substitution utilisée par le dernier message et grâce à notre fonction, nous avons pu déchiffrer le message. Voici, ci-dessous, le message en clair que nous avons trouvé :

> BRAVO, VOUS AVEZ GAGNE! LE CODE A FOURNIR EST: ELIZEBETH

Voici le code que nous avons réalisé :

```
vrai = ''
    dico = create_dico(indice)
    for c in message:
        if c.isalpha():
            if c in dico.keys():
                let = dico[c]
                vrai+=let
            else:
                vrai+=c
        else:
            vrai+=c
    return vrai
```

Cette fonction prend en paramètres : le message que nous souhaitons déchiffrer et le dictionnaire contenant toutes les substitutions possibles. Nous instancions également une variable locale, nommée `vrai`, qui contiendra le message en clair que cette fonction pourra déchiffrer. Dans cette fonction, comme pour avant, nous parcourons les caractères du message passé en paramètre, puis nous vérifions si ces caractères font parti de l'alphabet. Ensuite, nous vérifions que notre dictionnaire contient parmi ses clés, le caractère que nous sommes en train de lire. Si le caractère est contenu dans les clés du dictionnaire, nous prenons sa valeur et l'ajoutons à notre variable locale. A la fin de la fonction, nous retournons le message en clair.