Sécurisation des communications

TD3 - Chiffrement

Exercice 1 : Chiffrement de César

1. En vous basant sur la méthode du chiffrement de César, choisissez un message court que vous chiffrez avec la clé de votre choix. Transmettez votre message ainsi que la clé de chiffrement à votre binôme. Votre binôme doit retrouver votre message chiffré.

2. L'objectif est maintenant de créer une fonction Python capable de chiffrer et déchiffrer un message à partir d'une clé de chiffrement.

   Sans aide :

   Créez une fonction *cesar(message, cle*) qui chiffre un message *message* avec la clé de chiffrement *cle* selon la méthode du chiffrement de César.

   Indications : Le message chiffré devra être en majuscules Les caractères spéciaux (autres qu'une lettre de l'alphabet) ne doivent pas être modifiés.

   Créez une fonction *dechiffre(message, cle*) qui déchiffre un message chiffré *message* avec la clé de chiffrement *cle*.

   Avec aides :

   Créez une fonction *decale_lettre(lettre, cle)* qui va décaler la lettre *lettre* d'un nombre de rang égal à *cle*.

   Indications :

   Le message chiffré devra être en majuscules Les caractères spéciaux (autres qu'une lettre de l'alphabet) ne doivent pas être modifiés.

   Utilisez une variable *alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"* pour effectuer le décalage. Pour récupérer la position d'une lettre dans une chaine de caractère, vous pouvez utiliser la méthode *chaine.index(x)* qui va renvoyer la position de *x* dans la chaine de caractère *chaine*.

   Vous devrez utiliser l'opération % (modulo : reste de la division euclidienne) pour revenir au début de l'alphabet dans le cas où après décalage, votre lettre dépasse le Z.

   Exemple : *alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"*
   aphabet.index("C") -> 2

   Créez une fonction *cesar(message, cle*) qui chiffre un message *message* avec la clé de chiffrement *cle* selon la méthode du chiffrement de César.
   Utilisez la fonction *decale_lettre(lettre, cle)* précédemment définie.

Créez une fonction *dechiffre(message, cle*) qui déchiffre un message chiffré *message* avec la clé de chiffrement *cle*.

Contrôle des résultats :

Ce message a été chiffré avec le chiffrement de César et avec une clé de chiffrement = 3
Utilisez votre fonction déchiffre pour retrouver le message chiffré.

message = M'DLPH OD QVL

Exercice 2 : Chiffrement de Vigenère

Cette méthode a été mise au point durant la Renaissance pour contrer la cryptanalyse par la méthode des fréquences de lettres qui permettait de "casser" les clés de chiffrement assez facilement.

On choisit une clé sous la forme d'un mot ou d'une phrase qui donne le décalage à appliquer qui devient alors variable.

Supposons que la clé soit ABC, les décalages successifs seront 0, 1, 2, 0, 1, 2, 0...

Exemple :

Avec la clé ABC

Le message SUBSTITUTION devient: SVDSUKTVVIPP

```
SUBSTITUTION
ABCABCABCABC
------------
SVDSUKTVVIPP
```

1. En vous basant sur la méthode du chiffrement de Vigenère, choisissez un message court que vous chiffrez avec la clé de votre choix. Transmettez votre message ainsi que la clé de chiffrement à votre binôme. Votre binôme doit retrouver votre message chiffré.

Exercice 3 : Chiffrement XOR

Cette méthode repose sur l'opérateur logique XOR (ou exclusif), noté $\oplus$ et sur la répétition d'une clé de chiffrement.

Rappel de la table de l'opérateur XOR entre deux bits a et b :

a	b	a $\oplus$ b
0	0	0
0	1	1
1	0	1
1	1	1

Propriété de XOR : soit trois nombres entiers x, y et z
(codés sur 8 bits par exemple) tels que $x \oplus y = z$ alors on a aussi :
$z \oplus x = y$ et $z \oplus y = x$.

Méthode :

1. Je choisis mon message a chiffrer ainsi que ma clé de chiffrement
2. Je répète ma clé autant de fois que nécessaire pour avoir autant de bits dans mon message à coder que dans ma clé répétée.
3. Je convertis chaque lettre de mon message sur 8 bits et de ma clé en binaire grâce à leur code ASCII.
4. J'effectue un XOR bits à bits entre mon message et ma clé répétée.

Exemple :

Je souhaite chiffrer le message BONJOUR avec la cle NSI

Je répète ma clé autant de fois qu'il faut :

```
BONJOUR
NSINSIN
```

Je converti chaque lettre de mon message et de ma clé répétée en binaire :

```
BONJOUR : 01000010 01001111 01001110 01001010 01001111 01010101 01010010
NSINSIN : 01001110 01010011 01001001 01001110 01010011 01001001 01001110
```

J'effectue un XOR bits à bits entre ma clé répétée et mon message :

```
message: 01000010 01001111 01001110 01001010 01001111 01010101 01010010
clé:     01001110 01010011 01001001 01001110 01010011 01001001 01001110
         ———————————————————————————————————————————————————————— XOR
         00001100 00011100 00000111 00000100 00011100 00011100 00011100
```

J'obtiens un message chiffré qui n'a pas vraiment de sens converti en caractères (certains caractères ne sont pas des caractères visibles)

Pour déchiffrer ce message il suffit de refaire la même opération XOR entre le message chiffré et la clé de chiffrement répétée :

```
chiffré: 00001100 00011100 00000111 00000100 00011100 00011100 00011100
clé:     01001110 01010011 01001001 01001110 01010011 01001001 01001110
         ———————————————————————————————————————————————————————— XOR
          01000010 01001111 01001110 01001010 01001111 01010101 01010010
```

Je retombe bien sur mon message initial.

1. En vous basant sur la méthode du chiffrement XOR, choisissez un message court que vous chiffrez avec la clé de votre choix. Transmettez votre message ainsi que la clé de chiffrement à votre binôme. Votre binôme doit retrouver votre message chiffré. (Vous pouvez aller sur ce site pour la conversion)
2. Pourquoi peut-on utiliser le XOR et pas un OR ou un AND pour ce chiffrement ?
3. Créez une fonction *chiffre_xor(message, cle)* qui prend en entrée un message binaire ainsi que la clé de chiffrement répétée en binaire. Cette fonction doit renvoyer le message chiffré en binaire.

Le XOR en Python s'écrit avec le caractère "^"

Exemple :

```
>> a = 0
>> b = 1
>> a ^ b
1
```

4. Chiffrez le message suivant à l'aide de votre fonction :

   message :
   01001100011001010010000001100011011010000110100101100110011001100110011100
   10011001010110110101100101011011100111010000100000010110000100111101010
   0010

   cle :
   01010110011000010110001101100001011011100110001101100101011100110101010
   10011000010110001101100001011011100110001101100101011100110101011001100
   0001

5. Avez vous besoin de créer une nouvelle fonction pour déchiffrer un message ? Pourquoi ?

6. Pour contrôler vos résultats, essayez de déchiffrer le message que vous venez de chiffrer. Si vous tombez sur le même message, votre fonction fonctionne !

7. Difficile : Créez une fonction permettant de convertir un message binaire en chaine de caractère. Vérifiez votre fonction en convertissant le message de l'exercice 4.

   Aide : chr(), int()