2 Le chiffrement symétrique

2.1 Un exemple de chiffrement symétrique

2.1.1 Codage du mot

Je souhaite envoyer "Hello", qui donne en binaire : 010010000110010101101101100011011011111

Afin de convertir du texte en binaire, et inversement , il est possible d'utiliser des sites comme :

Conversion en binaire du texte : https://www.rapidtables.com/convert/number/ascii-to-binary.html ou mieux, une fonction python! (cf. ce qui a été réalisé en première à ce sujet)

Exercice 14.1

Créer une fonction ascii_to_binary(mot) qui prend en argument une chaîne de caractères, et qui renvoie une chaine de caractère composée de 0 et de 1, et qui correspond à la conversion en binaire du mot entré en argument. Chaque lettre devra être codé sur 8 bits.

```
>>> ascii_to_binary('Hello')
'01001000011001010110110001101101111'
```

2.1.2 Décodage

• Exercice 14.2

Créer une fonction **binary_to_ascii(mot)** qui prend en argument une chaîne de caractères constitutée de 0 et de 1 (multiple de 8), et qui renvoie le mot correspondant.

```
>>> binary_to_ascii('01001000011001011011000110110001101111')
'Hello'
```

2.1.3 Chiffrement

Choisissons maintenant un "mot" qui nous servira de clé de chiffrement. Je choisis par exemple "JB" qui donne en binaire :

0100101001000010

On va ensuite "recopier" autant de fois que nécessaire la clé pour avoir exactement autant de bits que le mot à coder :

mot à coder	0100100001100101011011000110110001101111
clé	0100101001000010010010100100001001001010

Nous allons ensuite, pour chiffrer le message, effectuer un XOR bit à bit :

mot à coder	0100100001100101011011000110110001101111
clé	010010100100001001001001001000010010010
XOR bit à bit	000000100010011100100110001011110001001

Exercice 14.3

Créer une fonction Codage(mot,cle) qui prend en argument une chaîne de caractères et une clé, et qui renvoie le mot codé.

C'est le message que nous allons envoyer! Si C l'intercepte et le lit, il aura " $\times 2$ '& .% "! et ne pourra pas lire le contenu.

2.1.4 Déchiffrement

B reçoit le message chiffré, et il connait la clé (car A lui a donné) :

On fait de nouveau un XOR bit à bit :

mot à décoder	000000100010011100100110001011110001001
clé	0100101001000010010010100100001001001010
XOR bit à bit	0100100001100101011011000110110001101111

Vous pouvez remarquer que nous avons bien retrouvé le code binaire d'origine! ouf!

A vous de jouer! 14.1

Imaginer un message court que vous souhaitez envoyer et convertissez-le en code binaire ASCII : chaque caractère est codé sur un octet.

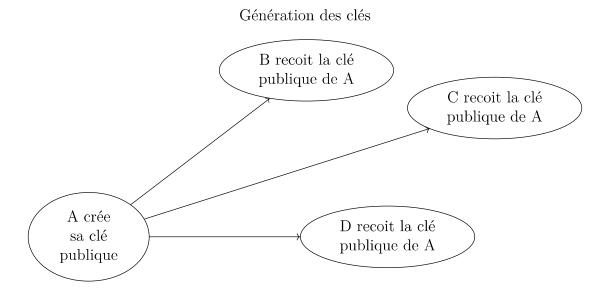
Choisissez une clé et chiffrez votre message. Remettez à votre voisin le message chiffré et la clé : celui-ci doit retrouver le message initial.

Le gros problème avec le chiffrement symétrique, c'est qu'il est nécessaire pour A et B de se mettre d'accord à l'avance sur la clé qui sera utilisée lors des échanges (et aussi se mettre d'accord sur l'algorithme de chiffrement). Le chiffrement asymétrique permet d'éviter ce problème.

- Les avantages d'utiliser "XOR":
- L'opération "ou exclusif" se calcule très rapidement.
- Le chiffrement fonctionne avec n'importe quel fichier binaire et pas seulement du texte.
- La même clef sert au chiffrement et au déchiffrement.

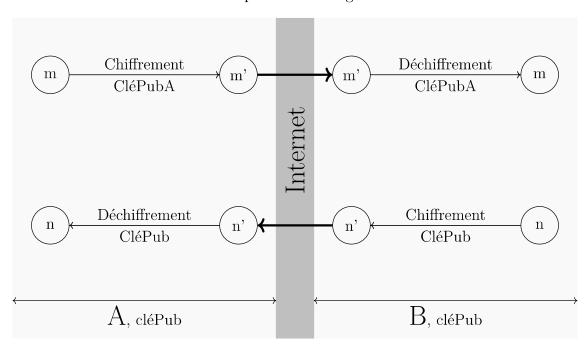
2.2 Principe du chiffrement symétrique

Supposons que (A) ait l'envie d'envoyer un message à (B), (C) et (D) de façon sécurisée. (A) doit créer une clé de chiffrement (clé publique) et la transmettre à (B):



La clé va servir à chiffre lors de l'envoi et à déchiffrer lors de la réception :

Envoi et réception de message avec XOR



2.3 Un autre algorithme

Analysez et testez ce programme :

```
def chiffrement(mess,cle):
    mess_code = ""
    position = 0
    for lettre in message:
        mess_code = mess_code + chr(ord(lettre) ^ cle[position])
        position = (position + 1) % len(cle)
    return mess_code
```

Remarque

Ici, la clé est exprimée sous la forme d'un tableau d'entiers!

Exercice 14.4

Utilisez les fonctions précédentes pour chiffrer la phrase "Veuillez transférer 1000 € sur mon compte offshore" en utilisant la clef "ŭšĠţŬkĠųŹŭkŴŲũűŵt".

Cette clef s'obtient en considérant les caractères dont l'unicode est [365, 353, 288, 355, 364, 489, 288, 371, 365, 489, 372, 370, 361, 369, 373, 357].

Exercice 14.5

On considère le message suivant, que l'on va essayer de décoder :

On sait que les 4 derniers caractères du message en clair sont 'ge !'. On sait que la clef est composée de 3 lettres.

Déterminer le message en clair, ainsi que la clé utilisée. Chrométrer le temps mis pour trouver la réponse. ***