1.2 Le chiffrement symétrique

NSI Tle - JB Duthoit

1.2.1 Un exemple de chiffrement symétrique

Codage du mot

Je souhaite envoyer "Hello", qui donne en binaire :
0100100001100101011011000110110001101111

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')
'0100100001100101011011000110110001101111'
```

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('0100100001100101011011000110110001101111')
'Hello'
```

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	0100101001000010010010100100001001001010
XOR bit à bit	0000001000100111001001100010111000100101

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.

```
>>> codage('Hello','JB')
"\x02'&.%"
```

C'est le message que nous allons envoyer ! Si C l'intercepte et le lit, il aura "\x02'& .% " ! et ne pourra pas lire le contenu.

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	000000100010011100100110001011100010 0101
clé	010010100100001001001010010000100 1001010
XOR bit à bit	010010000110010101101100011011000 1101111

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

Exercice 14.4
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.

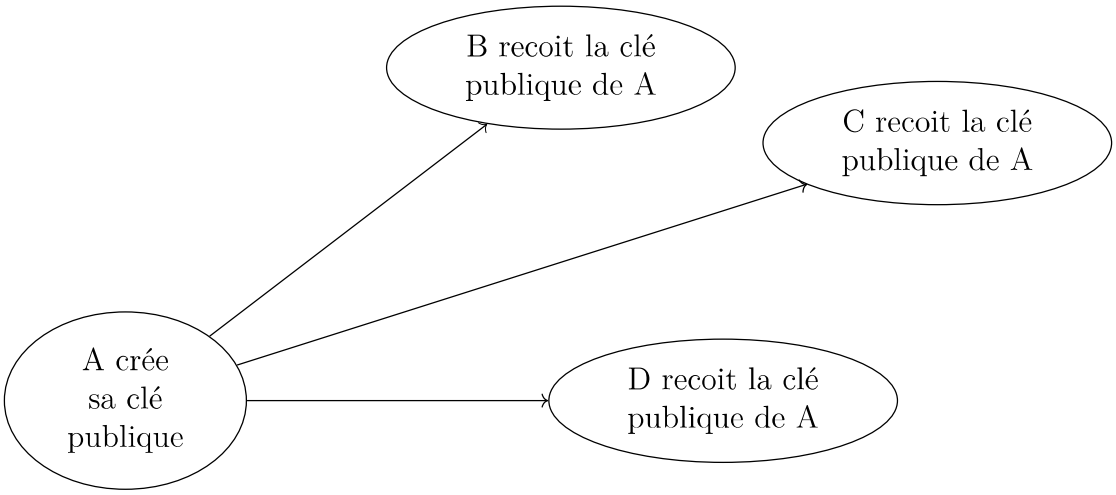
☞ Les avantages d'utiliser "XOR" :

- 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.

1.2.2 Principe du chiffrement symétrique

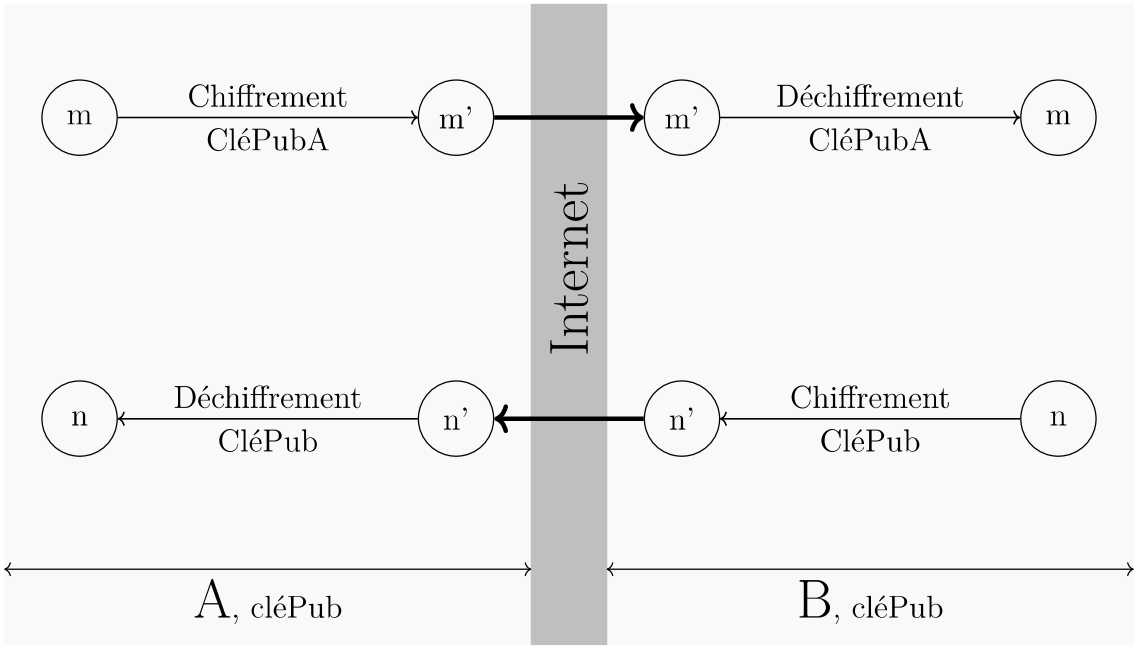
Supposons que (A) ait l'envie d'envoyer un message à (B), (C) et (D) de façon sécurisée. (A) doit créer une clé de chiffrement (clé publique) et la transmettre à (B) :

Génération des clés



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

Envoi et réception de message avec XOR



1.2.3 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 14.5

Utilisez les fonctions précédentes pour chiffrer la phrase "Veuillez transférer 1000 € sur mon compte off-shore" en utilisant la clef "ŭšĠţŬkĠąŹŭǩŴŲũűŵť".
Cette clef s'obtient en considérant les caractères dont l'unicode est [365, 353, 288, 355, 364, 489, 288, 371, 377, 365, 489, 372, 370, 361, 369, 373, 357].

Exercice 14.6

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

```
secret = '\x1a\x04$.\x19ix\x00*-\x05e9\x00 "V7±\x036+\x1fe;\x13e<\x9f&7\x12$?\x13ey'
```

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. ***