Sécurisation des communications

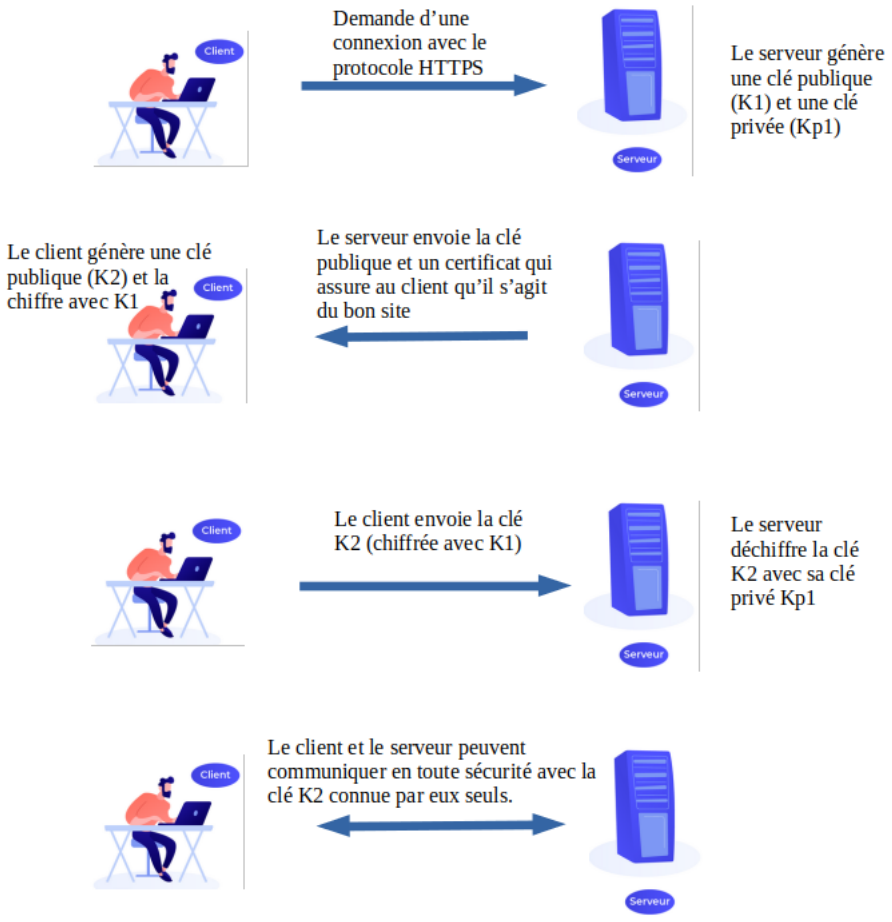
Introduction

Pour qu'un message entre deux machines dans un réseau informatique ne puisse pas être compréhensible s'il est intercepté, il faut qu'il soit chiffré.

Il y a deux manières principales pour le faire:

- Par un chiffrement **symétrique** qui utilise une clé unique, connue seulement de l'émetteur et du récepteur. L'émetteur chiffre le message avec la clé et le récepteur le déchiffre avec la même clé.
- Par un chiffrement **asymétrique** qui utilise un couple de clé, l'une publique connue de tout le monde, l'autre privée connue uniquement par le récepteur. L'émetteur chiffre le message avec la clé publique. Le récepteur le déchiffre avec la clé privée.

Ces deux méthodes de chiffrement sont utilisées lorsqu'un navigateur demande une page web avec le protocole **HTTPS**.
Dans le schéma ci-dessous, on a représenté les échanges de communications entre un client et un serveur avec le protocole HTTPS.



Les échanges entre client et serveur se font avec un chiffrement symétrique (clé K2) après que celle-ci ait été échangée avec un chiffrement asymétrique.

Le chiffrement symétrique

Dans un **chiffrement symétrique**, la clé utilisée par l'expéditeur pour chiffrer le message est la même que celle utilisée par le récepteur pour déchiffrer le message.

Un premier exemple : Le codage de César avec une clé de 13.

En ne considérant que les lettres en majuscule et non accentuées, notre alphabet se réduit à 26 lettres (ABCDEFGHIJKLMNOPQRSTUVWXYZ)
Avec un décalage de 13 : (A → N) et (N → A). On utilise bien la même clé, il s'agit donc d'un chiffrement symétrique.

Par exemple le message `'BONJOUR, COMMENT ALLEZ-VOUS'` se code par :
`'OBAWBHE, PBZZRAG NYYRM-IBHF'` (seules les lettres ont été codées...) et le déchiffrement nous redonne bien le message d'origine.

À FAIRE 1:

Écrire un programme qui réalise ce chiffrement de César avec une clé de 13. On pourra utiliser ce dictionnaire et la méthode get associée :

```
d = {chr(i+65): chr((i+13)%26+65) for i in range(26)}
```

Un second exemple : Utilisation de `XOR`

`XOR` est une opération de logique bit à bit qui renvoie 0 si les deux bits sont égaux et 1 sinon.
Une particularité de cette opération est : Si b_1 xor $b_2 = b_3$ alors b_1 xor $b_3 = b_2$ et b_2 xor $b_3 = b_1$
Ce qui permet un chiffrement symétrique :
Prenons le mot `'bonjour'` dont le code Unicode est donné par ce programme:

```
m = "bonjour"
 for c in m:
    print(ord(c),end=' ')
```

Soit : 98 111 110 106 111 117 114
Le mot `'nsi'` va nous servir de clé : (`'nsi'` → 110 115 105)
La méthode consiste alors à aligner le mot et la clé en la répétant autant de fois que nécessaire et d'effectuer un `XOR` entre les codes Unicode:

b	o	n	j	o	u	r
98	111	110	106	111	117	114
n	s	i	n	s	i	n
110	115	105	110	115	105	110

En pratiquant un XOR lettre par lettre entre les nombres obtenus (après une écriture en binaire...), on obtient les codes Unicode suivant: 12 28 7 4 28 28 28
(que l'on pourrait transformer en caractères mais ce n'est pas très utile...)

Pour décoder le message, il suffit alors de recommencer l'opération avec les codes Unicode du message chiffré en utilisant la même clé: (12 xor 110 → 98 donc 'b') etc.

En Python l'opérateur ∧ permet d'effectuer un xor directement sur deux entiers. Le programme suivant chiffre un message en utilisant cette méthode:

```
message = "Bonjour, comment allez-vous ?"
cle = "mystère"

def chiffre(message,key):
    c = []
    n = len(message)
    m = len(key)
    j = 0
    for i in range(n):
        c.append(ord(message[i]) ^ ord(key[j]))
        j = (j+1)%m

    return c

print(chiffre(message,cle))
```

Ce qui donne en console:[47, 22, 29, 30, 135, 7, 23, 65, 89, 16, 27, 133, 31, 0, 3, 13, 83, 21, 132, 30, 0, 23, 84, 5, 27, 157, 1, 69, 82]
Chaque nombre peut être converti en caractère (le contenu complet du message est chiffré, y compris les caractères de ponctuation et d'espace).

À FAIRE 2:

Modifier la fonction chiffre pour qu'elle renvoie le message chiffré (avec les caractères)

À FAIRE 3:

Écrire une fonction `dechiffre` qui prend en paramètres le message chiffré et la clé et qui renvoie le message déchiffré.

Chiffrement asymétrique

Pour reprendre l'exemple du code de César, il suffit de choisir une autre clé que 13 pour obtenir un chiffrement asymétrique, c'est à dire où la clé de chiffrement et de déchiffrement ne sont pas identiques.
Si la clé de chiffrement est 9, la clé de déchiffrement est de 17 (26 – 9)

À FAIRE 4:

Écrire un programme qui chiffre et déchiffre le message `'HELLO LES QUICHES'` pour illustrer le chiffrement asymétrique avec le code de César.

Dans la réalité, pour fabriquer des clés, on utilise des nombres qui se factorisent en produit de deux nombres premiers (9967*9973=99400891) . Le système RSA est basé sur ce principe.

Complément - Le système RSA simplifié

Le principe:
Prenons la lettre `'b'` dont le code Unicode est 98.

Méthode :
La clé publique pour notre exemple est : $(e,n) = (41,437)$
$n = 437$ est le produit de deux nombres premiers ($437 = 23 \times 19$), on considère alors l'entier $f = (23-1) \times (19-1) = 22 \times 18 = 396$ et on choisit $e = 41$ un nombre premier inférieur à f.

Pour coder la lettre `'b'` on détermine le reste de la division Euclidienne de 98^e par n.
Ce qui donne ici 110, c'est à dire la lettre `'n'`

Pour décoder la lettre `'n'` on utilise la clé privée qui est $(d,n) = (29,437)$ et on calcule le reste de la division Euclidienne de 110^d par n.

Méthode :
La clé privée pour notre exemple est : $(d,n) = (29,437)$
où d est un entier tel que le reste de la division de $e \times d = 41 \times 29$ par $f = 396$ est 1.
Des méthodes en mathématiques permettent de déterminer d.

Le reste de la division euclidienne de 110^{29} par 437 est 98 qui donne la lettre `'b'`.

À faire 5:
Écrire un programme qui chiffre et déchiffre un message en utilisant ce système avec les clés précédentes:
Voici d'autres clés: clé publique (103 , 697) ; clé privée (87 , 697)

Remarque :
la sécurité engendrée par un tel système réside dans la difficulté (par le calcul) à retrouver la clé privée dans un temps raisonnable et ce d'autant plus si les nombres utilisés sont grands...