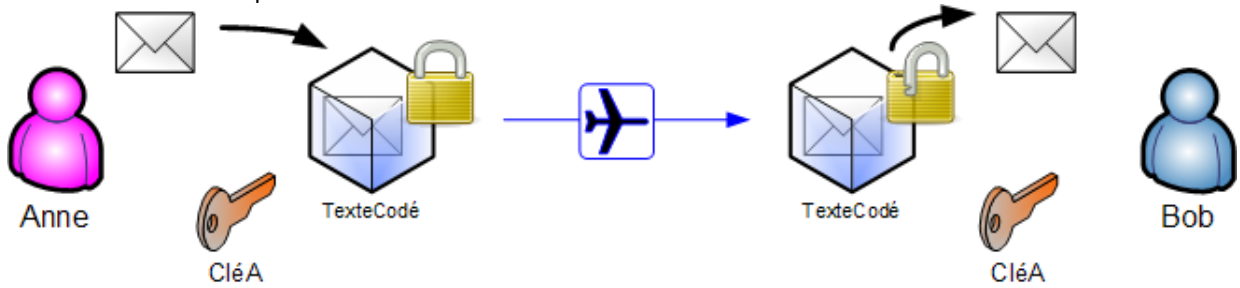
1. INTRODUCTION

Les communications, qu'elles soient entre humains, entre humains et machines, ou entre machines, peuvent être interceptées par différents moyens, parfois surprenants[1]. Il est donc important de les protéger, que ce soit pour protéger sa vie privée ou pour éviter un piratage. Pour protéger ces communications, on chiffre[2] les messages à l'aide d'une méthode de cryptographie, comportant une ou plusieurs clés secrètes. Les deux principes de chiffrement sont le chiffrement symétrique, où la même clé est utilisée pour le chiffrement et le déchiffrement du message, et le chiffrement asymétrique (objet du prochain cours), où deux clés différentes sont utilisées. Reste à assurer que les clés ne soient pas piratées, ce qui est le problème de l'authenticité des utilisateurs. La sécurité des communications dépend à la fois des méthodes de chiffrement et des protocoles d'échange des clés.

Assurer la sécurité des machines reliées à Internet c'est être sûr de l'identité de la machine distante, sécuriser les échanges, limiter l'accès à certaines données. L'une des solutions possibles est la mise en œuvre du protocole HTTPS

2. CHIFFREMENT SYMETRIQUE

Dans le chiffrement symétrique, la même clé permet à la fois de chiffrer et de déchiffrer le message. La clé secrète doit donc être connue préalablement par les deux entités communiquant. On peut remarquer que si la clé fait la taille du message (ou plus), alors le message ne peut pas être décrypté. Comme dit en introduction, il faut également échanger cette clé au préalable, sans qu'elles ne soient interceptées !



a. Des exemples historiques

Le chiffre de César est peut-être le plus connu. César chiffrait ces messages en décalant toutes les lettres de 3 (car C est la troisième lettre de l'alphabet). Ce chiffrement a été utilisé par les égyptiens en -2000. Plus généralement, on décale chaque lettre de l'alphabet d'un certain nombre de rangs fixe qui est appelé la clé de chiffrement. Par exemple avec la clé 3, A devient D, B devient E, etc.

Plus sophistiqué, le chiffre de Vigénère utilise un décalage alphabétique également, mais avec une clé de plusieurs lettres répétée. Si la clé est **NSI**, la première et la quatrième lettre du message seront décalées de 14, la deuxième et la cinquième de 19, la troisième et la sixième de 9 etc. Ce chiffre a résisté longtemps au décryptage, et a été utilisé jusque pendant la guerre de Sécession.

Enfin, le fameux code Enigma utilisé par l'Allemagne nazie pendant la deuxième guerre mondiale a été cassé par Alan Turing. Comme les alliés ne voulaient pas que les allemands changent de système, ils ont masqué leur découverte, en rajoutant du « bruit » à l'information recueillie. Par exemple, ils faisaient passer des avions de reconnaissance « par hasard » en vue d'un convoi de ravitaillement des forces de l'axe dont le trajet était connu grâce à Enigma. Et très probablement, les alliés ne ciblaient pas tous les convois repérés.

b. Méthodes actuelles

L'algorithme AES (Advanced Encryption Standard) est un standard de chiffrement symétrique. Il est peu coûteux en mémoire, facile à mettre en forme, et il n'existe pas actuellement de méthode plus efficace que la force brute pour le casser.

Ce standard, a une clé qui peut avoir une taille allant jusqu'à 256 bits ce qui laisse 2^{256} possibilités soit environ 10^{77} clés à tester ! Les attaques, alors, ne se portent pas sur le message lui-même mais sur les machines qui implémentent le programme de chiffrement.
Cette clé doit rester secrète car sa connaissance permet de déchiffrer un message. C'est l'un des inconvénients des chiffrements symétriques. Même si la clé est gardée secrète, un chiffrement symétrique doit avoir une clé suffisamment compliquée pour ne pas être attaquée. Dans le cas du chiffrement par décalage, il y a 25 clés possibles ce qui n'est pas compliqué à attaquer par force brute. Effectuer une attaque par force brute signifie ici essayer toutes les clés possibles sans stratégie jusqu'à trouver la clé.

[1] À l'époque des écrans cathodiques, il était possible « d'écouter » au travers d'un mur les variations de fréquence de balayage de l'écran pour trouver les caractères tapés au clavier… Seule solution, écrire dans une fenêtre cachée par une autre.

[2] Vocabulaire : chiffrer/déchiffrer s'utilise pour transformer un message clair en message codé à l'aide d'une clé. Décrypter, c'est casser le code sans connaître la clé.

3. Principe de fonctionnement

Pour chiffrer un message, *Anne* va utiliser une suite de caractère que l'on appelle "clé de chiffrement". Dans le cas du chiffrement symétrique, cette clé de chiffrement sera aussi utilisée par *Bob* pour déchiffrer le message envoyé par *Anne*. Dans ce cas, la clé de chiffrement est identique à la clé de déchiffrement. Concrètement comment cela se passe-t-il ?

Comme nous avons déjà eu l'occasion de le voir en première, toute "donnée informatique" peut être vue comme une suite de zéro et de un. Nous chercherons donc à chiffrer une suite de zéro et de un :

Soit le message "**Hello les TNSI**" ce qui nous donnera en binaire :

```
01001000 01100101 01101100 01101100 01101111 00100000 01101100 01100101 01110011 00100000 01010100 01001110
01010011 01001001
```

N.B. nous avons simplement utilisé le code ASCII de chaque caractère (par exemple, on peut vérifier que le H correspond bien à l'octet 01001000). Pour effectuer la "conversion" texte vers code binaire ASCII ou vice-versa, vous pouvez utiliser le site https://www.rapidtables.com/convert/number/ascii-to-binary.html

Choisissons maintenant un mot (ou une phrase) qui nous servira de clé de chiffrement, prenons pour exemple le mot "*Boissy*". "*Boissy*" nous donne en binaire :

```
01000010 01101111 01101001 01110011 01110011 01111001
```

Pour chiffrer le message nous allons effectuer un XOR bit à bit. Pour rappel, vous trouverez la table de vérité du XOR ci-dessous :

Table de vérité de l'opérateur XOR

a	b	a XOR b
0	0	0
0	1	1
1	0	1
1	1	0

Comme la clé est plus courte que le message, il faut "reproduire" la clé vers la droite autant de fois que nécessaire (si la taille du message n'est pas un multiple de la taille de la clé, on peut reproduire seulement quelques bits de la clé pour la fin du message):

```
⊕ 0100100001100101011011000110110001101111001000000110110001100101011100110010000001010100010011100101001101001001
  0100001001101111011010010111001101110011011110010100001001101111011010010111001101110011011110010100001001101111
  ----------------------------------------------------------------------------------------------------------------
  0000101000001010000001010001111100011100010110010010111000001010000110100101001100100111001101110001000100100110
```

Le signe + dans un cercle symbolise le XOR

Après ce XOR on obtient donc la suite de bits suivante :
0000101000001010000001010001111100011100010110010010111000001010000110100101001100100111001101110001000100100110

Soit la chaine de caractères suivante (si on cherche à afficher le message chiffré avec un éditeur de texte) :

Saut de ligne
Saut de ligne
Y.
S'7&

Maintenant ce message est prêt pour être envoyé à son destinataire *Bob*. Si *Pirate* intercepte le message et cherche à le lire avec un éditeur de texte, il obtiendra la suite de

Saut de ligne
Saut de ligne
Y.
S'7&

B a maintenant reçu le message chiffré, il possède la clé (toto), il va donc pouvoir déchiffrer le message en appliquant un XOR entre le message chiffré et la clé (on applique exactement la même méthode que ci-dessus).

```
⊕ 0000101000001010000001010001111100011100010110010010111000001010000110100101001100100111001101110001000100100110
  0100001001101111011010010111001101110011011110010100001001101111011010010111001101110011011110010100001001101111
  ----------------------------------------------------------------------------------------------------------------
  0100100001100101011011000110110001101111001000000110110001100101011100110010000001010100010011100101001101001001
```

On trouve le code binaire suivant :

```
01001000011001010110110001101100011011110010000001101100011001010111001100100000010101000100111001010011010010001
```

Vous pouvez remarquer que nous avons bien retrouvé le code binaire d'origine. Si vous ne voulez pas vous embêter à vérifier bit par bit, vous pouvez utiliser ce site qui vous permettra de repasser du code binaire ASCII au texte.
On retrouve bien le message d'origine : "**Hello les TNSI**", *Bob* a pu lire le message envoyé par *Anne* alors que pour *Pirate*, malgré le fait qu'il a pu intercepter le message, il n'a pas pu prendre connaissance de son contenu sans la clé.
Comme dit plus haut, la méthode la plus utilisée en matière de chiffrement symétrique se nomme AES (Advanced Encryption Standard). Cette méthode utilise une technique de chiffrement plus élaborée que ce qui a été vu ci-dessus, mais les grands principes restent identiques.
Le gros problème avec le chiffrement symétrique, c'est qu'il est nécessaire pour *Anne* et *Bob* de se mettre d'accord à l'avance sur la clé qui sera utilisée lors des échanges. Le *chiffrement asymétrique* permet d'éviter ce problème. Il sera l'objet du prochain cours…

4. Exercice 1

Après un chiffrement symétrique on obtient le message suivant : *ri*. Sachant que la clé de chiffrement est : 00001010 (la clé est directement donnée en binaire), déterminer le message d'origine.

5. Exercice 2

Le XOR est très utilisé dans les protocoles de chiffrement symétrique. En effet, c'est une opération qui est sa propre réciproque, ce qui n'est pas le cas du ET ni du OU. C'est à dire que : $A \oplus B = C \Leftrightarrow A \oplus C = B \Leftrightarrow C \oplus B = A$ (1)

a. Démontrer la propriété (1) en complétant les tables de vérité suivantes :

A	B	C

A	C	B

C	B	A

b. On va programmer un protocole de chiffrement symétrique utilisant le XOR.
On souhaite chiffrer un message (par exemple une chaîne de caractères) à l'aide d'une clé, qui peut aussi être une chaîne de caractères.
- Recopier la clé à la suite d'elle-même autant de fois que nécessaire, pour que la longueur totale soit égale à celle du message (on tronque si nécessaire).
- Transcrire le message en binaire (voir la fonction *ord()*)
- Transcrire la suite de clés en binaire
- Le message chiffré est obtenu avec un XOR des deux nombres binaires précédents.
- On peut alors retranscrire le message en caractères.
- Le déchiffrement se fait avec un XOR entre le message chiffré en binaire et le nombre binaire correspondant à la suite des clés.

Programmer ce chiffrement en Python (deux fonctions au moins, une pour chiffrer et une pour déchiffrer).
Le programme demandera la phrase à chiffrer ainsi que la clé de chiffrement. Le résultat affichera alors la phrase chiffrée.

c. Donner alors le chiffrement de la phrase suivante :
"*Le lycée Boissy d'Anglas est le meilleur lycée de France*" à l'aide de la clé "*NSI c'est top*"