SC_Sécurisation des communications

La généralisation rapide des communications par Internet engendre un besoin impérieux de sécurisation des informations et des technologies associées. D'où le rôle de plus en plus capital du chiffrement.

Pourtant, l'histoire du chiffrement ne date pas d'aujourd'hui puisqu'il faut remonter à la **civilisation babylonienne**, environ 3 000 ans avant notre ère, pour en trouver les premières traces. Quant à son application, elle s'est peu à peu étendue des seuls champs militaire et politique pour investir la sphère civile, notamment sous l'impulsion d'Internet et de l'explosion des volumes de données qui révolutionnent notre quotidien sous bien des aspects.

L'histoire du chiffrement retrace une épopée passionnante dans laquelle cryptographes (« crypteurs ») et cryptanalystes (« décrypteurs ») se livrent une bataille acharnée, éternel recommencement de développement d'un algorithme par les uns, de décodage par les autres, de développement d'un nouvel algorithme plus puissant, etc.

En savoir plus : https://www.thawte.fr/assets/documents/guides/history-cryptography.pdf

I/ Chiffrement à algorithme par substitution simple

Ce type de chiffrement est le plus ancien. Comme son nom l'indique, il s'agit de **substituer une lettre par une autre**, soit par un algorithme qui doit être connu de l'émetteur et du récepteur soit par un tableau de correspondance.

Exemple : l'algorithme ROT13 *(en anglais, ROTate by 13 places)*.
L'idée est très simple : il suffit de décaler la place de chaque lettre de l'alphabet de 13 places pour le chiffrement. Le *« A »* devient ainsi un *« N »*. Le déchiffrement est aussi très simple, il suffit également de décaler la lettre codée de 13 places également, le *« N »* redevient un *« A »* car il y a 26 lettres dans l'alphabet.

Un programme en Python

```
# Construction du dictionnaire de correspondance entre
# les lettres de l'alphabet et celles codées.
chiffre = { chr(i + 65) : chr((i + 13)%26 + 65) for i in range(26) }
print(chiffre)

# On ne code que les lettres.
message = "BONJOUR VOUS ALLEZ BIEN"
print("Message initial :", message)

message_code = ""
for lettre in message :
    # Récupère la valeur de la clé `lettre` dans le dictionnaire
    # si elle existe sinon la lettre (pour les espaces)
    message_code += chiffre.get(lettre,lettre)

print("Message codé :", message_code) # Affiche le message codé

message_decode = ""
for lettre in message_code :
    # Récupère la valeur de la clé `lettre` dans le dictionnaire
    # si elle existe sinon la lettre (pour les espaces)
    message_decode += chiffre.get(lettre,lettre)

print("Message décodé :", message_decode)
```

```
{'A': 'N', 'B': 'O', 'C': 'P', 'D': 'Q', 'E': 'R', 'F': 'S', 'G': 'T',
'Z', 'N': 'A', 'O': 'B', 'P': 'C', 'Q': 'D', 'R': 'E', 'S': 'F', 'T': '
'Z': 'M'}
Message initial : BONJOUR VOUS ALLEZ BIEN
Message codé : OBAWBHE IBHF NYYRM OVRA
Message décodé : BONJOUR VOUS ALLEZ BIEN
```

Tableau ASCII

Dec	Bin	Hex	Ch
64	0100 0000	40	@
65	0100 0001	41	A
66	0100 0010	42	B
67	0100 0011	43	C
68	0100 0100	44	D
69	0100 0101	45	E
70	0100 0110	46	F
71	0100 0111	47	G
72	0100 1000	48	H
73	0100 1001	49	I
74	0100 1010	4A	J
75	0100 1011	4B	K
76	0100 1100	4C	L
77	0100 1101	4D	M
78	0100 1110	4E	N
79	0100 1111	4F	O
80	0101 0000	50	P
81	0101 0001	51	Q
82	0101 0010	52	R
83	0101 0011	53	S
84	0101 0100	54	T
85	0101 0101	55	U
86	0101 0110	56	V
87	0101 0111	57	W
88	0101 1000	58	X
89	0101 1001	59	Y
90	0101 1010	5A	Z

La fonction ***chr(65)*** renvoie la lettre *« A »* par exemple.
A noter : la fonction ***ord('A')*** renverrait 65.

Remarque : on peut aussi imaginer un décalage asymétrique des lettres. Par exemple, on peut décaler les lettres de 17 places pour le chiffrement et donc de le redécaler de 9 places pour le déchiffrement (voir exercice).

Important : la sécurité du chiffrement repose donc ici l'algorithme.

Exemple : un chiffrement à l'aide d'une clé.
Contrairement à l'exemple précédent, il n'y a besoin d'aucun algorithme pour chiffrer / déchiffrer un message, il faut toutefois que l'émetteur et le récepteur possèdent le tableau des correspondances (clé).

Sous forme de dictionnaire :

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
S	E	T	R	O	G	Q	B	P	I	F	L	W	H	X	C	Y	Z	V	U	J	A	K	N	D	M

Sous forme de liste ou de chaîne de caractères :

```
Cle = "SETROGQBPIFLWHXCYZVUJAKNDM"
```

On peut se servir de la place de lettre dans la clé pour coder un texte avec des nombres par exemple.
Par exemple, la lettre *« T »* correspondrait au nombre *« 2 »* (si on commence à zéro).

Important : la sécurité du chiffrement repose donc ici la clé.

Au niveau de la **sécurité**, le principe d'un décalage à valeur constante d'une lettre n'assure de nos jours aucune protection, cet algorithme étant bien trop simple.
On pourrait penser que la version d'algorithme à substitution simple à l'aide d'une clé est plus sécurisée. En effet, il faudrait tester toutes les combinaisons possibles c'est à dire 26 x 25 x 24 x …. x 4 x 3 x 2 x 1 (factorielle de 26) ce qui fait beaucoup de combinaisons même pour un ordinateur actuel. C'est la méthode de la force brute.
On utilise en réalité la méthode de l'analyse fréquentielle qui consiste à repérer la répétition de lettres, de mots. En français, la lettre *« e »* apparaît le plus souvent, on a presque toujours un *« u »* après un *« q »* etc.
La performance de cette technique augmente avec le nombre et la taille des messages cryptés.

Concrètement, l'algorithme de substitution simple ne sont plus utilisées.

Pour augmenter la sécurité de la transmission des messages, on privilégie - en plus de l'algorithme de chiffrement - l'utilisation d'un système de clés : soit par un **chiffrement symétrique**, soit par un **chiffrement asymétrique**.
Ce système est mis au point dans les années 1970.

- Chiffrement symétrique : utilise une **clé unique**, connue uniquement par l'émetteur et le récepteur du message. Ainsi, l'émetteur chiffre son message avec la clé et le récepteur le déchiffre avec la même clé.
- Chiffrement asymétrique : utilise un **couple de clés**, l'une *publique* connue de tous et l'autre *privée* connue uniquement par le récepteur. L'émetteur chiffre son message avec la clé privée et le récepteur le décrypte avec sa clé privée.

II/ Chiffrement à clé symétrique

1/ Chiffrement manuel

Au XVe siècle, Leon Battista Alberti développa un prototype de chiffrement par **substitution poly alphabétique** qui, comme son nom l'indique, faisait intervenir de multiples alphabets de substitution. Il ouvrit ainsi la voie à une succession d'innovations dans ce domaine, dont la plus marquante fut celle du Français Blaise de Vigenère, aussi connue sous le nom de « *chiffre de Vigenère* ».

Il nécessite une **clé unique** qui doit être connue de l'émetteur et du récepteur (clé symétrique). On répète la clé jusqu'à ce qu'elle soit aussi longue que le texte à chiffrer.

Principe de chiffre de Vigenère

Exemple de message à coder

Texte clair	MEDAILLEDEBRONZE
Clé	OLYMPIQUEOLYMPIQ
Texte chiffré	APBMXTBYHSMPACHU

Pour coder la lettre « M », on utilise la lettre « O » de la clé.

Chiffrement :
Dans les colonnes, on repère le « M » et dans les lignes le « O ». Comme dans la bataille navale, leur intersection donne la lettre codée soit un « A ».

Déchiffrement :
On se place sur la ligne du « O » et on repère le « A » : la nom de la colonne est bien la lettre « M ».

Table de Vigenère

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

D'autres systèmes de chiffrement manuel apparaîtront notamment au Japon jusqu'à la première guerre mondiale, lien ici : https://www.thawte.fr/assets/documents/guides/history-cryptography.pdf

On voit très clairement l'aspect poly alphabétique de cet algorithme : une lettre codée pour donner toutes les lettres de l'alphabet ce qui améliore sensiblement la sécurité de ce chiffrement.
Il faut impérativement **connaître la clé** pour décoder le message mais **la transmission de la clé au récepteur est le point faible de cette méthode** car elle peut interceptée par un tiers.

2/ Chiffrement automatisé

Formulé manuellement jusqu'à la fin du XIXe siècle, le décryptage des chiffrements se compliqua davantage avec l'avènement des machines de chiffrement mécanique au début du XXe siècle. La plus célèbre d'entre elles fut sans aucun doute Enigma, une machine portable et puissante mise au point par l'ingénieur allemand Arthur Scherbius en 1918.

Depuis la Seconde Guerre mondiale, le cryptage et le décryptage sont eux aussi passés du mécanique au numérique. Outre les applications militaires traditionnelles, le raz-de-marée informatique dans le secteur privé engendra un besoin croissant de cryptage de transactions commerciales et autres usages civils.

Exemple : un chiffrement à l'aide du binaire et l'opérateur XOR (ou exclusif).

Rappels :

- l'opérateur logique XOR compare les nombres bit à bit, il faut donc convertir chaque caractère selon la table ascii (voire page 1) à l'aide de la fonction ***ord()*** en langage Python.
- Table de vérité de XOR : 0 XOR 0 = False ; 1 XOR 0 = True ; 0 XOR 1 = True et 1 XOR 1 = False.
- L'opérateur logique XOR se traduit par « ^ » en Python.

Cette technique s'appuie sur la propriété suivante de l'opérateur XOR : si C et D sont des nombres binaires alors on a l'égalité (C XOR D) XOR D = C. Par exemple, ***1011*** *XOR 0111 = 1100* et *1100 XOR 0111* = **1011** 😊. On retrouve bien le nombre de départ.

On suppose qu'une personne A souhaite envoyer un message à une personne B.

- **1ère étape** : une clé (*traduite en binaire*) est connue des deux personnes A et B.
- **2ème étape** : le message de la personne A est traduite en binaire, chaque lettre fera 1 octet.
- **3ème étape** : on applique l'opérateur XOR entre le message et la clé (qui sera répétée autant de fois que nécessaire pour avoir la même taille que le message, quitte à ne mettre que quelques bits de la clé à la fin).
- **4ème étape** : le message ainsi chiffré est envoyé à la personne B.
- **5ème étape** : on applique l'opérateur XOR entre le message chiffré et la clé (qui sera là aussi répétée autant de fois que nécessaire pour avoir la même taille que le message).
- **6ème étape** : on traduit en lettre (1 octet par lettre) le message déchiffré et il devient lisible.

Exemple de programme en Python

```
message = "Comment allez-vous ?"
message_dec = []
cle = "Coucou"
cle_dec = []

# Codage en décimal du message et de la clé
for lettre in message :
    message_dec.append(ord(lettre))
for lettre in cle :
    cle_dec.append(ord(lettre))

# Affichage du message / clé en décimal
print("Message en décimal :", message_dec)
print("Clé en décimal :", cle_dec)


# Chiffrement du message
def chiffre(message_d,cle_d) :
    message_dec_code= []
    long_cle = len(cle_d)

    # Parcours du message à coder
    for i in range(len(message)) :
        # Conversion en entier de chaque lettre du message et de la clé
        # et application du "ou exclusif" (a^b)
        # Remarque : l'instruction i % long_cle permet de faire en sorte
        # que la clé soit de la même taille que le message.
        message_dec_code.append(message_d[i] ^ cle_d[i%long_cle])

    return message_dec_code

message_dec_code = chiffre(message_dec,cle_dec)
print("Message codé en décimal :", message_dec_code)
```

```python
# Déchiffrement du message
def dechiffre(message_d_c,cle_d) :
    message_decode= []
    long_cle = len(cle_d)

    # Parcours du message à décoder
    for i in range(len(message)) :
        # Application du "ou exclusif" (a^b) sur la clé et le message codé
        # et conversion en lettre par la fonction chr().
        message_decode.append(chr(message_d_c[i] ^ cle_d[i%long_cle]))

    return message_decode

print("Message initial", dechiffre(message_dec_code, cle_dec))
```

```
Message en décimal : [67, 111, 109, 109, 101, 110, 116, 32, 97, 108, 108, 101, 122, 45, 118, 111, 117, 115, 32, 63]
Clé en décimal : [67, 111, 117, 99, 111, 117]
Message codé en décimal : [0, 0, 24, 14, 10, 27, 55, 79, 20, 15, 3, 16, 57, 66, 3, 12, 26, 6, 99, 80]
Message initial ['C', 'o', 'm', 'm', 'e', 'n', 't', ' ', 'a', 'l', 'l', 'e', 'z', '-', 'v', 'o', 'u', 's', ' ', '?']
```

Là encore, si le message chiffré est intercepté mais sans la clé, il est pratiquement indéchiffrable. Mais ici encore, se pose le **problème de la transmission de la clé en toute sécurité**.

III/ Chiffrement à clé asymétrique

Inventé vers 1976, le chiffrement asymétrique (ou chiffrement à clé publique) repose sur l'utilisation d'une clé composée de deux parties :

- Une partie privée appelée **clé privée**, conservée par le titulaire de la clé et qui ne sera **jamais communiquée**.
- Une partie publique appelée **clé publique**, connue de tous, y compris d'éventuels espions.

L'idée est que **tout message crypté transmis ne peut être décrypté qu'à l'aide des deux clés**.

On suppose qu'une personne A souhaite envoyer un message à une personne B.

- **1ère étape** : chaque personne génère une clé asymétrique dont ils conservent la partie privée mais échangent la partie publique.
- **2ème étape** : à l'aide de la clé publique de la personne B, la personne A chiffre son message.
- **3ème étape** : le message chiffré est envoyé à la personne B.
- **4ème étape** : la personne B déchiffre le massage à l'aide de sa clé privée.

Un exemple de fonction et de fonction réciproque : soit $f(x)= x^2$ avec x positif, sa fonction réciproque est $f^{-1}(x) = \sqrt{x}$ car par exemple, $f(3) = 9$ et $f^{-1}(9) = 3$.

Mathématiquement, le chiffrement asymétrique s'appuie sur la notion de ***fonction à sens unique***. Autrement dit, si on connait la fonction, la fonction réciproque est très compliquée à trouver.
Pour y parvenir, il faudra utilisée une donnée (appelée brèche secrète) : c'est le rôle de clé privée.

Exemple : l'algorithme RSA, lien ici : history-cryptography.pdf (thawte.fr) (page 11).

IV/ Le protocole HTTPS

L'échange de données par le **protocole HTTP** présente **plusieurs failles de sécurité** dont les deux principales sont :

- Les **données** transitent **en clair** et peuvent être lues par un pirate.
- Un **pirate** peut **se faire passer** pour le **serveur** et recevoir les données transmises par le client.

Le protocole SSL (maintenant TLS) permet de sécuriser l'ouverture de la session.

Le protocole HTTPS en quelques étapes.

- **1ère étape** : le client demande une connexion sécurisée au serveur.
- **2ème étape** : le serveur envoie alors sa clé publique et sa **signature**.
- **3ème étape** : le client vérifie la signature en faisant appel à des **autorités de certification**.
- **4ème étape** : après vérification, le client génère une clé de chiffrement symétrique (AES), sa **clé de session**, et la chiffre avec la clé publique du serveur : elle transite en toute sécurité.
- **5ème étape** : le serveur reçoit alors la clé symétrique et la déchiffre à l'aide de sa clé privée.
- **6ème étape** : client et serveur peuvent échanger en utilisant la clé de session (clé symétrique) qu'ils sont seuls à connaître.

En savoir plus sur les certificats, lien ici : https://www.certeurope.fr/blog/4-questions-pour-tout-comprendre-certificat-de-serveur/