Sécurisation des communications

Table des matières

1) 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.

2) Chiffrement symétrique

2.1) Différents codages

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.

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.

Exercice 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 autre exemple :

Utilisation de 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 b1 xor b2 = b3 alors b1 xor b3 = b2 et b2 xor b3=b1. Ce qui permet un chiffrement symétrique.

Table de vérité "XOR" :

E1	E2	S
0	0	0
0	1	1
1	0	1
1	1	0

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).

Exercice 2 :

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

Exercice 3 :

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

2.2) Décryptage par analyse fréquentielle (MP5)

a) Principe

Un texte qui a été chiffré via une substitution monoalphabétique (une lettre correspond à une seule autre) présente une sécurité très limitée, pour ne pas dire quasi nulle si le message est suffisamment long. En effet, ce type de chiffrement est totalement vulnérable à un procédé appelé l'analyse fréquentielle.

Exemple de cryptage par substitution mono-alphabétique :

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

Le principe de déchiffrement est simple. Des études statistiques sur un ensemble de textes (de longueur et de niveau de langage moyens) permettent de mettre en évidence le pourcentage moyen d'utilisation de chaque lettre. Dès lors, il suffit d'établir un tableau des fréquences du texte chiffré, et de le comparer avec le tableau des fréquences pour un texte français normal. Si on constate par exemple que, dans le texte chiffré, la lettre Q apparaît énormément, il y a de très fortes chances qu'elle corresponde au E (lettre très courante en français). Il faut alors procéder ainsi pour chaque lettre et évaluer les possibilités.

b) Les conditions pour une analyse fréquentielle efficace

— Le pourcentage varie selon les langues évidemment. Par exemple, en néerlandais, le Z est très courant, contrairement au français où il n'est quasiment jamais utilisé. Ce détail est primordial : si vous ne connaissez pas la langue dans laquelle le texte a été écrit, l'analyse des fréquences est inefficace voire impossible !

Pour la langue française, des études statistiques sur un ensemble de textes permettent de mettre en évidence le pourcentage d'utilisation de chaque lettre. Voici les résultats obtenus :

Lettre	A	B	C	D	E	F	G	H	I	J	K	L	M
%	8.40	1.06	3.03	4.18	17.26	1.12	1.27	0.92	7.34	0.31	0.05	6.01	2.96
Lettre	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
%	7.13	5.26	3.01	0.99	6.55	8.08	7.07	5.74	1.32	0.04	0.45	0.30	0.12

— Si le procédé de chiffrement n'est pas bijectif, c'est-à-dire si vous avez plusieurs correspondances possibles pour chaque lettre, encore une fois, l'analyse fréquentielle est impossible... En effet, si tel était le cas, le tableau des fréquences serait totalement faussé !
— Le texte chiffré soit suffisamment long, sans quoi l'établissement du tableau des fréquences.

c) Exercice

Voici un extrait d'une lettre de Denis Diderot, Principes philosophiques sur la matière et le mouvement :

```
BWFWKSAKWFIMWDKWFKDWKHZADGKGHZWKGFLKMHHGKWIMWDSESLA
WJWWLSALAFVAXXWJWFLWSMEGMNWEWFLWLSMJWHGKUWIMADQSVWTA
WFUWJLSAFUWKLIMWLGMKDWKUGJHKYJSNALWFLDWKMFKKMJDWKSMLJ
WKUWKLIMWLGMLWKDWKHSJLAUMDWKVWKUGJHKYJSNALWFLDWKMFW
KKMJDWKSMLJWKUWKLIMWVSFKUWLMFANWJKLGMLWKLWFLJSFKDSLAG
FGMAFFAKMGMWFLJSFKDSLAGFWLAFFAKMSDSXGAKUWLLWKMHHGKALA
GFVWKHZADGKGHZWKJWKKWETDWHWMLWLJWSUWDDWVWKYWGEWLJW
KIMASVEWLLWFLVWKHGAFLKKSFKSMUMFWVAEWFKAGFVWKDAYFWKKS
FKDSJYWMJFAHJGXGFVWMJVWKKMJXSUWKKSFKWHSAKKWMJGMHWMLW
LJWHSJDWFLADKVMJWHGKJWDSLAXVMFWESKKWSMFWSMLJWLGMLWKLV
SFKMFJWHGKJWDSLAXWFMFNSAKKWSMTSLLMHSJDSLWEHWLWJAWFFQW
KLWFMFJWHGKSTKGDMHSKEWEWDWKEGDWUMDWKSYJWYSLANWKFAVM
NSAKKWSMFAVWKUGJHKIMADJWFXWJEWKADKFWUGFUGANWFLHSKHDM
KVWLWFVSFUWSMJWHGKIMSMEGMNWEWFLVSFKMFUGJHKIMWDUGFIMUW
KLIMSHHSJWEEWFLADKJWYSJVWFLDSESLAWJWUGEEWZGEGYWFWUWKLI
MADKXGFLSTKLJSULAGFVWLGMLWKDWKIMSDALWKIMADMAKGFLWKK
WFLAWDDWKUWKLIMADKDSUGFKAVWJWFLUGEEWAFSDLWJSTDWVSFKDA
FKLSFLHJWKIMWAFVANAKATDWVWDWMJKHWUMDSLAGFUWKLIMADKJSA
KGFFWFLVMJWHGKJWDSLAXVMFSYJWYSLSMFSMLJWSYJWYSLUWKLIMAD
KGMTDAWFLIMWLSFVAKIMADKJSAKGFFWFLVWDAFVAXXWJWFUWVMUGJH
KSMEGMNWEWFLGMSMJWHGKDWTDGUVWESJTJWLWFVSKSVAKKGDMLAG
FUWKLIMADKSFWSFLAKKWFLHSJDSHWFKWWWLDWEGMNWEWFLYWFWJS
DIMASFAEWLGMKDWKUGJHKWLDWMJSULAGFHSJLAUMDAWJWVWKMFKK
MJDWKSMLJWKIMADWKVWLJMALLGMKUWKLIMWUWLLWAFVAXXWJWFU
WIMGAIMWXSMKKWWFWDDWEWEWESAKEGEWFLSFWWFWJWFVJSHSKDW
KDGAKVMEGMNWEWFLWJJGFWWK
```

Élaborer un programme qui vous aide à compléter le tableau suivant (les pourcentages seront arrondis à 10^{-2}) :
Attention : saisir le code en lettres majuscules et sans espace !!!

Lettre	A	B	C	D	E	F	G	H	I	J	K	L	M
Pourcentage d'apparition													
Lettre	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Pourcentage d'apparition													

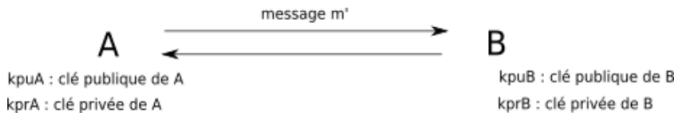
A l'aide des deux tableaux précédents, élaborer un programme qui décrypte le message .

3) Chiffrement asymétrique

Dans le cas du chiffrement asymétrique l'expéditeur (A) et le récepteur (B) n'ont pas besoin de partager une "clé secrète" :

— A possède une "clé privée" que l'on notera kprA et une "clé publique" que l'on notera kpuA. En aucun cas A ne devra divulguer sa clé privée à quiconque, elle devra rester strictement secrète. En revanche sa clé publique pourra être connue de tout le monde sans aucun problème.
— B possède une "clé privée" que l'on notera kprB et une "clé publique" que l'on notera kpuB. En aucun cas B ne devra divulguer sa clé privée à quiconque, elle devra rester strictement secrète. En revanche sa clé publique pourra être connue de tout le monde sans aucun problème.

Si A désire envoyer un message m à B, il va utiliser la clé publique de B afin de réaliser le chiffrement (m est chiffré en m'). Le message chiffré (m') va ensuite pouvoir transiter entre A et B. Une fois le message m' en sa possession, B va utiliser sa clé privée afin de pouvoir déchiffrer le message m' et ainsi obtenir le message m. Le processus peut être résumé par le schéma suivant :



Si P intercepte le message m', il sera incapable de déterminer m à partir de m' sans la clé privée de B.

Le chiffrement asymétrique repose sur des problèmes très difficiles à résoudre dans un sens et faciles à résoudre dans l'autre sens. Prenons un exemple : l'algorithme de chiffrement asymétrique RSA (du nom de ses 3 inventeurs : Rivest Shamir et Adleman), est très couramment utilisé, notamment dans tout ce qui touche au commerce électronique. RSA se base sur la factorisation des très grands nombres premiers. Si vous prenez un nombre premier A (par exemple A = 16813007) et un nombre premier B (par exemple B = 258027589), il facile de déterminer C le produit de A par B (ici on a A x B = C avec C = 4338219660050123). En revanche si je vous donne C (ici 4338219660050123) il est très difficile de retrouver A et B. En tous les cas, à ce jour, aucun algorithme n'est capable de retrouver A et B connaissant C dans un temps "raisonnable". Nous avons donc bien ici un problème relativement facile dans un sens (trouver C à partir de A et B) est extrêmement difficile dans l'autre sens (trouver A et B à partir de C). Les détails du fonctionnement de RSA sont relativement complexes (mathématiquement parlant) et ne seront pas abordés ici. Vous devez juste savoir qu'il existe un lien entre une clé publique et la clé privée correspondante, mais qu'il est quasiment impossible de trouver la clé privée de quelqu'un à partir de sa clé publique.

Remarque : Un nombre premier est un entier naturel qui admet exactement deux diviseurs distincts entiers et positifs. Ces deux diviseurs sont 1 et le nombre considéré, puisque tout nombre a pour diviseurs 1 et lui-même (comme le montre l'égalité n = 1 × n), les nombres premiers étant ceux qui n'en possèdent aucun autre. Par exemple, le nombre entier 7 est premier car 1 et 7 sont les seuls diviseurs entiers et positifs de 7.

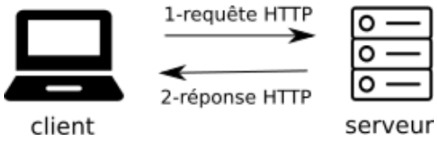
				0					
1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

4) Le protocole HTTPS

4.1) Contexte

Nous allons maintenant voir une utilisation concrète de ces chiffrements symétriques et asymétriques : le protocole HTTPS.

Avant de parler du protocole HTTPS, petit retour sur le protocole HTTP : un client effectue une requête HTTP vers un serveur, le serveur va alors répondre à cette requête (par exemple en envoyant une page HTML au client).

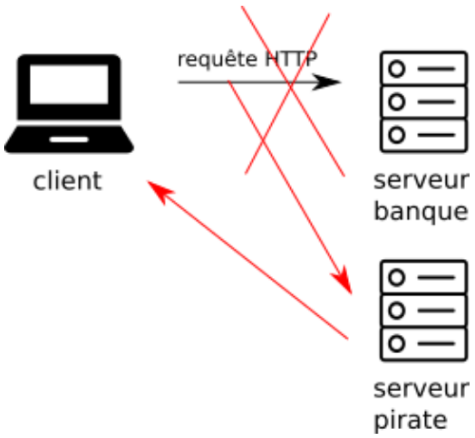


échange client-serveur protocole HTTP

Le protocole HTTP pose 2 problèmes en termes de sécurité informatique :

— Un individu qui intercepterait les données transitant entre le client et le serveur pourrait les lire sans aucun problème (ce qui serait problématique notamment avec un site de e-commerce au moment où le client envoie des données bancaires)

— grâce à une technique qui ne sera pas détaillée ici (le DNS spoofing), un serveur "pirate" peut se faire passer pour un site sur lequel vous avez l'habitude de vous rendre en toute confiance : imaginez vous voulez consulter vos comptes bancaires en ligne, vous saisissez l'adresse web de votre banque dans la barre d'adresse de votre navigateur favori, vous arrivez sur la page d'accueil d'un site en tout point identique au site de votre banque, en toute confiance, vous saisissez votre identifiant et votre mot de passe. Ainsi, un "pirate" va pouvoir récupérer votre identifiant et votre mot de passe ! En effet, vous avez saisi l'adresse web de votre banque comme d'habitude. A cause d'une attaque de type "DNS spoofing" vous avez été dirigé vers un site pirate, en tout point identique au site de votre banque. Dès vos identifiant et mot de passe saisis sur ce faux site, le pirate pourra les récupérer et se rendre avec sur le véritable site de votre banque.



Attaque DNS Spoofing

HTTPS est donc la version sécurisée de HTTP, le but de HTTPS est d'éviter les 2 problèmes évoqués ci-dessus. HTTPS s'appuie sur le protocole TSL (Transport Layer Security) anciennement connu sous le nom de SSL (Secure Sockets Layer)

4.2) Comment chiffrer les données circulant entre le client et le serveur ?

Les communications vont être chiffrées grâce à une clé symétrique.

Problème : comment échanger cette clé entre le client et le serveur ? Simplement en utilisant une paire clé publique / clé privée !

Voici le déroulement des opérations :

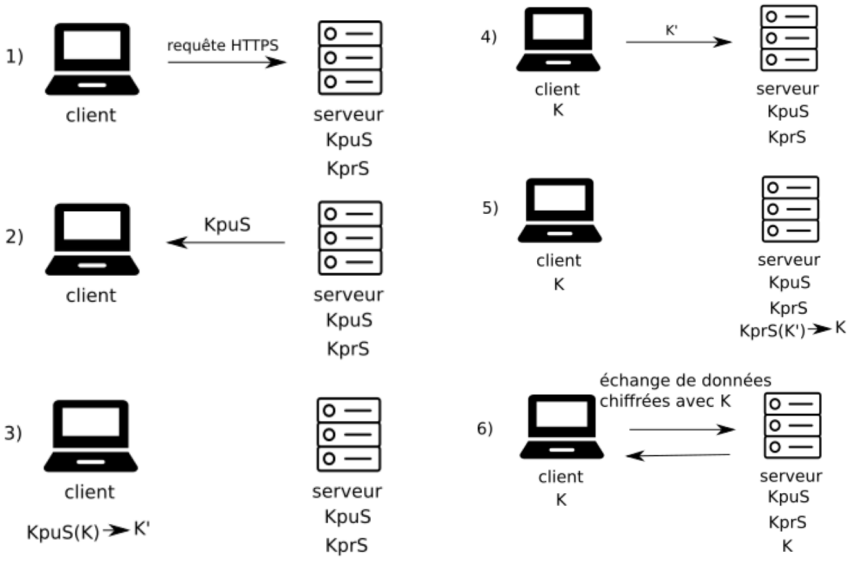
— le client effectue une requête HTTPS vers le serveur, en retour le serveur envoie sa clé publique (KpuS) au client

— le client "fabrique" une clé K (qui sera utilisé pour chiffrer les futurs échanges), chiffre cette clé K avec KpuS et envoie la version chiffrée de la clé K au serveur

— le serveur reçoit la version chiffrée de la clé K et la déchiffre en utilisant sa clé privée (KprS). À partir de ce moment-là, le client et le serveur sont en possession de la clé K

— le client et le serveur commencent à échanger des données en les chiffrant et en les déchiffrant à l'aide de la clé K (chiffrement symétrique).

On peut résumer ce processus avec le schéma suivant :



Ce processus se répète à chaque fois qu'un nouveau client effectue une requête HTTPS vers le serveur.

Comment éviter les conséquences fâcheuses d'une attaque de type DNS Spoofing ?

Pour éviter tout problème, il faut que le serveur puisse justifier de son "identité" ("voici la preuve que je suis bien le site de la banque B et pas un site "pirate""). Pour ce faire, chaque site désirant proposer des transactions HTTPS doit, périodiquement, demander (acheter dans la plupart des cas) un certificat d'authentification (sorte de carte d'identité pour un site internet) auprès d'une autorité habilitée à fournir ce genre de certificats (chaque navigateur web possède une liste des autorités dont il accepte les certificats). Comme dit plus haut, ce certificat permet au site de prouver son "identité" auprès des clients. Nous n'allons pas entrer dans les détails du fonctionnement de ces certificats, mais vous devez juste savoir que le serveur envoie ce certificat au client en même temps que sa clé publique (étape 2 du schéma ci-dessus). En cas d'absence de certificat (ou d'envoi de certificat non conforme), le client stoppe immédiatement les échanges avec le serveur. Il peut arriver de temps en temps que le responsable d'un site oublie de renouveler son certificat à temps (dépasse la date d'expiration), dans ce cas, le navigateur web côté client affichera une page de mise en garde avec un message du style "ATTENTION le certificat d'authentification du site XXX a expiré, il serait prudent de ne pas poursuivre vos échanges avec le site XXXX".