Sécurisation des communications

Capacités attendues

- ✓ Décrire les principes de chiffrement symétrique (clef partagée) et asymétrique (avec clef privée/clef publique). Décrire l'échange d'une clef symétrique en utilisant un protocole asymétrique pour sécuriser une communication HTTPS.
- ✓ Les protocoles symétriques et asymétriques peuvent être illustrés en mode débranché, éventuellement avec description d'un chiffrement particulier.

1 Introduction

La sécurisation des communications est un enjeu majeur à travers l'Histoire. La possession d'informations est généralement un facteur primordial de succès dans le cadre de conflits, que ceux-ci soient armés, économiques ou politiques.

Des méthodes permettant de garder ces informations secrètes ont donc été développées rapidement.

Afin d'empêcher la lecture par des personnes non concernées d'informations confidentielles, on utilise une méthode mathématique rendant tout message inintelligible à quiconque n'en possède pas la clé permettant de le déchiffrer.

On appelle cette méthode chiffrement .

Soit deux individus Alice et Bob qui cherchent à s'envoyer des messages par l'intermédiaire d'un réseau informatique. Alice et Bob désirent qu'une tierce personne (par exemple Pierrot) ne soit pas capable de lire les messages si par hasard ces derniers devaient être interceptés par Pierrot. Pour ce faire, Aliceva chiffrer le message . Toute personne qui ne possédera pas le moyen de déchiffrer ce message chiffré se verra dans l'impossibilité de comprendre le contenu du message (si Pierrot intercepte le message chiffré et qu'il ne possède pas le moyen de déchiffrer ce message, l'interception aura été totalement inutile puisque Pierrot sera dans l'incapacité de comprendre le contenu du message).

Il existe 2 grands types de chiffrement : le chiffrement symétrique et le chiffrement asymétrique .

2 Chiffrement symétrique

2.1 Chiffrement par décalage

L'un des codes le plus connu date de l'époque de César. Le *code de César*, ou *chiffrement par décalage*, consiste à décaler chaque lettre de l'alphabet d'un certain nombre de rangs fixe, appelé clé de chiffrement .

Par exemple, avec la clé 3 on obtient :

- alphabet d'origine : `ABCDEFGHIJKLMNOPQRSTUVWXYZ`
- alphabet modifié : `DEFGHIJKLMNOPQRSTUVWXYZABC`

Les armées de César utilisaient un tel code pour envoyer des messages sans risque que ceux-ci, s'ils étaient interceptés, puissent être lus par leur adversaire.

Par exemple, `"Bonjour"` devient `"Erqmrxu"`.

Afin de déchiffrer le message, il suffit d'utiliser le procédé inverse : on redécale les lettres dans l'ordre inverse, grâce à la clé de chiffrement.

Ce type de chiffrement, est appelé **chiffrement symétrique** car la clé qui permet de chiffrer le message est la même clé que celle qui permet de le déchiffer. Ce type de chiffrement possède une faille importante : si la clé est trop simple, il est facile, par la **force brute**, de chiffrer ou de déchiffrer le message.

Le code de César est aujourd'hui obsolète : n'importe quel ordinateur peut, en quelques millisecondes, déchiffrer un tel message en testant toutes les possibilités.

Pire encore, le code César est un type de chiffrement pour lequel les lettres se correspondent une à une. Il est possible de deviner la clé de chiffrement d'un message codé par un code César en analysant la fréquence d'occurrence de chaque lettre et en la comparant à l'occurrence naturelle des lettres dans la langue de notre choix.

De nombreux codes différents ont existé selon les périodes, chacun visant à améliorer les méthodes précédentes. On retiendra notamment le code de Vigénère dont la sécurité fut percée en 1863.

2.2 Chiffrement XOR

Soit le message `"Hello World!"`.

On peut le traduire en binaire en utilisant le code ASCII binaire de chaque caractère (voir le chapitre sur la *représentation des textes en machine* vu en première).

`010010000110010101101100011011000110111100100000010101110110111101110010011011000110010000100001`

(Pour faire la conversion : `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 `"toto"`qui se traduit en binaire :

`01110100011011110111010001101111`

Pour chiffrer le message nous allons effectuer un `XOR` bit à bit.

L'opération « ou exclusif », noté `xor` , est définie par A `xor` $B = (A$ `or` $B)$ `and` $\overline{A \texttt{ and } B}$, c'est-à-dire le $(A$ `or` $B)$ qu'on utilise en logique **mais** qui exclut le cas où A et B sont simultanément vrais.

Voici la table de vérité de cette opération :

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):

```
01001000011001010110110001101100011011110010000001010111011011110111001001101100011001000010000 1
XOR
01110100011011110111010001101111011101000110111101110100011011110111010001101111011101000110111 1
_______________________________________________________________________________________________
00111100000010100001100000000001100011011010011111001000110000000000000110000000110001000001001110
```

Maintenant ce message est prêt pour être envoyé à son destinataire B. Si Pierrot intercepte le message et cherche à le lire avec un éditeur de texte, il obtiendra une suite de caractère incompréhensible.

Bob 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).

```
00111100000010100001100000000001100011011010011111001000110000000000000110000000110001000001001110
XOR
01110100011011110111010001101111011101000110111101110100011011110111010001101111011101000110111 1
_______________________________________________________________________________________________
01001000011001010110110001101100011011110010000001010111011011110111001001101100011001000010000 1
```

Nous avons bien retrouvé le code binaire d'origine `"Hello World!"` !

Bob a pu lire le message envoyé par Alice alors que P, bien qu'il ait pu intercepter le message, n'a pas pu prendre connaissance de son contenu sans la clé.

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 chiffrement symétrique de nos jours

AES propose des clés dont la taille peut aller jusqu'à 256 bits, ce qui laisse 2^{256} possibilités de clés différentes, soit 10^77 clés différentes à tester.

À l'heure actuelle, la sécurité de l'information (et donc, par extension, la sécurité informatique), repose en grande partie sur la **robustesse des clés** utilisées pour chiffrer l'information. Les solutions proposées aujourd'hui ne sont donc pas parfaites. Avec une puissance de calcul suffisante, il pourrait devenir possible de systématiquement cracker un message protégé par AES.

Cette dernière affirmation est d'autant plus d'actualité avec les **progrès réalisés** dans le domaine de l'informatique quantique. Ce champ de l'informatique (pour lequel nous ne rentrerons pas dans les détails) vise à modifier la façon dont les calculs sont effectués par les machines.

Une fois perfectionné, ce nouveau type d'ordinateur pourrait théoriquement deviner extrêmement rapidement la clé utilisée pour chiffrer les messages à l'aide des méthodes de chiffrement actuellement utilisées par l'intégralité des acteurs du monde informatique (systèmes bancaires, militaires, médicaux, etc.).

Le gros problème avec le chiffrement symétrique, c'est qu'il est nécessaire pour Alice 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.

(TSVP)

3 Chiffrement asymétrique

Outre le problème de la possibilité de deviner par la force brute une clé, une autre faille existe dans le modèle du chiffrement symétrique.

Afin de pouvoir communiquer de manière sécurisée, les deux acteurs doivent posséder la clé en question. Or, comment s'assurer que lors de l'échange d'une clé secrète, celle-ci n'est pas volée par une tierce personne ? Ce problème est omniprésent avec Internet, puisqu'en théorie toutes les données transitant sur le réseau peuvent être interceptées.

En 1976, Whitfield Diffie et Martin Hellman présentent le principe d'un chiffrement à clé publique.

3.1 Fonctionnement

Alice et Bob souhaitent communiquer sans que leurs clés ne puisse être récupérée par un tiers.

- Alice fabrique :
  - un cadenas, sa **clé publique**
  - la clé pour l'ouvrir, sa **clé privée**

  Le cadenas doit être suffisamment compliqué pour ne pas être ouvert facilement sans clé. **La clé privée d'Alice, reste en sa possession**.
- Alice envoie à Bob sa clé publique. Comme le nom clé *publique* l'indique, cette clé est connue de tous.
- Bob utilise cette clé publique (d'Alice) pour chiffrer son message (ce qui reviendrait à enfermer dans une boîte son message à l'aide du cadenas envoyé par Alice). Connaître la clé publique d'Alice ne suffit pas à percer le message : pour ouvrir la boîte, il faut avoir en sa possession la clé qui correspond (et qui est restée avec Alice).
- Alice reçoit la boîte et l'ouvre avec sa clé privée à elle, qui n'a pas quitté sa maison et n'a donc pas pu être interceptée en cours de route.

En 1978, deux ans plus tard, Ronald Rivest, Adi Shamir et Leonard Adleman mettent en pratique cette théorie et créent un algorithme, **RSA** (du nom de ses créateurs), encore largement utilisé de nos jours.

Cet algorithme utilise des principes mathématiques tels que le petit théorème de Fermat afin d'assurer sa robustesse.

3.2 HTTPS

HTTPS fonctionne de la même manière que le chiffrement asymétrique décrite ci-dessus.

- le **client** envoie une **demande de connexion** sécurisée au serveur ;
- Le **serveur** envoie **sa clé publique ainsi que sa signature** ;
- Le **client vérifie la signature** en faisant appel à des autorités de certification extérieures (voir l'encadré ci-dessous).
- Après vérification que le serveur est authentique, le client génère une clé de chiffrement symétrique (AES), appelée **clé de session**, et la chiffre avec la clé publique du serveur.
- Le serveur reçoit la clé du client en toute sécurité.

Le client et le serveur peuvent maintenant s'échanger des données en utilisant la clé de session qu'ils sont les seuls à connaître. Pour désigner cet algorithme, on parle de poignée de main (handshake).

Pour information : les certificats d'authentification

Les serveurs achètent à ces organismes des certificats / signatures afin d'assurer leur authenticité.

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".

Récemment, un souci avec un organisme majeur (Let's Encrypt) dont les certificats avaient expiré ont rendu inaccessible une partie d'Internet.

Exercice 1

Qu'est-ce qu'Enigma en quelques lignes ?

Enigma est une machine électromécanique portative servant au chiffrement et au déchiffrement de l'information. Elle a été créée en 1919. Son utilisation la plus célèbre fut celle faite par l'Allemagne nazie et ses alliés, avant et pendant la Seconde Guerre mondiale, la machine étant réputée inviolable selon ses concepteurs. Néanmoins un nombre important de messages Enigma ont pu être déchiffrés près de sept ans avant la guerre puis pendant, grâce aux cryptanalystes britanniques, dont Alan Turing.

Exercice 2

Pour réaliser cet exercice, on se servira du chiffrement de Vigenère décrit ici.

Soient trois personnes Alice, Bob et Pierrot. Alice désire envoyer un message chiffré (chiffrement **symétrique**) à Bob. Pierrot est un pirate qui va essayer de déchiffrer un message qui ne lui est pas destiné.

Un élève joue le rôle d'Alice. Un autre jouera le rôle de Bob. Alice et Bob se mettent d'accord discrètement sur une clé de chiffrement/déchiffrement (choisissez un mot qui jouera le rôle de clé, ce mot doit rester secret).

Choisissez un message à faire parvenir à Bob puis procéder au chiffrement de ce message, notez le résultat du chiffrement (en binaire) sur une feuille. Donnez cette feuille à une ou un camarade tiers (qui ne connaît pas la clé, ce camarade jouera donc le rôle de Pierrot). Pierrot devra recopier le message avant de le transmettre à Bob. Pierrot devra essayer de trouver le message envoyé par Alice à Bob. Bob devra déchiffrer le message à l'aide de la clé.

Noter la méthode que Pierrot doit mettre en œuvre pour déchiffrer le message :

- Regarder dans la table de Vigenère toutes les possibilités pour une même lettre, et ce pour chaque lettre du message chiffré ;

- étudier les différentes combinaisons possibles pour identifier un texte qui ressemble à un message.

Qui obtient le résultat en premier entre Bob et Pierrot ?

C'est Bob car il possède la clef et n'a qu'à regarder la correspondance dans la table de Vigénère alors que Pierrot doit tester toutes les possibilités en espérant tomber sur la bonne.

Exercice 3

1. Après un chiffrement symétrique "XOR" 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.

   On donne l'extrait de la table ASCII suivant :

lettre	code binaire	lettre	code binaire
a	01100001	t	01110100
b	01100010	v	01110110
c	01100011	w	01110111
d	01100100	x	01111000
e	01100101	y	01111001
f	01100110	z	01111010
i	01101001	| (vertical bar)	01111100
r	01110010	{ (left opening brace)	01111101
s	01110011	~ (tilde)	01111110

   "ri", message chiffré, en binaire : 0111 0010 0110 1001 -> résultat du XOR entre le clef 0000 1010 et le message d'origine. On déduit du XOR que le message d'origine est : 0111 1000 0110 0011, soit "xc"

2. Un utilisateur B souhaite échanger un message chiffré avec un utilisateur A en utilisant un chiffrement **asymétrique**. A possède une clé publique `AKpub` et une clé privée `AKpriv`. B possède une clé publique `BKpub` et une clé privée `BKpriv`. B souhaite chiffrer un message m afin de pouvoir l'envoyer à A.

   (a) Quelle clé va être utilisée par B pour chiffrer le message m ?

   La clef publique de A (AKpub, celle « de l'autre » personne).

   (b) Quelle clé va être utilisée par A pour déchiffrer le message m ?

   La clef privée de A (AKpriv, la « sienne »).

3. Expliquer en quelques lignes le principe du protocole HTTPS (on s'intéressera uniquement à l'aspect Sécurité du protocole).

   HTTPS permet de chiffrer les données qui transitent sur le web grâce à une méthode de chiffrement asymétrique, accompagnée d'un certificat géré par un organisme extérieur dédié.

Exercice 4 – *(Exercice tiré du bac 2021)*

Pour chiffrer un message, une méthode, dite du masque jetable, consiste à le combiner avec une chaîne de caractères de longueur comparable. Une implémentation possible utilise l'opérateur XOR (ou exclusif).

Dans la suite, les nombres écrits en binaire seront précédés du préfixe `0b`.

1. Pour chiffrer un message, on convertit chacun de ses caractères en binaire (à l'aide du format Unicode), et on réalise l'opération `XOR` bit à bit avec la clé.

Après conversion en binaire, et avant que l'opération `XOR` bit à bit avec la clé n'ait été effectuée, Alice obtient le message suivant :

```
m = 0b 0110 0011 0100 0110
```

(a) Le message `m` correspond à deux caractères codés chacun sur 8 bits : déterminer quels sont ces caractères. On fournit pour cela la table ci-dessous qui associe à l'écriture hexadécimale d'un octet le caractère correspondant (figure 2).

Exemple de lecture : le caractère correspondant à l'octet codé `4A` en hexadécimal est la lettre `J`.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2	space	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

0110 0011 => 63 en hexa => caractère "c"
0100 0110 => 46 en hexa => caractère "F"
donc "cF"

(b) Pour chiffrer le message d'Alice, on réalise l'opération XOR bit à bit avec la clé suivante :

```
k = 0b 1110 1110 1111 0000
```

Donner l'écriture binaire du message obtenu.

0b 1000 1101 1011 0110

2. (a) Dresser la table de vérité de l'expression booléenne suivante :

```
(a XOR b) XOR b
```

A	B	A xor B	(A xor B) xor B
0	0	0	0
0	1	1	0
1	0	1	1
1	1	0	1

Qu'observe-t-on ?

(a XOR b) XOR b = A

(b) Bob connaît la chaîne de caractères utilisée par Alice pour chiffrer le message. Quelle opération doit-il réaliser pour déchiffrer son message ?

On peut remarquer que (a xor b) xor b permet de retrouver a, donc si a correspond au message non chiffré et a xor b correspond au message chiffré, un (a xor b) xor b permet donc de retrouver le message non chiffré. Si on appelle m le message non chiffré, m' le message chiffré et k la clé de chiffrement, un m' xor k permettra de retrouver m.

Source parmi d'autres : `https://dav74.github.io/site_nsi_term/`