Exercices - Sécurisation des communications

Exercice 1 Questions de cours

1. Quels sont les 3 objectifs de la sécurisation des communications ?
2. Quelle est la différence entre un chiffrement symétrique et un chiffrement asymétrique ?
3. Citer 2 chiffrements symétriques et 1 chiffrement asymétrique.
4. Quelle sont les avantages et inconvénients de ces deux types de chiffrement ?

Exercice 2

Tom souhaite envoyer un message x à Maë en utilisant le protocole de communication RSA.

- Tom possède une clé privée C_{Tom}^{Priv} et une clé publique C_{Tom}^{Pub}.
- Maë possède uen clé privée C_{Mae}^{Priv} et une clé publique C_{Mae}^{Pub}

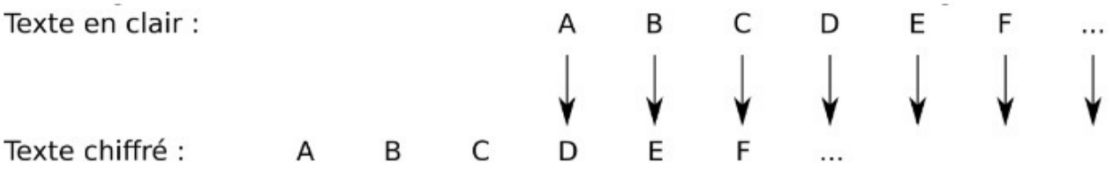
1. Décrire à l'aide d'un schéma le mécanisme d'envoi du message x utilisant le protocole RSA.
2. Max souhaite intercepter le message x. Expliquer pourquoi il ne peut pas déchiffrer ce message. Comment s'appelle ce genre d'attaque malveillante ?

Exercice 3

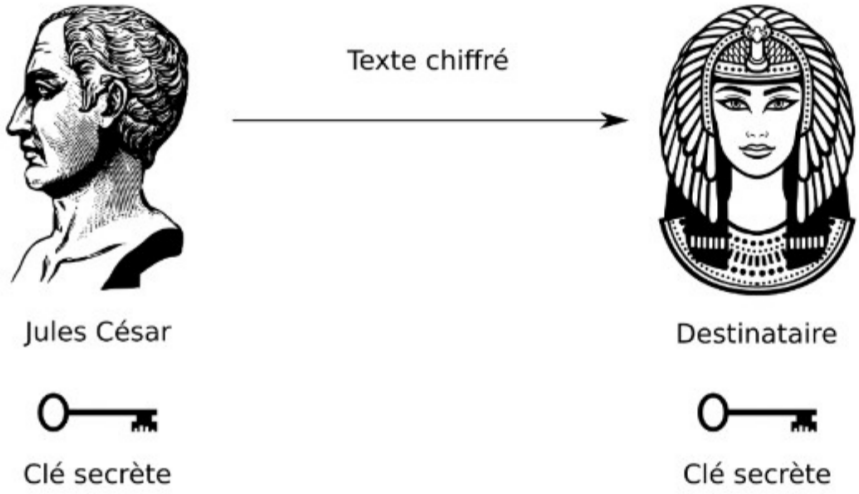
1. Décrire à l'aide d'un schéma commenté le fonctionnement d'un système d'authentification RSA par autorité de certification.
2. Décrire en quelques lignes le principe de fonctionnement du protocole HTTPS. On s'intéressera uniquement à l'aspect sécurité du protocole.

Exercice 4 Le chiffre de César

L'exemple le plus connu de chiffrement symétrique, et un des plus simples, est le chiffre de César. Cette méthode de communication chiffrée aurait été inventée par Jules César lui-même.
Pour obtenir le texte chiffré, le chiffre de César consiste à remplacer chaque lettre du texte en clair par la lettre obtenue après un décalage d'un nombre fixe de lettres dans l'alphabet.
Pour un décalage de 3 lettres, le A devient D, le B devient E... Comme le montre l'image ci-dessous



Le nombre 3, qui correspond au nombre de lettres à décaler, est appelé la clef secrète de chiffrement. Pour déchiffrer le message, il suffit de donner la clef de chiffrement au destinataire du message qui réalise l'opération inverse pour déchiffrer le message.



Jules César et son destinataire utilisent la même clef de chiffrement et elle n'est connue que par eux. Ce chiffrement est donc symétrique et cette clef est la clef partagée.

1. Déchiffrer manuellement le message chiffré « qf hwduytlwfumnj jxy zynqnxjj ifsx qjx sfanlfyjzwx bjg ! » avec la clef partagée 5.
2. Ecrire une fonction python permettant de déchiffrer un message à l'aide du code de César avec une clef partagée n.
   Vérifier le bon fonctionnement de votre fonction en déchiffrant le message de la question 1.

```
def dechiffrement_cesar(texte_chiffre, n):
        # ACOMPLETER
        return texte_dechiffre
```

3. Ecrire une fonction python permettant de chiffrer un message à l'aide du code de César avec une clef partagée n.
   Vérifier le bon fonctionnement de votre fonction en chiffrant le message de la question 2.

```
def chiffrement_cesar(texte_a_chiffrer, n):
        # ACOMPLETER
        return texte_chiffre
```

   Ce chiffrement fait apparaître quelques limites intéressantes, dont le principe peut s'appliquer à d'autres algorithmes plus modernes :

   - la clef 0 n'est pas utilisable, car elle ne chiffre pas le message (notion de clef faible) ;
   - si on se limite à l'alphabet sans ajouter de signe de ponctuation, le nombre de clefs possibles est limité à 25. Il est donc possible de toutes les tester et il s'agit alors d'une attaque dite par « force brute » ;
   - l'informatique permet également de réaliser une analyse de la fréquence des occurrences de chaque lettre dans une langue et de la comparer aux fréquences trouvées dans le message chiffré. On sait par exemple qu'en français, la lettre la plus fréquente est le « e ». Si la lettre la plus fréquente du message chiffré est le « f », la clef vaut alors certainement 1.

4. Ecrire deux versions d'une fonction Python qui renvoie la lettre la plus fréquente du texte passé en argument.
   Si des lettres ex æquo apparaissent, on choisit parmi les plus fréquentes la première qui apparait dans le message chiffré.

(a) Version utilisant un dictionnaire

```
def analyse_frequentielle_avec_dico(texte_chiffre):
        dico = {}
        #A COMPETER
```

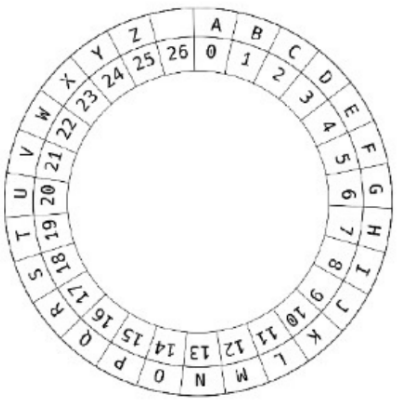
(b) Version utilisant une liste

```
def analyse_frequentielle_avec_liste(texte_chiffre):
        liste = []
        #A COMPETER
```

Exercice 5 **Le chiffre de Vernam**

Le chiffrement symétrique dit de Vernam utilise comme clef partagée une chaîne de caractères aussi longue que le message à chiffrer. L'algorithme consiste alors à :

- faire correspondre chaque lettre du message à chiffrer avec chaque lettre de la clef partagée ;
- convertir chaque lettre en nombre. Par exemple : 0 pour A, 1 pour B, 2 pour C ... Le caractère « espace » est aussi codé par le nombre 26. Les décalages sont alors modulo 27 : la lettre est codée par le reste de la division euclidienne par 27.
- Il est aussi possible d'utiliser le codage ASCII pour convertir chaque lettre en nombre : 65 pour A, 66 pour B ... ;
- décaler dans l'alphabet chaque lettre du message à chiffrer. Ce décalage est égal au nombre correspondant à la lettre qui a la même position dans la clef partagée. Cela revient à effectuer un chiffrement de César avec un décalage différent pour chaque lettre du message à chiffrer. Attention : pour garantir la sécurité de la méthode, la clef doit être choisie de façon aléatoire et ne pourra servir qu'une fois.



En utilisant le codage de l'illustration, le texte :

L	A		T	E	R	R	E		E	S	T		R	O	N	D	E

chiffré avec la clef :

Q	W	R	B	O	K	L	E	P	Y	C	F	O	R	A	H	Z	Q

devient :

A	W	Q	U	S	A	B	I	O	B	U	Y	N	H	O	U	B	U

Prenons l'exemple de la lettre S à chiffrer. Sa lettre correspondante dans la clef est C. Ceci occasionne un décalage de 2 dans l'alphabet et donc la lettre S devient U.

1. Ecrire une fonction Python qui calcule l'index de la lettre du texte à chiffrer (sens = 1) ou à déchiffrer (sens = -1) puis qui renvoie le texte chiffré ou déchiffré. Le texte et la clef ne doivent contenir que des lettres en minuscules et des espaces.

```
def vernam(texte, cle, sens=1):
        # A COMPLETER
        return texte_chiffre
```

Utilisé correctement, cet algorithme est théoriquement inviolable. Il faut néanmoins faire attention à certains écueils :

- une clef de chiffrement déjà utilisée ne doit pas être réutilisée puisque si deux messages ont été chiffrés avec la même clef de chiffrement, il suffit d'effectuer une soustraction pour annuler l'effet du masque et avec un peu plus de travail, de retrouver les deux textes en clair ;
- la clef se doit d'être aléatoire, par exemple en la générant avec un générateur aléatoire. Si ce n'est pas le cas, il est possible de retrouver la séquence qui l'a générée ;
- comme la clef, aussi grande que les données échangées, n'est pas réutilisable, il faut au préalable partager, par un autre canal sécurisé, une assez grande quantité de données pour permettre les futurs échanges. Ce système a été utilisé durant la guerre froide par le « Téléphone rouge » entre le Kremlin et la Maison-Blanche, qui était en fait plus un téléscripteur qu'un réel téléphone. La « clef », quant à elle, était échangée par valise diplomatique.

Exercice 6 Chiffrement XOR

L'objectif de l'exercice est d'étudier une méthode de cryptage d'une chaîne de caractères à l'aide du codage ASCII et de la fonction logique XOR.

1. Le nombre 65, donné ici en écriture décimale, s'écrit 01000001 en notation binaire. En détaillant la méthode utilisée, donner l'écriture binaire du nombre 89.
2. La fonction logique OU EXCLUSIF, appelée XOR et représentée par le symbole $\oplus$, fournit une sortie égale à 1 si l'une ou l'autre des deux entrées vaut 1 mais pas les deux.
   On donne ci-contre la table de vérité de la fonction XOR.

E_1	E_2	$E_1 \oplus E_2$
0	0	0
0	1	1
1	0	1
1	1	0

   Si on applique cette fonction à un nombre codé en binaire, elle opère bit à bit.

$$\begin{array}{rr} & 1100 \\ \oplus & 1010 \\ \hline = & 0110 \end{array}$$

   Poser et calculer l'opération : $11001110 \oplus 01101011$
3. On donne, ci-dessous, un extrait de la table ASCII qui permet d'encoder les caractères de A à Z.
   On peut alors considérer l'opération XOR entre deux caractères en effectuant le XOR entre les codes ASCII des deux caractères.
   Par exemple : 'F' XOR 'S' sera le résultat de $01000110 \oplus 01010011$.

Code ASCII Décimal	Code ASCII Binaire	Caractère
65	01000001	A
66	01000010	B
67	01000011	C
68	01000100	D
69	01000101	E
70	01000110	F
71	01000111	G
72	01001000	H
73	01001001	I
74	01001010	J
75	01001011	K
76	01001100	L
77	01001101	M

Code ASCII Décimal	Code ASCII Binaire	Caractère
78	01001110	N
79	01001111	O
80	01010000	P
81	01010001	Q
82	01010010	R
83	01010011	S
84	01010100	T
85	01010101	U
86	01010110	V
87	01010111	W
88	01011000	X
89	01011001	Y
90	01011010	Z

On souhaite mettre au point une méthode de cryptage à l'aide de la fonction XOR.
Pour cela, on dispose d'un message à crypter et d'une clé de cryptage de même longueur que ce message. Le message et la clé sont composés uniquement des caractères du tableau ci-dessus et on applique la fonction XOR caractère par caractère entre les lettres du message à crypter et les lettres de la clé de cryptage.
Par exemple, voici le cryptage du mot ALPHA à l'aide de la clé YAKYA :

Message à crypter	A	L	P	H	A
Clé de cryptage	Y	A	K	Y	A
	↓	↓	↓	↓	↓
Message crypté	'A' XOR 'Y'	'L' XOR 'A'	'P' XOR 'K'	…	…

Question : Ecrire une fonction `xor_crypt(message, cle)` qui prend en paramètres deux chaînes de caractères et qui renvoie la liste des entiers correspondant au message crypté.
Aide :

- On pourra utiliser la fonction native du langage Python `ord(c)` qui prend en paramètre un caractère `c` et qui renvoie un nombre représentant le code ASCII du caractère `c`.
- On considère également que l'on dispose d'une fonction écrite `xor(n1,n2)` qui prend en paramètre deux nombres $n1$ et $n2$ et qui renvoie le résultat de $n1 \oplus n2$.

4. On souhaite maintenant générer une clé de la taille du message à partir d'un mot quelconque. On considère que le mot choisi est plus court que le message, il faut donc le reproduire un certain nombre de fois pour créer une clé de la même longueur que le message.
   Par exemple, si le mot choisi est YAK pour crypter le message ALPHABET,
   la clé sera YAKYAKYA.
   Créer une fonction `generer_cle(mot,n)` qui renvoie la clé de longueur `n` à partir de la chaîne de caractères `mot`.
5. Recopier et compléter la table de vérité de $(E_1 \oplus E_2) \oplus E_2$.

E_1	E_2	$E_1 \oplus E_2$	$(E_1 \oplus E_2) \oplus E_2$
0	0	0	
0	1	1	
1	0	1	
1	1	0	

A l'aide de ce résultat, proposer une démarche pour déchiffrer un message chiffré.

Exercice 7 Chiffrement XOR

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) dont voici la table de vérité

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

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'ooctet 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

Figure 2

(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.

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

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

Exercice 8 Chiffrement AES

Le chiffrement AES (Advanced Encryption Standard) est un algorithme de chiffrement symétrique largement utilisé pour sécuriser les communications et le stockage des données. Il a été adopté comme standard en 2001 par le NIST (National Institute of Standards and Technology) et remplace l'ancien algorithme DES.

Principe du chiffrement AES

AES fonctionne sur des **blocs de données** de 128 bits et prend en entrée une clé de 128, 192 ou 256 bits. Le processus de chiffrement consiste en plusieurs tours (10, 12 ou 14 selon la taille de la clé) où les données subissent des transformations spécifiques pour assurer la confusion et la diffusion.

Structure d'un tour AES

Chaque tour de chiffrement comprend les étapes suivantes :

1. **SubBytes** : Substitution non linéaire de chaque octet via une table de substitution (S-Box).
2. **ShiftRows** : Décalage circulaire des lignes de la matrice d'état pour mélanger les données.
3. **MixColumns** (sauf au dernier tour) : Transformation linéaire appliquée aux colonnes pour diffuser l'information.
4. **AddRoundKey** : Application d'un XOR entre la matrice d'état et une sous-clé dérivée de la clé principale.

Le premier tour applique uniquement *AddRoundKey*, et le dernier tour n'inclut pas *MixColumns*.

Nombre de tours et sécurité

Le nombre de tours dépend de la taille de la clé :

— **AES-128** : 10 tours
— **AES-192** : 12 tours
— **AES-256** : 14 tours

Chaque tour renforce la résistance contre les attaques cryptographiques. Le chiffrement ne se limite pas à un simple XOR avec des sous-clés, mais repose sur un mélange de substitutions et de permutations pour améliorer la sécurité. En résumé, AES ne se contente pas d'un simple XOR avec des sous-clés différentes, mais génère des clés intermédiaires pour chaque tour via un mécanisme non linéaire combinant substitutions et opérations sur les bits.

AES est aujourd'hui l'un des algorithmes de chiffrement les plus sûrs et les plus utilisés dans le monde. Il est employé dans divers domaines comme le stockage de données, les communications sécurisées et la protection des fichiers.

Questions

1. Pourquoi AES utilise-t-il plusieurs tours de transformations au lieu d'une seule ?
2. Comment une clé plus longue (256 bits au lieu de 128 bits) améliore-t-elle la sécurité d'AES ?
3. Pourquoi le chiffrement AES est-il considéré comme sûr aujourd'hui ?