Informatique MP⋆, 2019-2020

Codage et algorithmes de cryptographie.

Table des matières

I Introduction à la cryptographie.

I.1 Cryptographie.

Le mot cryptographie vient des deux mots grecs « Cruptos » (qui signifie caché, dissimulé) et de « Graphein » (qui signifie écrire).
Le mot cryptographie est un terme générique désignant l'ensemble des techniques permettant de chiffrer des messages, c'est-à-dire permettant de les rendre inintelligibles sans une action spécifique. Le verbe crypter est parfois utilisé mais on lui préfèrera le verbe chiffrer.

Les quatre buts de la cryptographie :

- **Confidentialité :** mécanisme pour transmettre des données de telle sorte que seul le destinataire autorisé puisse les lire.
- **Intégrité :** mécanisme pour s'assurer que les données reçues n'ont pas été modifiées durant la transmission.
- **Authentification :** mécanisme pour permettre d'identifier des personnes ou des entités et de certifier cette identité.
- **Non-répudiation :** mécanisme pour enregistrer un acte ou un engagement d'une personne ou d'une entité de telle sorte que celle-ci ne puisse pas nier avoir accompli cet acte ou pris cet engagement.

Ainsi, la cryptographie permettra de transformer un texte en un autre texte, a priori, incompréhensible.
Le texte, une chaîne de caractères, sera transformé par un « calcul » en un autre texte. Le nouveau message obtenu est appelé cryptogramme par opposition au message de départ appelé message en clair.
Il faut bien sûr que le destinataire soit capable de retransformer le cryptogramme en message en clair.
L'action de coder le message s'appelle **chiffrement**, l'action de le décoder **déchiffrement**.

I.2 Clé.

En général un message est codé à l'aide d'une clé de chiffrement (un nombre ou un mot ou une phrase ou même parfois un texte) et décodé à l'aide d'une clé de déchiffrement.
Si les deux clés sont les mêmes, on parle de clés symétriques ; si les deux sont différentes, on parle de clés asymétriques.

I.3 Résumé.

Message de départ	Chiffrement → → Clé de chiffrement	Message codé	Déchiffrement → → Clé de déchiffrement	Message de départ

I.4 La scytale : un des plus anciens dispositifs de cryptographie.

Chez les Spartiates, la scytale, également connue sous le nom de bâton de Plutarque, était un bâton de bois utilisé pour lire ou écrire une dépêche chiffrée. La scytale est considérée comme le plus ancien dispositif de cryptographie militaire connue (les Egyptiens et les Phéniciens auraient aussi occasionnellement envoyé des messages chiffrés). Elle permettait l'inscription d'un message chiffré sur une fine lanière de cuir ou de parchemin que le messager pouvait porter à sa ceinture.

Après avoir enroulé la ceinture sur la scytale, le message était écrit en plaçant une lettre sur chaque circonvolution. Pour le déchiffrer, le destinataire devait posséder un bâton d'un diamètre identique à celui utilisé pour l'encodage. Il lui suffit d'enrouler la scytale autour de ce bâton pour obtenir le message en clair.

Il s'agit de l'un des plus anciens chiffrements de transposition ayant été utilisé. Plutarque raconte son utilisation par Lysandre de Sparte en 404 av. J.-C.

Exercice d'application 1. *Déchiffrer le message suivant : 'CVIXREEEIE_OSU_CUPFSR' sachant que ce texte sur la scytale correspond à un nombre de tours complets.*

I.5 Préliminaires Python.

Par la suite nous allons voir deux types de chiffrement assez simples : le code César et l'algorithme de Vigenère puis un chiffrement plus complexe : le chiffrement RSA.
L'objectif est de coder un texte et de savoir décoder celui-ci.
Nous utiliserons les deux instructions suivantes qui nous permettront de voir les lettres comme des chiffres.

- ***chr*** qui, à un chiffre (code ASCII), associe un caractère.
  Par exemple :

```
>>> chr(97)
'a'
>>> chr(98)
'b'
>>> chr(121)
'y'
>>> chr(122)
'z'
```

```
>>> chr(65)
'A'
>>> chr(66)
'B'
>>> chr(89)
'Y'
>>> chr(90)
'Z'
```

- ***ord*** qui est la fonction réciproque de ***chr*** et qui, à un caractère, associe son code ASCII.
  Par exemple :

```
>>> ord('a')
97
>>> ord('b')
98
>>> ord('y')
121
>>> ord('z')
122
```

```
>>> ord('A')
65
>>> ord('B')
66
>>> ord('Y')
89
>>> ord('Z')
90
```

II Le code César.

II.1 César.

Jules César était un général, homme politique et écrivain romain, né à Rome le 12 juillet ou le 13 juillet 100 av. J.-C. et mort le 15 mars 44 av. J.-C. Il aurait été assassiné par une conspiration, son propre fils Brutus lui portant le coup de grâce. César s'est illustré lors de la guerre des Gaules (et aussi dans les BD d'Astérix).

II.2 Principe.

Le code de César est la méthode de cryptographie la plus ancienne communément admise par l'histoire. Il consiste en une substitution mono-alphabétique : chaque lettre est remplacée ("substitution") par une seule autre ("mono-alphabétique"), selon un certain décalage dans l'alphabet. D'après Suétone, César avait coutume d'utiliser un décalage de 3 lettres : A devient D, B devient E, C devient F, Z devient C ... Il écrivait donc son message normalement, puis remplaçait chaque lettre par celle qui lui correspondait.

Exemple. *Si on prend, comme César, une clé de 3, on obtient :*

Av. code	*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*
Ap. code	*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*

- *Comment César aurait-il codé le mot « info » ?*

- *Coder le mot « cesar » avec une clé de 5.*

Exercice d'application 2.

1. *Décoder "wakyzout yaobgtzk" avec une clé de 6.*
2. *Proposer une méthode pour décoder le message : "foh ghfrghh idflohphqw", puis décoder le message.*

II.3 Sécurité.

Niveau sécurité, le code de César n'est pas fiable du tout (en particulier pour les grands textes), et ce pour deux raisons :

- Il n'existe que 25 façons différentes de crypter un message : puisqu'on ne dispose que de 26 lettres, il n'y a que 25 décalages possibles. Dès lors, des attaques exhaustives (tester tous les décalages un à un) ne demanderaient que très peu de temps.
- Le chiffre de César est très vulnérable à l'analyse des fréquences clés. La lettre "e" ayant, en langue française, une occurrence largement supérieure aux autres lettres, on devine ainsi la lettre correspondant au "e" et on a la clé pour décoder le message.

II.4 Implémentation en Python.

On verra l'implémentation du code César en TP.

On peut cependant en exposer le principe ici.
Dans un souci de simplification, notre code César ne modifiera que les lettres minuscules.

- Écrire une fonction ***decal(lettre , p)*** qui à *lettre* renvoie la lettre décalée de p si c'est une lettre minuscule. Par exemple *decal('m',3)* donne 'p'.
- Écrire une fonction ***cesar(ch,p)*** qui décale les lettres minuscules de *ch* de p lettres dans l'ordre alphabétique sans modifier les autres.
- Pour décoder un texte, si on connaît la clé, il suffit de prendre le décalage $26 - p$ et sinon, pour craquer le codage, il suffit de tester les 25 décodages possibles... (on pourra commencer par tester un mot pour trouver la clé).

III L'algorithme de Vigenère.

III.1 Vigenère.

Blaise De Vigenère est né le 5 avril 1523 à Saint-Pourçain-sur-Sioule et mort le 19 février 1596 à Paris. Il était diplomate, cryptographe, traducteur, alchimiste et astrologue .

III.2 Principe général.

Le code de Vigenère est un système de chiffrement par substitution poly-alphabétique, élaboré par Blaise de Vigenère. Utilisant généralement une clé et un tableau à double entrée, il permet de remplacer une lettre par une autre qui n'est pas toujours la même (contrairement au code César).
Pour ce chiffrement la clé est plus grande qu'un simple chiffre, il peut s'agir d'un mot, d'une phrase (voir d'un texte) si l'on veut coder avec plus de sécurité un grand texte.

III.3 La table de Vigenère.

La table ci-dessous est appelée table de Vigenère, c'est un outil indispensable pour coder un texte par la méthode de chiffrement de Vigenère. On va voir comment l'utiliser dans la partie suivante. (Inutile de l'apprendre évidemment !)

La 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

III.4 Exemple de chiffrement.

Exemple. *On part d'un texte de départ, par exemple : "AS MONTFERRAND" et d'une clé par exemple "JAUNE".*
Ensuite à chaque lettre du texte on associe la lettre de la clé, dont on aura répété le motif, qui est en même position.
Pour notre exemple on obtient :

Texte de départ	*A*	*S*		*M*	*O*	*N*	*T*	*F*	*E*	*R*	*R*	*A*	*N*	*D*
Clé (répétée)	*J*	*A*		*U*	*N*	*E*	*J*	*A*	*U*	*N*	*E*	*J*	*A*	*U*
Texte codé	*J*	*S*		*G*	...	...	...	...	...	...	...	...	...	...

La lettre A sera codée avec le code Cesar qui à la lettre A associe la lettre J.
En pratique, on code avec la lettre située à l'intersection de la ligne A et de la colonne J.

La lettre S sera codée avec le code Cesar qui à la lettre A associe la lettre A.
En pratique, on code avec la lettre située à l'intersection de la ligne S et de la colonne A.

La lettre M sera codée avec le code cesar qui à la lettre A associe la lettre U.
En pratique, on code avec la lettre située à l'intersection de la ligne M et de la colonne U.

La lettre O sera codée avec le code cesar qui à la lettre A associe la lettre N.
En pratique, on code avec la lettre située à l'intersection de ...

Exercice d'application 3. *Finir le codage ci-dessus.*

Exercice d'application 4.

1. *Décoder le message "JSGNW QT TXEM" avec la clé "ASM".*
2. *Ce code vous semble-t-il sûr ?*

III.5 Sécurité.

Bien que ce chiffrement soit beaucoup plus sûr que le code de César, il peut encore être facilement cassé.
En effet, lorsque les messages sont beaucoup plus longs que la clef, il est possible de repérer la longueur de la clef et d'utiliser, pour chaque séquence de la longueur de la clef, la méthode consistant à calculer la fréquence d'apparition des lettres, permettant de déterminer un à un les caractères de la clef...
Pour optimiser le chiffrement, il est donc préférable d'avoir une clé presque aussi longue que le texte à chiffrer.

Théorème III.1.
Le protocole de chiffrement de Vigenère avec une clé de ***longueur égale ou supérieure*** *à la longueur du message et utilisée* ***une et une seule*** *fois est totalement sûr.*

III.6 Implémentation en Python.

Exercice d'application 5. *Écrire un script Python qui donne la table de Vigenère.*

On verra l'implémentation de l'algorithme de Vigenère en TP.

Le but est de :

- Écrire une fonction ***code_vigenere(texte,cle)*** : qui code *texte* avec la clef *cle*.
- Écrire une procédure ***decode_vigenere(texte,cle)*** : qui décode *texte* avec la clef *cle*.

IV La cryptographie à clé publique.

Les Grecs, pour envoyer des messages secrets, rasaient la tête du messager, tatouaient le message sur son crâne et attendaient que les cheveux repoussent avant d'envoyer le messager effectuer sa mission !
Il est clair que ce principe repose uniquement sur le secret de la méthode.

IV.1 Le principe de Kerckhoffs.

Cette méthode rudimentaire va à l'encontre du principe de Kerckhoffs.
Auguste Kerckhoffs von Nieuwenhoff (1835 - 1903) est un cryptologue militaire néerlandais.
Le principe de Kerckhoffs s'énonce ainsi :

« La sécurité d'un système de chiffrement ne doit reposer que sur la clé.»

Cela se résume aussi par :

« L'ennemi peut avoir connaissance du système de chiffrement.»

Ce principe est novateur dans la mesure où, intuitivement, il semble opportun de dissimuler le maximum de choses possibles : clé et système de chiffrement utilisés. Mais l'objectif visé par Kerckhoffs est plus académique, il pense qu'un système dépendant d'un secret mais dont le mécanisme est connu de tous sera testé, attaqué, étudié, et finalement utilisé s'il s'avère intéressant et robuste.

IV.2 Factorisation des entiers.

Quels outils mathématiques répondent au principe de Kerckoffs ?
Un premier exemple est la toute simple multiplication ! En effet :

- si je vous demande combien font 5×7, vous répondez 35,
- si je vous demande de factoriser 35 vous répondez 5×7.
- si je vous demande de factoriser 1591, vous allez faire plusieurs tentatives...
- si je vous avais directement demandé de calculer 37×43, vous auriez rapidement trouvé 1591.

Pour des entiers de plusieurs centaines de chiffres le problème de factorisation ne peut être résolu en un temps raisonnable, même pour un ordinateur. C'est ce problème asymétrique qui est à la base de la cryptographie RSA (que nous détaillerons plus tard) : connaître p et q apporte plus d'information utilisable que $p \times q$, même si, en théorie, à partir de $p \times q$ on peut retrouver p et q, en pratique ce sera impossible en un temps raisonnable.

Formalisons ceci avec la notion de complexité. La complexité est le temps de calcul (ou le nombre d'opérations élémentaires) nécessaire pour effectuer une opération.

Pour calculer la somme de deux entiers à n chiffres, la complexité est d'ordre n (exemple : pour calculer 1234 + 2323, il faut faire 4 additions de chiffres).

La multiplication de deux entiers à n chiffres est de complexité d'ordre n^2. Par exemple pour multiplier 1234 par 2323 il faut faire 16 multiplications de chiffres (chaque chiffre de 1234 est à multiplier par chaque chiffre de 2323) et quelques additions.

Par contre la meilleure méthode de factorisation connue est de complexité d'ordre $\exp(4n^{\frac{1}{3}})$ (c'est moins que $\exp(n)$, mais plus que n^d pour tout d, lorsque n tend vers $+\infty$).

Voici un tableau pour avoir une idée de la difficulté croissante pour multiplier et factoriser des nombres à n chiffres :

n	multiplications	factorisations
3	9	320
4	16	572
5	25	934
10	100	5 528
50	2 500	2 510 835
100	10 000	115 681 968
200	40 000	14 423 748 780

IV.3 Fonctions à sens unique.

Il existe bien d'autres situations mathématiques asymétriques : les fonctions à sens unique. En d'autres termes, étant donnée une fonction f, il est possible connaissant x de calculer « facilement » $f(x)$; mais, connaissant un élément de l'ensemble image de f, il est souvent « difficile » ou impossible de trouver un antécédent.

Dans le cadre de la cryptographie, posséder une fonction à sens unique qui joue le rôle de chiffrement n'a que peu de sens. En effet, il est indispensable de trouver un moyen efficace afin de pouvoir déchiffrer les messages chiffrés. On parle alors de fonction à sens unique avec trappe secrète.

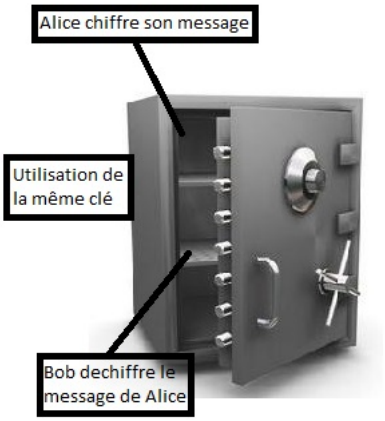
Prenons par exemple le cas de la fonction $f \ : \ x \mapsto x^3 \pmod{100}$.
- Connaissant x, trouver $y = f(x)$ est facile, cela nécessite deux multiplications et deux divisions.
- Connaissant y image par f d'un élément x, retrouver x est difficile.

Tentons de résoudre le problème suivant : trouver x tel que $x^3 \equiv 11 \pmod{100}$.
On peut pour cela :
- soit faire une recherche exhaustive, c'est-à-dire essayer successivement 1, 2, 3, ..., 99, on trouve alors : $71^3 = 357\,911 \equiv 11 \pmod{100}$,
- soit utiliser la trappe secrète : $y \mapsto y^7 \pmod{100}$ qui fournit directement le résultat :
$11^7 = 19\,487\,171 \equiv 71 \pmod{100}$.

La morale est la suivante : le problème est dur à résoudre, sauf pour ceux qui connaissent la trappe secrète. (Attention, dans le cas de cet exemple, la fonction f n'est pas bijective.)

IV.4 Chiffrement à clé privée.

Imaginons que Alice souhaite envoyer un message secret à Bob.
Les protocoles étudiés dans les paragraphes précédents étaient des chiffrements à clé privée.
De façon imagée, tout se passe comme si Alice pouvait déposer son message dans un coffre fort pour Bob ; Alice et Bob étant les seuls à posséder la clé du coffre.



En effet, jusqu'ici, les deux interlocuteurs se partageaient une même clé qui servait à chiffrer (et déchiffrer) les messages.
Cela pose bien sûr un problème majeur : ici, Alice et Bob doivent d'abord se communiquer la clé.

IV.5 Chiffrement à clé publique.

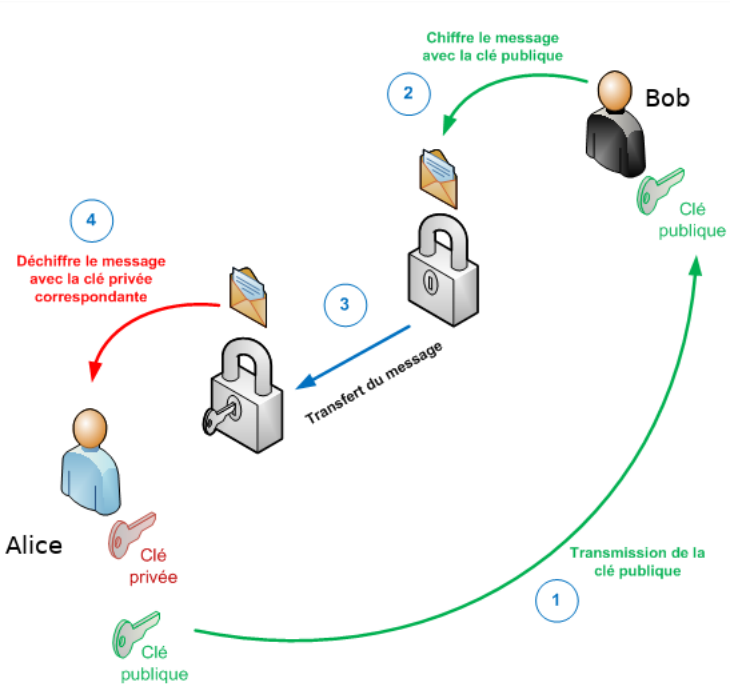
Les fonctions à sens unique à trappe donnent naissance à des protocoles de chiffrement à clé publique. L'association « clé » et « publique » peut paraître incongrue, mais il signifie que le principe de chiffrement est accessible à tous mais que le déchiffrement nécessite une clé qu'il faut bien sûr garder secrète.

De façon imagée, si Alice veut envoyer un message à Bob, elle dépose son message dans la boîte aux lettres de Bob, seul Bob pourra ouvrir sa boîte et consulter le message. Ici la clé publique est symbolisée par la boîte aux lettres, tout le monde peut y déposer un message, la clé qui ouvre la boîte aux lettres est la clé privée de Bob, qu'il doit conserver à l'abri.

En prenant appui sur l'exemple précédent, si le message initial est 71 et que la fonction f de chiffrement est connue de tous, le message transmis est 11 et le déchiffrement sera rapide si la trappe secrète 7 est connue du destinataire.

Les paramètres d'un protocole de chiffrement à clé publique sont donc :
- les fonctions de chiffrement et de déchiffrement : $\mathcal{C}$ et $\mathcal{D}$,
- la clé publique du destinataire qui va permettre de paramétrer la fonction $\mathcal{C}$,
- la clé privée du destinataire qui va permettre de paramétrer la fonction $\mathcal{D}$.

Dans le cadre de notre exemple Bob souhaite envoyer un message à Alice, ces éléments sont :
- $\mathcal{C} : x \mapsto x^? \pmod{100}$ et $\mathcal{D} : x \mapsto x^? \pmod{100}$,
- **3** : la clé publique d'Alice qui permet de définir complètement la fonction de chiffrement : $\mathcal{C} : x \mapsto x^3 \pmod{100}$,
- **7** : la clé privée d'Alice qui permet de définir complètement la fonction de déchiffrement : $\mathcal{D} : x \mapsto x^7 \pmod{100}$.



V Le chiffrement RSA.

Le chiffrement RSA (nommé par les initiales de ses trois inventeurs) est un algorithme de cryptographie asymétrique, très utilisé dans le commerce électronique, et plus généralement pour échanger des données confidentielles sur Internet. Cet algorithme a été décrit en 1977 par Ronald Rivest, Adi Shamir et Leonard Adleman.

Dans ce paragraphe, c'est Bob qui veut envoyer un message secret à Alice. La processus se décompose ainsi :

1. Alice prépare une clé publique et une clé privée,
2. Bob utilise la clé publique d'Alice pour crypter son message,
3. Alice reçoit le message crypté et le déchiffre grâce à sa clé privée.

V.1 Calcul de la clé publique et de la clé privée.

V.1.a Choix de deux nombres premiers.

Alice effectue, une fois pour toute, les opérations suivantes (en secret) :
- elle choisit deux nombres premiers distincts p et q (dans la pratique, ce sont de très grand nombres, jusqu'à des centaines de chiffres),
- elle calcule $n = p \times q$,
- elle calcule $\phi(n) = (p-1) \times (q-1)$.

Exemple :
- $p = 5$ et $q = 17$,
- $n = p \times q = 85$
- $\phi(n) = (p-1) \times (q-1) = 64$

Vous noterez que le calcul de $\phi(n)$ n'est possible que si la décomposition de n sous la forme $p \times q$ est connue. D'où le caractère secret de $\phi(n)$ même si n est connu de tous.

V.1.b Choix d'un exposant et calcul de son inverse.

Alice continue :
- elle choisit un exposant e tel que $pgcd(e, \phi(n)) = 1$,
- elle calcule l'inverse d de e modulo $\phi(n)$ c'est à dire : $d \times e \equiv 1 \pmod{\phi(n)}$.

Ce calcul se fait par l'algorithme d'Euclide étendu (voir exercices).

Exemple :
- Alice choisit par exemple $e = 5$ et on a bien $pgcd(e, \phi(n)) = pgcd(5, 64) = 1$,
- Alice cherche des entiers relatifs u et v qui vérifient $ue + v\phi(n) = 1$.

On utilisera l'algorithme d'Euclide étendu pour calculer u et v (coefficients de Bézout : cf exercices).
On trouve $u = 13$ et $v = -1$, car $5 \times 13 + 64 \times (-1) = 1$.
Donc $5 \times 13 \equiv 1 \pmod{64}$ et l'inverse de e modulo $\phi(n)$ est $d = 13$.

V.1.c Clé publique.

La clé publique d'Alice est constituée des deux nombres : $\boxed{n \text{ et } e}$

Et comme son nom l'indique Alice communique sa clé publique au monde entier.

Exemple : $n = 85$ et $e = 5$.

V.1.d Clé privée.

Alice garde pour elle sa clé privée : $\boxed{d}$

Alice détruit en secret p, q et $\phi(n)$ qui ne sont plus utiles. Elle conserve secrètement sa clé privée.

Exemple : $d = 13$.

V.2 Chiffrement du message.

Bob veut envoyer un message secret à Alice. Il se débrouille pour que son message soit un entier (quitte à découper son texte en blocs et à transformer chaque bloc en un entier).

V.2.a Message.

Le message est un entier m, tel que $0 \leqslant m < n$.

Exemple : Bob veut envoyer le message $m = 10$.

V.2.b Message chiffré.

Bob récupère la clé publique d'Alice : n et e avec laquelle il calcule, à l'aide de l'algorithme d'exponentiation rapide , le message chiffré : $\boxed{x \equiv m^e \pmod{n}}$.
Il transmet ce message x à Alice.

Exemple : $m = 10$, $n = 85$ et $e = 5$ donc

$$x \equiv m^e \pmod{n} \equiv 105 \pmod{85}$$

On peut ici faire les calculs à la main :
$10^2 \equiv 100 \equiv 15 \pmod{85}$
$10^4 \equiv (10^2)^2 \equiv 15^2 \equiv 225 \equiv 55 \pmod{85}$
$x \equiv 10^5 \equiv 10^4 \times 10 \equiv 55 \times 10 \equiv 550 \equiv 40 \pmod{85}$
Le message chiffré est donc $x = 40$.

V.2.c Déchiffrement.

Alice reçoit le message x chiffré par Bob, elle le décrypte à l'aide de sa clé privée d, par l'opération :
$\boxed{m \equiv x^d \pmod{n}}$
qui utilise également l'algorithme d'exponentiation rapide.
On pourrait prouver que, par cette opération, Alice retrouve bien le message original m de Bob (exercice ?).

Exemple : $x = 40$, $d = 13$, $n = 85$ donc $x^d \equiv (40)^{13} \pmod{85}$.
Calculons à la main $40^{13} \pmod{85}$, on note que $13 = 8 + 4 + 1$, donc $40^{13} = 40^8 \times 40^4 \times 40$.
$40^2 \equiv 1600 \equiv 70 \pmod{85}$
$40^4 \equiv (40^2)^2 \equiv 70^2 \equiv 4900 \equiv 55 \pmod{85}$
$40^8 \equiv (40^4)^2 \equiv 55^2 \equiv 3025 \equiv 50 \pmod{85}$
Donc :

$$x^d \equiv 40^{13} \equiv 40^8 \times 40^4 \times 40 \equiv 50 \times 55 \times 40 \equiv 10 \pmod{85}$$

qui est bien le message m de Bob.

V.3 Synthèse.

Système RSA

clé privée (n, d)
clé publique (n, e)

M est le message que l'on veut chiffrer.

Chiffrement : $C \equiv M^e \ [n]$

Déchiffrement : $M \equiv C^d \ [n]$

VI Principes de base en cryptographie.

- **Performance :** Un protocole sûr vaut mieux qu'un protocole efficace.
- **Simplicité :** Un protocole ne doit jamais essayer de faire plus que ce qu'il est sensé faire.
- **Le Maillon Faible :** Un protocole n'est jamais aussi sûr que sa composante la plus faible.
- **Raisonnement paranoïaque :** Un protocole avec une faiblesse, aussi petite qu'elle soit, est un protocole qui n'est plus sûr.
- **Modèle de sécurité :** Un protocole n'est jamais parfait. L'essentiel est d'obtenir le niveau de sécurité souhaité.
  Exemple : le réseau du Lycée Blaise Pascal

VII Exercices

Exercice 1. *Code César.*

Écrire une fonction **decal(lettre,p)** *qui à* lettre *renvoie la lettre décalée de p si c'est une lettre minuscule. Par exemple* decal('m',3) *donne 'p'.*

Écrire une fonction **cesar(ch,p)** *qui décale les lettres minuscules de* ch *de p lettres dans l'ordre alphabétique sans modifier les autres.*

Écrire une fonction **decod_cesar(ch)** *qui teste les 25 décodages possibles. Cette fonction renverra une liste de tuples contenant pour chaque tuple, le texte* ch *décodé et la clé correspondante.*

Décoder le message suivant : "s uij jubbucudj vqsybu b ydvehcqjygku".

La méthode que nous avons utilisée s'appelle la méthode de force brute (vérification de toutes les combinaisons possibles). Cette méthode est très lente (bien qu'aisée pour le code César qui n'a que 25 clés possibles).

1. *Sur un codage par décalage, la lettre la plus fréquente a de grandes chances d'être un "e", on peut ainsi en déduire le décalage.*
   Copier le fichier txt0.txt *situé dans votre répertoire de travail.*
   Écrire un code Python qui va compter le nombre de fois qu'apparaît chaque lettre dans le fichier txt0.txt *et qui renvoie* [['a', 106], ['b', 90], ['c', 140],...] *et en déduire le déchiffrement du message. De quel livre est-ce un extrait ?*

2. *Essayer cette méthode avec le fichier* txt1.txt *(copier le fichier dans le même répertoire que ci-dessus) et constatez qu'elle ne fonctionne pas.*
   Utilisez Python pour remplir le tableau suivant :

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*
...	...	...	...	...	...	...	...	...	...	...	...	...

   Comment en déduire la clé probable ?
   Décoder alors le fichier txt1.txt.
   Quel est sa particularité ? De quel livre est-ce un extrait ?

Exercice 2. *L'algorithme de Vigenère.*

1. *Uniquement si vous ne pouvez pas directement faire les questions 2 et 3.*
   Écrire une fonction cle_vig(texte,cle) *qui répète les caractères de la clé* cle *jusqu'à ce qu'elle atteigne la longueur du texte* texte*.*

2. *Écrire une fonction* code_vig(texte,cle) *qui code le texte* texte *avec la clef* cle *avec l'algorithme de Vigenère.*

3. *Écrire une fonction* decode_vig(texte,cle) *qui décode le texte* texte *avec la clef* cle *avec l'algorithme de Vigenère.*

4. ⋆ *(A ne traiter que si tout le reste a été fait) Le document* txt2.txt *est un texte codé avec l'algorithme de Vigenère dont on ignore la longueur p de la clé. Pour la trouver, on commence par créer un nouveau texte ne contenant que les lettres codées, ensuite on extrait une lettre sur p et on regarde la répartition statistique, on peut deviner alors la longueur de la clé, puis la clé...*
   On peut trouver la longueur de la clé avec l'indice de coincidence (inventé par William F. Friedman en 1920). Dans un (sous) texte, on calcule $IC = \sum_{i=0}^{25} \frac{n_i(n_i-1)}{n(n-1)}$ *où* n_i *est le nombre d'occurrences de la ième lettre et n le nombre total de lettres du texte. En français, l'indice de coincidence vaut environ* $0,078$*. Dans le cas de lettres uniformément distribuées (contenu aléatoire sans biais), l'indice se monte à* $0,0385$*.*
   Déterminer la longueur de la clé qui a servi à chiffrer le fichier txt2.txt*, puis, grâce à l'analyse des fréquences, déterminer la clé et déchiffrer le fichier.*

Exercice 3. *Le chiffrement RSA.*

1. *Écrire une fonction récursive de l'algorithme d'exponentiation rapide* $puiss\_rapide(m,e,n)$ *qui renvoie* $m^e \pmod n$*.*
   On pourra utiliser les égalités suivantes : $m^{2k} = (m^k)^2$ *et* $m^{2k+1} = m \times (m^k)^2$ *et les appliquer modulo(n).*

2. *L'algorithme d'Euclide étendu.*

   (a) *L'algorithme d'Euclide repose sur le principe que* $pgcd(a,b) = pgcd(b, a \pmod b)$ *et renvoie* $pgcd(a,b)$*.*
   Écrire une fonction $euclide(a,b)$ *qui renvoie* $pgcd(a,b)$*.*

   (b) *L'algorithme d'Euclide étendu permet d'obtenir les coefficients de Bézout u et v tels que* $au + bv = pgdc(a,b)$*. Pour cela, on définit deux suites* (x_i) *et* (y_i) *qui vont aboutir aux coefficients de Bézout.*

   $$\textit{L'initialisation est :} \quad x_0 = 1\,, \quad x_1 = 0\,, \quad y_0 = 0\,, \quad y_1 = 1$$

   et la formule de récurrence pour $i \geqslant 1$ *:*

   $$x_{i+1} = x_{i-1} - q_i x_i \qquad y_{i+1} = y_{i-1} - q_i y_i$$

   où q_i *est le quotient de la division euclidienne de* a_i *par* b_i*.*
   Écrire une fonction $euclide\_etendu(a,b)$ *qui renvoie d'abord* δ *le* $pgcd(a,b)$ *puis les coefficients de Bezout sous la forme d'un triplet* (δ,u,v)*.*

3. *La mise en oeuvre du chiffrement RSA est maintenant relativement simple.*

(a) *A l'aide de la fonction euclide_etendu, écrire une fonction cle_privee(p, q, e) qui renvoie la clé privée.*

(b) *Le chiffrement d'un message m est possible par tout le monde, connaissant la clé publique (n, e).*
Écrire une fonction codage_rsa(m, n, e) qui renvoie x le message m codé.

(c) *Seul le détenteur de la clé privée peut maintenant déchiffrer le message crypté x, à l'aide de sa clé privée d. Écrire une fonction decodage_rsa(x, n, d) qui renvoie m le message original.*