# TD 12 : L'algorithme RSA
**Avril 2013**

On étudie ici l'algorithme RSA, une méthode de chiffrement des données utilisée très largement dans le monde depuis les années 80, pour sécuriser notamment les cartes de crédit. Ce sera l'occasion de manipuler des chaînes de caractères avec Maple, et aussi de faire un peu d'arithmétique.

## 1 La cryptographie asymétrique

**Cryptographie symétrique** Vous connaissez sans doute déjà des algorithmes de chiffrement des données : le chiffre de César (on remplace chaque lettre par la i-ième suivante dans l'alphabet, avec i fixé), le chiffre de Vigenère, la machine Enigma utilisée par les Allemands pendant la deuxième guerre mondiale, etc... Les exemples sont nombreux.

On se rend compte que toutes ces méthodes se fondent sur le même principe de base. On fait subir au message à transmettre une transformation plus ou moins alambiquée, mais dont la *clé* est connue du destinataire, et qui lui permet de déchiffrer le message. On dit que ces méthodes relèvent de la cryptographie *symétrique*, puisque la clé est commune à la personne qui envoie et à celle qui reçoit le message.

Ces méthodes, même si elles peuvent être théoriquement inviolables sans la clé, nécessitent que les deux parties se mettent d'accord au préalable sur la clé avant de pouvoir communiquer de façon cachée. Ce point de détail, vous l'imaginez, peut se révéler gênant.

**Deux clés différentes** La cryptographie asymétrique pallie ce problème en distinguant la clé avec laquelle le message est chiffré (la *clé publique*) et celle avec laquelle on déchiffre le message codé (la *clé privée*). On peut voir la clé publique comme un cadenas ouvert distribué un peu partout, et la clé privée comme la clé correspondant au cadenas.

Imaginons que vous vouliez m'envoyer un message secret. Il vous faut tout d'abord récupérer ma clé publique, disponible dans un annuaire de clés, et de vous en servir pour crypter votre message. Ce message, moi seul pourrai l'ouvrir avec ma clé privée.

Le problème qui se pose tout de suite, c'est de trouver une fonction-*trappe*, c'est à dire une fonction bijective facile à calculer, mais dont l'inverse est très très dur, voire impossible à déterminer. Dans le cas de RSA, cette fonction trappe, c'est la multiplication de deux nombres premiers. En effet, le temps de calcul de la factorisation d'un entier $n$ en facteurs premiers augmente exponentiellement avec la taille de $n$ ; du moins on ne sait pas encore faire mieux.

## 2 Chaînes de caractères en Maple

Maple donne un type aux suites de symboles écrites entre deux *quotes* : ". Vous pouvez omettre ces quotes, mais seulement pour les suites de lettres sans espaces. Par exemple, `"sapajou"`, `URUbu` et `"Comment est votre blanquette?"` ont le type `string`, mais pas `crème caramel`, qui a un caractère interdit et un espace.

LA fonction utile sur les chaînes, c'est `substring`. En effet, `substring(chaine,i)` renvoie le i-ème caractère de la chaîne donnée en argument. Vous en savez assez pour faire l'exercice suivant.

**Exercice 1 (*ROT13*)**

*La méthode ROT13 est une variante du chiffre de César, dans laquelle chaque lettre est remplacée par la lettre 13 rangs plus loin dans l'alphabet (modulairement, pour ne pas déborder). On ne travaillera qu'avec des lettres de l'alphabet en minuscules.*

1. *Ecrire une fonction qui prend un mot en entrée, et qui renvoie la liste des lettres qui le compose.*
2. *Ecrire une fonction qui, à partir d'une liste de lettres, renvoie la liste des entiers correspondant au rang de la lettre de la liste de base. Procéder comme suit :*
   (a) *Définir une liste [a,b,c,d,...,z].*
   (b) *Ecrire une fonction* `rechercher:= proc(x,l)` *qui renvoie le rang où apparaît* `x` *dans*

*la liste* `l`

(c) *En déduire une fonction* `rangLettre := proc (x)`*, puis conclure.*

3. *Ecrire l'inverse de la fonction précédente*
4. *Ecrire l'inverse de la fonction de la question 1. On utilise* `cat(a,b)` *pour concaténer les chaînes* `a` *et* `b`
5. *Combiner toutes ces fonctions pour écrire* `rot13:=proc(s)`*, qui prend une chaîne et qui renvoie la chaîne cryptée par ROT13.*
6. *Décoder* `bqrabgerobaurhegbvyrsngnyrzoyrzr`

# 3 Le cryptosystème RSA

## 3.1 Remplissage

Un algorithme cryptographique nécessite une entrée (le message à crypter) sous une forme particulière, et rien ne garantit que le message que vous voulez transmettre est sous la bonne forme. Une étape préalable à l'algorithme est ce qu'on appelle le *remplissage* (padding, en anglais). Cela consiste en la transformation (bijective, impérativement) de votre message en une entrée sous une meilleure forme.

**Exercice 2**

*L'algorithme RSA crypte des entiers, il faut donc trouver un moyen de transformer bijectivement une suite de lettres en suite d'entiers, sans ambigüité. Ce que l'on va faire, c'est d'abord découper le texte à coder en blocs de deux lettres. On codera ensuite chaque bloc par l'entier formé des deux codes des symboles du bloc mis côte-à-côte. On doit définir ce code nous-même. Pour que notre système soit bien bijectif, on va faire en sorte que chaque symbole ait un code à deux chiffres (donc entre 10 et 99).*

1. *Je vous propose le code suivant (c'est celui que j'utiliserai pour les messages à déchiffrer de la fin du TD) : a=10, b=11, ..., z=35, A=36, ..., Z=61, ' '=62, '0'=63, '1'=64, ..., '9'=73. Ecrire une fonction* `code := proc(symbole)` *en vous inspirant de l'exercice sur ROT13*
2. *Ecrire une fonction* `decoupe:=proc(mot)` *qui renvoie la liste des blocs de deux symboles composant le mot. Utiliser la fonction* `length(s)` *qui donne la longueur d'une string. Si la taille du mot n'est pas un multiple de 2, lui rajouter un espace à la fin.*
3. *Ecrire la fonction* `codebloc:=proc(s)`*, qui code un bloc de 2 lettres.*
4. *Un peu plus délicat : coder* `decodebloc:=proc(n)`*, l'inverse de cette fonction.*
5. *Ecrire enfin* `codemot:=proc(s)`*, qui renvoie une liste d'entiers donnés par* `codebloc`*, et écrire* `decodemot := proc(l)`*, qui donne un mot à partir d'une liste de codes de blocs.*

## 3.2 Principe de l'algorithme

Avec vos connaissances en arithmétique, vous pouvez comprendre le fonctionnement de l'algorithme, qui n'est pas si compliqué que ça.

**Création des clés** Avant de pouvoir recevoir des messages codés avec RSA, il faut posséder une clé publique, et la clé privée correspondante. On procède comme suit :

1. On choisit deux nombres premiers au hasard $p$ et $q$ (plus ils sont grands, plus le cryptage est sûr, mais plus le temps de calcul est long)
2. On calcule $\phi(n)$, l'indicatrice d'Euler du produit $n = pq$
3. On choisit un entier $e$ au hasard, premier avec $\phi(n)$. On l'appelle l'exposant d'encryptage.
4. On choisit enfin $d$, un entier tel que $ed \bmod \phi(n) = 1$. C'est l'exposant de décryptage.

Une fois ces étapes accomplies, on dispose d'une clé publique : le couple $(e, n)$, et d'une clé privée, le couple $(d, n)$. Voyons maintenant comment on se sert de ces clés pour crypter des entiers.

**Cryptage et décryptage** On suppose disposer d'un message sous la forme d'un entier $M$. On définit alors le cryptage du message $M$ par la clé publique $(e, n)$ comme $C = M^e \bmod n$. On retrouve l'original $M$ à partir de $C$ et de la clé privée $(d, n)$ par $M = C^d \bmod n$.

**Exercice 3 (*Un peu d'arithmétique*)**

*Voyons si vous avez compris le principe.*

1. *Dans la création des clés, comme $p$ et $q$ sont des nombres premiers, peut-on exprimer plus simplement $\phi(pq)$ ?*
2. *Est-on garanti de l'existence d'un entier $d$ vérifiant ce que l'on veut sur le produit $ed$ ?*
3. *Comme on les a définies, est-ce que l'opération de décryptage est bien l'inverse de l'opération de cryptage ?*

## 3.3 Programmation

On peut maintenant se lancer sans retenue dans la programmation de cet algorithme. Quelques remarques :

- Il est *crucial* que vos entiers $p$ et $q$ donnent un produit $n$ à 5 chiffres au moins, pour que la taille des messages à coder ou décoder soit toujours inférieure à $n$.
- Ne pas prendre non plus de trop gros entiers, ça ne fait qu'augmenter votre temps de calcul.
- Pour trouver des nombres premiers au hasard, une bonne idée est de combiner les fonctions `ithprime` et `rand`.

**Exercice 4**

*Les questions suivantes ne peuvent pas forcément toutes être résolues en une seule procédure. Réfléchissez, si besoin avec une feuille de papier et un crayon, à ce que vous avez besoin avant de coder ; vous ferez moins d'erreurs.*

1. *Créer un jeu de clés, publique et secrete.*
2. *Avec la convention du code de l'exercice 2, écrire une fonction qui code un mot avec RSA, et une autre qui le décode.*
3. *Envoyez un message crypté à votre voisin de gauche, avec sa clé publique ; et avec votre clé privée, décryptez le message que vient de vous envoyer votre voisin de droite.*

Pour tester vos procédures, je vous donne les exemples suivant, avec, à chaque fois, le résultat que vous devez obtenir :

1. Avec la clé publique $(505823; 961801)$, le mot `maple` est codé par $[396970, 108540, 62531]$ ;
2. De même, la clé $(38161; 847879)$ crypte le message `"Double double toil and trouble"` en $[627030, 55006, 592960, 521189, 552767, 479739, 508823, 8181, 672447, 745423, 656498, 104894, 466743, 55006, 592960]$ ;
3. Avec la clé privée $(265933; 488039)$, le message $[416527, 98416, 256794, 413062, 371856, 317612, 77689, 329251]$ se décrypte en `"The answer is 42"`.

**Exercice 5 (*Bonus : Casser RSA*)**

*Pour retrouver le $d$ de la clé privée avec le $e$ de la clé publique, il faut connaître $\phi(n) = \phi(pq) = (p-1)(q-1)$, ce qui suppose que l'on sache factoriser $n$. Si c'est presque impossible pour $n$ grand (même impossible tout court, rassurez-vous, ne jetez pas encore votre carte de crédit), c'est faisable pour des entiers plus petits. Essayez de décrypter les messages suivants, où seules les clés publiques sont données.*

1. *Le message : [9197, 6284, 12836, 8709, 4584, 10239, 11553, 4584, 7008, 12523, 9862, 356, 5356, 1159, 10280, 12523, 7506, 6311] a été crypté avec la clé publique (12413 ; 13289)*
2. *On a crypté [2692487, 2081187, 2739461, 808175, 642354, 3151712, 2669957, 3594969, 3736381] avec (27583 ; 4003997)*
3. *Attention, celui-ci est ardu. Clé publique : (163119273 ; 755918011) ; message crypté : [671828605, 407505023, 288441355, 679172842, 180261802]*