*Agrégation externe de mathématiques, session 2006*
*Épreuve de modélisation, option calcul scientifique : méthodes numériques et symboliques*

---

# (*583*) CRYPTOGRAPHIE ET FACTORISATION

**Résumé :** Ce texte comporte deux parties : dans la première, on expose l'exemple du code RSA, qui repose sur le fait qu'on ne sait pas factoriser rapidement un nombre entier.
Dans la seconde, on présente l'algorithme $\rho$ de Pollard, qui permet de factoriser un entier $n$ en $O(N^{1/4})$ opérations "élémentaires" (alors que l'algorithme naïf, qui consiste à diviser $N$ par les entiers $\leq \sqrt{N}$, est en $N^{1/2}$ telles opérations.)

**Thème applicatif, mots clefs :** Cryptographie, RSA, factorisation d'entiers, méthode $\rho$ de Pollard.

---

▶ *Il est rappelé que le jury n'exige pas une compréhension exhaustive du texte. Vous êtes laissé(e) libre d'organiser votre discussion comme vous l'entendez. Des suggestions de développement, largement indépendantes les unes des autres, vous sont proposées en fin de texte. Vous n'êtes pas tenu(e) de les suivre. Il vous est conseillé de mettre en lumière vos connaissances à partir du fil conducteur constitué par le texte. Le jury demande que la discussion soit accompagnée d'exemples traités sur ordinateur.*

## 1. Introduction

Il y a à peine quelques années, le problème de la sécurité des transmissions de données semblait être l'apanage des seuls militaires. Ce n'est plus le cas, avec l'essor des techniques numériques dans le commerce (Internet, les cartes de crédit, les télécommunications, les décodeurs de télévision, etc.)

Les méthodes empiriques traditionnelles se sont révélées trop fragiles ; elles reposaient souvent sur le schéma suivant : on choisit un livre, une fois pour toutes, et on considère la permutation des vingt-six lettres de l'alphabet définie par les vingt-six premières lettres de ce livre.

Le codage d'un texte consiste alors à appliquer cette permutation $\sigma$ au texte à coder, et le décodage à appliquer la permutation réciproque $\sigma^{-1}$ au texte à décoder.

En numérique, si par exemple le message à transmettre est codé sur 64 bits, on peut employer cette technique en considérant une permutation $\sigma \in S_{64}$. C'est ce genre d'idées qui est sous-jacent au procédé DES, dû à IBM, et qui est encore le plus utilisé en informatique.

Le talon d'Achille de ce genre de cryptosytème est le suivant : si quelqu'un sait coder, il sait aussi décoder, car il est très facile de trouver la permutation réciproque d'une permutation sur 26, 64, 128 ou même 256 lettres.

C'est pourquoi l'apparition de cryptosystèmes dits *à clé publique*, à la fin du siècle dernier, a fait sensation.

Ces cryptosystèmes, comme le nom l'indique, sont tels que le *codage* est public : tout le monde connaît l'algorithme pour coder le message. Mais on ne peut pas en déduire le décodage.

En fait, cela revient à construire une permutation $\sigma$ d'un ensemble à $N$ éléments, mais ici $N$ est gigantesque (de l'ordre de $10^{500}$, par exemple.) On ne peut même pas écrire la liste de ces éléments, et la méthode habituelle pour trouver la permutation réciproque d'une permutation donnée ne peut plus s'appliquer.

Dans le paragraphe suivant, nous décrivons le cryptosystème le plus célèbre de ce genre : la méthode RSA, du nom de ses inventeurs Rivest, Shamir et Adelman.

## 2. RSA

Cette méthode est basée sur le théorème suivant :

**Théorème.-** *Soit $G$ un groupe fini d'ordre $N$ et $m$ un entier premier à $N$. L'application $f_m$ : $G \to G$ définie par $x \mapsto x^m$ est bijective, et son application réciproque est l'application $f_k$ définie par $x \mapsto x^k$, où $k$ est l'inverse de $m$ dans le groupe multiplicatif $(\mathbf{Z}/N\mathbf{Z})^*$ (i.e. l'ensemble des éléments inversibles de $\mathbf{Z}/N\mathbf{Z}$).*

Si l'on connaît $N$ et $m$, on en déduit donc rapidement $k$, par exemple par l'algorithme d'Euclide étendu, qui permet d'obtenir des entiers $k$ et $l$ tels que $mk + lN = 1$.

Par ailleurs, pour $x \in G$, le calcul de $x^k$ se fait lui aussi rapidement, en $O(\log k)$ opérations dans le groupe $G$, en écrivant $k$ en binaire. L'algorithme RSA consiste à fournir au codeur un groupe $G$ dans lequel il sait calculer, mais dont il ne connaît pas l'ordre $N$, et un certain entier $m$. Le codage consiste alors, à partir d'un message $x$ identifié à un élément de $G$, à calculer $x^m$. Le décodage nécessite de connaître $k$, inverse de $m$ dans $(\mathbf{Z}/N\mathbf{Z})^*$. Dans les cas utilisés dans la pratique, on ne sait pas comment trouver $k$ sans connaître $N$.

Donnons l'exemple habituel de groupe $G$ utilisé : soient deux nombres premiers $p$ et $q$, et soient $M = pq$, et $G = (\mathbf{Z}/M\mathbf{Z})^*$. Il suffit de connaître $M$ pour savoir calculer dans $G$ (et rapidement : les additions et multiplications se font en $O(\log^2 M)$ opérations élémentaires sur les chiffres de $M$), mais on ne connaît l'ordre de $G$, à savoir $(p-1)(q-1)$ que si l'on connaît $p$ et $q$, c'est-à-dire la décomposition en facteurs premiers de $M$.

On ne peut pas choisir $m = 2$ pour coder. Mais si l'on a choisi $p$ et $q$ tels que $p \equiv q \equiv 2 \mod 3$, on peut alors choisir $m = 3$ ; $x \mapsto x^m$ est bien une permutation de $G$, et on trouve immédiatement $k = \dfrac{2(p-1)(q-1)+1}{3}$.

La force de la méthode RSA réside dans le fait qu'à l'heure actuelle on ne connaît pas de méthode rapide de factorisation d'un entier : il faut actuellement des mois et plusieurs centaines de stations de travail travaillant en parallèle pour factoriser un entier de l'ordre de 150 chiffres.

**Quelques remarques.**

- Supposons que le mot à coder soit un entier $x < \sqrt[3]{M}$. On retrouve alors rapidement $x$ à partir de $x^3$, sans rien savoir de la factorisation de $M$.
- Supposons que l'on ait un même entier $x$ à coder pour, par exemple, trois destinataires différents. On pourrait penser qu'il est plus prudent de considérer trois entiers $M_1, M_2$ et $M_3$ distincts (et $> x$) pour le codage RSA. En fait, il n'en est rien : une personne

mal intentionnée qui connaît les trois messages codés $x^3 \mod M_1, x^3 \mod M_2$ et $x^3 \mod M_3$ reconstitue immédiatement $x$ par le théorème chinois !

– Au lieu de prendre $m = 3$, on peut prendre tout $m$ premier à $(p-1)(q-1)$. L'intérêt de prendre $m$ petit est que le codage est d'autant plus rapide, mais $k$ est alors grand, et le décodage est plus lent. Par ailleurs, comme les deux remarques précédentes le montrent, $m$ petit a des effets pervers. C'est pourquoi il a été proposé de prendre d'autres valeurs de $m$, même très grandes, mais pour lesquelles le codage reste rapide, par exemple $m = 65537$.

## 3. Factorisation d'entiers

La méthode RSA est sûre dans la mesure où l'on ne sait pas factoriser rapidement un entier.

La méthode naïve pour factoriser un entier $M$ est en $O(M^{1/2})$ opérations : on divise $M$ par tous les entiers inférieurs à $\sqrt{M}$.

Nous décrivons dans ce paragraphe une méthode probabiliste de factorisation d'un entier $M$ en $O(M^{1/4})$ opérations.

Elle est basée sur le phénomène dit *paradoxe des anniversaires* : si une assemblée a au moins 23 personnes, il y a plus d'une chance sur deux pour que deux personnes soient nées le même jour. Plus généralement, soit $p$ un entier $\geq 1$, et soient $q$ éléments de $\mathbf{Z}/p\mathbf{Z}$ tirés "au hasard". Quelle taille doit avoir $q$ pour que la probabilité $P$ que deux d'entre eux soient égaux soit supérieure à $1/2$ ?

La probabilité pour qu'ils soient distincts est $Q = 1 - P$

$$= (1-1/p)(1-2/p)\dots(1-(q-1)/p)$$

$$\sim (1-q/(2p))^{q-1} \sim e^{-q(q-1)/(2p)}.$$

On a donc $P > 1/2$ dès que $q(q-1)/(2p) > \log 2$, soit $q \sim 1.18\sqrt{p}$.

Soit donc $n$ un entier à factoriser, et soit $p$ le plus petit nombre premier divisant $n$. Dans ce qui suit, notons $\bar{a}$ la classe de $a \mod p$. Soit $f : \mathbf{Z} \to \mathbf{Z}$ l'application définie par $x \mapsto x^2 + 1$. On part d'un entier $a$ choisi au hasard, et on pose $a_k = f^k(a) \mod n$, où $f^k$ est égal au $k$-ième itéré de $f$ (i.e. $f^k = f^{k-1} \circ f$, avec $f^1 = f$.)

Pour $k$ assez grand (en $O(\sqrt{p})$, d'après ce qui précède), la suite $(a_l)_{l \leq k}$ est telle qu'il existe $i < j \leq k$ tels que $\bar{a_i} = \bar{a_j}$. Le pgcd de $a_i - a_j$ et de $n$ est donc divisible par $p$, et en fait certainement égal à $p$. Néanmoins, il n'est pas question de conserver tous les $a_i$, et surtout de calculer tous les pgcd des $a_i - a_j$.

La clé de l'algorithme est le fait que

$$\bar{a_i} = \bar{a_j} \Rightarrow \bar{a_{i+s}} = \bar{a_{j+s}},$$

pour tout $s \geq 0$.

La suite $(\bar{a_l})$ est donc périodique à partir d'un certain rang (à savoir $i$), de période $j - i$.

On peut donc supposer $j > 2i$ ; en prenant $l = j - 2i$, on obtient $\bar{a_{j-i}} = \bar{a_{2(j-i)}}$.

Il existe donc $k$, de l'ordre de $O(\sqrt{p})$, tel que $\overline{a_k} = \overline{a_{2k}}$.
L'algorithme consiste donc à calculer simultanément $a_k$ et

$$b_k = a_{2k} = (f^2)(b_{k-1}),$$

jusqu'à ce que l'on ait $\text{pgcd}(a_k - a_{2k}, n) > 1$.

## Suggestions pour le développement

▶ *Soulignons qu'il s'agit d'un menu à la carte et que vous pouvez choisir d'étudier certains points, pas tous, pas nécessairement dans l'ordre, et de façon plus ou moins fouillée. Vous pouvez aussi vous poser d'autres questions que celles indiquées plus bas. Il est très vivement souhaité que vos investigations comportent une partie traitée sur ordinateur et, si possible, des représentations graphiques de vos résultats.*

- Le candidat pourra choisir de développer les points qu'il désire sur le texte. En ce qui concerne le passage sur machine, il pourra par exemple programmer l'algorithme de factorisation naïf et l'algorithme proposé à la fin du texte, et les comparer. Il pourra s'il préfère programmer la méthode RSA, pour $M$ et $m$ quelconques.
- Plusieurs points du texte sont affirmés sans démonstration. Le candidat est invité à les expliciter.
- Le candidat pourra par exemple démontrer le théorème de la section 2, et détailler les remarques concluant cette section. Par ailleurs, pourquoi la connaissance de l'ordre de $G$ implique-t-elle la connaissance de $p$ et $q$ ? Pourquoi ne peut-on pas choisir $m = 2$ ?
- Dans la section trois, il pourra détailler l'argument conduisant à la démonstration du paradoxe des anniversaires. Pourquoi *n'est-il pas question de conserver tous les $a_i$, et surtout de calculer* ... ?
- Pourrait-on choisir d'autres fonctions que $x \mapsto x^2 + 1$ ? Que se passe-t-il si l'on prend une fonction affine $x \mapsto ax + b$ ?